Décompositions de graphes et algorithmes efficaces

Michaël Rao

THÈSE

présentée à

l'Université Paul Verlaine – Metz

pour obtenir le titre de
Docteur de l'Université Paul Verlaine – Metz

Spécialité : Informatique

Michaël Rao

Décompositions de graphes et algorithmes efficaces

Soutenue à Metz, le 16 juin 2006

Composition du jury :

Président:	
Jean Fonlupt	Professeur à l'Université Pierre et Marie Curie – Paris 6
Directeur de thèse:	
Dieter Kratsch	Professeur à l'Université Paul Verlaine – Metz
Rapporteurs:	
Victor Chepoi	Professeur à l'Université de la Méditerranée – Aix-Marseille II
Michel Habib	Professeur à l'Université Denis-Diderot – Paris 7
Examinateurs:	
Andreas Brandstädt	Professeur à l'Université de Rostock
Grégory Kucherov	Directeur de recherche au CNRS, LIFL Lille

Laboratoire d'Informatique Théorique et Appliquée

Table des matières

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse Dieter Kratsch qui a su me montrer la voie de l'algorithmique de graphes en soutenant et en encadrant mes travaux depuis la licence.

J'exprime toute ma gratitude à Jean Fonlupt qui m'a fait l'honneur de présider le jury de cette thèse.

Je remercie également Victor Chepoi et Michel Habib, rapporteurs de cette thèse, pour leur lecture attentive et pour leurs commentaires qui ont permis d'améliorer le manuscrit.

Merci à Andreas Brandstädt et Grégory Kucherov d'avoir accepté de faire partie du jury de cette thèse.

Ma reconnaissance s'adresse également à mes coauteurs : Hans Bodlaender, Andreas Brandstädt, Dieter Kratsch, Grégory Kucherov, Fabien de Montgolfier, Pascal Ochem et Jeremy Spinrad. Sans oublier toutes les autres personnes avec qui j'ai eu des discussions scientifiques, en particulier Michel Habib, Mathieu Liedloff, Ross McConnell et Haiko Müller.

Je tiens à remercier les membres du département d'informatique de l'UFR MIM, qui m'ont intégré dans l'équipe pédagogique.

Merci également aux membres du Laboratoire d'Informatique Théorique et Appliquée de Metz. Je pense en particulier à mes amis doctorants : Mihaela, Mathieu, David, Thomas et Nicolas. Sans oublier les stagiaires que j'ai côtoyés : Pascal, Alain, Khalil et Serge.

Je tiens à remercier mes amis Christelle, Gilles, Lionel, Magali, Marc, Mihaela, Sébastien, Stéphane et Stéphanie pour leur soutien moral.

Enfin, je remercie mes parents, mon frère, ma soeur et ma famille pour leur soutien sans faille depuis toujours.

Introduction

Les décompositions de graphes jouent un rôle important en théorie et en algorithmique des graphes. La preuve d'une des plus importantes conjectures dans le domaine, la conjecture forte des graphes parfaits de Claude Berge, l'a encore montré récemment.

Le principal argument donné pour motiver l'étude des décompositions de graphes en algorithmique est qu'elles peuvent aider à résoudre efficacement certains problèmes sur les graphes.

Décomposition des graphes

On voit facilement comment on peut décomposer un ensemble, une liste, un arbre... Dans le cas d'un ensemble ou d'une liste, on peut les séparer en deux. Pour un arbre enraciné, on peut le séparer en ses sous-arbres enracinés aux différents fils de la racine et sa racine. Comme il n'y a pas de relation complexe entre les différentes parties de la décomposition, la structure originale peut être recréée à partir des sous-structures obtenues après la décomposition. Pour un ensemble, il s'agira de faire une union, pour une liste une concaténation, et pour un arbre enraciné, d'ajouter les sous-arbres comme sous-arbre de la racine.

Avec les graphes, cela se complique. Si l'on sépare l'ensemble des sommets en deux (ou plus), la relation d'adjacence peut ne pas être facilement reconstruite entre les différentes parties. En effet il faut dans le cas général une information additionnelle en espace $\Theta(n^2)$.

On est donc intéressé par des séparations telles que la relation entre les différentes parties soit simple. Une décomposition de graphes peut devenir intéressante si, par exemple, l'information additionnelle peut être stockée en espace $O(n)$.

Il y a globalement deux familles de décompositions de graphes. On peut chercher une séparation dans le graphe, et on disposera d'un paramètre pour évaluer la séparation, ou bien on peut essayer de chercher un motif connu dans le graphe.

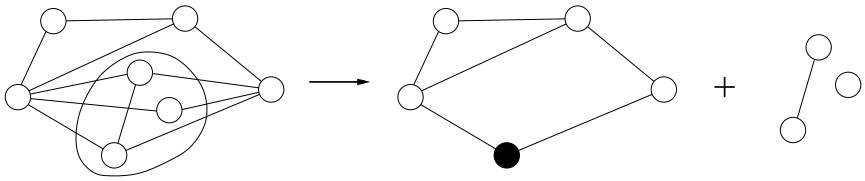
Décomposition modulaire et familles partitives

Le **module** est un exemple bien connu d'un motif que l'on peut trouver dans un graphe. Il s'agit d'un motif assez naturel que l'on peut retrouver dans plusieurs structures (ordres, graphes, hypergraphes...) : un sous-ensemble du domaine qui ne peut pas être distingué

par les éléments en dehors de l'ensemble. Ce sous-ensemble peut donc être « factorisé » dans la structure. Les modules ont beaucoup été étudiés et utilisés dans la littérature, et apparaissent sous différents noms : ensembles homogènes, intervalles, X-joints...

Dans le cas d'un graphe non orienté $G = (V, E)$, un **module** est un sous-ensemble non vide $M \subseteq V$ tel que tous les sommets $v \notin M$ sont soit adjacents à tous les sommets de M, soit adjacents à aucun sommet de M.

Certains modules sont triviaux (si $|M| = 1$ ou $M = V$). Ceux qui nous intéressent sont ceux qui permettent de décomposer le graphe d'une façon non triviale.



Tous les graphes n'ont pas des modules non triviaux, auquel cas ils sont dits **premiers** ou **indécomposables**. On peut décomposer entièrement le graphe en se servant de la décomposition par modules non triviaux, jusqu'à ce que tous les graphes obtenus soient premiers.

L'application récursive ne nous donne pas directement l'unicité de la décomposition. Mais Tibor Gallai [Gal67] a montré que l'unicité peut être obtenue en rajoutant certaines contraintes. Cette décomposition, unique et utilisant la factorisation des modules comme opération de base, est appelée la **décomposition modulaire**, et est souvent représentée par un arbre enraciné, **l'arbre de décomposition modulaire**. De plus, cette décomposition peut être calculée en temps linéaire [CH94, MS99].

L'unicité que l'on obtient pour la décomposition modulaire provient de certaines propriétés des modules. Une **famille partitive** (définie page 17) est une famille de sous-ensembles d'un ensemble qui possède certaines propriétés des modules. La famille des modules d'un graphe $G = (V, E)$ est une famille partitive de l'ensemble V. Les familles partitives ont été étudiées par Michel Chein, Michel Habib et Marie-Catherine Maurer [Hab81, CHM81].

Les familles partitives peuvent être représentées uniquement par un arbre enraciné. Dans le cas de la famille des modules, cet arbre est exactement l'arbre de décomposition modulaire.

Décomposition en coupes et familles bi-partitives

La **coupe** (définie page 57) est un autre motif, qui généralise le module. Elle peut être utilisée comme base pour définir la **décomposition en coupes**. William Cunningham [Cun82] montre que la décomposition en coupe devient unique sous certaines conditions. Il fait cela en montrant que les coupes respectent les propriétés du **cadre de décomposition**, défini par Jack Edmonds et lui même [CE80].

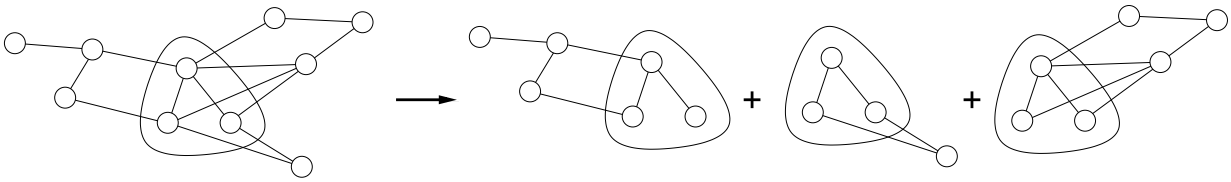
Une **famille bi-partitive** est une famille de bi-partitions d'un ensemble. Elles peuvent être vues comme la transposition des familles partitives sur les bi-partitions. Elles sont également uniquement représentables par un arbre, mais non enraciné.

Certaines propriétés des cadres de décomposition coïncident avec les propriétés des familles bi-partitives : il s'agit de deux concepts très proches, mais utilisant des formalismes différents. Toute décomposition respectant le cadre de décomposition avec les propriétés d'intersection et de transitivité correspond à une famille bi-partitive ; la famille de toutes les coupes d'un graphe est donc une famille bi-partitive.

Fabien de Montgolfier montre dans sa thèse [Mon03] qu'un autre motif, le **bi-joint**[1], respecte les propriétés des familles bi-partitives. Il en résulte qu'on peut définir une décomposition unique en utilisant le bi-joint, la **décomposition bi-joint**.

Décompositions paramétrées

Un exemple de l'autre famille de décompositions est le séparateur utilisé dans la **largeur arborescente**. La largeur arborescente a été introduite, puis beaucoup étudiée par Neil Robertson et Paul Seymour [RS86]. Un **séparateur** est un ensemble de sommets S tel que $G[V \setminus S]$ ne soit pas connexe. Le paramètre correspondant est la taille du séparateur. Lorsque l'on divisera le graphe suivant un séparateur S, on le divisera en l'ensemble des graphes $G_1, \ldots, G_l$, avec $G_i = G[S \cup C_i]$, où $C_1, \ldots C_l$ sont les composantes connexes de $G[V \setminus S]$.



La **décomposition arborescente** est l'application récursive de la décomposition par séparateur. La **largeur arborescente** d'un graphe est le plus petit entier k tel que le graphe soit entièrement décomposable par des séparateurs de taille au plus k (une définition équivalente sera donnée à la page page 13).

On peut donc dire que plus la largeur arborescente d'un graphe est petite, plus le graphe se prête à la décomposition arborescente. Les graphes de largeur arborescente 1 sont exactement les arbres et les forêts.

On remarque que si k est fixé, alors l'information nécessaire à stocker pour reconstruire le graphe G après sa séparation en G_1 et G_2 par un séparateur de taille au plus k est de taille constante. En effet il y a au plus k sommets dans G_1 et k sommets dans G_2 qui proviennent du séparateur, et il suffit de savoir quel sommet de G_1 correspond à quel sommet de G_2.

[1] Originellement appelé 2-joint par Fabien de Montgolfier.

Tout au long de cette thèse, on parlera beaucoup d'une autre largeur, la **largeur de clique** (définie page 14), introduite par Bruno Courcelle, Joost Engelfriet et Grzegorz Rozenberg [CER93], qui est une sorte de généralisation de la largeur arborescente, dans le sens où si une classe de graphes a une largeur arborescente bornée, alors sa largeur de clique est bornée. On parlera aussi de la **largeur NLC**, qui est une largeur similaire à la largeur de clique.

On introduira deux nouvelles largeurs, la **largeur modulaire** et la **largeur bi-modulaire**, comme généralisation de la décomposition modulaire, et on verra qu'elles sont aussi des largeurs similaires à la largeur de clique.

On peut remarquer que tous les graphes sont décomposables par des séparateurs, mais plus le paramètre sera grand, moins le graphe se prête à la décomposition. Tandis que pour un motif donné, soit le graphe possède le motif, soit il ne le possède pas. La frontière entre les deux familles de décompositions n'est pas claire. Par exemple, si on fixe un entier k, on peut considérer qu'une section de paramètre au plus k est un motif. De l'autre coté, on verra dans le chapitre 6 une généralisation paramétrée du module, le **k-module**, qui sera le motif de base pour la largeur modulaire.

Attaquer un problème en utilisant une décomposition

Le principal argument donné pour l'étude des décompositions en algorithmique est qu'elles permettent généralement de résoudre des problèmes efficacement.

L'objet de cette thèse est d'étudier certaines décompositions de graphes, et de comprendre comment ces décompositions peuvent être utilisées, comme un outil, pour résoudre un problème difficile en général sur une sous-classe de graphes. Le but n'est pas de montrer qu'il faut utiliser une certaine décomposition pour résoudre un problème sur une classe de graphes, mais de montrer comment une décomposition peut aider, simplifier aussi bien le travail que les algorithmes, puisqu'une grande partie du travail a déjà été effectuée.

Les problèmes étudiés sont pour la plupart NP-complets, il faut donc se restreindre à une classe de graphes pour obtenir un algorithme efficace pour résoudre le problème. L'étude se passera en deux étapes : l'étude du comportement du problème par la décomposition, puis l'étude de la décomposition de la classe de graphes.

Par exemple prenons le problème ENSEMBLE STABLE et la décomposition modulaire. On peut voir assez facilement que si l'on veut trouver la taille d'un plus grand ensemble stable du graphe original, on doit pouvoir résoudre la version pondérée du problème sur les graphes premiers de la décomposition.

Si l'on dispose d'une classe de graphes $\mathcal{G}$ telle que l'on sait résoudre efficacement le problème ENSEMBLE STABLE PONDÉRÉ sur les graphes premiers de $\mathcal{G}$, alors on dispose d'un algorithme efficace pour le problème ENSEMBLE STABLE PONDÉRÉ sur la classe $\mathcal{G}$.

Rolf Möhring et Franz-Josef Radermacher [MR84] donnent une vue d'ensemble de l'utilisation possible de la décomposition modulaire et de comment elle peut servir pour résoudre certains problèmes.

Dans le cas des décompositions paramétrées, les classes naturelles sont celles des graphes qui ont une largeur bornée. Si l'on veut utiliser une telle décomposition pour résoudre un problème sur une certaine classe de graphes, il faudra commencer par montrer que tous les graphes de cette classe ont une largeur bornée.

Beaucoup de problèmes ont été étudiés dans le cas de la décomposition arborescente. On connaît des classes de problèmes qui peuvent être résolus en temps polynomial si la largeur arborescente est bornée. Le travail a ensuite été effectué, ou généralisé, sur la décomposition en cliques.

Présentation des chapitres

Le chapitre 1 donnera les notions préliminaires sur les graphes et les ensembles. Il donnera les définitions des familles partitives et bi-partitives, ainsi que les théorèmes nécessaires.

Le chapitre 2 sera consacré à la décomposition modulaire. On y verra comment elle peut servir pour nous aider à résoudre certains problèmes. Comme illustration, on verra des algorithmes linéaires pour la classe des graphes sans P_5 et gem induit. Le travail présenté dans ce chapitre a été réalisé en collaboration avec Hans Bodlaender, Andreas Brandstädt, Dieter Kratsch et Jeremy Spinrad, a fait l'objet d'une présentation à la conférence FCT'2003 à Malmö, et d'une publication dans *Theoretical Computer Science* [BBK03].

Le chapitre 3 parlera de la décomposition en coupes. Après la présentation de la décomposition et des diverses propriétés, on donnera des algorithmes pour résoudre différents problèmes en utilisant des primitives sur les graphes premiers. Le travail a donné lieu à une présentation à la conférence WG'2004 [Rao04].

Le chapitre 4 parlera de la décomposition bi-joint. Il présentera les avancées dans l'étude de la décomposition depuis la thèse de Fabien de Montgolfier. On présentera en application une preuve que la classe des graphes sans gem et co-gem est de largeur de clique bornée. Le travail, réalisé en collaboration avec Fabien de Montgolfier, a donné lieu à une présentation à la conférence ICGT'2005 [MR05].

Le chapitre 5 parlera d'algorithmes polynomiaux sur les classes de graphes de largeur de clique bornée. On y introduira une nouvelle famille de problèmes de partitionnement, et on montrera que tous les problèmes dans cette famille peuvent être résolus en temps polynomial sur une classe de graphes de largeur de clique bornée.

Enfin le chapitre 6 présentera deux généralisations des décompositions modulaire, en coupes et bi-joint, appelés décompositions k-modulaire et k-bimodulaire. On y verra que ces décompositions sont reliées à la décomposition en cliques.

Chapitre 1

Préliminaires

1.1 Ensembles

Une partition $\{V_1, V_2, \ldots, V_k\}$ d'un ensemble V est un ensemble de sous-ensembles de V tel que $\bigcup_{i=1}^{k} V_i = V$ et $V_i \cap V_j = \emptyset$ pour tout $i \neq j$. L'**ensemble des parties** de V, noté $\mathcal{P}(V)$, est l'ensemble des sous-ensembles de V. Pour $k \in \mathbb{N}$, $\mathcal{P}_k(V)$ désigne l'ensemble de sous-ensembles de V de taille k.

Un **multi-ensemble** est un ensemble où chaque élément peut apparaître plus d'une fois. Il sera noté par $\langle a, b, c \ldots \rangle$. La **multiplicité** d'un élément a dans un multi-ensemble est le nombre de fois où a apparaît dans le multi-ensemble.

1.2 Graphes

1.2.1 Définitions et terminologie

Un **graphe** G est un couple (V, E), où V est l'ensemble des **sommets** et E, l'ensemble des **arêtes**, est un ensemble de sous-ensembles de V de taille 2. On désignera par $V(G)$ et $E(G)$, respectivement, l'ensemble des sommets et l'ensemble des arêtes d'un graphe G. Soient $n(G) = |V(G)|$ et $m(G) = |E|$. Si il n'y a pas d'ambiguïté, on notera simplement V, E, n et m. On dit que les sommets u et v d'un graphe G sont **adjacents** si $\{u, v\} \in E(G)$. Le **voisinage** de v, noté $N_G(v)$, est l'ensemble des sommets de G adjacents au sommet v, et le **voisinage fermé** de v, noté $N_G[v]$, désigne $N_G(v) \cup \{v\}$. Le **degré** d'un sommet v dans un graphe G, noté $d_G(v)$, est $|N_G(v)|$. Là encore, si il n'y a pas d'ambiguïté, on utilisera les notations $N(v)$, $N[v]$ et $d(v)$. Un **sommet isolé** est un sommet de degré 0, et un **sommet dominant** est un sommet de degré $n-1$. Dans cette thèse nous ne parlerons que de graphes finis, c'est à dire des graphes pour lesquels V et E sont des ensembles finis.

Le **complémentaire** de G est le graphe $(V, \mathcal{P}_2(V) \setminus E)$, noté $\overline{G}$. Un graphe H est **sous-graphe** d'un graphe G si $V(H) \subseteq V(G)$ et $E(H) \subseteq E(G)$. Soient G un graphe et $V' \subseteq V$. Le **sous-graphe induit** par V' est le graphe $(V', E \cap \mathcal{P}_2(V'))$, et sera noté $G[V']$. Deux graphes H et G sont **isomorphes** si il existe une bijection f entre $V(H)$ et $V(G)$

telle que $\{u, v\} \in E(H)$ si et seulement si $\{f(u), f(v)\} \in E(G)$. On dit que G est **sans** H **induit** si aucun sous-graphe de G n'est isomorphe à H.

Un graphe est **complet** si pour tout $u, v \in V$, $\{u, v\} \in E$. Un sous-ensemble $V' \subseteq V$ est un **ensemble stable** (respectivement une **clique**) si $G[V']$ est sans arête (respectivement, un graphe complet). Une **coloration** d'un graphe $G = (V, E)$ est une partition de V en ensembles stables. Le nombre de couleurs d'une coloration est le nombre d'ensembles dans la partition.

Un **chemin** dans un graphe G est une séquence $(v_1, \ldots, v_k)$ de sommets telle que $\forall i \in \{1, \ldots, k-1\}$, $\{v_i, v_{i+1}\} \in E$. Un chemin est **sans corde** si le graphe induit par $\{v_1, \ldots, v_k\}$ n'a pas d'autres arêtes que $\{v_i, v_{i+1}\}$ pour $i \in \{1, \ldots, k-1\}$. La **longueur** du chemin est $k-1$. Un **cycle** est un chemin $(v_1, \ldots, v_k)$, $k \geq 3$, tel que $\{v_1, v_k\} \in E$. Un cycle est sans corde si le sous-graphe induit par $\{v_1, \ldots, v_k\}$ n'a pas d'autres arêtes. On désignera également par **cycle induit** un cycle sans corde. La **taille** du cycle est k. Un **trou** est un cycle induit de taille au moins 5.

La distance entre le sommet u et le sommet v est la longueur minimum parmi tous les chemins entre u et v, et sera notée $d_G(u, v)$. Un graphe est **connexe** si pour tout sommet u et v, il existe un chemin entre u et v dans le graphe. Un graphe est **co-connexe** si son complémentaire est connexe.

Pour $k \in \mathbb{N}$, K_k est le graphe complet de k sommets, C_k est le cycle sans corde de k sommets et P_k est le chemin sans corde de k sommets. La figure 1.1 présente quelques graphes que l'on rencontrera. On remarque que les graphes P_4, C_5 et taureau sont isomorphes à leur complémentaire. Le complémentaire de la maison est le P_5.

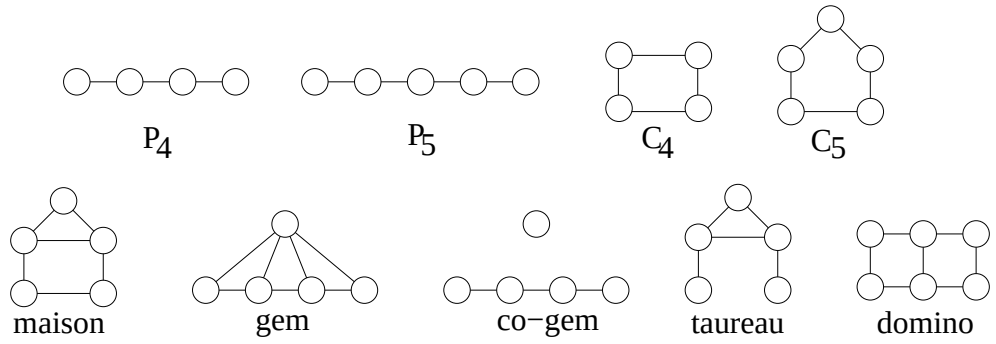


Fig. 1.1 : – Quelques petits graphes.

Un **graphe orienté** est un couple (V, A), où V désigne l'ensemble des sommets et A l'ensemble des **arcs**. Un arc est un couple (u, v) de sommets tel que $u \neq v$. Sauf indication contraire, tous les graphes seront supposés non orientés.

1.2.2 Classes de graphes

Une **classe de graphes** est un ensemble de graphes. Elle peut être finie ou infinie. Une classe de graphes $\mathcal{G}$ est **héréditaire** si pour tout $G \in \mathcal{G}$, et pour tout sous-graphe induit

H de G, alors $H \in \mathcal{G}$.

Toute classe de graphes héréditaire peut être définie par un ensemble (qui peut être infini) de sous-graphes induits interdits : il suffit d'interdire tous les graphes qui ne sont pas dans la classe. Soit $\mathcal{H}$ un ensemble de graphes. La **classe des graphes sans $\mathcal{H}$ induit** est l'ensemble des graphes qui ne possèdent aucun graphe de $\mathcal{H}$ comme sous-graphe induit.

Brandstädt, Le et Spinrad [BLS99] présentent une collection de classes de graphes, avec les différentes caractérisations et inclusions connues.

1.2.2.1 Arbres

Un **arbre** est un graphe connexe sans cycle. Les sommets d'un arbre pourront être appelés des **noeuds**. Un **arbre enraciné** est un arbre avec un noeud appelé **racine**. Tous les noeuds autres que la racine ont un **père**, le premier noeud autre que lui même dans l'unique chemin entre lui et la racine. Un noeud u est un fils d'un noeud v si v est le père de u. Un noeud est une **feuille** s'il n'a pas de fils, sinon il s'agit d'un **noeud interne**.

Un graphe est une **forêt** s'il n'a pas de cycle. On voit facilement qu'une forêt est une union disjointe d'arbres, et que la classe des forêts est héréditaire.

1.2.2.2 Graphes bipartis

Un graphe est un graphe **biparti** s'il existe une partition $\{X, Y\}$ de V telle que X et Y soient des ensembles stables. On les notera parfois (X, Y, E) plutôt que (V, E). Un graphe biparti est **complet** si $E = \big\{\{u, v\} : u \in X \text{ et } v \in Y\big\}$. Le graphe biparti complet tel que $|X| = a$ et $|Y| = b$ est noté $K_{a,b}$. Les arbres et les forêts sont des graphes bipartis, et la classe des graphes bipartis est héréditaire. On peut noter qu'un graphe est un graphe biparti si et seulement si il n'a pas de cycle induit de taille impaire.

1.2.2.3 Graphes chordaux

Un graphe est un graphe **chordal** s'il ne possède pas de cycle induit de taille au moins 4 (c'est à dire s'il est sans C_k, $k \geq 4$). Les arbres et les forêts sont des graphes chordaux. Un sommet v est **simplicial** si $N(v)$ est une clique. Un **ordre d'élimination parfait** d'un graphe G est un ordonnancement de ses sommets $(v_1, v_2, \ldots, v_n)$ tel que pour tout $i \in \{1, 2, \ldots n\}$, v_i est simplicial dans $G[\{v_i, \ldots, v_n\}]$.

Théorème 1.1. *[FG65, Gol80] Un graphe est chordal si et seulement si il possède un ordre d'élimination parfait.*

Un graphe est **faiblement chordal** si ni lui, ni son complémentaire ne possèdent de cycle induit de taille au moins 5. Un graphe chordal est donc un graphe faiblement chordal.

1.2.2.4 Graphes parfaits

On voit facilement qu'il faut au moins k couleurs pour colorier un graphe possédant une clique de taille k. La réciproque n'est pas vraie ; il faut au moins 3 couleurs pour colorier

un C_5, alors que sa plus grande clique est de taille 2.

Un graphe G est **parfait** si pour tout sous-graphe induit H de G, le nombre de couleurs minimum pour colorier H est égal à la taille d'une plus grande clique de H. Les graphes bipartis, chordaux, faiblement chordaux sont des graphes parfaits.

Shannon a introduit les graphes parfaits dans le cadre de théorie de l'information [Sha56]. Golumbic donne une bonne présentation des graphes parfaits, et de différentes sous-classes [Gol80].

Théorème 1.2. *[Lov72] Un graphe est parfait si et seulement si son complémentaire est parfait.*

Un graphe G est un graphe **Berge** s'il n'a pas de trou de taille impaire, et $\overline{G}$ n'a pas de trou de taille impaire. Claude Berge a conjecturé qu'un graphe est un graphe parfait si et seulement si il est un graphe Berge [Ber61]. Cette conjecture a finalement été montrée récemment.

Théorème 1.3. *[CRS02] Un graphe est parfait si et seulement si il est un graphe Berge.*

La preuve utilise certaines décompositions de graphes. Il est facile de voir que si un graphe est parfait, alors c'est un graphe Berge. Pour montrer l'autre sens, les auteurs montrent que pour tout graphe Berge G, G est soit dans une classe basique (G ou $\overline{G}$ est un graphe biparti, ou le graphe de lignes d'un graphe biparti), soit G est décomposable par une certaine décomposition (G ou $\overline{G}$ a soit un 2-*join*, soit une partition *skew*). Tous les graphes basiques sont parfaits, et un graphe Berge non parfait ne possède pas de 2-*join*, de partition *skew*, et son complément non plus, ce qui montre le théorème.

1.3 Problèmes de graphes

1.3.1 Notions de complexité

Un algorithme **déterministe** est un algorithme qui, à chaque moment, n'a qu'une seule possibilité de continuer son exécution. Il s'agit du sens usuel qu'on donne à un algorithme ; dans cette thèse tous les algorithmes donnés sont déterministes. Un algorithme **non déterministe** est un algorithme qui à chaque étape peut avoir plusieurs possibilités pour suivre son exécution.

On dit qu'un problème est **polynomial** (ou est dans la classe de problèmes **P**) s'il existe un algorithme déterministe qui résout le problème sur toutes les entrées, et qui s'exécute en un nombre d'opérations $f(n)$, où n est la taille de l'entrée, et f est une fonction bornée par un polynôme. Un problème est dans la classe **NP** (pour « *non deterministic polynomial* ») s'il existe un algorithme non déterministe résolvant le problème, et s'exécutant en un nombre polynomial d'opérations.

On dit qu'un problème Π est **NP-complet** si tout problème dans NP peut se réduire polynomialement à Π, c'est à dire que si on sait résoudre Π en temps polynomial, alors

on sait résoudre tous les problèmes dans NP en temps polynomial. Le premier problème à avoir été démontré NP-complet est le problème SAT [Coo71].

SAT
INSTANCE : une formule booléenne B.
QUESTION : Existe-t-il une assignation des variables de B qui satisfait la formule ?

La démonstration de la NP-complétude du problème SAT a été faite en simulant une machine de Turing non déterministe par un problème SAT. Donc si on sait résoudre SAT en temps polynomial sur une machine déterministe, on peut connaître le résultat de l'algorithme non déterministe exécuté par la machine. On peut noter que le problème 3-SAT, une restriction du problème SAT où la formule booléenne est en forme normale conjonctive et chaque clause possède au plus 3 littéraux, est également NP-complet. Par contre il existe un algorithme pour le problème 2-SAT s'exécutant en temps linéaire [APT79]. Garey et Johnson dressent une liste de problèmes NP-complets dans un livre qui est encore aujourd'hui une référence dans le domaine [GJ78].

La théorie de la complexité traite des problèmes de **décisions**, c'est à dire des problèmes où la sortie est soit « oui », soit « non ». La plupart des problèmes de décisions possèdent une version **d'optimisation**, au moins aussi dure que la version de décision. Par exemple le problème de décision demande si un certain paramètre de la structure est inférieur ou égal à k donnée en entrée, tandis que la version d'optimisation demandera de minimiser le paramètre. La plupart des problèmes présentés seront des problèmes d'optimisation.

On voit facilement que tout problème de P est dans la classe des problèmes NP. Une grande question (si ce n'est la plus grande) dans le domaine de l'algorithmique est : est-ce que P $\neq$ NP ? C'est à dire, est-ce que le non déterminisme apporte quelque chose ? Il semble naturel de penser que oui, comme le pense la plus grande partie de la communauté scientifique. Mais cela n'a pas encore été démontré, et cela ne le sera sans doute pas dans l'immédiat.

Si on assume que P $\neq$ NP, il n'y a donc pas d'algorithme polynomial pour résoudre un problème NP-complet. Il peut donc devenir rapidement impossible à traiter en pratique. Il y a néanmoins plusieurs moyens pour attaquer un tel problème.

1.3.2 Problèmes NP-complets sur les graphes

Beaucoup de problèmes sont connus comme étant NP-complets sur les graphes. On présente quelques problèmes que l'on rencontrera le long de la thèse.

Soit $G = (V, E)$ un graphe. On note par $\alpha(G)$ la taille d'un plus grand ensemble stable de G, et par $\omega(G)$ la taille d'une plus grande clique. Le problème suivant est NP-complet.

Ensemble stable
INSTANCE : un graphe G et $k \in \mathbb{N}$.
QUESTION : est ce que $\alpha(G) \geq k$?

En prenant le complémentaire, on voit qu'il en est de même pour le problème CLIQUE, où l'on demande si $\omega(G) \geq k$. On parlera souvent de la version d'optimisation du problème ENSEMBLE STABLE, qui est la suivante.

Ensemble stable (optimisation)
INSTANCE : un graphe G.
SORTIE : $\alpha(G)$.

Une variante du problème précédent est celui où il est demandé en sortie un ensemble stable de taille maximum.

Une coloration est **minimum** s'il n'existe pas de coloration du graphe avec strictement moins de couleurs. Le **nombre chromatique**, noté $\chi(G)$, d'un graphe est la taille d'une coloration minimum du graphe. Le problème suivant est NP-complet.

Nombre chromatique
INSTANCE : un graphe G et $k \in \mathbb{N}$.
QUESTION : est ce que $\chi(G) \leq k$?

La version d'optimisation est la suivante.

Nombre chromatique (optimisation)
INSTANCE : un graphe G.
SORTIE : $\chi(G)$.

Le problème sur le complémentaire, qui demande de partitionner l'ensemble de sommets en cliques, est le problème PARTITION EN CLIQUES. On notera par $\kappa(G)$ la plus petite partition de V en cliques.

Un sous-ensemble $D \subseteq V$ est un **ensemble dominant** de G si tout sommet $v \notin D$ a au moins un voisin dans D. On note par $\gamma(G)$ la taille d'un plus petit ensemble dominant. Le problème de décider si un graphe possède un ensemble dominant de taille au plus k est NP-complet.

Ensemble dominant (optimisation)
INSTANCE : un graphe G.
SORTIE : $\gamma(G)$.

Différentes variantes de ce problème sont également NP-complet. On dit qu'un sous-ensemble $D \subseteq V$ est un **ensemble dominant total** si tout sommet $v \in V$ a au moins un voisin dans D. Un sous-ensemble $D \subseteq V$ est un **ensemble dominant connexe**, un **ensemble dominant indépendant**, ou un **clique dominante** si D est un ensemble dominant, et D est connexe, un ensemble stable ou une clique, respectivement. On note par $\gamma_{\mathrm{t}}(G)$, $\gamma_{\mathrm{c}}(G)$, $\gamma_{\mathrm{i}}(G)$ et $\gamma_{\mathrm{cl}}(G)$, la taille minimum d'un ensemble dominant total, d'un ensemble dominant connexe, d'un ensemble dominant indépendant et d'une clique dominante de G, respectivement. Les problèmes où l'on demande de décider si un graphe donné G possède une de ces variantes d'ensemble dominant d'une taille au plus k donné sont tous NP-complets.

1.3.3 Attaquer un problème difficile

Plusieurs approches peuvent être envisagées pour attaquer un problème difficile.

Une première approche, quand on a à faire à un problème d'optimisation, est de dire que l'on ne veut pas nécessairement la solution exacte au problème. Cette relaxation permet parfois de faire passer le problème dans le polynomial. Un exemple significatif est le problème de coloration des arêtes d'un graphe avec un nombre minimum de couleurs[1]. Ce problème est NP-complet. On voit immédiatement que le nombre de couleurs nécessaires est au moins Δ (le degré maximum d'un sommet dans le graphe). D'autre part, Vizing [Viz64] a montré que tout graphe possède une coloration d'arêtes avec $\Delta + 1$ couleurs, et cette coloration peut être obtenue en temps polynomial. Malheureusement, tous les problèmes d'optimisation ne s'approximent pas aussi bien.

Dans certains cas on ne souhaite pas se contenter d'une approximation. On peut se servir des algorithmes exponentiels, qui donnent des résultats sur de petites entrées. Une recherche peut être faite pour les améliorer ; un algorithme en temps $O(1,21^n)$ sera bien meilleur en pratique qu'un algorithme en $O(2^n)$.

Enfin, la possibilité que l'on emprunte dans cette thèse quand on a à faire à des problèmes difficiles, est de restreindre les entrées du problème à certaines classes d'instances. Pour les problèmes de graphes, cela revient à se restreindre à des classes de graphes, comme les graphes parfaits, les graphes bipartis, les arbres... Il est par exemple bien connu que les problèmes ENSEMBLE STABLE, CLIQUE, COLORATION MINIMUM et PARTITION EN CLIQUE sont polynomiaux sur les graphes parfaits [GLS84].

L'objet de cette thèse est d'étudier certaines décompositions de graphes, et expliquer comment ces décompositions peuvent être utilisées, comme un outil, pour résoudre un problème difficile en général sur une sous-classe de graphes. Le but n'est pas de montrer qu'il faut utiliser une décomposition pour avoir un algorithme polynomial ; on fait ceci pour simplifier le travail et les algorithmes.

1.4 Largeur arborescente et largeur de clique

Il existe plusieurs définitions équivalentes de la largeur arborescente. La classe des **k-arbres**, pour $k \in \mathbb{N}^*$, est définie récursivement. Un graphe complet avec k sommets est un k-arbre. Si G est un k-arbre, et on ajoute à G un sommet adjacent à une clique de taille k, le nouveau graphe est un k-arbre. Un graphe est un **k-arbre partiel** s'il est sous-graphe (pas forcement induit) d'un k-arbre. La **largeur arborescente** d'un graphe G est le plus petit k tel que G soit un k-arbre partiel.

Le problème où l'on demande de décider si un graphe a une largeur arborescente d'au plus k donné est NP-complet [ACP87]. Mais pour tout k fixé, il existe un algorithme linéaire pour décider si un graphe a une largeur arborescente d'au plus k [Bod96].

[1] On demande d'attribuer une couleur à chaque arête tel que deux arêtes incidentes à un même sommet aient des couleurs différentes.

La largeur NLC et la largeur de clique sont deux paramètres qui ont été introduits pour généraliser la largeur arborescente. La largeur NLC a été introduite par Wanke [Wan94], et la largeur de clique par Courcelle, Engelfriet et Rozenberg [CER93].

Elles généralisent la largeur arborescente dans le sens où si une classe de graphes a une largeur arborescente bornée, alors elle a une largeur de clique (et une largeur NLC) bornée. Le contraire n'est pas vrai. Un graphe complet de k sommets ($k > 1$) a une largeur arborescente de $k-1$, une largeur de clique de 2 et une largeur NLC de 1.

Un **graphe étiqueté** est un graphe avec une fonction d'étiquetage $\mathrm{lab}_G : V \to \mathbb{N}$ sur les sommets. Si toutes les étiquettes sont dans $\{1, \ldots, k\}$, le graphe sera dit **k-étiqueté**. On dit que deux graphes étiquetés G et H, avec fonction d'étiquetage respective lab_G et lab_H, sont **isomorphes** s'il existe une bijection f entre $V(G)$ et $V(H)$, telle que $\mathrm{lab}_G(v) = \mathrm{lab}_H(f(v))$, et $\{u, v\} \in E(G) \iff \{f(u), f(v)\} \in E(H)$ pour tout $u, v \in V(G)$. Si $V' \subseteq V(G)$, le sous-graphe étiqueté induit par V' est le sous-graphe induit par V', avec la fonction d'étiquetage lab_G restreinte à V'. Le sous graphe étiqueté induit par V' sera noté par $G[V']$. Enfin, $\mathrm{unlab}(G)$ représente le graphe G sans la fonction d'étiquetage.

Soient G et H deux graphes étiquetés, et soit $i, j \in \mathbb{N}$, $i \neq j$.

- $\cdot_i$ représente le graphe d'un sommet, étiqueté par i.
- $\rho_{i \to j}(G)$ représente le graphe étiqueté G dans lequel les étiquettes des sommets étiquetés i ont été changés à j.
- $\eta_{i,j}(G)$ représente le graphe étiqueté G dans lequel on a ajouté toutes les arêtes entre les sommets étiquetés i et les sommets étiquetés j.
- $G \oplus H$ représente l'union disjointe de G et H.

Une **k-expression** est une expression définissant un graphe, avec les étiquettes dans $\{1, \ldots, k\}$. Une k-expression t pour un graphe étiqueté G est une k-expression définissant G. On notera par $G(t)$ le graphe définit par l'expression t.

La **largeur de clique** d'un graphe étiqueté G est le plus petit k tel qu'il existe une k-expression (construite avec les opérations $\cdot$, η, ρ et $\oplus$) définissant G. La largeur de clique d'un graphe non étiqueté est la largeur de clique du graphe G tel que tous les sommets soient étiquetés 1. On notera par $\mathrm{cwd}(G)$ la largeur de clique du graphe G.

La largeur NLC est également définie à l'aide d'opérations sur les graphes étiquetés.

- $\cdot_i$ représente le graphe d'un sommet, étiqueté par i.
- Soient $G_1 = (V_1, E_1, \mathrm{lab}_1)$ et $G_2 = (V_2, E_2, \mathrm{lab}_2)$ deux graphes k-étiquetés, et soit $S \subseteq \{1, \ldots, k\}^2$. $G_1 \times_S G_2$ représente le graphe étiqueté (V, E, lab) tel que $V = V_1 \cup V_2$,

$$E = E_1 \cup E_2 \cup \big\{\{u, v\} : u \in V_1, v \in V_2 \text{ et } (\mathrm{lab}_1(u), \mathrm{lab}_2(v)) \in S\big\}$$

  et pour tout $u \in V$,

$$\mathrm{lab}(u) = \begin{cases} \mathrm{lab}_1(u) \text{ si } u \in V_1 \\ \mathrm{lab}_2(u) \text{ si } u \in V_2. \end{cases}$$

- Soient $R : \{1, \ldots, k\} \to \{1, \ldots, k\}$ et $G = (V, E, \mathrm{lab})$. Alors $\rho_R(G) = (V, E, \mathrm{lab}')$ tel que $\mathrm{lab}'(u) = R(\mathrm{lab}(u))$, pour tout $u \in V$.

La **largeur NLC** d'un graphe étiqueté est le plus petit k tel qu'il existe une k-expression (utilisant les opération $\cdot$, $\times$ et ρ) définissant G. Elle sera notée NLC-wd(G). La largeur NLC d'un graphe non étiqueté est la largeur NLC du graphe avec tous les sommets étiquetés 1.

On peut remarquer que les graphes de largeur de clique 1 sont les graphes sans arêtes. Les graphes de largeur de clique 2 et les graphes de largeur NLC 1 sont exactement les cographes (définit section 2.1.5). Les problèmes où l'on demande de trouver la largeur de clique ou la largeur NLC d'un graphe sont NP-complets [FRR06, GW05]. Mais on peut décider en temps polynomial si la largeur de clique est au plus 3 [CHL00], et si la largeur NLC est au plus 2 [Joh00].

Les deux expressions suivantes sont des expressions pour le graphe C_5. La première utilisant les opérations de la largeur de clique, la deuxième les opérations de la largeur NLC.

$$\eta_{1,3}\left(\rho_{3\to 2}\left(\eta_{2,3}\left(\eta_{1,2}(\cdot_1 \oplus \cdot_2) \oplus \eta_{1,3}(\cdot_1 \oplus \cdot_3)\right)\right) \oplus \cdot_3\right)$$

$$\left((\cdot_1 \times_{\{(1,2)\}} \cdot_2) \times_{\{(2,2)\}} \cdot_1 (\times_{\{(1,2)\}} \cdot_2)\right) \times_{\{(1,1)\}} \cdot_1$$

Le C_5 n'étant pas un cographe, ses largeurs de clique et NLC sont respectivement de 3 et de 2.

On peut remarquer la similitude entre l'opération ρ de la largeur NLC et l'opération ρ de la largeur de clique. L'opération $\times$ joue le rôle de l'opération $\oplus$ et η en même temps. Une k-expression pour la largeur de clique peut donc être facilement transformée en expression pour la largeur NLC, avec le même nombre d'étiquettes. L'inverse n'est en général vrai, mais on peut construire une $2k$-expression pour la largeur de clique à partir d'une k-expression pour la largeur NLC.

Théorème 1.4. *[Joh98] Soit G un graphe. Alors* $\text{NLC-wd}(G) \leq \text{cwd}(G)$ *et* $\text{cwd}(G) \leq 2 \times \text{NLC-wd}(G)$.

On dira que deux paramètres de graphes p et q sont **similaires** s'il existe deux fonctions f et h de $\mathbb{N}$ dans $\mathbb{N}$, telles que pour tout graphe G, $p(G) \leq f(q(G))$ et $q(G) \leq h(p(G))$. On voit facilement que la relation de similitude est une relation d'équivalence. La largeur de clique et la largeur NLC sont donc deux paramètres similaires. Ce n'est pas le cas de la largeur de clique et de la largeur arborescente.

Théorème 1.5. *[CR05]. Soit G un graphe. Alors* $\text{cwd}(G) \leq 3 \times 2^{\text{twd}(G)-1}$.

Soit $G = (V, E)$ un graphe. Le **graphe d'incidence** de G est le graphe $(V \cup E, R)$, où $\{v, e\} \in R$ si $v \in V$, $e \in E$ et v est une extrémité de e. Il est noté $I(G)$. Le lemme suivant montre que largeur arborescente d'un graphe et la largeur de clique du graphe d'incidence sont deux paramètres similaires :

Lemme 1.6. $\text{cwd}(I(G)) \leq 3 \times 2^{\text{twd}(G)-1}$ *et* $\text{twd}(G) \leq 6 \times \text{cwd}(I(G)) - 1$.

Démonstration. On sait que la largeur arborescente du graphe d'incidence d'un graphe G est égale à la largeur arborescente de G [LR04]. Donc si G est un graphe de largeur arborescente au plus k, alors la largeur de clique de son graphe d'incidence $I(G)$ est au plus $3 \times 2^{k-1}$.

Si la largeur de clique de $I(G)$ est au plus k, la largeur arborescente de $I(G)$, et donc celle de G, est au plus $6k-1$, car un graphe d'incidence n'a pas de $K_{3,3}$ induit [GW00]. □

Récemment, Oum et Seymour ont introduit la largeur de rang, comme paramètre alternatif à la largeur de clique [OS06]. Ils montrent qu'il s'agit d'une largeur similaire à la largeur de clique.

Soit G un graphe. Soit T un arbre tel qu'il y ait une bijection entre les feuilles de T et les sommets de G, et que chaque noeud de T soit de degré au plus 3. Chaque arête de T induit une partition de V. Soit $M_{T,e}$ la matrice d'adjacence entre les deux classes de cette partition. La largeur de rang de G suivant T est $\mathrm{rwd}(G,T) = \max_{e \in E(T)} \mathrm{rang}(M_{T,e})$, où $\mathrm{rang}(M)$ est le rang de la matrice M. La **largeur de rang** de G est $\mathrm{rwd}(G) = \min_T \mathrm{rwd}(G,T)$.

Théorème 1.7. *[OS06] Pour tout graphe G,* $\mathrm{rwd}(G) \leq \mathrm{cwd}(G) \leq 2^{\mathrm{rwd}(G)+1} - 1$.

Les auteurs donnent également un algorithme polynomial, pour un $k \in \mathbb{N}$ fixé, qui étant donné un graphe G, soit donne un arbre T tel que la largeur de rang de G suivant T soit au plus $3k+1$, soit conclut que $\mathrm{rwd}(G) > k$. Oum [Oum05] donne d'autres algorithmes d'approximation pour la largeur de rang, et Courcelle et Oum [CO04] donnent un algorithme exact qui décide si la largeur de rang est au plus k, k fixé.

1.5 Familles d'ensembles

Soient $X \subseteq V$ et $Y \subseteq V$. On dit que X et Y se **chevauchent** si $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$ et $X \cap Y \neq \emptyset$. On notera par $X \Delta Y$, la **différence symétrique** de X et Y, l'ensemble $(X \setminus Y) \cup (Y \setminus X)$. Une **famille** désignera un ensemble de sous-ensembles d'un ensemble fixé V.

1.5.1 Familles arborescentes

Définition 1.5.1. Soit V un ensemble. Une famille $\mathcal{F}$ de sous-ensembles de V est une **famille arborescente**[2] si :

1. $\emptyset \notin \mathcal{F}$ et $V \in \mathcal{F}$,
2. pour tout $v \in V$, $\{v\} \in \mathcal{F}$,
3. pour tout X et pour tout Y dans $\mathcal{F}$, X et Y ne se chevauchent pas.

[2]Cette notion n'a aucun rapport avec la largeur arborescente.

Une famille arborescente peut être représentée de façon unique par un arbre enraciné, appelé **l'arbre d'inclusion** de la famille. Cet arbre a pour ensemble de noeuds $\mathcal{F}$, pour racine V, et un noeud Y est fils d'un noeud X si $Y \subsetneq X$ et il n'existe pas de $Z \in \mathcal{F}$ tel que $Y \subsetneq Z \subsetneq X$. Les feuilles de l'arbre d'inclusion seront donc les singletons de $\mathcal{F}$. On remarque que la taille de l'arbre est proportionelle aux nombres d'éléments de la famille. La figure 1.2 donne en exemple une famille arborescente avec son arbre d'inclusion.

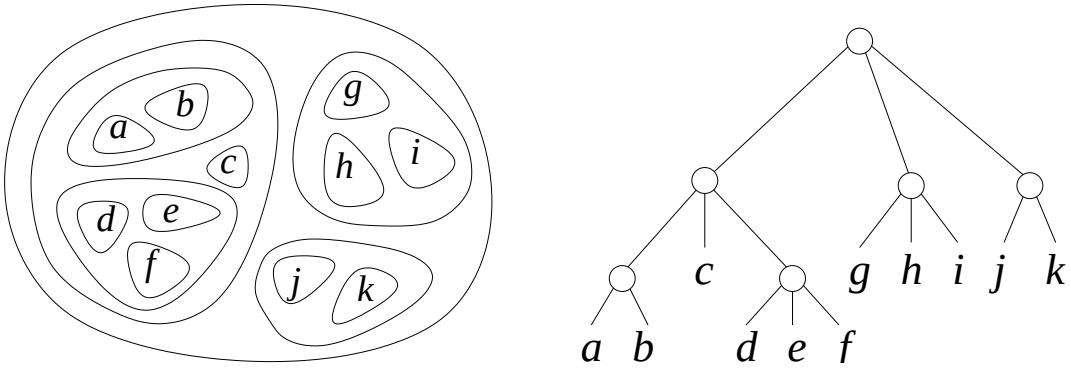


Fig. 1.2 : – Une famille arborescente et son arbre d'inclusion.

1.5.2 Familles partitives

Les familles partitives ont été introduites et étudiées par Chein, Habib et Maurer [Hab81, CHM81]. Ce sont des familles qui synthétisent certaines propriétés des modules dans les graphes et hypergraphes. Möhring et Radermacher en donnent une bonne présentation [MR84].

Définition 1.5.2. Soit $\mathcal{F} \subseteq \mathcal{P}(V)$ une famille de sous-ensembles de V. $\mathcal{F}$ est **partitive** si :

- $V \in \mathcal{F}$, $\emptyset \notin \mathcal{F}$ et pour tout $v \in V$, $\{v\} \in \mathcal{F}$.
- Pour tout $A \in \mathcal{F}$ et $B \in \mathcal{F}$, telles que A chevauche B, on a :
  1. $A \cap B \in \mathcal{F}$.
  2. $A \cup B \in \mathcal{F}$.
  3. $A \setminus B \in \mathcal{F}$ et $B \setminus A \in \mathcal{F}$.
  4. $A \Delta B \in \mathcal{F}$.

Une famille de sous-ensembles de V est une famille **faiblement partitive** si elle respecte toutes les conditions, excepté la condition (4).

On peut noter qu'une famille arborescente est une famille partitive. On dit qu'un membre X d'une famille partitive ou faiblement partitive $\mathcal{F}$ est **fort** s'il n'existe pas de $Y \in \mathcal{F}$ tel que X et Y se chevauchent. On voit immédiatement que la famille $\mathcal{F}'$ des membres forts d'une famille partitive ou faiblement partitive $\mathcal{F}$ est une famille arborescente.

Le théorème suivant nous montre qu'il suffit d'ajouter une information aux noeuds de T pour qu'il puisse représenter toute la famille $\mathcal{F}$.

Théorème 1.8. *[Hab81, CHM81] Soit $\mathcal{F}$ une famille partitive, et $\mathcal{F}'$ la famille des éléments forts de $\mathcal{F}$. Les noeuds de l'arbre d'inclusion T de $\mathcal{F}'$ peuvent être étiquetés* **premiers** *ou* **dégénérés** *tel que*

- *pour tout $X \in \mathcal{F} \setminus \mathcal{F}'$, il existe un noeud* **dégénéré** *Y de T tel que X soit une union de fils de Y, et*
- *toute union de fils d'un noeud* **dégénéré** *de T appartient à $\mathcal{F}$.*

Par convention, on étiquettera tous les noeuds possédant exactement deux fils par `dégénéré`. Dans ce cas, l'étiquetage obtenu est unique. L'arbre T avec les étiquettes `premier` et `dégénéré` est appelé **l'arbre représentatif** de la famille $\mathcal{F}$.

Une famille partitive sur V, qui peut avoir un nombre de membres exponentiel en $|V|$ (par exemple la famille de tous les sous-ensembles de V), est donc représentable en espace linéaire. La figure 1.3 donne par exemple l'arbre représentatif de la famille $\mathcal{F} = \big\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{a,b\},\{d,e,f\},\{a,b,c\},\{c,d,e,f\},\{a,b,d,e,f\},\{a,b,c,d,e,f\}\big\}$. Les membres de la famille qui ne sont pas forts sont $\{a,b,c\}$,$\{c,d,e,f\}$ et $\{a,b,d,e,f\}$.

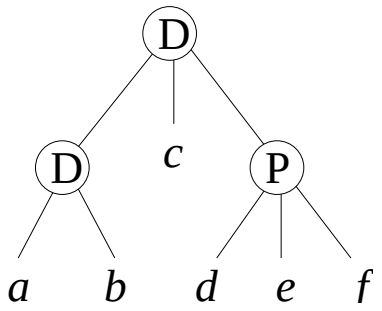


Fig. 1.3 : – L'arbre représentatif de la famille partitive $\mathcal{F}$ (D = `dégénéré` et P = `premier`).

Dans le cas des familles faiblement partitives, l'arbre d'inclusion possède un troisième type de noeuds :

Théorème 1.9. *[Hab81] Soit $\mathcal{F}$ une famille faiblement partitive, et $\mathcal{F}'$ la famille des éléments forts de $\mathcal{F}$. Les noeuds de l'arbre d'inclusion T de $\mathcal{F}'$ peuvent être étiquetés* **premiers**, **dégénérés** *ou* **linéaires**, *et les fils des noeuds* **linéaires** *peuvent être ordonnés tel que*

- *toute union de fils d'un noeud* **dégénéré** *de T appartient à $\mathcal{F}$,*
- *pour tout noeud* **linéaire** *de T et pour tout $a,b \in \{1,\ldots,k\}$ tel que $a \leq b$, $\bigcup_{i=a}^{b} X_i$ appartient à $\mathcal{F}$, et*
- *pour tout $Y \in \mathcal{F}$, il existe un noeud* **dégénéré** *X tel que Y soit une union de fils de X, ou il existe un noeud* **linéaire** *X et $a,b \in \{1,\ldots,k\}$, $a \leq b$, tel que $Y = \bigcup_{i=a}^{b} X_i$.*

1.5.3 Familles arborées

Une **bi-partition** $\{X, Y\}$ d'un ensemble V est une partition de V en deux ensembles X et Y (c'est à dire $X \cap Y = \emptyset$ et $X \cup Y = V$). Soient $\{X, X'\}$ et $\{Y, Y'\}$ deux bi-partitions de V. On dit que $\{X, X'\}$ et $\{Y, Y'\}$ se **chevauchent** si $X \cap Y \neq \emptyset$, $X \cap Y' \neq \emptyset$, $X' \cap Y \neq \emptyset$ et $X' \cap Y' \neq \emptyset$.

Définition 1.5.3. Soit $\mathcal{F}$ une famille de bi-partitions de V. $\mathcal{F}$ est une **famille arborée** si :
- $\{V, \emptyset\} \notin \mathcal{F}$ et pour tout $v \in V$, $\big\{\{v\}, V \setminus \{v\}\big\} \in \mathcal{F}$,
- pour tout $\{X, X'\}$ et $\{Y, Y'\}$ dans $\mathcal{F}$, $\{X, X'\}$ et $\{Y, Y'\}$ ne se chevauchent pas.

Les familles arborées sont une transposition des familles arborescentes sur les familles de bi-partitions. Elles sont aussi uniquement représentables par un arbre, à la différence qu'il est non enraciné. Chaque bi-partition correspond à une arête de l'arbre, et chaque élément de V correspond à une feuille de l'arbre.

Soit $T = (V_T, E_T)$ un arbre (non enraciné) avec l'ensemble de feuilles $V \subseteq V_T$, et soit e une arête de T. $T' = (V_T, E_T \setminus \{e\})$ a deux composantes connexes T_e^1 et T_e^2. Pour $i \in \{1, 2\}$, soit C_e^i l'ensemble des noeuds de T_e^i qui sont des feuilles de T. Chaque arête e de T définit une bi-partition $\{C_e^1, C_e^2\}$ de V.

Lemme 1.10. *[Bun71] Soit $\mathcal{F}$ une famille arborée sur l'ensemble V. Alors il existe un unique arbre $T = (V_T, E_T)$ avec V comme ensemble de feuilles, tel que :*
- *pour tout $e \in E_T$, $\{C_e^1, C_e^2\} \in \mathcal{F}$, et*
- *pour tout $\{X, Y\} \in \mathcal{F}$, il existe $e \in E_T$ tel que $\{X, Y\} = \{C_e^1, C_e^2\}$.*

Cet arbre sera appelé **l'arbre d'inclusion** de la famille arborée. La figure 1.4 donne en exemple une famille arborée avec son arbre d'inclusion.

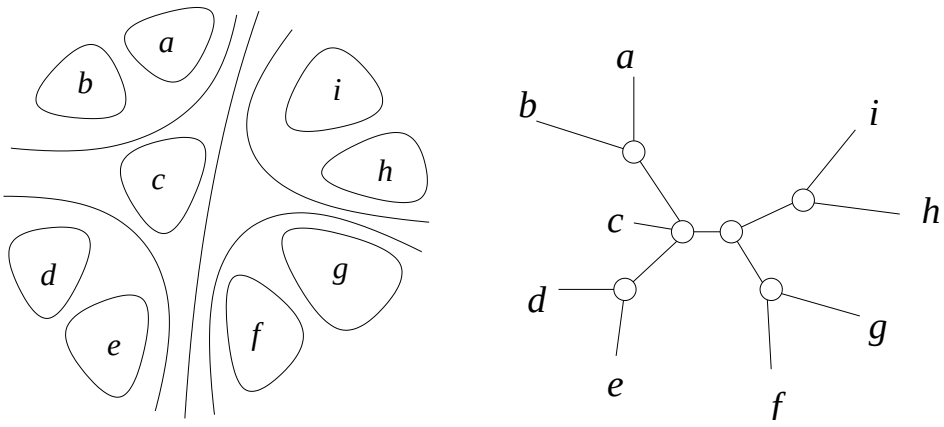


Fig. 1.4 : – Une famille arborée et son arbre d'inclusion.

1.5.4 Familles bi-partitives

Définition 1.5.4. Soit $\mathcal{F}$ une famille de bi-partitions de V. $\mathcal{F}$ est une **famille bi-partitive** si :

- $\{V, \emptyset\} \notin \mathcal{F}$ et pour tout $v \in V$, $\big\{\{v\}, V \setminus \{v\}\big\} \in \mathcal{F}$,
- pour tout $\{X, X'\}$ et $\{Y, Y'\}$ dans $\mathcal{F}$ tels que $\{X, X'\}$ et $\{Y, Y'\}$ se chevauchent, alors
  1. $\{X \cup Y, X' \cap Y'\} \in \mathcal{F}$,
  2. $\{X \cup Y', X' \cap Y\} \in \mathcal{F}$,
  3. $\{X' \cup Y, X \cap Y'\} \in \mathcal{F}$,
  4. $\{X' \cup Y', X \cap Y\} \in \mathcal{F}$,
  5. $\{X \Delta Y, X' \Delta Y\} \in \mathcal{F}$.

Une famille de bi-partitions de V est une famille **faiblement bi-partitive** si elle respecte toutes les conditions, excepté la condition (5).

Les familles bi-partitives sont une transposition des familles partitives aux familles de bi-partitions. On verra qu'il y a de fortes relations entre elles : on peut facilement transformer une famille partitive en famille bi-partitive, et inversement. Une bonne présentation des familles arborées et bi-partitives peut être trouvée dans la thèse de Fabien de Montgolfier [Mon03].

Les familles arborées sont des familles bi-partitives. Un membre d'une famille bi-partitive est **fort** s'il ne chevauche aucun autre membre de la famille. La famille des membres forts d'une famille bi-partitive est une famille arborée, et donc peut être représentée de façon unique par un arbre non enraciné.

Soit α un noeud interne d'un arbre T. La forêt $T - \alpha$ a $d(\alpha)$ composantes connexes $T_\alpha^1, \ldots, T_\alpha^{d(\alpha)}$. Pour tout $i \in \{1, \ldots, d(\alpha)\}$, on note par C_α^i l'ensemble des noeuds de T_α^i qui sont des feuilles de T. On voit que $\{C_\alpha^1, \ldots, C_\alpha^{d(\alpha)}\}$ est une partition de V.

Théorème 1.11. *[Mon03] Soit $\mathcal{F}$ une famille bi-partitive sur V, et soit T l'arbre d'inclusion des membres forts de $\mathcal{F}$. Les noeuds internes de $T = (V_T, E_T)$ peuvent être étiquetés* `premiers` *ou* `dégénérés` *tel que :*

- *pour tout $e \in E_T$, $\{C_e^1, C_e^2\}$ est une bi-partition forte de $\mathcal{F}$,*
- *pour tout noeud* `dégénéré` *α, et pour tout $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$, $\{\bigcup_{i \in I} C_\alpha^i, \bigcup_{i \notin I} C_\alpha^i\}$ est une bi-partition de $\mathcal{F}$, et*
- *pour tout $\{X, Y\} \in \mathcal{F}$, il existe un noeud* `dégénéré` *α et $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$ tel que $\{X, Y\} = \{\bigcup_{i \in I} C_\alpha^i, \bigcup_{i \notin I} C_\alpha^i\}$.*

Par convention, si $d(\alpha) \leq 3$, le noeud sera étiqueté `dégénéré`. Dans ce cas, l'arbre étiqueté sera unique, et sera appelé **l'arbre représentatif** de la famille. La figure 1.5 donne par exemple l'arbre représentatif de la famille $\mathcal{F} = \Big\{\big\{\{v\}, V \setminus \{v\}\big\} : v \in V\Big\} \cup \Big\{\big\{\{a, b\}, \{c, d, e, f, g\}\big\}, \big\{\{a, b, c, g\}, \{d, e, f\}\big\}, \big\{\{d, e\}, \{a, b, c, f, g\}\big\}, \big\{\{d, f\}, \{a, b, c, e,$

$g\}\}, \{\{e,f\},\{a,b,c,d,g\}\}\Big\}$, avec $V = \{a,b,c,d,e,f,g\}$. Les membres de la famille qui ne sont pas forts sont $\{\{d,e\},\{a,b,c,f,g\}\}$, $\{\{d,f\},\{a,b,c,e,g\}\}$ et $\{\{e,f\},\{a,b,c,d,g\}\}$.

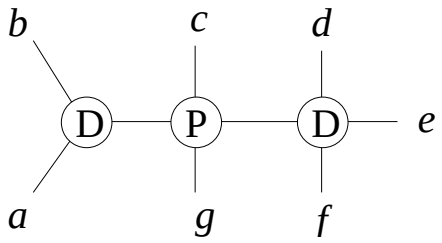


Fig. 1.5 : – L'arbre représentatif de la famille bi-partitive $\mathcal{F}$ (D = `dégénéré` et P = `premier`).

Pour les familles faiblement bi-partitives, un troisième type de noeuds apparaît :

Théorème 1.12. *[Mon03] Soit $\mathcal{F}$ une famille faiblement bi-partitive sur V, et soit T l'arbre d'inclusion des membres forts de $\mathcal{F}$. Les noeuds internes de $T = (V_T, E_T)$ peuvent être étiquetés* ***premiers***, ***dégénérés*** *ou* ***linéaires***, *et les voisins des noeuds* ***linéraires*** *peuvent être ordonnés, tel que*

- *pour tout $e \in E_T$, $\{C_e^1, C_e^2\}$ est une bi-partition forte de $\mathcal{F}$,*
- *pour tout noeud* ***dégénéré*** *α, et pour tout $\emptyset \subsetneq I \subsetneq \{1,\ldots,d(\alpha)\}$, $\{\bigcup_{i\in I} C_\alpha^i, \bigcup_{i\notin I} C_\alpha^i\}$ est une bi-partition de $\mathcal{F}$,*
- *pour tout noeud* ***linéaire*** *α, et pour tout $a,b \in \{1,\ldots,d(\alpha)\}$ tel que $a \leq b$, $\{\bigcup_{i\in\{a,\ldots,b\}} C_\alpha^i, V \setminus \bigcup_{i\notin\{a,\ldots,b\}} C_\alpha^i\}$ est une bi-partition de $\mathcal{F}$,*
- *pour tout $\{X,Y\} \in \mathcal{F}$, il existe un noeud* ***dégénéré*** *α et $\emptyset \subsetneq I \subsetneq \{1,\ldots,d(\alpha)\}$ tel que $\{X,Y\} = \{\bigcup_{i\in I} C_\alpha^i, \bigcup_{i\notin I} C_\alpha^i\}$, ou il existe un noeud* ***linéaire*** *α et $a,b \in \{1,\ldots,d(\alpha)\}$, $a \leq b$, tel que $\{X,Y\} = \{\bigcup_{i\in\{a,\ldots,b\}} C_\alpha^i, V \setminus \bigcup_{i\notin\{a,\ldots,b\}} C_\alpha^i\}$.*

1.5.5 Relation entre les familles

Il existe de fortes relations entre les familles partitives et les familles bi-partitives.

Lemme 1.13. *[Mon03] Soit $\mathcal{F}$ une famille de sous-ensembles de V, et soit $\mathcal{F}'$ la famille de bi-partitions de V suivante :*

$$\mathcal{F}' = \big\{\{A, V \setminus A\} : A \in \mathcal{F} \text{ et } A \neq V\big\}.$$

Alors

- *$\mathcal{F}$ est partitive si et seulement si $\mathcal{F}'$ est bi-partitive,*
- *$\mathcal{F}$ est faiblement partitive si et seulement si $\mathcal{F}'$ est faiblement bi-partitive.*

Le lemme suivant donne un moyen similaire pour transformer une famille bi-partitive en famille partitive, et inversement.

Lemme 1.14. *Soient V un ensemble et $v \in V$. Soient $\mathcal{F}$ une famille de bi-partitions de V et $\mathcal{F}' = \left\{ V' : v \notin V' \text{ et } \{V', V \setminus V'\} \in \mathcal{F} \right\}$. Alors $\mathcal{F}$ est une famille bi-partitive si et seulement si $\mathcal{F}'$ est une famille partitive.*

Dans ce cas, soit T' l'arbre représentatif de $\mathcal{F}'$. Alors l'arbre représentatif T de $\mathcal{F}$ est isomorphe à T' avec un noeud additionnel pour l'élément v, adjacent à la racine de T'. De plus, un noeud interne de T est étiqueté **dégénéré** *(respectivement* **premier***) si et seulement si le noeud correspondant dans T' est étiqueté* **dégénéré** *(respectivement* **premier***).*

Démonstration. $\mathcal{F}$ respecte la condition (1) et la condition (2) de la définition 1.5.4 si et seulement si $\mathcal{F}'$ respecte les conditions (1) et (2) de la définition 1.5.2. Il suffit de montrer que la condition (3) de la définition 1.5.2 est vérifiée par $\mathcal{F}$ si et seulement si la condition (3) de la définition 1.5.4 est vérifiée par $\mathcal{F}'$. Soient $\{X, V \setminus X\}$ et $\{Y, V \setminus Y\}$ deux bi-partitions de V. Sans perte de généralité, on suppose $v \notin X$ et $v \notin Y$. Suivant la définition, si deux bi-partitions $\{X, V \setminus X\}$ et $\{Y, V \setminus Y\}$ se chevauchent, alors X et Y se chevauchent. Inversement si X et Y se chevauchent, les bi-partitions doivent se chevaucher puisque $v \in (V \setminus X)$ et $v \in (V \setminus Y)$: les deux ensembles $V \setminus X$ et $V \setminus X$ se chevauchent également.

Ainsi les deux bi-partitions $\{X, V \setminus X\}$ et $\{Y, V \setminus Y\}$ se chevauchent si et seulement si les deux ensembles X et Y se chevauchent. Dans ce cas les bi-partitions $\{X \cap Y, V \setminus (X \cap Y)\}$, $\{X \cup Y, V \setminus (X \cup Y)\}$, $\{X \setminus Y, V \setminus (X \setminus Y)\}$, $\{Y \setminus X, V \setminus (Y \setminus X)\}$ et $\{X \Delta Y, V \setminus (X \Delta Y)\}$ sont dans $\mathcal{F}$ si et seulement si $X \cap Y$, $X \cup Y$, $X \setminus Y$, $Y \setminus X$ et $X \Delta Y$ sont dans $\mathcal{F}'$.

Pour la deuxième partie du lemme, on montre que l'arbre T est l'arbre représentatif de $\mathcal{F}$. Soit $\{X, V \setminus X\}$ une bi-partition de V telle que $v \notin X$. La bi-partition $\{X, V \setminus X\}$ est un élément fort $\mathcal{F}$ si et seulement si X est un élément fort de $\mathcal{F}'$. Dans ce cas, il existe un noeud α pour X dans T'. Si $X \neq V \setminus \{v\}$, la bi-partition $\{X, V \setminus X\}$ est la bi-partition $\{C_e^1, C_e^2\}$ pour l'arête e entre entre α et son père dans T'. Si $X = V \setminus \{v\}$, alors $\{X, V \setminus X\}$ est la bi-partition $\{C_e^1, C_e^2\}$ pour l'arête e qu'on a ajoutée dans T entre la racine et la feuille pour v.

On note qu'un noeud α dans l'arbre représentatif de $\mathcal{F}$ est **dégénéré** si et seulement si il existe une bi-partition $\{X, V \setminus X\} \in \mathcal{F}$ dans $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$ telle que $|I| \in \{2, \ldots, d(\alpha) - 2\}$ et $X = \bigcup_{i \in I} C_\alpha^i$. Sans perte de généralité, on suppose $v \notin X$. Une telle partition existe si et seulement si $X \in \mathcal{F}'$, et ainsi α est **dégénéré** dans T'. □

Chapitre 2

Décomposition modulaire

2.1 Définition

2.1.1 Modules d'un graphe

Un **module** d'un graphe est un ensemble M de sommets non vide tel que pour chaque sommet v n'appartenant pas à M, tous les sommets de M sont adjacents à v, ou aucun sommet de M n'est adjacent à v.

La notion de module apparaît sous différents noms dans la littérature : **intervalle** selon Fraïsé [Fra84], **closed set** selon Gallai [Gal67], **partitive set** dans le livre de Golumbic [Gol80] ou encore **ensemble homogène**.

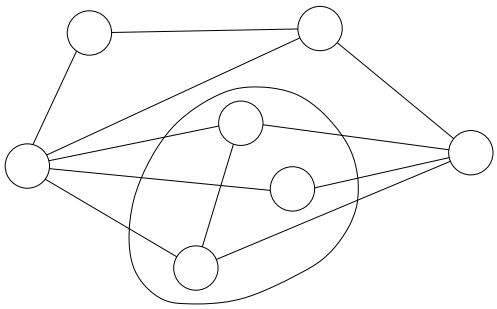


Fig. 2.1 : – Exemple d'un module dans un graphe.

On peut remarquer qu'un ensemble de sommets de taille 1 ou $|V|$ est toujours un module. Ces modules particuliers sont appelés **modules triviaux**. Un graphe est **premier**, ou **indécomposable**, par rapport à la décomposition modulaire si tous ses modules sont triviaux. On dira simplement qu'il est premier s'il n'y a pas d'ambiguïté sur la décomposition considérée. Le plus petit graphe premier non trivial[1] est le P_4.

[1] Il existe un sous-ensemble $M \subseteq V$ non vide qui n'est pas un module.

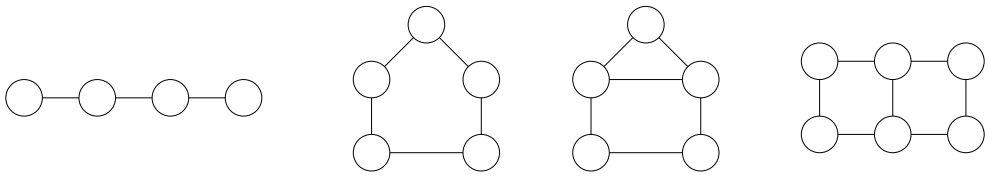


Fig. 2.2 : – Quelques graphes premiers.

2.1.1.1 Propriétés simples des modules

Proposition 2.1. *Soit $G = (V, E)$ un graphe, M un module de G, et V' un sous-ensemble de V tel que $M \cap V' \neq \emptyset$. Alors $M \cap V'$ est un module de $G[V']$.*

Proposition 2.2. *Soit $G = (V, E)$ un graphe, M un module de G et V' un sous-ensemble de M. Alors V' est un module de G si et seulement si V' est un module de $G[M]$.*

Proposition 2.3. *M est un module de G si et seulement si M est un module de $\overline{G}$.*

2.1.1.2 Contraction et substitution

Si un graphe possède un module non trivial, alors il a une redondance de l'information d'adjacence dont on peut se servir pour coder le graphe ; si un sommet u est adjacent à un sommet d'un module M, alors il en sera de même pour tous les sommets de M.

Soit G un graphe et M un module non trivial. Alors tous les graphes $G - (M \setminus \{v\})$, pour $v \in M$, sont isomorphes. Ce graphe sera la **contraction** du module M dans G, et sera note $G\,{}^{v}_{M}$.

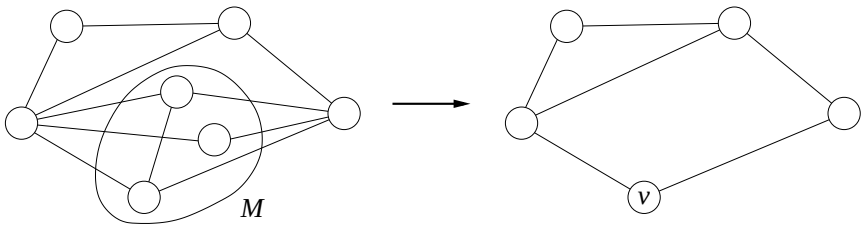


Fig. 2.3 : – Contraction d'un module.

L'opération inverse de la contraction est la **substitution** d'un sommet par un graphe. Elle a été étudiée en premier par Sabidussi [Sab59]. Soit $G = (V_G, E_G)$ et $H = (V_H, E_H)$ deux graphes tels que $V_G \cap V_H = \emptyset$ (sinon on prend une copie disjointe de H), et soit $v \in V_G$. Le graphe $G\,{}^{H}_{v}$ est le graphe d'ensemble de sommets $(V_G \setminus \{v\}) \cup V_H$, et l'ensemble d'arêtes $(E_G \cap \mathcal{P}_2(V_G \setminus \{v\})) \cup E_H \cup \big\{\{u, w\} : u \in V_H \text{ et } w \in N_G(v)\big\}$. On voit immédiatement que si $V_H \neq \emptyset$, alors V_H est un module de $G\,{}^{H}_{v}$. Si $v_1, \ldots v_r$ sont des sommets du graphe G, différents deux à deux, et $H_1, \ldots, H_r$ sont des graphes, alors l'ordre des substitutions des v_i par les H_i n'a pas d'importance : on notera le graphe $G\,{}^{H_1 \ \cdots \ H_r}_{v_1 \ \cdots \ v_r}$ obtenu après les substitutions. Si M est un module de G, alors $G = (G\,{}^{v}_{M})\,{}^{G[M]}_{v}$.

On peut répéter la contraction d'un module non trivial jusqu'à ce que l'ensemble des graphes obtenus soient tous premiers. L'ensemble des opérations peut être représenté par un arbre, et on peut rajouter des informations supplémentaires afin de pouvoir reconstruire le graphe originel avec des opérations de substitutions. Mais cette décomposition ne serait pas unique ; par exemple le graphe complet de trois sommets $\{a, b, c\}$ peut être décomposé par le module $\{a, b\}$, par le module $\{b, c\}$ et par le module $\{a, c\}$. Après l'opération de contraction, les graphes obtenus sont tous triviaux. On a donc trois façons différentes pour décomposer le même graphe.

2.1.2 Théorème de décomposition modulaire

Le théorème suivant donne un moyen de décomposer de façon unique un graphe par des modules. Gallai l'a énoncé initialement pour l'étude des graphes de comparabilité. On reverra qu'il y a une forte relation entre la décomposition modulaire et cette classe de graphes.

Théorème 2.4 (Théorème de décomposition modulaire [Gal67]). *Soit $G = (V, E)$ un graphe ayant au moins deux sommets. Alors exactement une des conditions suivantes est respectée :*

(a) *G n'est pas connexe,*

(b) *$\overline{G}$ n'est pas connexe,*

(c) *G et $\overline{G}$ sont connexe ; il existe un sous-ensemble U de V et une unique partition $\mathcal{P}$ de V tels que :*
- *$|U| \geq 3$,*
- *$G[U]$ est un sous-graphe premier maximal[2] de G,*
- *pour tout ensemble S appartenant à la partition $\mathcal{P}$, S est un module et $|S \cap U| = 1$.*

À chaque cas du théorème de décomposition modulaire correspond une décomposition :

(a) si G n'est pas connexe, alors G est décomposé en les graphes induits par ses composantes connexes,

(b) si $\overline{G}$ n'est pas connexe, alors G est décomposé en les graphes induits par les composantes connexes de $\overline{G}$,

(c) sinon, G est décomposé en les graphes induits par les classes de la partition $\mathcal{P}$.

La décomposition modulaire est l'application récursive de cette décomposition, jusqu'à ce qu'il n'y ait plus que des graphes d'un seul sommet.

2.1.2.1 Arbre de décomposition modulaire

La décomposition modulaire d'un graphe G peut être représentée par un arbre enraciné. La racine correspond à la décomposition de G, et les sous-arbres enracinés aux fils d'un

[2] C'est à dire qu'il n'existe pas de $U' \supsetneq U$ tel que $G[U']$ soit premier.

noeud h correspondent à la décomposition des graphes obtenus par la décomposition du graphe correspondant au noeud h.

Si on est dans le cas (a) du théorème de décomposition, le noeud de l'arbre sera étiqueté `parallèle`. Si le cas est le (b), il sera étiqueté `série`. Enfin, dans le cas (c), le noeud sera étiqueté `premier`.

Chaque noeud h de l'arbre correspond à un module V_h de G. On notera G_h le graphe induit par V_h. Si h est une feuille, alors G_h est un graphe possédant un seul sommet, et si h est la racine de l'arbre, $G_h = G$.

Soit h un noeud de l'arbre, et soit $h_1, \ldots, h_k$ ses fils dans l'arbre. Le **graphe caractéristique** de h, noté G_h^*, est le graphe $G^{v_1 \ \ldots \ v_k}_{V_{h_1} \ \ldots \ V_{h_k}}$, c'est à dire le graphe obtenu après la contraction des modules $V_{h_1}, \ldots, V_{h_k}$ en sommets $v_1, \ldots v_k$.

Si le noeud est étiqueté `parallèle`, son graphe caractéristique sera un graphe sans arêtes. S'il est étiqueté `série`, il s'agira d'un graphe complet. Enfin, si le noeud est `premier`, le graphe caractéristique sera un graphe premier.

Le lemme suivant montre que l'arbre de décomposition modulaire, avec les graphes caractéristiques, sera toujours de taille linéaire en la taille du graphe original.

Lemme 2.5. *Soit $G = (V, E)$ un graphe et $T = (V_T, E_T)$ son arbre de décomposition modulaire. Alors on a $\sum_{h \in V_T} n(G_h^*) \leq 2 \times n(G)$ et $\sum_{h \in V_T} m(G_h^*) \leq m(G)$.*

Démonstration. Pour chaque noeud interne h, le nombre de sommets de G_h^* est égal au nombre de fils de h dans l'arbre T. Au total, $\sum_{h \in V_T} n(G_h^*)$ est donc borné par le nombre de noeuds dans T, qui est au plus deux fois le nombre de feuilles de T, car chaque noeud interne a au moins 2 fils.

On montre par récurrence sur l'arbre de décomposition que $\sum_{h \in V_T} m(G_h^*) \leq m(G)$. Si l'arbre n'a qu'une feuille, alors $\sum_{h \in V_T} m(G_h^*) = m(G) = 0$. Sinon, soit r la racine de l'arbre, $h_1, \ldots, h_k$ ses fils, T_i le sous-arbre enraciné en h_i, pour $i \in \{1, \ldots, r\}$, et M_i l'ensemble des feuilles de T_i. On voit que $G^* = G_r^*$. Alors par récurrence, pour tout $i \in \{1, \ldots, k\}$, $\sum_{h \in V(T_i)} m(G_h^*) \leq m(G_{h_i})$. $\{M_1, \ldots, M_k\}$ est une partition de V en modules. Chaque arête est donc soit contenue dans un module M_i, soit a une extrémité dans M_i, et l'autre dans M_j, avec $i \neq j$. Soit E' l'ensemble des arêtes qui ne sont pas contenues dans un module. $\{E', E(G_{h_1}), \ldots, E(G_{h_k})\}$ est une partition de E. Comme G^* est isomorphe a $G[\{v_1, \ldots v_k\}]$, où $v_i \in M_i$ pour tout $i \in \{1, \ldots, k\}$, on a $m(G^*) \leq |E'|$. Au final, on obtient :

$$\sum_{h \in V_T} m(G_h^*) = m(G^*) + \sum_{i \in \{1,\ldots,k\}} \sum_{h \in V(T_i)} m(G_h^*) \leq |E'| + \sum_{i \in \{1,\ldots,k\}} |E(G_{h_i})| \leq m(G)$$

□

Ce lemme aura son importance par la suite, au moment où on voudra se servir de la décomposition modulaire pour résoudre un problème sur un graphe. Par exemple, si on effectue une opération linéaire sur chaque graphe caractéristique apparaissant dans la décomposition modulaire d'un graphe G, le temps total sera linéaire en la taille du

graphe G. Par contre, si on effectue une opération en temps constant sur chaque graphe caractéristique (ce qui arrive par exemple si chaque graphe caractéristique a une taille bornée), le temps total sera en $O(n)$, car il y a $O(n)$ feuilles dans l'arbre de décomposition.

2.1.3 Exemple de décomposition

On donne en exemple la décomposition modulaire du graphe de la figure 2.4. Pour commencer, on remarque que ce graphe n'est pas connexe. Ses deux composantes connexes sont $\{a, b, c, d, e, f, g, h\}$ et $\{i, j, k, l, m, n\}$. Le graphe est donc décomposé en deux graphes, qui correspondent aux graphes induits par ces ensembles de sommets. La racine de l'arbre de décomposition modulaire sera donc un noeud étiqueté `parallèle`. Il aura deux fils, qui seront les racines des sous-arbres correspondants à la décomposition de ces deux composantes connexes.

On regarde le graphe induit par $\{a, b, c, d, e, f, g, h\}$. Ce graphe est connexe et co-connexe. On regarde ses modules maximaux différents de V, qui sont $\{a\}$, $\{b, c\}$, $\{d\}$, $\{e\}$ et $\{f, g, h\}$. Ce sous-graphe est donc décomposé en cinq graphes, qui sont les graphes induits par $\{a\}$, $\{b, c\}$, $\{d\}$, $\{e\}$ et $\{f, g, h\}$. Si on prend un sommet par ensemble de la partition (par exemple a, b, d, e, h) on obtient un C_5, qui est le graphe caractéristique de ce sous-graphe. Le noeud de l'arbre correspondant sera donc étiqueté `premier`, avec comme graphe caractéristique un C_5. Il aura cinq fils, chaque fils correspondant à un sommet du C_5. Trois de ses fils seront des feuilles (correspondant aux sommets a, d et e).

Un des fils correspondra au sous-graphe induit par $\{b, c\}$. Ce sous-graphe n'est pas co-connexe, donc le noeud correspondant sera `série`. Le dernier fils correspondra au sous-graphe induit par $\{f, g, h\}$, qui n'est pas connexe, donc le noeud sera `parallèle`, puis le sous-graphe induit par $\{f, g\}$ sera décomposé par l'opération `série`.

On fait de même pour le sous-graphe induit par $\{i, j, k, l, m, n\}$, qui est connexe et co-connexe, et dont le graphe caractéristique est un P_4. Ceci nous donnera l'autre sous-arbre fils de la racine. Au final on obtiendra l'arbre de décomposition modulaire donné en figure 2.5.

2.1.4 Familles des modules comme famille partitive

Proposition 2.6. *Soient X et Y des modules du graphe G, tels que X chevauche Y. Alors $X \cap Y$, $X \cup Y$, $X \setminus Y$, $Y \setminus X$ et $X \Delta Y$ sont des modules de G.*

On a donc immédiatement :

Théorème 2.7. *Soit G un graphe. La famille de tous les modules de G est une famille partitive.*

On dit qu'un module X est **fort** s'il est un élément fort de la famille partitive des modules, c'est à dire s'il n'existe pas d'autre module Y qui chevauche X.

On voit que dans chacun des cas du théorème de décomposition 2.4, on décompose le graphe suivant des modules forts. De plus, ces modules forts sont des éléments maximaux

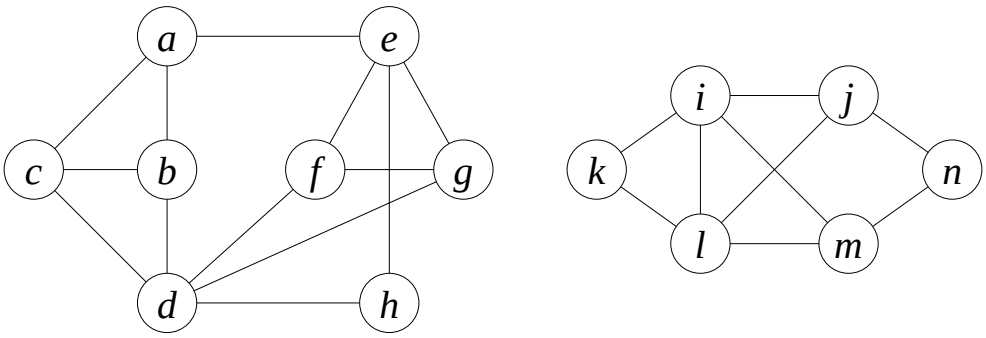


Fig. 2.4 : – Le graphe de la section 2.1.3.

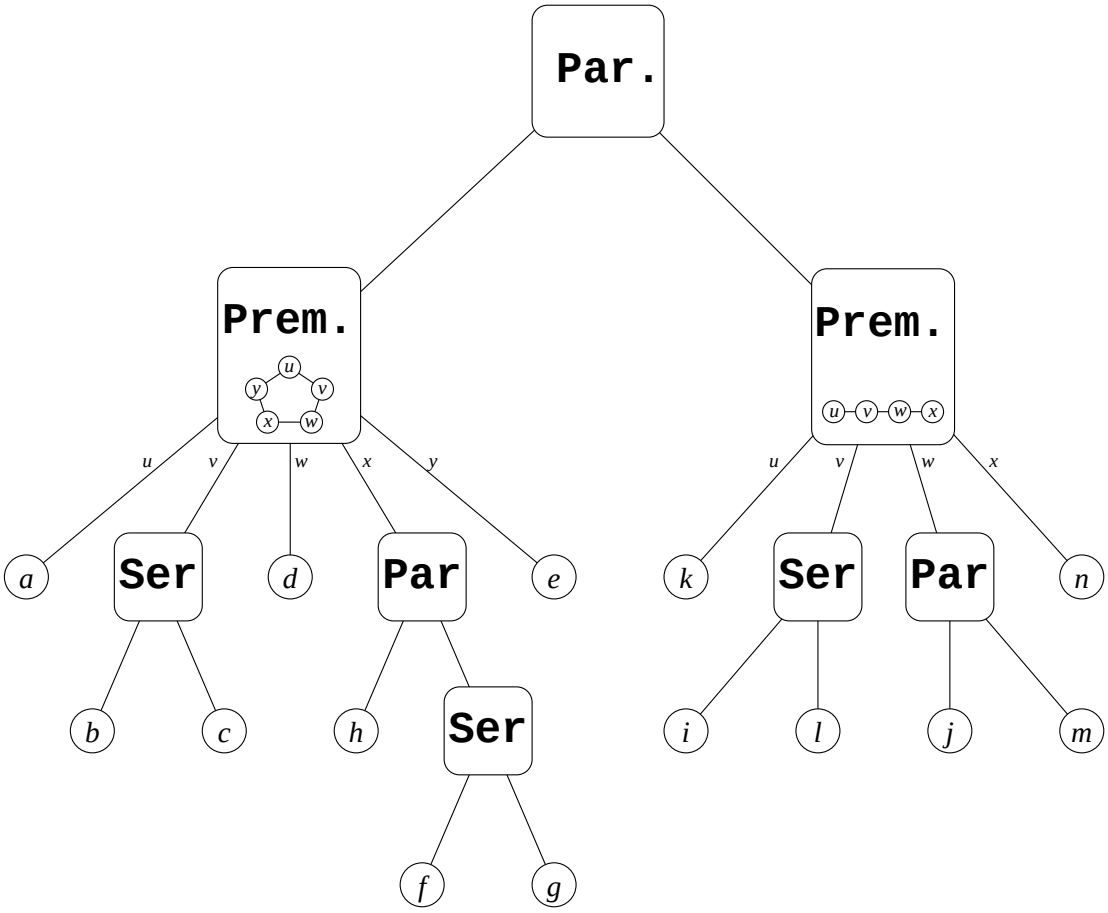


Fig. 2.5 : – L'arbre de décomposition modulaire.

(par inclusion) parmi les modules forts du graphe autre que le module V. Donc chaque noeud de l'arbre de décomposition modulaire défini en section 2.1.2 correspond à un module fort, et tous les modules forts sont représentés dans cet arbre.

L'arbre représentatif de la famille partitive des modules d'un graphe est donc isomorphe à l'arbre de décomposition modulaire du graphe. Les noeuds `dégénérés` de l'arbre représentatif des modules correspondent aux noeuds `séries` et `parallèles` de l'arbre de décomposition, et les noeuds `premiers` correspondent aux noeuds `premiers`.

L'arbre de décomposition modulaire est donc une représentation de tous les modules d'un graphe. Un graphe peut avoir un nombre exponentiel de modules (par exemple le graphe complet de n sommets possède $2^n - 1$ modules), mais l'arbre permet de les représenter tous de façon implicite en espace linéaire.

2.1.5 La classe des cographes

Une classe de graphes est attachée en particulier à la décomposition modulaire. Cette classe est celle des graphes se décomposant entièrement par cette décomposition.

On dit qu'un graphe est un **cographe** s'il n'y a aucun noeud `premier` dans son arbre de décomposition modulaire. Dans ce cas, son arbre de décomposition modulaire sera appelé son **co-arbre**.

Soit $G = (V, E)$ un graphe, et $u \in V$. Un **faux jumeau** (respectivement, **vrai jumeau**) de u est un sommet v tel que $N(v) = N(u)$ ($N[v] = N[u]$). On dira qu'on ajoute un faux jumeau (vrai jumeau) à un graphe G si on ajoute un nouveau sommet w adjacent à tous les sommets dans $N_G(u)$ ($N_G[u]$), où u est un sommet du graphe. Ces deux opérations sont des extensions de sommets.

Théorème 2.8 ([CLS81, Sum73]). *Soit G un graphe. Les propositions suivantes sont équivalentes :*

(0) G est un cographe.
(1) G est complètement décomposable par rapport à la décomposition modulaire.
(2) G est sans P_4 induit.
(3) Pour tout sous-graphe induit H de G, soit H est non connexe ou soit $\overline{H}$ est non connexe.
(4) G peut être obtenu d'un seul sommet par une séquence d'extensions de sommets : ajout d'un vrai jumeau ou ajout d'un faux jumeau.

On peut noter la proposition suivante :

Proposition 2.9. *Un graphe G est un cographe si et seulement si tout sous-graphe induit de G avec au moins 3 sommets possède un module non trivial.*

Démonstration. Si G n'est pas un cographe, alors il possède un P_4 induit, qui ne possède pas de module non trivial.

Si G est un cographe, alors pour tout sous-graphe induit H, H est non connexe ou $\overline{H}$ est non connexe. Supposons que H n'est pas connexe. Si H est un graphe sans arête,

alors n'importe quel sous-ensemble de 2 sommets est un module non trivial. Sinon une composante connexe d'au moins 2 sommets est un module non trivial. Si $\overline{H}$ n'est pas connexe, le même argument peut être utilisé sur $\overline{H}$. □

2.1.6 Graphes premiers

On peut déduire du théorème 2.8 la proposition suivante :

Proposition 2.10. *Tout graphe premier d'au moins 4 sommets possède un P_4 induit.*

Les seuls graphes premiers qui sont des cographes ont donc au plus 2 sommets. Il s'agit des graphes K_1, K_2 et $\overline{K_2}$. Le théorème suivant est une extension de la proposition 2.10

Théorème 2.11 ([CI98]). *Soit $G = (V, E)$ un graphe premier d'au moins 4 sommets. Soit $W \subseteq V$ l'ensemble des sommets $x \in V$ tel qu'il existe un $X \subseteq V$ induisant un P_4 avec $x \in X$. Alors $|V \setminus W| \leq 1$. De plus, si $V \setminus W = \{x\}$, alors il existe $X \subseteq V$ induisant un taureau, avec $x \in X$ et $d_{G[X]}(x) = 2$.*

La classe des graphes premier n'est pas héréditaire, mais néanmoins on a le théorème suivant :

Théorème 2.12 ([ER90]). *Soit G un graphe premier d'un moins 5 sommets. Alors G a un sous-graphe induit H premier, tel que $1 \leq n(G) - n(H) \leq 2$.*

2.1.7 Décomposition d'autres structures discrètes

La décomposition modulaire peut être étendue à d'autres objets mathématiques que les graphes. La notion de modules et de décomposition modulaire a été introduite pour l'étude des ordres. Ces notions se sont ensuite généralisées aux graphes ; la décomposition modulaire d'un ordre correspond à la décomposition modulaire du graphe orienté représentant cet ordre. Par la suite, elle a été généralisée aux hypergraphes et aux k-structures.

2.1.7.1 Graphes orientés

Un sous-ensemble $M \subseteq V$ est un module d'un graphe orienté $G = (V, A)$ si pour tout sommet $u \notin M$ et pour tout $v, w \in M$, $(u, v) \in A \iff (u, w) \in A$ et $(v, u) \in A \iff (w, u) \in A$.

La famille des modules d'un graphe orienté forme une famille faiblement partitive. L'arbre de décomposition comporte donc un autre type de noeud, le noeud `linéaire`. Les graphes caractéristiques de ces noeuds sont des ordres totaux (c'est à dire un graphe orienté (V, A) tel que la relation d'adjacence soit anti-symétrique, anti-réflexive et transitive).

2.1.7.2 Hypergraphes

Un **hypergraphe** est un couple $(V, \mathcal{E})$ où $\mathcal{E} \subseteq \mathcal{P}(V)$. Un hypergraphe est simple si $\forall X, Y \in \mathcal{E}, X \not\subseteq Y$. Un **comité** dans un hypergraphe est un sous-ensemble $M \subseteq V$ tel que pour tout $X, Y \in \mathcal{E}$ tels que $X \cap M \neq \emptyset$ et $Y \cap M \neq \emptyset$, $(X \setminus M) \cup (Y \cap M) \in \mathcal{E}$.

Théorème 2.13. *[CHM81, MR84] L'ensemble des comités d'un graphe simple est une famille partitive.*

2.1.7.3 k-structures

Une k-structure[3], pour $k \geq 2$, est un triplet (V, E, k) où V est l'ensemble des sommets, et $E : V^k \setminus \{\{x\}^k : x \in V\} \to \mathbb{N}$. Un sous-ensemble $M \subseteq V$ est un module d'une k-structure (V, E, k) si pour tout $e_1, \ldots, e_k \in V$, avec $\{e_1, \ldots, e_k\}$ chevauchant M, pour tout $i \in \{1, \ldots, k\}$ tel que $e_i \in M$, et pour tout $e' \in M$,

$$E(e_1, \ldots, e_k) = E(e_1, \ldots, e_{i-1}, e', e_{i+1}, \ldots e_k).$$

Un graphe orienté est assimilable à une 2-structure telle que les arcs aient deux valeurs possibles (« présente » et « non-présente »). Dans ce cas les modules du graphes correspondent exactement aux modules de la 2-structure. Les k-structures sont des généralisations des k-hypergraphes (hypergraphes (V, E) tels que pour tout $X \in \mathcal{E}$, $|X| = k$).

Théorème 2.14. *[ER90] L'ensemble des modules d'une 2-structure est une famille faiblement partitive.*

Théorème 2.15. *[EM94] L'ensemble des modules d'une k-structure, pour $k \geq 3$, est une famille partitive.*

Les graphes caractéristiques des noeuds `dégénérés` sont des k-structures telles que $E(u, v) = E(u', v')$ pour tout $u \neq v$, et $u' \neq v'$. Dans le cas $k = 2$, les graphes caractéristiques des noeuds `linéaires` sont des 2-structures telles qu'il existe un ordre des sommets $v_1, \ldots, v_n$ et $k, l \in \mathbb{N}$, $k \neq l$, avec $E(v_i, v_j) = k$ si $i < j$, et $E(v_i, v_j) = l$ si $i > j$. On voit facilement que les graphes caractéristiques des graphes orientés sont des cas particuliers de ceux des 2-structures.

2.1.8 Algorithmes calculant la décomposition modulaire

La recherche d'algorithmes de plus en plus efficaces pour la décomposition modulaire, et ses variantes, a produit beaucoup de publications depuis quelques décennies.

On peut remarquer qu'il est facile de trouver un module non trivial maximal d'un graphe par élagage. En utilisant cette observation et le théorème de décomposition, on voit immédiatement que la décomposition modulaire peut être calculée en temps polynomial. La suite des articles sur les algorithmes de décompositions avait comme but de diminuer

[3] Attention à ne pas les confondre avec les structures du chapitre 5, bien qu'elles aient des similitudes.

le temps, de généraliser les structures sur lesquelles on calculait la décomposition, et enfin de simplifier les algorithmes.

Le premier algorithme sur les graphes non orientés est donné par [CJS72] et fonctionne en temps $O(n^4)$. L'article [MS89] présente un algorithme incrémental en temps $O(n^2)$ pour les graphes non orientés. Cet algorithme sera ensuite généralisé aux graphes orientés et aux 2-structures [McC95]. La publication [Spi92] donne un algorithme en temps $O(n + m\alpha(m, n))$ en passant par une structure intermédiaire, le P_4-*tree*. Le temps linéaire sera enfin atteint [CH94, MS99].

Depuis, les auteurs cherchent à simplifier les algorithmes de décomposition. [HMP04, Mon03] donnent un algorithme linéaire simple pour les graphes non orientés en passant par une permutation factorisante (une permutation des sommets telle que tous les sommets des modules soient consécutifs). [MM05] présente un algorithme linéaire pour les graphes orientés.

Dans le cas particulier des cographes, [CPS85] présentent un algorithme linéaire incrémental de reconnaissance, produisant le co-arbre si le graphe est un cographe. Très récemment, des algorithmes plus simples on été présentés [BCH03, HP05].

2.2 Décomposition modulaire et classes de graphes

2.2.1 Classes de graphes stables par substitutions

On dit qu'une classe $\mathcal{G}$ est **stable par substitutions** si pour tout $G, H \in \mathcal{G}$, et $v \in V(G)$, alors $G_v^H \in \mathcal{G}$.

Un graphe de **comparabilité** est un graphe qui admet une orientation transitive. Les graphes orientés transitifs sont stables par substitutions, et donc les graphes (non orientés) de comparabilités le sont aussi.

On voit facilement que les cographes sont stables par substitutions. Cela est aussi le cas pour les graphes faiblement chordaux, et les graphes parfaits, car une substitution ne peut pas créer un cycle sans cordes avec au moins 5 sommets. On verra que cela est également le cas des graphes de permutation. Par contre, les graphes chordaux ne sont pas stables par substitutions car une substitution peut faire apparaître un C_4.

Plus généralement, toute classe de graphes qui peut être définie par un ensemble de sous-graphes interdits premiers est stable par substitutions.

2.2.2 Classes de graphes joliment décomposables

Dans le but d'utiliser la décomposition modulaire pour résoudre un problème, on doit se restreindre à un classe de graphes « joliment » décomposables par la décomposition modulaire. « Joliment » veut dire qu'on a une meilleure caractérisation des graphes premiers que de la classe considérée. Par exemple les graphes premiers peuvent être de taille bornée ou appartenir à une classe plus restreinte. La classe des cographes est l'exemple le plus

naturel : l'ensemble des graphes premiers autorisés est vide. On a quelques autres exemples de classes de graphes joliment décomposables.

2.2.2.1 Graphes P_4-réductibles et P_4-clairsemés

Un graphe est **P_4-réductible** si chaque sommet appartient à au plus un P_4. Un graphe est **P_4-clairsemé**[4] si tous les sous-graphes induits de 5 sommets induisent au plus un P_4. On voit facilement que les graphes P_4-réductibles sont des généralisations des cographes, et les graphes P_4-clairsemés sont des généralisations des graphes P_4-réductibles.

Un graphe est une **araignée maigre** si on peut partitionner son ensemble de sommets en un ensemble stable S et une clique C de telle façon que $|S| = |C|$ ou $|S| = |C| + 1$, chaque sommet de S ait exactement un voisin dans C, et chaque sommet de C ait au plus un voisin dans S. Un graphe est une **araignée grasse** si son complément est une araignée maigre. Un graphe est une **araignée** s'il s'agit d'une araignée maigre ou grasse.

Théorème 2.16. *[GV97] Les graphes premiers apparaissant dans la décomposition modulaire des graphes P_4-réductibles sont des P_4 et des taureaux.*

Théorème 2.17. *[Hoa83, JO92] Les graphes premiers apparaissant dans la décomposition modulaire des graphes P_4-clairsemés sont des araignées.*

Vanherpe [Van99] présente d'autres classes de graphes généralisants les cographes, et leur comportement par rapport à la décomposition modulaire.

2.2.2.2 Graphes sans P_5 et $\overline{P_5}$ induit

Théorème 2.18 ([GR97]). *Soit G un graphe. Alors G est sans P_5 et $\overline{P_5}$ induit si et seulement si pour tous les graphes premiers H dans la décomposition modulaire :*
- *soit H est un C_5,*
- *soit H est sans P_5, $\overline{P_5}$ et C_5 induit.*

2.2.2.3 Graphes sans P_5 et gem induit

Brandstädt et Kratsch [BK02] donnent une caractérisation, utilisant la décomposition modulaire, des graphes sans P_5 et gem induit. Un graphe est un graphe **co-biparti couplé** si son ensemble de sommets V est partitionnable en deux cliques K_1 et K_2, avec $|K_1| = |K_2|$ ou $|K_1| = |K_2| - 1$, tels que chaque sommet d'une clique a au plus un voisin dans l'autre clique, et qu'il y ait au plus un sommet dans chaque clique qui n'ait pas de voisin dans l'autre clique. Un graphe est un graphe **spécifique** s'il est un sous-graphe induit du complément d'un des trois graphes donnés en figure 2.6.

Un graphe est **co-chordal** si son complémentaire est un graphe chordal. Un sommet v est un sommet **co-simplicial** de G si v est un sommet simplicial dans $\overline{G}$ (c'est-à-dire si le non-voisinage $V \setminus N[v]$ est un ensemble stable de G).

On définit la notion **d'extension**[5] d'un sommet dans un graphe. Soit G un graphe et v

[4] *P_4-sparse* en anglais.

[5] On peut noter que cette extension est une composition en coupe (définie à la page 60).

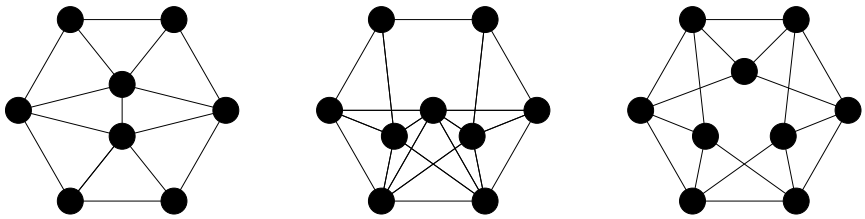


Fig. 2.6 : – Compléments des graphes spécifiques.

un sommet de G. On note $\text{ext}(G, v)$ le graphe G' résultant de G après le remplacement de v par un C_5 $(v_1, v_2, v_3, v_4, v_5)$ tel que v_2, v_4 et v_5 ont le même voisinage dans $G - v$ que v dans G, et v_1 et v_3 n'ont pas de voisin dans $G - v$. Pour $U \subseteq V$, soit $\text{ext}(G, U)$ le résultat de l'application de l'extension ext sur tous les sommets de U. Notons que le graphe résultant ne dépend pas de l'ordre dans lequel on étend les sommets de U.

Pour $k \geq 0$, soit $\mathcal{G}_k$ la classe des graphes premiers $G' = \text{ext}(G, Q)$, obtenus en prenant un graphe co-chordal sans gem induit G, et en étendant une clique Q de k sommets co-simpliciaux de G. $\mathcal{G}_0$ est la classe des graphes premiers co-chordaux sans gem induit. La **classe 3** est l'ensemble des graphes $\bigcup_{k \geq 0} \mathcal{G}_k$.

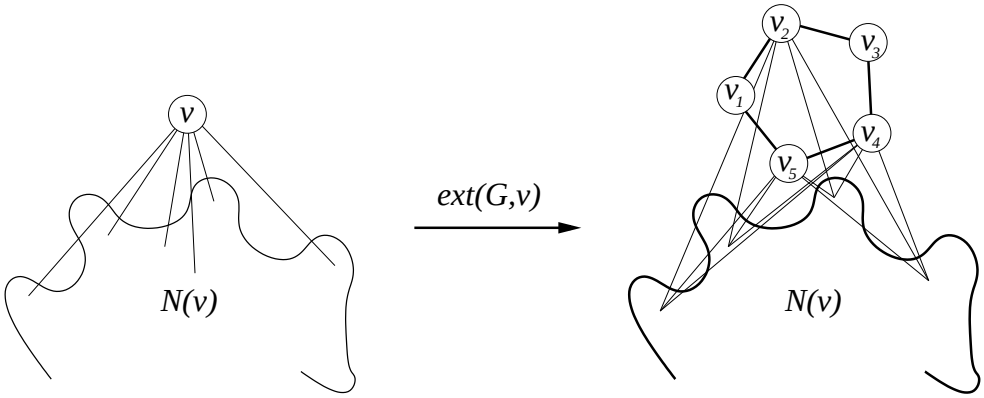


Fig. 2.7 : – L'extension d'un sommet par un C_5.

Théorème 2.19 ([BK02]). *Un graphe connexe et co-connexe est un graphe sans P_5 et sans gem si et seulement si les conditions suivantes sont respectées :*

- *Tous les modules de G, autres que V, induisent des cographes (c'est à dire que tous les noeuds de l'arbre de décomposition autre que la racine, sont des noeuds* **séries** *ou* **parallèles**).
- *Pour le graphe caractéristique G^* de G, une des conditions suivantes est respectée :*
  - *G^* est un graphe co-biparti couplé.*
  - *G^* est un graphe spécifique.*
  - *il existe un $k \geq 0$ tel que G^* soit dans $\mathcal{G}_k$.*

2.2.3 Relation avec la largeur de clique

La classe des cographes est exactement la classe des graphes de largeur de clique au plus 2, et il s'agit également de la classe de largeur NLC 1.

Théorème 2.20. *[CO00] Soit G un graphe. Alors* $\mathrm{cwd}(G) = \max\{\mathrm{cwd}(H) : H$ *est un graphe caractéristique d'un noeud interne dans la décomposition modulaire de* $G\}$.

Les classes des graphes de largeur de clique au plus k, pour $k \in \mathbb{N}$ fixé, sont donc des classes stables par substitutions. Il en est de même pour les classes de largeur NLC au plus k [Joh98]. Par contre cela n'est pas le cas des graphes de largeur arborescente bornée, puisque la classe des cographes contient tous les graphes complets.

Les modules et la décomposition modulaire sont souvent utilisés comme outils pour montrer que certaines classes de graphes sont de largeur de clique bornée. Brandstädt, Le et Mosca montrent notamment que :

Lemme 2.21. *[BLM05] Les graphes co-chordaux sans gem induit ont une largeur de clique d'au plus* 4. *Les graphes de la classe 3 ont une largeur de clique d'au plus* 5.

On voit facilement que les graphes co-bipartis couplés sont de largeur de clique au plus 4, et que les graphes spécifiques sont de largeur de clique bornée. On a donc :

Théorème 2.22. *[BLM05] Les graphes sans P_5 et gem induit sont de largeur de clique au plus* 5.

2.3 Applications de la décomposition modulaire

Une forte motivation pour travailler sur la décomposition modulaire est qu'elle sert à résoudre des problèmes de graphes. Paradoxalement, la littérature évoquant comment se servir de la décomposition modulaire pour résoudre un certain problème n'est pas très abondante, si on la compare à celle sur le calcul de la décomposition elle même. Möhring et Radermacher [MR84] donnent une vue de différentes applications de la décomposition modulaire, sur les graphes et sur d'autres structures.

La décomposition peut servir, par exemple, à résoudre les problèmes MAXIMUM CUT [BJ00], MINIMUM FILL IN et à trouver la largeur arborescente du graphe [BT03].

Malheureusement, la décomposition modulaire n'aide pas dans tous les cas. Si un problème peut être résolu avec cette décomposition, alors il doit être polynomial sur la classe des cographes. Dans ce cas, l'algorithme pour le problème utilisant le co-arbre est un point de départ pour trouver, si possible, l'algorithme travaillant sur l'arbre de décomposition modulaire. Par exemple, les problèmes NOMBRE ACHROMATIQUE et LIST COLORING sont NP-complets sur les cographes [Bod89, JS97]. Il se peut aussi qu'un problème soit polynomial sur les cographes, et NP-complet sur la classe des graphes dont les seuls graphes premiers apparaissant dans la décomposition modulaire sont des P_4, comme COLORATION $\lambda_{2,1}$ [BBF02].

Nous présentons dans cette section une collection de problèmes et leur algorithme utilisant la décomposition modulaire. Certains de ces résultats sont précédemment connus et utilisées, directement ou indirectement, dans des algorithmes.

2.3.1 Orientation transitive

Un graphe est un graphe de **comparabilité** s'il existe une orientation transitive de ses arêtes. On a donc deux problèmes : celui de décider si un graphe est un graphe de comparabilité, et celui de trouver une orientation transitive d'un graphe de comparabilité.

Lemme 2.23. *[Gol80] Soit G un graphe et $\mathcal{D}$ son arbre de décomposition modulaire. Alors G est un graphe de comparabilité si et seulement si le graphe caractéristique de chaque noeud premier de $\mathcal{D}$ est un graphe de comparabilité.*

A priori, le lemme précédent ne nous apporte rien ; pour savoir si un graphe possède une orientation transitive, il faut savoir si les graphes caractéristiques de sa décomposition modulaire possèdent une orientation transitive. Mais, pour un graphe premier, ce problème est plus facile à traiter, car un graphe premier de comparabilité possède uniquement deux orientations transitives, dont l'une est l'opposée de l'autre.

Lemme 2.24. *[Gol80] Soit G un graphe de comparabilité premier avec au moins 2 sommets. Alors G possède exactement deux orientations transitives distinctes, dont l'une est l'opposée de l'autre.*

La décomposition modulaire peut donc servir à reconnaître les graphes de comparabilité. Pour cela il suffit de calculer l'arbre de décomposition du graphe, et de vérifier que chaque graphe caractéristique dans l'arbre est un graphe de comparabilité. De plus, si on garde l'orientation transitive des graphes caractéristiques, on a une structure compacte contenant toutes les orientations transitives du graphe.

Une orientation transitive d'un graphe de comparabilité peut être trouvée en temps linéaire [MS99]. Calculer la fermeture transitive d'un graphe orienté est équivalent à la multiplication de matrices [FM71, Mur71]. (Pour tester si un graphe est transitif, il suffit de vérifier si la matrice d'adjacence booléenne avec des 1 sur la diagonale est égale à son carré.) Les graphes de comparabilités peuvent donc être reconnus en temps $O(n^\alpha)$, où α désigne le meilleur temps connu pour la multiplication de matrices (actuellement $2.376\ldots$).

Soit π une permutation des nombres de 1 à n. On dénote par $G[\pi]$ le graphe de sommets $\{1,\ldots,n\}$, avec i et j adjacent si $(i-j)\times(\pi_i^{-1}-\pi_j^{-1})<0$. Un graphe G est un **graphe de permutation** s'il existe une permutation π telle que G est isomorphe à $G[\pi]$.

Lemme 2.25. *[PLE71] Un graphe G est un graphe de permutation si et seulement si G et $\overline{G}$ sont des graphes de comparabilité.*

Les graphes de permutation peuvent être reconnus en temps linéaire [MS99], grâce à un algorithme de décomposition modulaire linéaire, et un algorithme linéaire d'orientation transitive d'un graphe de comparabilité et de son complément.

2.3.2 Ensemble stable

Le problème ENSEMBLE STABLE PONDÉRÉ est une généralisation du problème ENSEMBLE STABLE, où l'on a une fonction de poids $w : V \to \mathbb{N}$ sur les sommets du graphe. Le **poids** d'un sous-ensemble $S \subseteq V$ est $\sum_{v \in S} w(v)$. On note par $\alpha_w(G)$ le poids maximum d'un ensemble stable de G. On note par $\omega_w(G)$ le poids maximum d'une clique de G, c'est-à-dire $\alpha_w(\overline{G})$.

Ensemble stable pondéré (optimisation)
INSTANCE : un graphe G et $w : V \to \mathbb{N}$ une fonction de poids.
SORTIE : $\alpha_w(G)$.

Les opérations à résoudre pour chaque noeud de l'arbre de décomposition modulaire sont données par le lemme suivant.

Lemme 2.26. *[Chv75, MR84] Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $\alpha_w(G_h) = w(v_h)$,*
2. *Si h est un noeud* ***parallèle****, alors $\alpha_w(G_h) = \sum_{i=1}^{r} \alpha_w(G_{h_i})$,*
3. *Si h est un noeud* ***série****, alors $\alpha_w(G_h) = \max_{i \in \{1,\ldots,r\}} \alpha_w(G_{h_i})$,*
4. *Si h est un noeud* ***premier****. Soit $G_h^* = (V^*, E^*)$ le graphe caractéristique de h, avec $V_h^* = \{v_1, \ldots, v_r\}$ et tel que v_i soit le sommet représentatif du module V_{h_i}. Soit $w^* : V^* \to \mathbb{N}$ une fonction de poids telle que $w^*(v_i) = \alpha_w(G_{h_i})$. Alors $\alpha_w(G_h) = \alpha_{w^*}(G_h^*)$.*

On voit que si on arrive à résoudre ENSEMBLE STABLE PONDÉRÉ sur les graphes premiers de la décomposition, on peut donc résoudre ENSEMBLE STABLE PONDÉRÉ (et donc ENSEMBLE STABLE) sur le graphe originel. De plus, si on peut résoudre les sous-problèmes en temps polynomial, alors on pourra résoudre le problème originel en temps polynomial.

L'algorithme précédent nous permet de calculer $\alpha_w(G)$ en utilisant la décomposition modulaire. Cet algorithme peut être modifié pour calculer un ensemble stable de poids maximum à la place de $\alpha_w(G)$. Dans l'algorithme suivant, $S_w(G)$ représente un ensemble stable de poids maximum du graphe G.

Lemme 2.27. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $S_w(G_h) = \{v_h\}$,*
2. *Si h est un noeud* ***parallèle****, alors $S_w(G_h) = \bigcup_{i=1}^{r} S_w(G_{h_i})$,*
3. *Si h est un noeud* ***série****, $S_w(G_h) = S_w(G_{h_j})$, où $\alpha_w(G(h_j)) = \max_{i \in \{1,\ldots,r\}} \alpha_w(G_{h_i})$,*
4. *Si h est un noeud* ***premier****. Soit $G_h^* = (V^*, E^*)$ le graphe caractéristique de h, avec $V_h^* = \{v_1, \ldots, v_r\}$ et tel que v_i soit le sommet représentatif du module V_{h_i}. Soit $w^* : V^* \to \mathbb{N}$ une fonction de poids telle que $w^*(v_i) = \alpha_w(G_{h_i})$. Soit S^* un ensemble stable de poids maximum de G^* avec la fonction de pondération w^*. Alors $S_w(G_h) = \bigcup_{i : v_i \in S^*} S_w(G_{h_i})$.*

Lorsque que l'on veut utiliser la décomposition modulaire pour résoudre le problème ENSEMBLE STABLE sur une certaine classe de graphes $\mathcal{G}$, il faut et il suffit de résoudre le problème ENSEMBLE STABLE PONDÉRÉ sur les graphes premiers de la classe de graphes $\mathcal{G}$. Ceci est avantageux uniquement si on a une meilleure caractérisation de ces graphes premiers. Il est donc important d'avoir des théorèmes de structure des classes de graphes.

Remarque. *Pour le problème* ENSEMBLE STABLE*, les opérations à effectuer pour les décompositions séries et parallèles sont des cas particuliers de l'opération à effectuer pour un noeud premier. Nous verrons que cela n'est pas toujours le cas. Le fait que le graphe caractéristique soit premier dans le cas des noeuds premiers peut simplifier l'opération, car par exemple le graphe est connexe et co-connexe.*

Pour résoudre le problème ENSEMBLE STABLE en utilisant la décomposition modulaire, on a du passer à une généralisation du problème qui est ENSEMBLE STABLE PONDÉRÉ. On verra que c'est ce qui arrive en général lorsque qu'on essaye de résoudre un problème en utilisant la décomposition modulaire. Pour certains problèmes, le sous-problème à résoudre sur les graphes premiers sera le problème originel. Pour d'autre, il s'agira de la version pondérée. Enfin, pour une dernière catégorie, il s'agira de problèmes encore plus généraux.

Remarque. *L'application du lemme 2.26 sur le complémentaire du graphe nous donne directement les sous-problèmes à résoudre pour le problème* CLIQUE PONDÉRÉE*.*

2.3.3 Coloration

Une **coloration pondérée** d'un graphe $G = (V, E)$ avec une fonction de poids $w : V \to \mathbb{N}$ est un multi-ensemble $\mathcal{C} = \langle V_1, \ldots, V_k \rangle$ d'ensembles stables de G, tel que pour tout $v \in V$, $|\langle V' \in \mathcal{C} : v \in V' \rangle| \geq w(v)$. Le **nombre chromatique pondéré** d'un graphe G avec une fonction de poids w est la plus petite taille d'une coloration pondérée de G et w. Il sera noté $\chi_w(G)$. Une **coloration pondérée minimum** de (G, w) est une coloration pondérée de (G, w) avec un nombre minimum de couleurs. On appellera également une coloration pondérée minimum de G avec fonction de poids w une coloration minimum de (G, w). Une **partition en cliques pondérée** de (G, w) est une coloration pondérée de $(\overline{G}, w)$. On notera par $\kappa_w(G) = \chi_w(\overline{G})$.

Coloration pondérée minimum (optimisation)
INSTANCE : un graphe G et une fonction de poids $w : V \to \mathbb{N}$.
SORTIE : une coloration minimum de (G, w).

Lemme 2.28. *[Chv75, MR84] Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $\chi_w(G_h) = w(v_h)$,*
2. *Si h est un noeud* ***parallèle****, alors $\chi_w(G_h) = \max_{i \in \{1,\ldots,r\}} \chi_w(G_{h_i})$,*
3. *Si h est un noeud* ***série****, alors $\chi_w(G_h) = \sum_{i=1}^{r} \chi_w(G_{h_i})$,*

4. *Si h est un noeud* **premier**. *Soit $G_h^* = (V^*, E^*)$ le graphe caractéristique de h, avec $V^* = \{v_1, \ldots, v_r\}$ et tel que v_i soit le sommet représentatif du module V_{h_i}. Soit $w^* : V^* \to \mathbb{N}$ une fonction de poids telle que $w^*(v_i) = \chi_w(G_{h_i})$. Alors $\chi_w(G_h) = \chi_{w^*}(G_h^*)$.*

Remarque. *L'application du lemme 2.28 sur le complémentaire du graphe nous donne directement les sous-problèmes à résoudre pour le problème* PARTITION EN CLIQUES.

2.3.4 Problèmes de domination

Nous allons commencer par des variantes de problèmes de domination, qui ont des sous-problèmes plus faciles que le problème DOMINATION. Le problème ENSEMBLE DOMINANT CONNEXE, qui demande pour un graphe G et un $k \in \mathbb{N}$ si G possède un ensemble dominant connexe de taille au plus k, est NP-complet [GJ78]. On peut remarquer que tout graphe connexe a un ensemble dominant connexe. Voici la version d'optimisation du problème :

Ensemble dominant connexe (optimisation)
INSTANCE : un graphe $G = (V, E)$.
SORTIE : taille minimum d'un ensemble dominant connexe de G, $+\infty$ s'il n'en possède pas.

On note par $\gamma_c(G)$ la taille d'un plus petit ensemble dominant connexe de G. Les sous-problèmes à résoudre sur l'arbre de décomposition modulaire sont donnés par le lemme suivant.

Lemme 2.29. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $\gamma_c(G_h) = 1$,*
2. *Si h est un noeud* **parallèle**, *alors $\gamma_c(G_h) = +\infty$,*
3. *Si h est un noeud* **série**, *alors s'il existe $i \in \{1, \ldots, r\}$ tel que G_{h_i} ait un sommet dominant, alors $\gamma_c(G_h) = 1$, sinon $\gamma_c(G_h) = 2$,*
4. *Si h est un noeud* **premier**, *alors $\gamma_c(G_h) = \gamma_c(G_h^*)$, où $G_h^* = (V^*, E^*)$ est le graphe caractéristique de h.*

Démonstration. Les affirmations 1, 2 et 3 sont immédiates. On suppose donc que h est `premier`. Soit D^* un ensemble dominant connexe de G_h^*. On construit l'ensemble D en prenant un sommet de V_{h_i} pour tous les sommets v_i dans D^*. Alors $|D| = |D^*|$. Comme D^* est connexe et possède au moins 2 sommets, D domine tout le graphe G_h.

Inversement, soit D est un ensemble dominant de G_h, et soit D^* l'ensemble des v_i tel que $D \cap V_{h_i} \neq \emptyset$. Alors $|D^*| \leq |D|$. Comme chaque sommet est dominé dans G_h par D, si $D \cap V_{h_j} = \emptyset$, alors il existe k tel que $\{v_j, v_k\} \in E(G_h^*)$ et $D \cap V_{h_k} \neq \emptyset$. Tout sommet de G_h^* sera donc dominé par un sommet dans D^*. □

On voit que dans ce cas, le sous-problème à résoudre sur les graphes caractéristiques des noeuds premiers est le problème original. Le problème suivant est également NP-complet.

Clique dominante (optimisation)
INSTANCE : un graphe $G = (V, E)$.
SORTIE : taille minimum d'une clique dominante de G, $+\infty$ s'il n'en possède pas.

On note par $\gamma_{\mathrm{cl}}(G)$ la taille de la plus petite clique dominante de G. On a le lemme suivant, la preuve est similaire à la preuve précédente.

Lemme 2.30. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $\gamma_{cl}(G_h) = 1$,*
2. *Si h est un noeud* ***parallèle****, alors $\gamma_{cl}(G_h) = +\infty$,*
3. *Si h est un noeud* ***série****, alors s'il existe $i \in \{1, \ldots, r\}$ tel que G_{h_i} ait un sommet dominant, alors $\gamma_{cl}(G_h) = 1$, sinon $\gamma_{cl}(G_h) = 2$,*
4. *Si h est un noeud* ***premier****, alors $\gamma_{cl}(G_h) = \gamma_{cl}(G_h^*)$, où $G_h^* = (V^*, E^*)$ est le graphe caractéristique de h.*

Ensemble stable dominant (optimisation)
INSTANCE : un graphe $G = (V, E)$.
SORTIE : taille minimum d'un ensemble stable dominant de G.

On note par $\gamma_{\mathrm{i}}(G)$ la taille d'un plus petit ensemble dominant connexe de G. Le problème suivant est la version pondérée du problème :

Ensemble stable dominant pondéré (optimisation)
INSTANCE : un graphe $G = (V, E)$ et une fonction de poids w sur V.
SORTIE : poids minimum $\gamma_{\mathrm{i}w}(G)$ d'un ensemble stable dominant de G.

Lemme 2.31. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

1. *Si h est une feuille, alors $\gamma_{iw}(G_h) = w(v_h)$,*
2. *Si h est un noeud* ***parallèle****, alors $\gamma_{iw}(G_h) = \sum_{i=1}^{r} \gamma_{iw}(G_{h_i})$,*
3. *Si h est un noeud* ***série****, alors $\gamma_{iw}(G_h) = \min_{i \in \{1,\ldots,r\}} \gamma_{iw}(G_{h_i})$,*
4. *Si h est un noeud* ***premier****. Soit $G_h^* = (V^*, E^*)$ le graphe caractéristique de h, avec $V^* = \{v_1, \ldots, v_r\}$ et tel que v_i soit le sommet représentatif du module V_{h_i}. Soit $w^* : V^* \to \mathbb{N}$ une fonction de poids telle que $w^*(v_i) = \gamma_{iw}(G_{h_i})$. Alors $\gamma_{iw}(G_h) = \gamma_{iw^*}(G_h^*)$.*

Pour finir, on donne l'algorithme pour ENSEMBLE DOMINANT. Soit $G = (V, E)$ un graphe, $f : V \to \{1, 2\}$ un étiquetage des sommets de V et $D \subseteq V$. Soit :

$$a(v) = \begin{cases} 2 & \text{si } f(v) = 2 \text{ et } v \text{ est un sommet isolé de } G[D], \\ 1 & \text{sinon,} \end{cases}$$

$$w(f, G, D) = \sum_{v \in D} a(v),$$

$$\gamma(f, G) = \min_{D \subseteq V} w(f, G, D) \quad \text{où } D \text{ est un ensemble stable de } G.$$

Lemme 2.32. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

- *Si h est une feuille, alors $\gamma(G_h) = 1$.*
- *Si h est un noeud interne étiqueté* **`parallèle`***, alors $\gamma(G_h) = \sum_{i=1}^{r} \gamma(G_{h_i})$.*
- *Si h est un noeud interne étiqueté* **`série`***, alors s'il existe un $i \in \{1, \ldots, r\}$ tel que G_{h_i} ait un sommet dominant, alors $\gamma(G_h) = 1$, sinon $\gamma(G_h) = 2$.*
- *Si h est un noeud interne étiqueté* **`premier`***. Soit $V(G_h^*) = \{v_1, \ldots, v_r\}$, et soit f l'étiquetage des sommets de G_h^* tel que $f(v_i) = 1$ si G_{h_i} a un sommet dominant, 2 sinon. Alors $\gamma(G_h) = \gamma(f, G_h^*)$.*

Démonstration. Les observations pour les noeuds étiquetés `séries` et `parallèles` sont immédiates. Soit h un noeud `premier`. Soit D^* un sous-ensemble dominant de G_h^* qui minimise $w(F, G_h^*, D^*)$. On construit un sous-ensemble D de $V(G_h)$ de la manière suivante. Si $v_i \in D^*$ et $f(v_i) = 1$, alors on ajoute à D un sommet dominant de G_{h_i}. Si $v_i \in D^*$, $f(v_i) = 2$ et v_i a un voisin dans $G_h^*[D^*]$, alors on ajoute un sommet de G_{h_i} à D. Enfin si $v_i \in D^*$, $f(v_i) = 2$ et v_i n'a pas de voisin dans $G_h^*[D^*]$, on ajoute un sommet de G_{h_i} et un sommet de G_{h_j} à D, où $j \in \{1, \ldots, r\}$ tel que v_i et v_j soient adjacents dans G_h^* (on sait qu'un tel j existe, car le graphe est premier). Il est facile de voir que $|D| = w(f, G_h^*, D^*)$, et que D est un ensemble dominant de G_h. Donc $\gamma(G_h) \leq \gamma(f, G_h^*)$.

Soit D un ensemble dominant minimum de G_h. Soit $i \in \{1, \ldots, r\}$ tel que $D \cap V(G_{h_i}) \neq \emptyset$. Si G_{h_i} a un sommet dominant, ou s'il existe $j \neq i$ tel que $D \cap V(G_{h_j}) \neq \emptyset$ et $\{v_i, v_j\} \in E(G_h^*)$, alors $|D \cap V(G_{h_i})| = 1$ (sinon on pourrait construire un ensemble dominant plus petit que D). Sinon $|D \cap V(G_{h_i})| = 2$. Donc $w(f, G_h^*, D^*) = |D|$, où D^* est l'ensemble des v_i, pour lesquels $D \cap V(G_{h_i}) \neq \emptyset$. Donc $\gamma(f, G_h^*) \leq \gamma(G_h)$. □

Le problème qu'il faut résoudre sur les graphes premiers est donc une généralisation du problème DOMINATION, mais il ne s'agit pas de sa version pondérée. On peut remarquer que dans ce cas, on n'est pas obligé de calculer les valeurs pour tous les noeuds de l'arbre. En effet, si le graphe est connexe, les seules informations nécessaires pour calculer la taille d'un des plus petits ensembles dominants sont les informations de la racine (l'étiquette et le graphe caractéristique), et, pour chaque fils de la racine, de savoir si le sous-graphe induit par le module comporte un sommet dominant (qui peut être trouvé en temps $O(n + m)$). Si le graphe n'est pas connexe, il faut calculer la taille d'un des plus petits ensembles dominants de chaque composante connexe.

2.3.5 Chemin induit

Chemin induit
INSTANCE : un graphe $G = (V, E)$.
QUESTION : quel est le plus grand k tel que G ait un P_k induit ?

Ce problème est NP-complet (référence [GT23] dans [GJ78]). On note $\mathrm{ip}(G)$ le plus grand k tel que G ait un P_k induit.

Lemme 2.33. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*
- *Si h est une feuille, alors* $\mathrm{ip}(G_h) = 1$.
- *Si h est un noeud interne étiqueté* ***parallèle****, alors* $\mathrm{ip}(G_h) = \max_{i\in\{1,\ldots,r\}} \mathrm{ip}(G_{h_i})$.
- *Si h est un noeud interne étiqueté* ***série****, alors si pour tout $i \in \{1, \ldots, r\}$, $G(h_i)$ est complet, alors* $\mathrm{ip}(G_h) = 2$, *sinon* $\mathrm{ip}(G_h) = \max\big(3, \mathrm{ip}(G_{h_1}), \ldots, \mathrm{ip}(G_{h_r})\big)$.
- *Si h est un noeud interne étiqueté* ***premier****, alors*

$$\mathrm{ip}(G_h) = \max\left(3, \mathrm{ip}(G_h^*), \mathrm{ip}(G_{h_1}), \ldots, \mathrm{ip}(G_{h_r})\right).$$

La preuve est immédiate, car le graphe P_k, pour $k \geq 4$, est premier, et donc ne peut pas être décomposé par la décomposition modulaire du graphe. Le lemme nous donne directement un algorithme récursif pour calculer $\mathrm{ip}(G)$ en utilisant la décomposition modulaire de G : le problème à résoudre sur les graphes caractéristiques est le problème original.

2.3.6 Feedback vertex set

Soit $G = (V, E)$ un graphe. Un ensemble $F \subseteq V$ est un **FVS** (pour *feedback vertex set*) s'il contient au moins un sommet de chaque cycle de G.

Feedback vertex set
INSTANCE : un graphe $G = (V, E)$.
QUESTION : quelle est la taille d'un FVS de taille minimum de G ?

Ce problème est NP-complet (référence [GT7] dans [GJ78]). On notera $\mu(G)$ la taille d'un plus petit FVS de G. Un ensemble V' est une **couverture de sommets** si pour toute arête $\{u, v\} \in E$, $u \in V'$ ou $v \in V'$. On note $\beta(G)$ la taille minimum d'une couverture de sommets de G. Il est connu que $\beta(G) = n - \alpha(G)$ (voir par exemple [GJ78]). Calculer $\beta(G)$ revient donc à calculer $\alpha(G)$.

Soit $G = G'\begin{smallmatrix} G_1 & \cdots & G_r \\ v_1 & \cdots & v_r \end{smallmatrix}$, avec $G' = (V', E')$ et pour tout $i \in \{1, \ldots, r\}$, $G_i = (V_i, E_i)$, et soit $f : V' \to \{1, 2, 3, 4\}$ un étiquetage des sommets de G'. Dans cette section on appellera **le poids** de f la valeur suivante :

$$w(f, G', G_1, \ldots, G_r) = \sum_{i=1}^{r} a(f(v_i), G_i)$$

où $G = (V, E)$ et

$$a(e, G) = \begin{cases} \mu(G) & \text{si} \quad e = 1, \\ \beta(G) & \text{si} \quad e = 2, \\ |V| - 1 & \text{si} \quad e = 3, \\ |V| & \text{si} \quad e = 4 \end{cases}$$

On dit qu'un étiquetage f est **valide** s'il respecte les conditions suivantes :
- l'ensemble des sommets étiquetés 4 forment un FVS de G'.
- si un sommet est étiqueté 1, alors tous ses voisins sont étiquetés 4.
- si un sommet est étiqueté 2, alors tous ses voisins sont étiquetés 4 sauf au plus un, qui est étiqueté 3.

On note :

$$\mu'(G', G_1, \ldots, G_r) = \min_f w(f, G', G_1, \ldots, G_r) \quad \text{où } f \text{ est un étiquetage valide de } G'.$$

Lemme 2.34. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*
- *Si h est une feuille, alors $\mu(G_h) = 0$.*
- *Si h est un noeud interne étiqueté* ***parallèle****, alors $\mu(G_h) = \sum_{i=1}^{r} \mu(G_{h_i})$.*
- *Si h est un noeud interne étiqueté* ***série****, alors*

$$\mu(G_h) = \min_{i \in \{1,2,\ldots,r\}} (\min(\mu(G_{h_i}), \beta(G_{h_i}) - 1) + n(G_h) - n(G_{h_i})).$$

- *Si h est un noeud interne étiqueté* ***premier****, alors*

$$\mu(G_h) = \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r}).$$

Démonstration. Soit f un étiquetage valide. Soit, pour tout $i \in \{1, \ldots, r\}$, F_i un FVS minimum de G_{h_i}, C_i une couverture de sommets minimum de G_{h_i}, $V_i' \subset V_{h_i}$ tel que $|V_i'| = |V_{h_i}| - 1$, et $V_i = V_{h_i}$. On définit :

$$A_i = \begin{cases} F_i & \text{si} \quad f(v_i) = 1, \\ C_i & \text{si} \quad f(v_i) = 2, \\ V_i' & \text{si} \quad f(v_i) = 3, \\ V_i & \text{si} \quad f(v_i) = 4, \end{cases}$$

et $F = \bigcup_{i=1}^{r} A_i$. On peut remarquer que $|F| = w(f, G_h^*, G_{h_1}, \ldots, G_{h_r})$.

On montre que F est un FVS de G_h. Soit $C = (u_1, u_2, \ldots, u_k)$ un cycle de G_h. Supposons que aucun u_i n'appartient à F. On définit $u_1', u_2', \ldots, u_k'$ tel que pour tout $j \in \{1, \ldots, k\}$, $u_j \in V_i$ si et seulement si $u_j' = v_i$.
- Si tous les v_j sont différents, alors $C' = (u_1', \ldots, u_k)$ est un cycle de G_h^*. Or nous savons que l'ensemble des sommets étiquetés 4 forment un FVS de G_h^*. Donc il y a au moins un sommet du cycle C' qui est étiqueté 4. Il y a donc au moins un sommet de C dans F. Contradiction.

- Sinon il existe i tel que v_i apparaît au moins deux fois dans $(u'_1, \ldots, u'_k)$. Donc $f(v_i) \in \{1, 2\}$.
  - si $f(v_i) = 1$ alors tous les voisins de v_i sont étiquetés 4. Comme aucun sommet de C n'est dans F, tous les u'_j, $j \in \{1, 2, \ldots, k\}$ sont égaux. Donc $C \subseteq V_i$. Or $F \cap V_i$ est un FVS de G_{h_i}, d'où contradiction.
  - sinon $f(v_i) = 2$. Il y a au plus un sommet dans les ensembles homogènes maximaux correspondants aux voisins de v_i qui n'est pas dans F. Alors soit tous les u'_j, $j \in \{1, 2, \ldots, k\}$ sont égaux, et on a une contradiction (car une couverture de sommet est un FVS), soit un et un seul sommet de C' est différent de v_i. Donc il existerait un chemin de taille supérieure ou égale à 2 dans G_{h_i} qui ne passe par aucun sommet de F. Contradiction avec le fait que $F \cap V_i$ est une couverture de sommet de G_{h_i}.

Donc F est un FVS de G_h, et $\mu(G_h) \leq |F|$. Comme cela est vrai pour tout étiquetage valide, $\mu(G_h) \leq \mu(G_h^*, G_{h_1}, \ldots, G_{h_r})$.

Soit F un FVS de $G = G_h^* \begin{smallmatrix} G_{h_1} & \cdots & G_{h_r} \\ v_1 & \cdots & v_r \end{smallmatrix}$. Alors pour tout i, $F \cap V_i$ est un FVS de G_{h_i}. Soit $f : V_h^* \to \{1, 2, 3, 4\}$ l'étiquetage suivant :

$$f(v_i) = \begin{cases} 1 & \text{si } F \cap V_i \text{ n'est pas une couverture de sommets de } G_{h_i}, \\ 2 & \text{si } F \cap V_i \text{ est une couverture de sommets de } G_{h_i} \text{ et} \\ & |F \cap V_i| < |V_i| - 1, \\ 3 & \text{si } |F \cap V_i| = |V_i| - 1, \\ 4 & \text{si } F \cap V_i = V_i. \end{cases}$$

On remarque que $w(f, G_h^*, G_{h_1}, \ldots, G_{h_r}) \leq |F|$, car $\mu(G) \leq \beta(G) \leq |V| - 1$ pour tout graphe $G = (V, E)$.

L'ensemble des sommets étiquetés 4 forme un FVS de G_h^*, sinon on pourrait construire un cycle de G_h à partir d'un cycle C' de G_h^* en prenant un sommet non couvert par F par ensemble homogène correspondant à un sommet du cycle C'.

Soit i tel que $F \cap V_i$ ne soit pas une couverture de sommets de G_{h_i}, alors pour tout j tel que v_i et v_j soient ajdacents, $V_j \subseteq F$, sinon il existerait un cycle de taille 3 dont les sommets ne sont pas dans F. Donc si $f(v_i) = 1$, alors tous les voisins de v_i sont étiquetés 4.

Soit i tel que $|F \cap V_i| < |V_i| - 1$, alors il ne peut pas y avoir deux sommets parmi tous les sommets des modules correspondants aux voisins de v_i qui ne soient pas dans F, sinon on pourrait construire un cycle de taille 4 dont les sommets ne sont pas dans F. Donc tous les voisins de v_i sont étiquetés 4 par f, sauf au plus un qui est étiqueté 3.

L'étiquetage f est valide, donc $w(f, G_h^*, G_{h_1}, \ldots, G_{h_r}) \geq \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r})$, et par conséquence $|F| \geq \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r})$. Comme cela est vrai pour tout FVS de G_h, on a $\mu(G_h) \geq \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r})$. On a finalement $\mu(G_h) = \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r})$.

On peut conclure :

- Si h est une feuille, alors G_h est un sommet isolé, et $\mu(G_h) = 0$.

- Si h est étiqueté `parallèle`, alors G_h^* est un graphe sans arêtes, l'étiquetage $f : V \in V_h^* \to 1$ est valide, et son poids est minimum.
- Si h est étiqueté `série`, alors G_h^* est un graphe complet. Un étiquetage valide donne une valeur différente de 4 pour au plus 2 sommets. Ainsi, soit un sommet est étiqueté 1 et tous les autres sont étiquetés 4, soit un sommet est étiqueté 2, un autre 3 et le reste 4. Les autres étiquetages valides ont un poids supérieur ou égal.
- Si h est étiqueté `premier`, on a bien $\mu(G_h) = \mu'(G_h^*, G_{h_1}, \ldots, G_{h_r})$.

□

2.3.7 Nombre de séparateurs minimaux

Soit $G = (V, E)$ un graphe, et soit $u, v \in V$ tels que u et v soient dans la même composante connexe de G. Un sous-ensemble $S \subseteq V \setminus \{u, v\}$ est un **uv-séparateur** si u et v sont dans deux composantes différentes de $G - S$. S est un **uv-séparateur minimal** si S est un uv-séparateur et pour tout $S' \subsetneq S$, S' n'est pas un uv-séparateur. S est un **séparateur minimal** s'il existe $u, v \in V$ tel que S soit un uv-séparateur minimal. On remarque que $\emptyset$ est un séparateur minimal si et seulement si le graphe n'est pas connexe.

Soit $s(G)$ le nombre de séparateurs minimaux de G.

Lemme 2.35. *Soit h un noeud de l'arbre de décomposition modulaire, et soit $h_1, \ldots, h_r$ ses fils si h est un noeud interne.*

- *Si h est une feuille, alors $s(G_h) = 0$.*
- *Si h est un noeud interne étiqueté `parallèle`, alors $s(G_h) = 1 + \sum_{i=1}^{r} s(G_{h_i})$.*
- *Si h est un noeud interne étiqueté `série`, alors $s(G_h) = \sum_{i=1}^{r} s(G_{h_i})$.*
- *Si h est un noeud interne étiqueté `premier`, alors $s(G_h) = s(G^*) + \sum_{i=1}^{r}(s(G_{h_i}) - a_i)$, où $a_i = 1$ si G_{h_i} n'est pas connexe et $N_{G_h^*}(v_i)$ est un séparateur minimal de G_h^*, $a_i = 0$ sinon.*

Démonstration. Si h est étiqueté `parallèle`, alors $\emptyset$ est un séparateur minimal, et les autres séparateurs minimaux sont des séparateurs minimaux des G_{h_i}. Les graphes G_{h_i} sont connexes, donc n'ont pas $\emptyset$ comme séparateur.

Soit h est étiqueté `série`. Si il existe un uv-séparateur minimal S, alors u ne doit pas être adjacent à v, et donc il existe $i \in \{1, \ldots, r\}$ tel que $u, v \in V_{h_i}$. Dans ce cas $V_{h_j} \subseteq S$ pour tout $j \neq i$, et $S \cap V_{h_i}$ est un uv-séparateur minimal dans G_{h_i}. Ainsi chaque séparateur minimal de G_h correspond à un séparateur minimal d'un G_{h_i}.

Enfin soit h étiqueté `premier`. Soit $u, v \in V_h$. Il y a deux cas à traiter.

Cas 1 : il existe un $i \in \{1, \ldots, r\}$ tel que $u, v \in V_{h_i}$. Un uv-séparateur minimal S devra forcément inclure tous les V_{h_j}, avec $\{v_i, v_j\} \in E(G_h^*)$ et pour tout $j \neq i$, $S \cap V_{h_j} \neq \emptyset$ si et seulement si $\{v_i, v_j\} \in E(G_h^*)$, car il est minimal. De plus, si u et v sont dans la même composante connexe de G_{h_i}, alors $S \cap V_{h_i}$ est un uv-séparateur minimal dans G_{h_i}.

Cas 2 : $u \in V_{h_i}$ et $v \in V_{h_j}$, avec $i \neq j$. Soit S un uv-séparateur minimal. Soit $S^* = \{v_k : V_{h_k} \subseteq S\}$. S^* est un $v_i v_j$-séparateur minimal dans G_h^*. De plus pour tout k tel que $v_k \notin S^*$, $S \cap V_{h_k} = \emptyset$, sinon il ne serait pas minimal.

Chaque séparateur minimal de G_h correspond donc à un séparateur minimal de G_h^*, ou à un séparateur minimal d'un G_{h_i}, $i \in \{1, \ldots, r\}$. Supposons que S soit un séparateur minimal pour un u et v dans le cas 1, et pour u' et v' dans le cas 1, tel que $u, v \in V_{h_i}$, et $u', v' \in V_{h_j}$, avec $i \neq j$. Alors on aurait $N_{G_h^*}(v_i) = N_{G_h^*}(v_j)$, et donc $\{v_i, v_j\}$ serait un module de G_h^*. Contradiction car G_h^* est premier.

Supposons que S soit un séparateur minimal pour un u et v dans le cas 1, et pour u' et v' dans le cas 2. Soit $i \in \{1, \ldots, r\}$ tel que $u, v \in V_{h_i}$. Alors $S \cap V_{h_i} = \emptyset$, sinon S ne serait pas du cas 2, et donc G_{h_i} ne serait pas connexe. De plus $S^* = N_{G_h^*}(v_i)$ est un séparateur minimal de G_h^*. Il y a donc exactement $\sum_{i=1}^{r} a_i$ séparateurs qui sont du cas 1 et 2. □

L'article [NP06] donne un algorithme calculant le nombre de séparateurs minimaux utilisant la décomposition modulaire, dans le cas particulier des graphes P_4-clairsemés.

2.4 Algorithmes sur les graphes sans P_5 et gem induit

Brandstädt *et al.* montrent que la classe des graphes sans P_5 et gem induit a une largeur de clique bornée par 5 [BLM04]. Ainsi beaucoup de problèmes peuvent se résoudre en temps linéaire sur cette classe de graphes, si la k-expression est donnée avec le graphe. C'est le cas en particulier pour tous les problèmes $LinEMSOL(\tau_1)$ [CMR00] (voir chapitre 5). Les problèmes ENSEMBLE STABLE PONDÉRÉ et CLIQUE PONDÉRÉE en font partie. Le meilleur temps pour trouver cette k-expression est $O(n^2)$, ce qui nous donne un temps $O(n^2)$ pour ces problèmes.

Il existe également des algorithmes polynomiaux pour les problèmes COLORATION MINIMUM et PARTITION EN CLIQUES, mais pour une largeur de clique d'au plus 5, l'exposant r dans la complexité temporelle $O(n^r)$ de l'algorithme est supérieur à 2000.

Dans la suite, on va présenter des algorithmes linéaires pour les problèmes ENSEMBLE STABLE PONDÉRÉ, CLIQUE PONDÉRÉE, COLORATION MINIMUM et PARTITION EN CLIQUES.

La « classe 1 » désignera la classe des graphes co-bipartis couplés, la « classe 2 » celle des graphes spécifiques, et la « classe 3 » l'union des classes $\mathcal{G}_k$, pour $k \in \mathbb{N}$.

2.4.1 Reconnaissance

Le théorème 2.19 ne nous donne pas seulement une caractérisation des graphes premiers apparaissant dans la décomposition modulaire d'un graphe sans P_5 et gem induit, mais présente également une caractérisation des graphes sans P_5 et gem induit, via la structure de leur arbre de décomposition modulaire. Ce théorème peut donc nous servir à reconnaître un graphe sans P_5 et gem induit.

Lemme 2.36. *La classe des graphes co-bipartis couplés premiers peut être reconnue en temps linéaire.*

Démonstration. Si G a au plus 3 sommets, ou si il a moins de $n^2/4$ arêtes, alors G ne peut pas être un graphe co-bipartis couplé premier. Si G a 4 sommets, alors on teste s'il est un P_4 (le C_4 n'est pas premier).

Sinon, on partitionne le graphe en deux cliques K_1 et K_2, en appliquant l'algorithme linéaire de reconnaissance des graphes bipartis sur $\overline{G}$ de [Gol80]. Finalement on vérifie que $|K_1| - |K_2| \in \{-1, 0, 1\}$, qu'un sommet dans une clique ait au plus un voisin dans l'autre clique, et qu'il y ait au plus un sommet par clique qui n'ait pas de voisin dans l'autre clique.

Cet algorithme est linéaire car un graphe passant le premier test a $\Omega(n^2)$ arêtes, et il est correct car si le graphe est co-biparti couplé, alors la partition K_1, K_2 est unique, puisque toute clique chevauchant K_1 est de taille au plus 2. □

Comme les graphes spécifiques ont au plus 9 sommets, l'appartenance peut être vérifiée en temps $O(1)$.

Si on a un graphe sans P_5 et gem induit, on sait que chaque graphe caractéristique d'un noeud premier dans l'arbre de décomposition modulaire est dans une des trois classes du théorème 2.19. Grâce aux observations précédentes, on peut savoir, en temps linéaire, à quelle classe appartient chaque graphe caractéristique.

Dans le cas de la reconnaissance, il faut également tester dans le dernier cas si le graphe appartient bien à la classe 3. Ceci peut se faire en temps $O(n^2)$

Théorème 2.37. *[BBK03] Il existe un algorithme en temps $O(n^2)$ qui reconnaît les graphes chordaux sans co-gem induit.*

Tous les C_5 d'un graphe de la classe 3 peuvent être exhibés en temps linéaire (théorème 2.40). Avec l'algorithme du théorème 2.37, on obtient :

Lemme 2.38. *[BBK03] Il existe un algorithme en temps $O(n^2)$ qui reconnaît les graphes de la classe 3.*

Au final, en utilisant le théorème de structure des graphes sans P_5 et gem induit, on obtient :

Théorème 2.39. *[BBK03] Les graphes sans P_5 et gem induit peuvent être reconnus en temps $O(n^2)$.*

2.4.2 Algorithme linéaire pour ENSEMBLE STABLE PONDÉRÉ

Le lemme 2.26 nous dit que pour résoudre le problème ENSEMBLE STABLE PONDÉRÉ sur les graphes sans P_5 et gem induit, il suffit de savoir résoudre le problème sur chaque graphe caractéristique des noeuds premiers dans l'arbre de décomposition modulaire. Pour cela, on commence par utiliser l'algorithme de la section 2.4.1 pour savoir, en temps linéaire, à quelle classe de graphes appartient le graphe premier.

2.4.2.1 Classe 1 : Graphes co-bipartis couplés

Soit $\{K_1, K_2\}$ une partition de V en deux cliques, obtenue par l'algorithme linéaire du lemme 2.36. Soient a_1 et a_2 deux sommets de poids maximum dans K_1, et soient b_1 et b_2 deux sommets de poids maximum dans K_2.

Comme chaque sommet a au plus un voisin dans l'autre clique, on a :

$$\alpha'_w(G) = \max\left\{w(a_i) + w(b_i) : i, j \in \{1,2\} \text{ et } \{a_i, b_i\} \notin E\right\}.$$

2.4.2.2 Classe 2 : Graphes spécifiques

Les graphes spécifiques ont au plus 9 sommets, donc le poids maximum d'un ensemble stable peut être trouvé en temps $O(1)$, en essayant tous les ensembles stables du graphe.

2.4.2.3 Classe 3

Théorème 2.40. *Il existe un algorithme linéaire qui exhibe tous les C_5 d'un graphe de la classe 3.*

Démonstration. Premièrement, on calcule l'ensemble V' des sommets de degré 2 de G. Pour tout sommet w avec exactement deux voisins u et v dans V', on teste si $N[u] \cup N[v]$ induit un C_5.

Par définition de la classe 3, tous les C_5 sont disjoints deux à deux. Les sommets adjacents à un C_5 sont adjacents à exactement 3 sommets du C_5, donc ont un degré d'au moins 3. Tous les C_5 peuvent donc être trouvés par cette procédure. De plus on parcourt au plus 3 fois la liste l'adjacence de chaque sommet, donc l'algorithme est linéaire. □

Théorème 2.41. *Le problème* ENSEMBLE STABLE PONDÉRÉ *peut être résolu en temps linéaire sur les graphes de la classe 3.*

Démonstration. Pour commencer, on calcule en temps linéaire tous les sommets de degré 2 et tous les C_5 de G en utilisant l'algorithme du théorème 2.40.

Si G est co-chordal, alors S est un ensemble stable de poids maximum si et seulement si S est une clique de poids maximum de $(\overline{G}, w)$. Un ordre d'élimination parfait $v_1, \ldots, v_n$ de $\overline{G}$ peut être calculé en temps $O(n+m)$ [MS99]. Pour une clique maximum S du graphe $\overline{G}$, $S = N^i_{\overline{G}}[v_i]$ pour un $i \in \{1, \ldots, n\}$, avec $N^i_{\overline{G}}[v_i] = N_{\overline{G}}[v_i] \cap \{v_i, v_{i+1}, \ldots v_n\}$. $w(S) = W^i - \sum_{u \in N^i_{\overline{G}}(v_i)} w(u)$, avec $W^i = \sum_{j=i}^{n} w(v_j)$. Ainsi $\alpha_w(G) = \omega_w(\overline{G})$ peut être calculé en temps linéaire.

Si G est un C_5, alors le poids maximum d'un ensemble stable peut être calculé en temps constant. Sinon, pour tout C_5 $C = (v_1, v_2, v_3, v_4, v_5)$ de G, il y a deux sommets v_1 et v_3 non adjacents de degré 2, et trois sommets v_2, v_4, v_5 de degré au moins 3. Soit S un ensemble stable de G. Si S contient un sommet de degré au moins 3 d'un C_5, alors ce n'est le cas que pour un C_5, disons C. Tout ensemble stable maximal de ce type contient l'ensemble stable A des sommets non adjacents à C, et deux sommets non adjacents de C. Il y a au plus $O(n)$ ensembles stables de ce type, et leurs poids peut être calculé en temps total $O(n)$, puisque $w(S) = w(A) + w(x) + w(y)$, où x et y sont deux sommets non adjacents du C_5.

Si S ne contient aucun sommet de degré 2 d'un C_5 de G, alors S est un ensemble stable du graphe co-chordal $G' = G - V'$, où V' est l'ensemble des sommets de degré 2 des C_5 de G. On a déjà vu que leur poids maximum pouvait être calculé en temps linéaire. □

Corollaire 2.42. *Il existe un algorithme linéaire pour le problème* ENSEMBLE STABLE PONDÉRÉ *sur les graphes sans P_5 et gem induit.*

2.4.3 Algorithme linéaire pour CLIQUE PONDÉRÉE

On veut donc résoudre CLIQUE PONDÉRÉE sur les graphes caractéristiques des noeuds premiers.

2.4.3.1 Classe 1 : Graphes co-bipartis couplés

Soit $G = (V, E)$ un graphe co-biparti couplé, et K_1, K_2 une partition de V en deux cliques. Chaque clique maximale par inclusion de G est soit K_1, soit K_2, soit une arête du graphe. Ainsi, $\omega_w(G)$ peut être calculé en temps linéaire.

2.4.3.2 Classe 2 : Graphes spécifiques

Les graphes spécifiques ont au plus 9 sommets, donc le poids maximum d'une clique peut être trouvé en temps $O(1)$, en essayant toutes les cliques du graphe.

2.4.3.3 Classe 3

Pour commencer on considère le problème sur les graphes co-chordaux.

Lemme 2.43. *Le problème* CLIQUE PONDÉRÉE *peut être résolu en temps linéaire sur les graphes co-chordaux.*

Démonstration. L'algorithme linéaire de Frank [Fra76] pour calculer le poids maximum d'un ensemble stable d'un graphe chordal G peut être utilisé pour calculer le poids maximum d'une clique sur un graphe co-chordal $\overline{G}$ en temps $O(n^2)$. On le modifie pour pouvoir calculer $\omega_w(G)$ en temps linéaire.

On rappelle l'algorithme de Frank. Il commence par calculer un ordre d'élimination parfait $v_1, \dots v_n$ du graphe d'entrée $G = (V, E)$, avec $w(v_i) \geq 0$ le poids de v_i pour $i \in \{1, \dots n\}$.

Un ensemble stable de poids maximum est construit de la façon suivante. Initialement, soit $c\_w(v_i) = w(v_i)$, pour tout $i \in \{1, \dots, n\}$. Pour tout i de 1 à n, si $c\_w(v_i) > 0$, alors on colorie v_i en rouge, et on soustrait $c\_w(v_i)$ à $c\_w(v_j)$ pour tout $v_j \in N[v_i] \cap \{v_i, \dots v_n\}$. Quand tous les sommets ont été traités, soit $I = \emptyset$ et, pour tout i de n à 1, si v_i est rouge et non adjacent à aucun sommet de I, alors $I = I \cup \{v_i\}$. Quand tous les sommets ont été traités, l'algorithme termine en renvoyant un ensemble de stable de poids maximum I de (G, w).

On va simuler l'algorithme de Frank sur $\overline{G}$ en temps linéaire par rapport à la taille de G, où l'entrée G est un graphe co-chordal. Pour commencer, l'algorithme calcule un ordre d'élimination parfait $v_1, \dots v_n$ de $\overline{G}$ en temps linéaire (voir [MS99]). Soit $w(v_i)$ le poids du sommet v_i, pour $i \in \{1, \dots, n\}$.

Le poids maximum d'une clique de G est construit de la façon suivante. Soit $W' = 0$ et $s(v_i) = 0$ pour tout $i \in \{1, \ldots, n\}$. Pour tout i de 1 à n, si $w(v_i) - W' + s(v_i) > 0$, alors on colorie v_i en rouge, et affecte à $W = w(v_i) + s(v_i)$, et on ajoute $w(v_i) - W' + s(v_i)$ à $s(v_j)$ pour tout $v_j \in N(v_j) \cap \{v_{i+1}, \ldots, v_n\}$.

Quand tous les sommets ont été traités, soit $K = \emptyset$, et pour chaque i de n à 1, si v_i est rouge et adjacent à tous les sommets de K, alors $K = K \cup \{v_i\}$. Au final, l'algorithme renvoie la clique de poids maximum K de (G, w).

On voit immédiatement que l'algorithme s'exécute en temps linéaire. Sa correction vient du fait que lorsque que l'on traite le sommet v_i, la différence $W' - s(v_i)$ est précisément la valeur que l'algorithme original de Frank appliqué à $\overline{G}$ aurait soustrait à $c\_w(v_i)$ du commencement jusqu'au traitement de v_i. Ainsi, notre algorithme simule l'algorithme de Frank sur $\overline{G}$ et est correct. □

Théorème 2.44. *Le problème* CLIQUE PONDÉRÉE *peut être résolu en temps linéaire sur les graphes de la classe 3.*

Démonstration. Pour commencer, on calcule en temps linéaire tous les sommets de degré 2 et tous les C_5 de G en utilisant l'algorithme du théorème 2.40.

Si G est co-chordal, alors on applique l'algorithme du lemme 2.43 pour avoir le résultat en temps linéaire. Si G est un C_5, alors le poids maximum d'une clique peut être calculé en temps constant.

Sinon, soit u un sommet de degré 2 d'un C_5. Il y a deux cliques maximales par inclusion de G qui contiennent u, correspondant aux deux arêtes incidentes à u. Le poids maximum de toutes ces cliques peut être calculé en temps $O(m)$. Toutes les autres cliques maximales de G sont des cliques du graphe co-chordal obtenues après avoir supprimé tous les sommets de degré 2 des C_5 de G. Leur poids maximum peut être calculé en temps linéaire avec l'algorithme du lemme 2.43. □

Corollaire 2.45. *Il existe un algorithme linéaire pour le problème* CLIQUE PONDÉRÉE *sur les graphes sans P_5 et gem induit.*

2.4.4 Algorithme linéaire pour COLORATION MINIMUM

On veut résoudre COLORATION MINIMUM PONDÉRÉE sur les graphes caractéristiques des noeuds premiers.

2.4.4.1 Classe 1 : Graphes co-bipartis couplés

Cette classe est une classe de graphes parfaits, car co-bipartis.

Lemme 2.46. *[GLS84] Soit G un graphe parfait et w une fonction de poids sur les arêtes de G. Alors $\chi_w(G) = \omega_w(G)$ et $\kappa_w(G) = \alpha_w(G)$.*

Leur nombre chromatique pondéré est donc égal au poids maximum d'une clique. On a déjà vu en section 2.4.3 qu'il pouvait être calculé en temps linéaire sur les graphes co-bipartis couplés.

2.4.4.2 Classe 2 : Graphes spécifiques

Paradoxalement, pour cette classe de graphes le résultat n'est pas immédiat. On montre que le problème peut être résolu en temps constant pour chaque graphe de taille $O(1)$, si on assume que les opérations sur les nombres entiers se font en temps $O(1)$. Pour cela, on se ramène à un problème de programmation linéaire en nombres entiers.

On fixe un graphe $G = (V, E)$, et $w : V \to \mathbb{N}$ une fonction de poids.

Soit $\mathcal{I}$ l'ensemble des stables maximaux de G. On construit le problème de programmation linéaire en nombres entiers suivant.

$$\begin{aligned} \text{minimiser} \quad & \textstyle\sum_{I \in \mathcal{I}} x_I && && (2.1)\\ \text{tel que} \quad & && && \\ & \textstyle\sum_{I \in \mathcal{I}: v \in I} x_I \geq w(v) && \text{pour tout } v \in V && (2.2)\\ & x_I \in \mathbb{N} && \text{pour tout } I \in \mathcal{I}. && (2.3) \end{aligned}$$

On désigne par $\overrightarrow{x}$ le vecteur contenant pour chaque $I \in \mathcal{I}$ la valeur x_I.

Soit z la valeur optimale de ce problème de programmation linéaire entière. Alors z est égal au nombre de couleurs nécessaires pour colorier (G, w). Si on a une coloration $\mathcal{C}'$ de (G, w), alors construit la coloration $\mathcal{C}$ en remplaçant chaque stable I' de $\mathcal{C}$ par un stable maximum I contenant I'. $\mathcal{C}$ est également une coloration de (G, w), et a le même nombre de couleurs. Soit x_I le nombre de couleurs assignées à I. Alors x_I est un entier non négatif. Pour chaque $v \in V$, comme v a au moins $w(v)$ couleurs, $\sum_{I \in \mathcal{I}: v \in I} x_I \geq w(v)$. $\sum_{I \in \mathcal{I}} x_I$ est égal au nombre de couleurs de $\mathcal{C}$. Inversement, soit $\overrightarrow{x}$ une solution optimale du programme linéaire en nombres entiers. Pour chaque $I \in \mathcal{I}$, on peut prendre un ensemble de x_I couleurs pour colorier les sommets de I. Comme I est un stable, et que pour chaque sommet v, $\sum_{I \in \mathcal{I}: v \in I} x_I \geq w(v)$, cela nous donne une coloration de (G, w), avec z couleurs.

La relaxation du problème, après suppression de la condition 2.3, est le problème de programmation linéaire suivant :

$$\begin{aligned} \text{minimiser} \quad & \textstyle\sum_{I \in \mathcal{I}} x_I && && (2.4)\\ \text{tel que} \quad & && && \\ & \textstyle\sum_{I \in \mathcal{I}: v \in I} x_I \geq w(v) && \text{pour tout } v \in V && (2.5) \end{aligned}$$

Soit $\overrightarrow{x'}$ une solution optimale de cette relaxation, et soit $z' = \sum_{i \in \mathcal{I}} x'_I$.

Notons que puisque le graphe G est fixé, ce problème de programmation linéaire possède un nombre constant de variables (le nombre de stables maximaux de G), et un nombre constant de contraintes (le nombre de sommets de G). Il peut donc être résolu en temps constant (par exemple en énumérant et essayant tous les sommets du polyèdre définit par le programme). On peut écrire le programme linéaire sous la forme $\max\{cx : Ax \leq b\}$, tel que chaque élément de la matrice A est soit 0, soit 1. Soit Δ la valeur maximum des

déterminants des sous-matrices de la matrice A. Δ est bornée par une constante[6]. Soit $k = |\mathcal{I}|$.

On peut maintenant utiliser un résultat de Cook, Gerards, Schrijver et Tardos, le théorème 17.2 dans [Sch86]. Ce théorème nous dit que le problème de programmation linéaire en nombres entiers a une solution optimale $\overrightarrow{x''}$, telle que pour tout $I \in \mathcal{I}$, $|x'_I - x''_I| \leq q\Delta$.

Premièrement, on trouve une solution optimale $\overrightarrow{x'}$ du programme linéaire relaxé. Ensuite on énumère tous les vecteurs d'entiers $\overrightarrow{x''}$ tels que pour tout $I \in \mathcal{I}$, $|x'_I - x''_I| \leq q\Delta$. Parmi les vecteurs qui vérifient les conditions 2.2, on choisit le vecteur solution minimum. Par le théorème 17.2 de [Sch86], c'est une solution optimale du programme linéaire en nombres entiers. Cette méthode prend un temps constant, car q et Δ sont fixés si le graphe est fixé, donc le nombre de vecteurs à tester est constant.

Une implémentation telle quelle de cette procédure ne sera pas pratique, puisqu'il y a de l'ordre de $(q\Delta)^q$ vecteurs à tester. En pratique, on peut par exemple résoudre le programme linéaire, puis se servir de la solution comme point de départ d'une procédure « branch and bound ».

Comme il y a un nombre constant de graphes de taille bornée, on a donc un algorithme en temps $O(1)$ pour trouver le nombre chromatique pondéré d'un graphe de taille bornée. C'est aussi le cas des graphes spécifiques, qui ont au plus 9 sommets.

On a un algorithme en temps $O(1)$ pour calculer le nombre chromatique pondéré des graphes de taille borné. Ainsi, on peut résoudre COLORATION MINIMUM en temps linéaire sur les graphes pour lesquels les graphes premiers apparaissant dans la décomposition modulaire ont une taille bornée. Ceci améliore une remarque de McDiarmid et Reed [MR00], qui observaient que le problème COLORATION PONDÉRÉE MINIMUM pouvait être résolu en temps polynomial sur les graphes de taille constante.

2.4.4.3 Classe 3

Soit G un graphe de la classe 3, avec une fonction de pondération w. Premièrement on calcule en temps linéaire tous les sommets de degré 2 et tous les C_5 de G, en utilisant l'algorithme du théorème 2.40.

Si G est sans C_5 induit, G est donc un graphe co-chordal, et $\chi_w(G) = \omega_w(G)$ peut être calculé en temps linéaire par l'algorithme présenté au lemme 2.43.

Sinon, $G \in \mathcal{G}_k$, avec $k \geq 1$. Si G est un C_5, alors la technique utilisée dans le cas des graphes spécifiques, présentée dans la section précédente, peut être utilisée pour calculer la coloration pondérée minimum en temps constant.

Finalement, soit $C = (v_1, v_2, v_3, v_4, v_5)$ un C_5 de G, et soit v_1 et v_3 ses sommets de degré 2. Par définition des graphes de la classe 3, l'ensemble A des sommets non adjacents à C est un ensemble stable, $B = V \setminus (C \cup A) = N(v_2) \setminus C = N(v_4) \setminus C = N(v_5) \setminus C$, et $G[B]$ est un cographe. En conséquence il y a précisément 4 ensembles stables maximaux de G contenant au moins un des trois sommets v_2, v_4 ou v_5 : il s'agit de $\{v_1, v_4\} \cup A$,

[6]Si G est un graphe spécifique, $\Delta \leq 3$.

$\{v_2, v_4\} \cup A$, $\{v_2, v_5\} \cup A$ et $\{v_3, v_5\} \cup A$. Tous les autres ensembles stables maximaux incluent l'ensemble stable $\{v_1, v_3\}$ de C. Plus précisément, $\{v_1, v_3\} \cup A'$ est un ensemble stable de G si et seulement si A' est un ensemble stable maximal de $G - C$. (Notons que $\{v_1, v_3\} \cup A$ est un ensemble stable de G.)

Lemme 2.47. *Soit $k \geq 1$, et $G \in \mathcal{G}_k$. Soit $C = (v_1, v_2, v_3, v_4, v_5)$ un C_5 de G tel que v_1 et v_3 soient de degré 2. Soit w une fonction de poids sur les sommets de G. Alors il existe une coloration pondérée minimum $\mathcal{S}$ de (G, w) avec précisément $\max\big(w(v_2), w(v_4) + w(v_5)\big)$ ensembles stables contenant au moins un sommet de $\{v_2, v_4, v_5\}$.*

Démonstration. Soit $\mathcal{S}$ une coloration pondérée minimum de (G, w). Soit $C = (v_1, v_2, v_3, v_4, v_5)$ un C_5 de G tel que, si $G \neq C_5$, v_1 et v_3 soient les sommets de degré 2. Puisque $N(v_1) \setminus C = N(v_3) \setminus C = \emptyset$ et $N(v_2) \setminus C = N(v_4) \setminus C = N(v_5) \setminus C = B$, on suppose que tous les ensembles stables de $\mathcal{S}$ contiennent soit aucun, soit deux sommets de C.

En conséquence on étudie les colorations pondérées du C_5 $C = (v_1, v_2, v_3, v_4, v_5)$ de G avec comme poids pour les sommets w, où tous les ensembles stables sont des non-arêtes de C. On les appellera des coloration pondérées partielles de C. Notons que toutes les colorations pondérées de $C = (v_1, v_2, v_3, v_4, v_5)$ doivent posséder au moins $w(v_2)$ ensembles stables contenant v_2, et au moins $w(v_4) + w(v_5)$ ensembles stables contenant v_4 ou v_5.

Soit $\mathcal{S}'$ une coloration pondérée de G' contenant le nombre minimum possible d'ensembles stables S avec $S \cap \{v_2, v_4, v_5\} \neq \emptyset$. Notons q le nombre d'ensembles stables de S de $\mathcal{S}'$ avec $S \cap \{v_2, v_4, v_5\} \neq \emptyset$, et supposons que contrairement à l'affirmation du lemme, $q > \max\big(w(v_2), w(v_4) + w(v_5)\big)$. Soit $c(v)$ le nombre d'ensembles stables de $\mathcal{S}'$ contenant le sommet v. Alors, $q > w(v_4) + w(v_5)$ implique $c(v_4) > w(v_4)$ ou $c(v_5) > w(v_5)$. Sans perte de généralité, on suppose $c(v_4) > w(v_4)$. En conséquence il existe un ensemble stable $S' \in \mathcal{S}'$ contenant v_4, et on a soit $S \subseteq \{v_2, v_4\} \cup A$, soit $S \subseteq \{v_2, v_3\} \cup A$. Dans les deux cas, on remplace S' par $\{v_1, v_3\} \cup A$. Cela diminue de 1 le nombre d'ensembles stables contenant v_4, et éventuellement de 1 le nombre d'ensembles stables contenant v_2. Ainsi nous obtenons une coloration pondérée $\mathcal{S}''$ de G avec $q-1$ ensemble stables S vérifiant $S \cap \{v_2, v_4, v_5\} \neq \emptyset$. Ceci contredit le choix de $\mathcal{S}'$.

Par conséquent, $q = \max\big(w(v_2), w(v_4) + w(v_5)\big)$. □

Corollaire 2.48. *Soit G un C_5 $(v_1, v_2, v_3, v_4, v_5)$, et w une fonction de poids sur les sommets. Alors il existe une coloration pondérée minimum de (G, w) avec précisément $\max\big(w(v_2), w(v_4) + w(v_5)\big)$ ensembles stables contenant au moins un sommet de $\{v_2, v_4, v_5\}$.*

Pour étendre une coloration pondérée partielle d'un C_5 à une coloration pondérée de G, seulement deux paramètres sont importants :

- le nombre de copies de $\{v_1, v_3\}$ dans la coloration pondérée partielle, dénoté par s, et
- le nombre d'ensembles stables du C_5 différents de $\{v_1, v_3\}$ de la coloration partielle, dénoté par t.

Pour chacun des s ensembles stables $\{v_1, v_3\}$ du C_5, chaque $\{v_1, v_3\} \cup A'$, où A' est un ensemble stable maximal de $G - C$, est un ensemble maximal de G qui étend $\{v_1, v_3\}$. Pour chacun des t ensembles stables S différents de $\{v_1, v_3\}$ du C_5, $S \cup A$ est le seul ensemble stable maximal de G étendant S.

Par le lemme 2.47, pour chaque C_5 de G il y a une coloration pondérée minimum de G avec $t = \max\big(w(v_2), w(v_4) + w(v_5)\big)$ ensembles stables contenant au moins un des sommets de $\{v_2, v_3, v_4\}$. En prenant une de ces coloration pondérée minimum, on peut enlever les sommets v_1 et v_3 des ensembles stables contenant les deux sommets, jusqu'à obtenir la plus petite valeur possible pour s dans une coloration partielle de C avec $t = \max\big(w(v_2), w(v_4) + w(v_5)\big)$. Par le corollaire 2.48, il existe une coloration pondérée partielle avec $\chi(G[C])$ ensembles stables, tel que $\max\big(w(v_2), w(v_4) + w(v_5)\big)$ ensembles stables contiennent au moins un sommet de $\{v_2, v_3, v_4\}$, et ainsi $s = \chi(G[C]) - t$ peut être calculée en temps constant.

On présente maintenant l'algorithme qui calcule une coloration pondérée minimum de (G, w) pour un graphe G de $\mathcal{G}_k$, $k \geq 1$. Il enlève au plus k fois un C_5 précalculé au graphe courant, jusqu'à ce que le graphe restant n'ait plus de C_5, et donc soit co-chordal. Alors, une coloration minimum pondérée du graphe co-chordal peut être obtenue en temps linéaire en utilisant l'algorithme pour la clique pondérée maximum présenté au lemme 2.43.

A chaque étape, c'est à dire quand on supprime un C_5 $C = (v_1, v_2, v_3, v_4, v_5)$ du graphe courant G' avec la fonction de pondération w', l'algorithme procède de la façon suivante : il calcule, en temps constant, une coloration pondérée minimum partielle de C telle que $t = \max\big(w(v_2), w(v_4) + w(v_5)\big)$, et telle que s soit le plus petit que possible. Alors il enlève tous les sommets de C, et il enlève tous les sommets de A avec un poids inférieur ou égal à t, et soustrait t aux poids de tous les autres sommets de A. On note G'' le nouveau graphe obtenu, et w'' la nouvelle fonction de pondération. Alors l'algorithme résout récursivement le problème coloration minimum sur (G'', w''). Enfin, le nombre minimum d'ensembles stables dans une coloration de (G', w') est obtenu par la formule suivante

$$\chi_{w'}(G') = t + \max\Big(s, \chi_{w''}(G'')\Big).$$

L'algorithme enlève au plus $k \leq n$ fois un C_5. Chaque coloration pondérée partielle d'un C_5 peut être calculée en temps constant. Pour le graphe co-chordal final, la coloration pondérée minimum peut être calculée en temps linéaire. Ainsi, le temps total cet l'algorithme pour la classe 3 est linéaire. En résumé, on obtient :

Corollaire 2.49. *Il existe un algorithme linéaire pour le problème* COLORATION PONDÉRÉ MINIMUM *sur les graphes sans P_5 et gem induit.*

2.4.5 Algorithme linéaire pour PARTITION EN CLIQUES

On veut résoudre PARTITION EN CLIQUES PONDÉRÉE sur les graphes caractéristiques des noeuds premiers.

2.4.5.1 Classe 1 : Graphes co-bipartis couplés

Cette classe est une classe de graphes parfaits, car co-bipartis.

Par le lemme 2.46, $\kappa_w(G) = \alpha_w(G)$. En utilisant l'algorithme linéaire pour ENSEMBLE STABLE PONDÉRÉ pour la classe des graphes co-bipartis couplé, présenté en section 2.4.2.1, $\kappa_w(G)$ peut donc être calculé en temps linéaire.

2.4.5.2 Classe 2 : Graphes spécifiques

La technique présentée dans la section précédente pour calculer une coloration pondérée minimum d'un graphe fixé peut être appliquée sur le complémentaire pour calculer une partition en cliques minimum en temps constant.

2.4.5.3 Classe 3

Soit G un graphe de la classe 3, avec une fonction de pondération w. Premièrement on calcule en temps linéaire l'ensemble des sommets de degré 2 et tous les C_5 de G, en utilisant l'algorithme du théorème 2.40.

Si G est sans C_5 induit, donc est un graphe co-chordal, $\kappa_w(G) = \alpha_w(G)$ peut être calculé en temps linéaire par l'algorithme présenté en section 2.4.2.3.

Sinon, $G \in \mathcal{G}_k$, avec $k \geq 1$. Si G est un C_5, alors la technique utilisée dans le cas de la coloration des graphes spécifiques, présentée dans la section 2.4.4.2, peut être utilisée pour calculer une partition minimum en cliques en temps constant.

Finalement, soit $C = (v_1, v_2, v_3, v_4, v_5)$ un C_5 de G, et soit v_1 et v_3 ses sommets de degré 2. Par définition des graphes de la classe 3, l'ensemble A des sommets non adjacents à C est un ensemble stable, $B = V \setminus (C \cup A) = N(v_2) \setminus C = N(v_4) \setminus C = N(v_5) \setminus C$, et $G[B]$ est un cographe. En conséquence il y a précisément 4 ensembles stables maximaux de G contenant au moins un des trois sommets v_2, v_4, v_5, soit $\{v_1, v_4\} \cup A$, $\{v_2, v_4\} \cup A$, $\{v_2, v_5\} \cup A$ et $\{v_3, v_5\} \cup A$. tous les autres ensembles stables maximaux incluent l'ensemble stable $\{v_1, v_3\}$ de C. Plus précisément, $\{v_1, v_3\} \cup A'$ est un ensemble stable de G si et seulement si A' est un ensemble stable maximal de $G - C$. (Notons que $\{v_1, v_3\} \cup A$ est un ensemble stable de G.)

Lemme 2.50. *Soit $k \geq 1$, et $G \in \mathcal{G}_k$. Soit $C = (v_1, v_2, v_3, v_4, v_5)$ un C_5 de G tel que v_1 et v_3 soient de degré 2. Soit w une fonction de poids sur les sommets de G. Alors il existe une partition en cliques pondérée minimum de (G, w) avec précisément $w(v_1) + w(v_3)$ cliques contenant au moins un sommet de $\{v_1, v_3\}$.*

Démonstration. Considérons une partition en cliques minimum $\mathcal{K}$ de G. Chaque clique de $\mathcal{K}$ contenant v_1 ou v_3 est un sous-ensemble de C.

Supposons que $\mathcal{K}$ contienne plus que $w(v_1) + w(v_3)$ cliques contenant v_1 ou v_3. Alors on peut obtenir une partition en cliques minimum $\mathcal{K}$ avec précisément $w(v_1) + w(v_3)$ cliques contenant v_1 ou v_3 en appliquant les opérations de remplacement suivantes. Soit $c(v)$ le nombre de cliques de $\mathcal{K}$ contenant v. On remplace $c(v_1) - w(v_1)$ cliques K de $\mathcal{K}$ contenant v_1 par $K \setminus \{v_1\}$, et de même on remplace $c(v_3) - w(v_3)$ cliques K de $\mathcal{K}$ contenant v_3 par $K \setminus \{v_3\}$. □

Corollaire 2.51. *Soit G un C_5 $(v_1, v_2, v_3, v_4, v_5)$, et w une fonction de poids sur les sommets. Alors il existe une partition en cliques pondérée minimum de (G, w) avec précisément $w(v_1) + w(v_3)$ cliques contenant au moins un des sommets $\{v_1, v_3\}$.*

On étudie les partitions en cliques pondérée d'un C_5 C avec la fonction de pondération w. On appelle une partition en cliques pondérée du C_5 une partition partielle en cliques de C. Pour étendre une partition partielle en cliques de C à G, seulement deux paramètres sont importants :

- le nombre de cliques de la partition partielle en cliques de C contenant soit v_1 soit v_3, dénoté par s, et
- le nombre de cliques ne contenant pas v_1 ou v_3, dénoté par t.

Pour étendre une partition partielle en cliques du C_5 $C = (v_1, v_2, v_3, v_4, v_5)$ à une partition en cliques pondérée minimum de G, on doit étendre les t cliques $\{v_4, v_5\}$, $\{v_4\}$, $\{v_5\}$ ou $\{v_2\}$ du C_5 en ajoutant à chacune de ces t cliques une clique B' de $G[B]$. On suppose que chaque sommet $a \in A$ apparaît dans exactement $w(a)$ cliques de $\mathcal{K}$ (on note qu'aucune de ces cliques contient un sommet de C).

L'algorithme calcule une partition en cliques minimum de (G, w) pour un graphe G de $\mathcal{G}_k$, $k \geq 1$, en supprimant k fois un C_5 du graphe courant, jusqu'à ce qu'il n'y ait plus aucun C_5 dans le graphe, et ainsi est un graphe co-chordal. Étant donné un C_5 C, par le corollaire 2.51, il existe une partition partielle en cliques de C avec $s = w(v_1) + w(v_3)$ cliques contenant v_1 ou v_3, et $t = \kappa_w(G[C]) - s$ cliques $\{v_4, v_5\}$, $\{v_4\}$, $\{v_5\}$ ou $\{v_2\}$.

L'algorithme résout le problème de partitionnement en cliques pondérée récursivement. Soit G' le graphe courant, avec w' la fonction de pondération. L'algorithme calcule $s = w'(v_1) + w'(v_3)$ et t. Récursivement, l'algorithme calcule $\kappa_{w''}(G' - C)$, où $w''(v) = w'(v)$ pour tous les sommets v de $G - C$. Alors le nombre minimum de cliques dans une partition en cliques de (G', w') est obtenu par la formule :

$$\kappa_{w'}(G') = s + \sum_{a \in A} w'(a) + \max\left(t, \kappa_{w''}(G' - C) - \sum_{a \in A} w'(a)\right),$$

où A est l'ensemble des sommets non adjacents à C dans G'. On note qu'il y a exactement $\sum_{a \in A} w'(a)$ cliques de la partition en cliques de $(G' - C, w'')$ qui contiennent un sommet de A, et donc ne peuvent pas être étendues en ajoutant une clique de C. Il y a $\kappa_{w''}(G' - C) - \sum_{a \in A} w'(a)$ cliques qui peuvent être étendues en ajoutant $\{v_4, v_5\}$, $\{v_4\}$, $\{v_5\}$ ou $\{v_2\}$.

Ainsi, l'algorithme supprime $k < n$ fois un C_5, et calcule la partition en cliques pondérée d'un graphe co-chordal, qui peut être fait en temps linéaire (corollaire 2.42). En conséquence, le temps total de l'algorithme est linéaire. En résumé, on obtient :

Corollaire 2.52. *Il existe un algorithme linéaire pour le problème* PARTITION EN CLIQUES MINIMUM *sur les graphes sans P_5 et gem induit.*

Chapitre 3

Décomposition en coupes

La décomposition en coupes peut être vue comme une généralisation de la décomposition modulaire. Elle a été introduite par Cunningham [Cun82], qui a prouvé son unicité, et qui a donné un algorithme polynomial pour la calculer.

3.1 Décomposition en coupes

3.1.1 Coupes d'un graphe

Une **coupe** dans un graphe est une partition de l'ensemble des sommets en deux ensembles V_1 et V_2 tel que tous les sommets de V_1 ayant au moins un voisin dans V_2 ont le même voisinage dans V_2. On peut remarquer que si X est un module, alors $\{X, V \setminus X\}$ est une coupe.

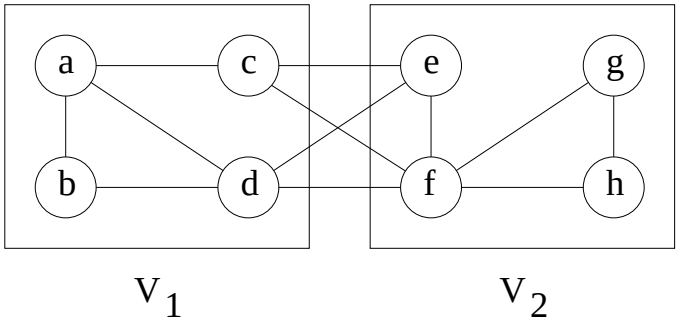


Fig. 3.1 : – Un graphe avec une coupe $\{V_1, V_2\}$.

Une coupe est **triviale** si $|V_1| = 1$ ou $|V_2| = 1$. Attention toutefois, un module non trivial peut être une coupe triviale.

Un graphe est **premier** par rapport à la décomposition en coupes si toutes ses coupes sont triviales.

Théorème 3.1. *[Cun82] La famille de toutes les coupes d'un graphe connexe est une famille bi-partitive.*

Il est important que le graphe soit connexe. Considérons par exemple le graphe non connexe formé par un chemin a, b, c, d et d'un sommet additionnel e de degré 0. Les bi-partitions $\big\{\{a, e\}, \{b, c, d\}\big\}$ et $\big\{\{c, e\}, \{a, b, d\}\big\}$ sont des coupes qui se chevauchent, mais la bi-partition $\big\{\{a, c, e\}, \{b, d\}\big\}$ n'est pas une coupe.

3.1.2 Cadre de décomposition

Initialement, Cunningham [Cun82] a défini la décomposition en coupes et a montré son unicité sans parler de familles bi-partitives. Edmonds et lui même avaient défini un cadre de décomposition avec certaines propriétés [CE80]. Ils avaient montré que toutes les décompositions respectant les propriétés des cadres de décompositions ont une unique décomposition. Cunningham a montré que les coupes respectaient ce cadre et la propriété d'intersection.

Soit $\mathcal{N}$ une classe et E une fonction définie sur $\mathcal{N}$ telles que, pour tout $N \in \mathcal{N}$, $E(N)$ est un ensemble fini, appelé l'ensemble des cellules de N. Soit $\rightarrow$ une relation associant les éléments N de $\mathcal{N}$ aux sous-ensembles de taille 2 $\{N_1, N_2\}$ de $\mathcal{N}$. Cette relation est notée $N \rightarrow \{N_1, N_2\}$.

Définition 3.1.1. [CE80] Le triplet $(\mathcal{N}, E, \rightarrow)$ est un **cadre de décomposition** si les conditions suivantes sont respectées :

(F1) Si $N \rightarrow \{N_1, N_2\}$, alors il existe $e \notin E(N)$ et une partition $\{E_1, E_2\}$ de $E(N)$ avec $|E_i| \geq 2$ pour $i \in \{1, 2\}$, tels que $E(N_i) = E_i \cup \{e\}$ pour $i \in \{1, 2\}$.
$\{N_1, N_2\}$ est appelée une **décomposition simple** de N, e le **marqueur** et $\{E_1, E_2\}$ la **section**[1] de N correspondant à la décomposition simple.

(F2) Pour une section $\{E_1, E_2\}$ de $N \in \mathcal{N}$ et $e \notin E(N)$, il y a exactement une décomposition simple $\{N_1, N_2\}$ de N avec le marqueur e correspondant à $\{E_1, E_2\}$.
Étant donné une section $\{E_1, E_2\}$ de N et $e \notin E(N)$, on note par $N(E_i; e)$, $i \in \{1, 2\}$, l'unique élément de $\mathcal{N}$ tel que $E(N(E_i; e)) = E_i \cup \{e\}$ et $N \rightarrow \{N(E_1; e), N(E_2; e)\}$.

(F3) Soit $\{E_1, E_2\}$ une section de $N \in \mathcal{N}$, soit $A \subseteq E_1$ et $E \notin E(N)$. Alors $\{A, E(N) \setminus A\}$ est une section de N si et seulement si $\{A, (E_1 \cup \{e\}) \setminus A\}$ est une section de $N(E_1, e)$.

(F4) Soient $\{E_1, E_2\}$, $\{E_3, E_4\}$ deux sections de $N \in \mathcal{N}$ telles que $E_3 \subseteq E_1$, et soit $e, f \in E(N)$, $e \neq f$. Alors

$$N(E_1; e)(N_3; f) = N(E_3; f)$$

et

$$N(E_1; e)(E_1 \setminus E_3 \cup \{e\}; f) = N(E_4; f)(E_4 \setminus E_2 \cup \{f\}; e).$$

On dit qu'une section est **forte** si elle ne chevauche aucune autre section de N. Cunnhingham et Edmonds montrent que si on a un cadre de décomposition, et un ensemble de

[1] Dans les articles [Cun82, CE80], les coupes et les sections sont appelées des **split**. Dans sa thèse, Lanlignel a choisi le mot **coupes** pour les *splits* de [Cun82]. Pour éviter les confusions, nous avons choisi **section** pour les splits des cadres de décomposition.

sections dont aucune section ne chevauche une autre, alors la décomposition est unique par cet ensemble de sections. Plus particulièrement, si l'ensemble de sections est l'ensemble des sections fortes, alors on a l'unicité de la décomposition. Mais, uniquement avec les propriétés F1–F4, nous n'en savons pas plus sur la structure des graphes obtenus à la fin de la décomposition.

Ils ont également introduit deux autres propriétés sur les cadres de décomposition. On dit qu'un cadre de décomposition a la **propriété d'intersection** si pour toutes sections $\{E_1, E_2\}$ et $\{E_3, E_4\}$ de $N \in \mathcal{N}$ telles que $|E_1 \cap E_3| \geq 2$ et $E_1 \cup E_3 \neq E(N)$, alors $\{E_1 \cap E_3, E_2 \cup E_4\}$ est une section de N. De plus, on dit qu'un cadre de décomposition a la **propriété de transitivité** si pour e_1, e_2 et e_3 tels que $\big\{\{e_1, e_2\}, E(N) \setminus \{e_1, e_2\}\big\}$ et $\big\{\{e_2, e_3\}, E(N) \setminus \{e_2, e_3\}\big\}$ sont des sections de $N \in \mathcal{N}$, alors $\big\{\{e_1, e_3\}, E(N) \setminus \{e_1, e_3\}\big\}$ est une section de N.

Lemme 3.2. *Soit $(\mathcal{N}, E, \rightarrow)$ un cadre de décomposition avec la propriété d'intersection, et soit $N \in \mathcal{N}$. Soit $\mathcal{F}'$ la famille des sections de N, et soit $\mathcal{F} = \mathcal{F}' \cup \mathcal{T}$, où $\mathcal{T}$ est l'ensemble des bi-partitions triviales de $E(N)$. Alors la famille $\mathcal{F}$ des sections de N est faiblement bi-partitive.*

De plus, si le cadre de décomposition possède la propriété de transitivité, alors $\mathcal{F}$ est bi-partitive.

Démonstration. Supposons que $\{E_1, E_2\}$ et $\{E_3, E_4\}$ soient 2 sections de N qui se chevauchent. Si $|E_1 \cap E_3| = 1$, alors $\{E_1 \cap E_3, E_2 \cup E_4\}$ est une bi-partition triviale de $E(N)$, et donc est dans $\mathcal{F}$. Sinon, comme le cadre de décomposition possède la propriété d'intersection, $\{E_1 \cap E_3, E_2 \cup E_4\}$ est une section de N, et donc est dans $\mathcal{F}$. De même, $\{E_1 \cap E_4, E_2 \cup E_3\}$, $\{E_2 \cap E_3, E_1 \cup E_4\}$ et $\{E_2 \cap E_4, E_1 \cup E_3\}$ sont des sections.

Supposons maintenant que le cadre possède également la propriété de transitivité. On sait, comme il possède la propriété d'intersection, que $\{E_1 \cap E_3, E(N) \setminus (E_1 \cap E_3)\}$, $\{E_1 \setminus E_3, E(N) \setminus (E_1 \setminus E_3)\}$ et $\{E_3 \setminus E_1, E(N) \setminus (E_3 \setminus E_1)\}$ sont des sections. On décompose successivement N par ces 3 sections, en utilisant respectivement les marqueurs a, b et c. Soit $N' \in \mathcal{N}$ issu de ces décompositions, possédant les marqueurs a, b et c. $\big\{\{a, b\}, E(N) \setminus \{a, b\}\big\}$ et $\big\{\{a, c\}, E(N) \setminus \{a, c\}\big\}$ sont des sections de N', car $\{E_1, E(N) \setminus E_1\}$ et $\{E_3, E(N) \setminus E_3\}$ sont des sections de N. Donc, par la propriété de transitivité, $\big\{\{b, c\}, E(N) \setminus \{b, c\}\big\}$ est une section de N', et par la propriété F3, $\{E_1 \Delta E_3, E(N) \setminus (E_1 \Delta E_3)\}$ est une section de N. □

En fait, seules les propriétés d'intersection et de transitivité sont importantes pour avoir une famille bi-partitive. Les propriétés F1–F4 sont nécessaires pour que l'approche récursive puisse toujours arriver à une même décomposition. Elles permettent aussi de trouver toutes les sections fortes de N en décomposant récursivement N par des décompositions simples. En effet, comme aucune section ne chevauche une section forte de N, la section correspondra à une section de N_1 ou N_2 après une décomposition simple de N en N_1 et N_2.

Cunnhingham et Edmonds donnent la structure des graphes obtenus à la fin de la décomposition. Il s'agit soit de graphes premiers (qui n'ont aucune section), soit de graphes **brittles** (toutes les bi-partitions sont des sections), ou de graphes **semi-brittles** (il existe

un ordonnancement $e_1, \dots e_n$ des éléments de $E(N)$ tel que les sections soient de la forme $\Big\{\{e_{i+1}, \dots e_j\}, \{e_{j+1}, e_i\}\Big\}$, où les indices sont modulo n). On retombe donc bien sur les noeuds `premiers`, `dégénérés` et `linéaires` dans l'arbre représentatif d'une famille faiblement bi-partitive.

Ils donnent également quelques exemples de cadres de décomposition, dont la décomposition en graphes 3-connexes des graphes 2-connexes. Plus tard, Cunningham donnera deux autres exemples : la décomposition en coupes [Cun82], et la décomposition de fonctions submodulaires [Cun83].

3.1.3 Décomposition en coupes simples

Soit $\{V_1, V_2\}$ une coupe non triviale d'un graphe $G = (V, E)$, et soit W_1 l'ensemble des sommets dans V_1 qui ont au moins un voisin dans V_2 et soit W_2 l'ensemble des sommets dans V_2 qui ont au moins un voisin dans V_1. La **décomposition simple** de G par la coupe $\{V_1, V_2\}$ est la décomposition en $G_1 = (V_1 \cup \{v\}, E_1)$ et $G_2 = (V_2 \cup \{v\}, E_2)$, avec $E_i = (E \cap \mathcal{P}_2(V_i)) \cup \{\{u, v\} : u \in W_i\}$, pour $i \in \{1, 2\}$.

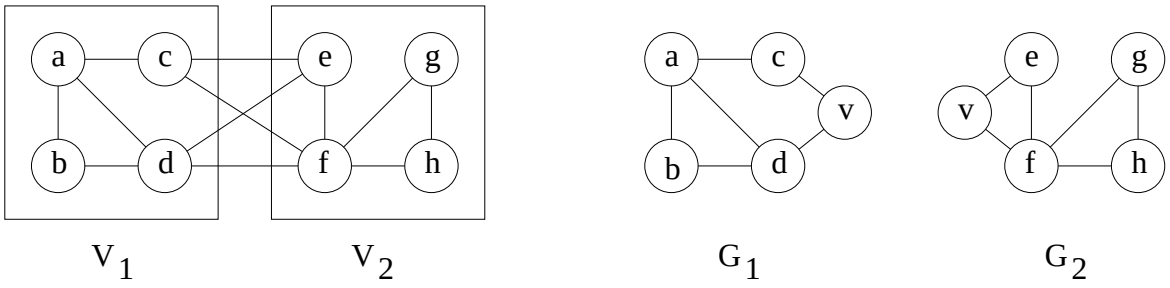


Fig. 3.2 : – Un graphe avec une coupe $\{V_1, V_2\}$, et sa décomposition en coupes simples par la coupe $\{V_1, V_2\}$.

Cunningham [Cun82] montre que la décomposition en coupes simples d'un graphe orienté fortement connexe est un cadre de décomposition, et possède la propriété d'intersection. Il peut en être déduit que l'ensemble des coupes d'un graphe orienté fortement connexe est une famille faiblement bi-partitive.

Théorème 3.3. *[Cun82] Le triplet $(\mathcal{G}, V, \rightarrow)$, où $\mathcal{G}$ est l'ensemble des graphes orientés fortement connectés, V la fonction donnant l'ensemble des sommets d'un graphe, et $\rightarrow$ la relation de la décomposition en coupes simples, est un cadre de décomposition avec la propriété d'intersection.*

L'opération inverse de la décomposition simple est la **composition simple**. Soit G_1 et G_2 deux graphes tels que $V(G_1) \cap V(G_2) = \{v\}$. Alors $G_1 * G_2$ représentera le graphe d'ensemble de sommets $(V(G_1) \cup V(G_2)) \backslash \{v\}$, et d'ensemble d'arêtes $(E(G_1) \cap \mathcal{P}_2(V(G_1))) \cup (E(G_2) \cap \mathcal{P}_2(V(G_2))) \cup \big\{\{u_1, u_2\} : u_1 \in N_{G_1}(v)$ et $u_2 \in N_{G_2}(v)\big\}$. On voit immédiatement que si G a une décomposition simple en G_1 et G_2, alors $G = G_1 * G_2$.

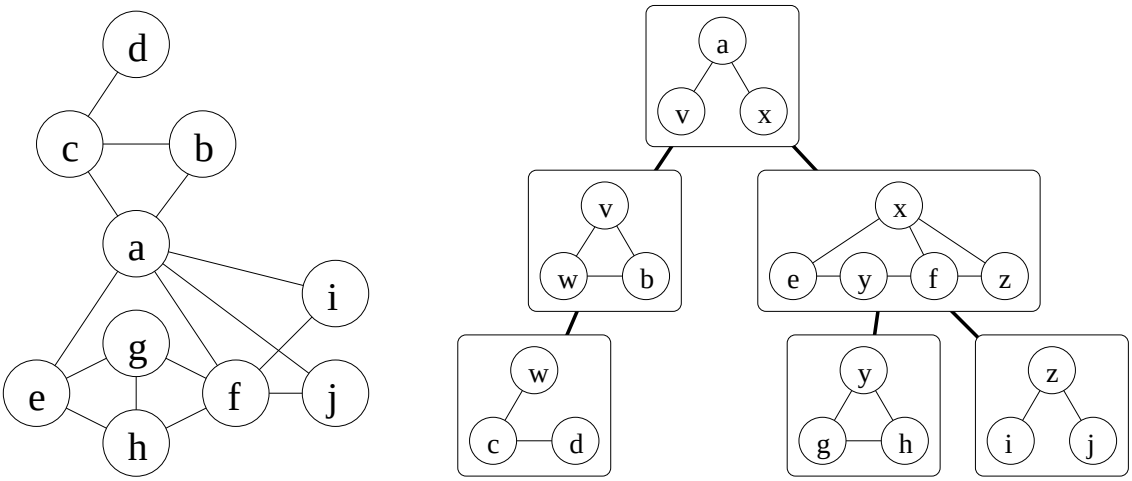


Fig. 3.3 : – Un graphe et son arbre de décomposition en coupes. Les marqueurs sont v, w, x, y et z.

3.1.4 Arbre de décomposition en coupes

L'application récursive de la décomposition simple sur un graphe G nous donne une décomposition par des coupes du graphe G. Mais cette décomposition n'est pas unique. Néanmoins, Cunningham et Edmonds montrent que si on applique la décomposition simple uniquement sur des coupes fortes, l'arbre de décomposition obtenu sera unique. Cunningham parle de **décomposition première**, non unique, pour les décompositions par coupes telles que chaque composante obtenues à la fin de la décomposition est un graphe premier, et de **décomposition standard**, unique, pour la décomposition par coupes fortes. Sans information complémentaire, la décomposition en coupes désignera la décomposition standard.

Pour chaque composante H à la fin de la décomposition, soit H est un graphe premier, soit toutes les bi-partitions de $V(H)$ sont des coupes de H. Dans le second cas, H est soit un graphe complet, soit une **étoile** (c'est-à-dire un graphe $K_{1,k}$, $k \in \mathbb{N}$).

Théorème 3.4. *[Cun82] Chaque graphe connexe a une décomposition unique, telle que chaque composante est un graphe premier, une étoile ou graphe complet.*

On peut obtenir l'arbre de décomposition par les coupes fortes en calculant une décomposition première, puis en recomposant les noeuds voisins de l'arbre qui ne correspondent pas à des coupes fortes ; il s'agit de l'algorithme donné par Cunningham. On peut facilement savoir quelles arêtes correspondent à une coupe non forte.

Lemme 3.5. *[Cun82] Soient G_1 et G_2 des graphes connexes, tels que $V(G_1) \cap V(G_2) = \{v\}$. Alors $G_1 * G_2$ est un graphe complet ou est une étoile si et seulement si une des conditions suivantes est vraie :*

- *G_1 et G_2 sont complets,*
- *G_1 et G_2 sont des étoiles, et v est le centre d'exactement une d'elles.*

3.1.5 Graphes distance héréditaire

Les graphes distance héréditaire sont pour la décomposition en coupes ce que les cographes sont pour la décomposition modulaire : la classe des graphes entièrement décomposables.

Un graphe est un graphe **distance héréditaire** si tous les chemins induits entre deux sommets du graphe ont la même longueur. Un **sommet pendant** d'un graphe est un sommet de degré 1. On dira qu'on ajoute un sommet pendant à un graphe si on ajoute un sommet adjacent à exactement un sommet du graphe.

Le théorème suivant donne d'autres caractérisations de la classe des graphes distance héréditaire :

Théorème 3.6. *[BM84, HM90] Les propositions suivantes sont équivalentes :*
(1) G est complètement décomposable par la décomposition en coupes.
(2) G est un graphe distance héréditaire.
(3) G est sans maison, trous, domino et gem induit[2]*.*
(4) G peut être obtenu d'un seul sommet par une séquence d'extensions de sommets : ajout d'un vrai jumeau, ajout d'un faux jumeau ou ajout d'un sommet pendant.

3.1.6 Algorithmes de décomposition

Le premier algorithme pour la décomposition en coupes a été donné par Cunningham [Cun82], avec un temps en $O(n^3)$. La complexité a été améliorée en $O(nm)$ [GHS89], puis en $O(n^2)$ dans [MS94]. Finalement Dahlhaus [Dah00] donne un algorithme linéaire.

Si on dispose d'un algorithme pour trouver une coupe non triviale d'un graphe, alors on peut s'en servir pour calculer la décomposition en coupes : il suffit de décomposer tant qu'il existe une coupe non triviale, puis de fusionner les coupes non fortes pour obtenir l'arbre unique de décomposition.

Cunnhingham donne un algorithme en $O(n^2)$ qui, pour $(x_1, y_1), (x_2, y_2) \in A$ recherche par raffinement une coupe $\{V_1, V_2\}$ du graphe orienté (V, A) telle que $x_1, y_2 \in V_1$ et $x_2, y_1 \in V_2$. En ne testant que les arêtes d'arbres couvrants, il arrive à un algorithme en $O(n^4)$ pour le calcul de la décomposition en coupes. Dans le cas des graphes non orientés, il arrive à $O(n^3)$ en fixant $x_2 = y_1$ et $y_2 = x_1$. Gabor, Hsu et Supowit [GHS89] arrivent ensuite au temps $O(nm)$ dans le cas des graphes non orientés, en améliorant à $O(m)$ le temps pour trouver une coupe telle que $x \in V_1$ et $y \in V_2$, pour une arête $\{x, y\}$ donnée.

Ma et Spinrad [MS94] obtiennent le temps $O(n^2)$ en utilisant récursivement une sous-routine qui recherche un partitionnement $F_1, \ldots, F_k$ de V tel que si $\{V_1, V_2\}$ est une coupe avec $a, b \in V_1$, alors il existe un $i > 1$ tel que $V_2 \in F_i$. Finalement Dahlhaus [Dah00] obtient le temps linéaire en réduisant le problème à celui de trouver les classes d'équivalences dans la fermeture transitive de la relation de chevauchement de sous-ensembles.

[2] Également appelé *HHDG-free.*

3.2 Classes de graphes et décomposition en coupes

Si l'on veut utiliser la décomposition en coupes comme base d'un algorithme, on doit se restreindre à une classe de graphes joliment décomposables par cette décomposition.

La première des classes joliment décomposables est bien sûr la classe des graphes distance héréditaire : tous les graphes caractéristiques sont soit des graphes complets, soit des étoiles.

On dit qu'une classe de graphes $\mathcal{G}$ est **stable** par la composition en coupes si pour tout $G, H \in \mathcal{G}$, alors $G * H \in \mathcal{G}$. Les graphes distance héréditaire sont bien sûr stables par la composition en coupes. Bixby [Bix84] montre que les graphes parfaits le sont aussi. Comme la composition ne peut pas créer de trous ou d'anti-trous, les graphes faiblement chordaux sont stables par la composition en coupes. Cela n'est pas le cas des graphes chordaux car la composition peut créer un C_4.

Dans quelques cas, on recherche plus qu'une classe stable par la composition en coupes. On veut exhiber une sous-classe $\mathcal{G}'$ de $\mathcal{G}$ telle que les graphes caractéristiques des décompositions des graphes de $\mathcal{G}$ soient dans $\mathcal{G}'$.

Graphes cercle

Un graphe est un **graphe cercle** s'il s'agit d'un graphe d'intersection[3] de cordes d'un cercle. Gabor, Hsu et Supowit [GHS89] montrent qu'un graphe est un graphe cercle si et seulement si chaque graphe caractéristique est un graphe cercle, et que les graphes cercle premiers ont une unique représentation par intersection de cordes d'un cercle (résultat également montré par Bouchet [Bou87]). Ils donnent également un algorithme en temps $O(nm)$ pour reconnaître les graphes cercle. Ce résultat sera amélioré à $O(n^2)$ par Spinrad [Spi94]. On peut faire un parallèle entre ce résultat utilisant la décomposition, et le résultat de décomposition des graphes de permutation utilisant la décomposition modulaire.

Graphes de parité

Un graphe est un **graphe de parité** si les longueurs des chemins induits entre 2 sommets ont toutes la même parité (soit paires, soit impaires). Pour $k \in \{5, 6 \ldots\}$, soit H_k^* le graphe d'ensemble de sommets $\{v_1, \ldots, v_k\}$ et d'ensemble d'arêtes $\big\{\{v_1, v_3\}\big\} \cup \big\{\{v_i, v_{i+1}\} : i \in \{1, \ldots, k\}\big\}$ (les indices sont modulo k). Burlet et Uhry [BU84] montrent qu'un graphe est un graphe de parité si et seulement si c'est un graphe sans trou impair, gem et H_{2k+1}^*, $k \geq 2$, induit.

Cicerone et Stephano [CS99] montrent que la classe des graphes de parité sont exactement les graphes tels que tous les graphes caractéristiques dans la décomposition en coupes sont soit des graphes complets, soit des graphes bipartis. Cette caractérisation permet de reconnaître les graphes de parité en temps linéaire [Dah00].

[3] Le graphe d'intersection d'une famille $\mathcal{E}$ de sous-ensembles est le graphe $\Big(\mathcal{E}, \big\{\{X, Y\} : X, Y \in \mathcal{E} \text{ et } X \cap Y \neq \emptyset\big\}\Big)$.

Un graphe est **biparti faiblement chordal** s'il est biparti et faiblement chordal.

Théorème 3.7. *Un graphe est sans maison, trou et gem induit*[4] *si et seulement si les graphes caractéristiques de la décomposition en coupes sont soit des graphes bipartis faiblement chordaux, soit des graphes complets.*

Démonstration. Les graphes sans maison, trou et gem induit sont des graphes faiblement chordaux, car ils ne possèdent pas de trou, et leurs complémentaires ne possèdent pas de C_5 et de P_5. Ce sont également des graphes de parité car ils respectent la caractérisation par sous-graphes induits interdits des graphes de parité. Donc tous les graphes caractéristiques dans la décomposition en coupes d'un graphe sans maison, trou et gem induit sont soit des graphes bipartis faiblement chordaux, soit des graphes complets.

Inversement, si tous les graphes caractéristiques dans une décomposition en coupes de G sont soit des graphes bipartis faiblement chordaux, soit des graphes complets, alors G est un graphe sans maison, trou et gem induit. En effet, une composition en coupes ne peux pas créer de maison, de trou ou de gem, car se sont des graphes premiers par la décomposition en coupes. □

Il existe un algorithme en temps $O(\min(m\log(n), n^2))$ pour la reconnaissance des graphes bipartis faiblement chordaux [Lub87], ce qui nous donne un algorithme de reconnaissance des graphes sans maison, trou et gem induit en temps $O(\min(m\log(n), n^2))$.

3.3 Notations complémentaires

Dans l'arbre de décomposition modulaire, qui est enraciné, chaque noeud correspond à un sous-graphe du graphe originel. Il est donc possible de résoudre un problème récursivement sur l'arbre de décomposition modulaire. L'arbre de décomposition en coupes n'est pas enraciné, donc on n'a pas directement un sous-graphe correspondant à chaque noeud de l'arbre. Pour cette raison, il est plus commode de choisir une racine à cet arbre, ainsi il sera possible d'avoir une correspondance entre les noeuds de l'arbre enraciné et des sous-graphes du graphe originel.

Soit G un graphe connexe et T son arbre de décomposition en coupes. On rappelle que pour un noeud h de T, G_h^* est le graphe caractéristique du noeud h.

On choisit arbitrairement une racine r pour T. Pour tout noeud $h \neq r$, on note par v_h l'unique marqueur commun à G_h^* et $G_{h'}^*$, où h' est le père de h. Pour chaque feuille h de T, on définit $G_h = G_h^*$. On définit récursivement pour chaque noeud interne h de T, $G_h = G_h^* * G_{h_1} * \ldots * G_{h_k}$, où $h_1, h_2, \ldots, h_k$ désignent les fils de h. On appelle G_h le graphe correspondant au sous-arbre de T enraciné en h. On note par V_h l'ensemble des sommets de G_h. On voit immédiatement que $G_r = G$.

[4] Également appelé *HHG-free*.

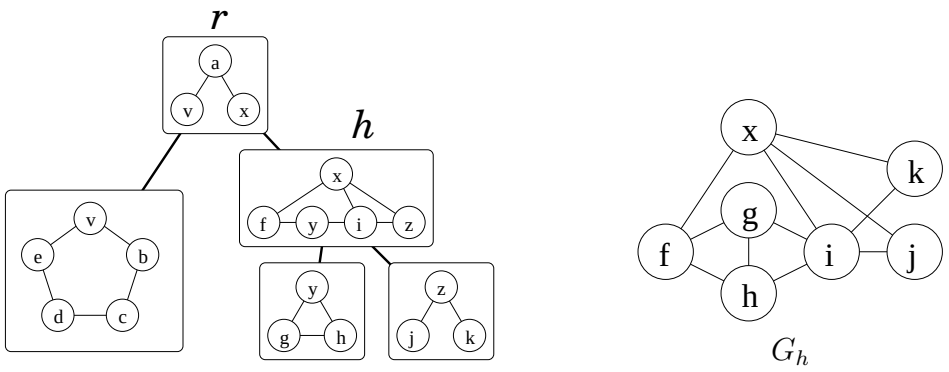


Fig. 3.4 : – Le graphe G_h ($v_h = x$).

3.4 Relation avec la largeur de clique

Nous savons que la largeur de clique d'un graphe est le maximum des largeurs de clique des graphes premiers de sa décomposition modulaire. Dans cette section, on montre un théorème similaire pour la décomposition en coupes : si la largeur de clique de tous les graphes premiers de la décomposition en coupes est au plus k, alors la largeur de clique du graphe est au plus $2k + 1$.

Théorème 3.8. *Soit G un graphe. Alors* $\mathrm{cwd}(G) \leq 2 \times k + 1$*, où k est la largeur de clique maximum des graphes représentatifs des noeuds internes de la décomposition en coupes de G.*

Démonstration. Soit T l'arbre de décomposition en coupes de G et pour tous les noeuds h de T, soit t_h^* une k-expression de G_h^* avec les étiquettes dans $\{1, \ldots, k\}$. On construit une $(2k+1)$-expression pour G en utilisant T. On choisit arbitrairement une racine r dans T et on construit récursivement une $(2k+1)$-expression t_h pour $G_h - v_h$, avec les étiquettes dans $\{0, 1, \ldots, 2k\}$, telle que $\forall v \in N_{G_h}(v_h)$, $\mathrm{lab}_{G(t_h)}(v) = 1$ et $\forall v \notin N_{G_h}(v_h)$, $\mathrm{lab}_{G(t_h)}(v) = 0$.

Soit h un noeud de T, soit q son nombre de fils (éventuellement nul) et soit $h_1, \ldots, h_q$ les fils de h. On construit récursivement t_{h_i} pour tout $i \in \{1, \ldots, q\}$. Pour $h \neq r$, on construit une $2k$-expression t'_h de $G_h^* - v_h$ depuis t_h^*, avec les étiquettes dans $\{1, \ldots, 2k\}$, telle que $\forall v \in N_{G_h^*}(v_h)$, $\mathrm{lab}_{G(t'_h)}(v) = \mathrm{lab}_{G(t_h^*)}(v)$, sinon $\mathrm{lab}_{G(t'_h)}(v) = \mathrm{lab}_{G(t_h^*)}(v) + k$. Soit $t'_r = t_h^*$.

On construit t_h depuis t'_h de la façon suivante. Pour tout $i \in \{1, \ldots, q\}$, on substitue dans t'_h la sous-expression introduisant v_{h_i} avec l'étiquette l par $\rho_{1 \to l}(t_{h_i})$. Finalement on re-étiquette toutes les étiquettes de $\{k+1, \ldots 2k\}$ en 0 et toutes les étiquettes de $\{1, \ldots, k\}$ en 1. (On remarque que si h est une feuille, alors $t_h = t'_h$.)

On montre que pour tout $h \neq r$, t_h est une $(2k+1)$-expression pour $G_h - v_h$ avec la propriété demandée, et t_r est une $(2k+1)$-expression pour $G_r = G$. Pour tout $i \in \{1, \ldots, k\}$, soit $A_i = N_{G_{h_i}}(v_{h_i})$ et soit $B_i = V_{h_i} \setminus N_{G_{h_i}}[v_{h_i}]$. Soit $C = V_h^* \setminus \{v_1, \ldots, v_q\}$. On note que $\{C, \bigcup_{i=1}^{q} A_i, \bigcup_{i=1}^{q} B_i\}$ est une partition des sommets de G_h.

On n'ajoute jamais d'arêtes entre des sommets étiquetés 0 et d'autres sommets dans t_h, donc pour tout $i \in \{1, \ldots, q\}$, $G(t_h)[A_i \cup B_i] = G(t_{h_i}) = G_{h_i} - v_{h_i}$, et il n'y a pas d'arêtes

entre B_1, B_2, ..., B_q et C dans $G(t_h)$. Pour tout $j \in \{1, \ldots, q\}$, A_i est un module dans $G_h - \bigcup_{i=1}^{q} B_i$ et dans $G(t_h) - \bigcup_{i=1}^{q} B_i$. Ainsi si $h \neq r$ alors $G(t_h) - \bigcup_{i=1}^{q} B_i = G_h - v_h - \bigcup_{i=1}^{q} B_i$ et $G(t_r) - \bigcup_{i=1}^{q} B_i = G - \bigcup_{i=1}^{q} B_i$. □

3.5 Algorithmes utilisant la décomposition en coupes

Comme dans le cas de la décomposition modulaire, on est ici intéressé par l'utilisation de la décomposition en coupes pour résoudre un certain problème sur un graphe. Pour cela, il faut formuler un problème, éventuellement plus général, que l'on doit résoudre sur chaque graphe caractéristique de la décomposition du graphe. La décomposition en coupes pouvant être vue comme une généralisation de la décomposition modulaire, le sous-problème à résoudre doit être, soit le même problème que le problème correspondant pour la décomposition modulaire, soit un problème plus général.

Certains problèmes peuvent directement être résolus sur les décompositions en coupes simple. Pour le nombre chromatique, on donnera un algorithme calculant le nombre chromatique pondéré récursivement sur l'arbre de décomposition en coupes enraciné. Le lemme suivant montre que l'arbre de décomposition en coupes, avec les graphes caractéristiques, sera toujours de taille linéaire en la taille du graphe original. Il s'agit de l'équivalent du lemme 2.5, pour la décomposition en coupe.

Lemme 3.9. *Soit $G = (V, E)$ un graphe et $T = (V_T, E_T)$ son arbre de décomposition en coupes. Alors on a $\sum_{h \in V_T} n(G_h^*) \leq 3 \times n(G) - 4$ et $\sum_{h \in V_T} m(G_h^*) \leq m(G) + n(G) - 3$.*

Démonstration. On voir facilement par récurrence qu'un arbre de décomposition en coupes d'un graphe de n sommets a au plus $n - 2$ noeuds. Donc, comme chaque sommet des graphes caractéristiques de la décomposition en coupes est soit un sommet originel, soit un marqueur (apparaissant deux fois), la somme des nombres de sommets des graphes caractéristiques est au plus $3n - 4$.

De plus, comme les graphes sont connexes, chaque décomposition simple ajoute au plus une arête au nombre total d'arêtes dans les graphes caractéristiques. Donc la somme du nombre d'arêtes des graphes caractéristiques est bornée par $m + n - 3$. □

3.5.1 Clique et ensemble stable

Cunningham [Cun82] a donné un algorithme pour calculer le poids maximum d'un ensemble stable dans un graphe pondéré, en utilisant la décomposition en coupes. Cicerone et Stephano [CS99] donnent pour leur part un algorithme pour calculer le poids maximum d'une clique. L'algorithme donné est le suivant : le poids maximum d'une clique est le poids maximum d'une clique dans les graphes caractéristiques de la décomposition en coupes. Le marqueur u dans le graphe caractéristique H a comme poids $\max_{v \in V \cap N_{H'}(u)} w(v)$, où H' est l'unique autre graphe caractéristique possédant le marqueur u. Cet algorithme n'est pas correct : la figure 3.5 en présente un contre exemple.

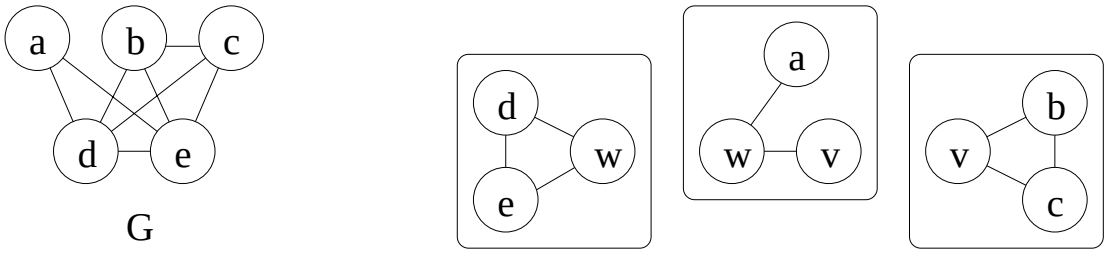


Fig. 3.5 : – Gauche : un graphe de parité pour lequel l'algorithme pour la taille de la plus grande clique donné dans [CS99] échoue. Tous les poids sont de 1. Droite : son arbre de décomposition en coupes, qui est également son unique décomposition première. Les marqueurs sont v et w. $\omega(G) = 4$ et la réponse de l'algorithme est 3.

Le lemme 3.10 donne l'algorithme de Cunningham. Le lemme 3.11 donne un algorithme pour calculer le poids maximum d'une clique, en utilisant la décomposition en coupes.

Soit $G = (V, E)$ un graphe tel que $G = G_1 * G_2$, $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ et $V_1 \cap V_2 = \{v\}$. Soit $w : V \to \mathbb{N}$ une fonction de pondération.

Lemme 3.10 ([Cun82]). *Soit* $a = \alpha_w(G_2 - N_{G_2}[v])$ *et* $a' = \alpha_w(G_2 - v)$. *Alors*

$$\alpha_w(G) = \alpha_{w|_{v \to a'-a}}(G_1) + a.$$

Lemme 3.11. *Soit* $a = \omega_w(G_2[N_{G_2}(v)])$. *Alors* $\omega_w(G) = \max\Big(\omega_w(G_2 - v), \omega_{w|_{v\to a}}(G_1)\Big)$.

Démonstration. On a immédiatement $\omega_w(G) \geq \omega_w(G_2 - v)$. De plus on a $\omega_w(G) \geq \omega_w\Big(G[(V_1 \setminus \{v\}) \cup N_{G_2}(v)]\Big)$, et $\omega_w\left(G[(V_1 \setminus \{v\}) \cup N_{G_2}(v)]\right) = \omega_{w|_{v\to a}}(G_1)$, car $N_{G_2}(v)$ est un module de $G[(V_1 \setminus \{v\}) \cup N_{G_2}(v)]$. En conséquence $\omega_w(G) \geq \max\Big(\omega_w(G_2 - v), \omega_{w|_{v\to a}}(G_1)\Big)$.

En outre, on a $\omega_w(G) = \max\Big(\omega_w(G_2 - v), \omega_{w|_{v\to a}}(G_1)\Big)$ puisque si C est une clique de G alors soit $C \subseteq V_2 \setminus \{v\}$ et $w(C) \leq \omega_w(G_2 - v)$, soit $C \subseteq (V_1 \setminus \{v\}) \cup N_{G_2}(v)$ et $w(C) \leq \omega_{w|_{v\to a}}(G_1)$. □

Les lemmes 3.10 et 3.11, respectivement, peuvent être utilisés pour calculer $\alpha_w(G)$ et $\omega_w(G)$ d'un graphe G en utilisant sa décomposition en coupes T. L'idée est celle donnée par Cicerone et Stephano [CS99]. On prend une coupe $\{V_1, V_2\}$ du graphe, avec comme décomposition simple G_1 et G_2, de marqueur v, tels que G_2 est un graphe premier, un graphe complet ou une étoile. (C'est à dire on prend une coupe correspondant à une arête de l'arbre de décomposition incidente à une feuille.) On calcule $a = \alpha_w(G_2 - N_{G_2}[v])$ et $a' = \alpha_w(G_2 - v)$, puis on calcule récursivement $\alpha_{w|_{v\to a'-a}}(G_1)$. D'après le lemme, on a $\alpha_w(G) = \alpha_{w|_{v\to a'-a}}(G_1) + a$. Au final, on aura calculé au plus deux fois le poids maximum d'un ensemble stable sur chaque graphe caractéristique de la décomposition.

3.5.2 Problèmes de domination

Nous étudions dans cette section le problème de domination et ses variantes. Soit $G = (V, E)$ un graphe tel que $G = G_1 * G_2$, $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ et $V_1 \cap V_2 = \{v\}$. Soit $w : V \to \mathbb{N}$ une fonction de pondération.

Le lemme suivant nous montre comment calculer la taille minimum d'un ensemble dominant connexe, avec une décomposition en coupes simple.

Lemme 3.12. *Soit* $a_1 = \min_{u \in V_1} w(u)$ *et* $a_2 = \min_{u \in V_2} w(u)$. *Alors on a :*

$$\gamma_{c_w}(G) = \begin{cases} \min\left(\gamma_{c_w}(G_1 - v), \gamma_{c_w}(G_2 - v), a_1 + a_2\right) & \text{si } N_{G_1}[v] = V_1 \text{ et } N_{G_2}[v] = V_2, \\ \gamma_{c_{w|_{v \to a_1}}}(G_2) & \text{si } N_{G_1}[v] = V_1 \text{ et } N_{G_2}[v] \neq V_2, \\ \gamma_{c_{w|_{v \to a_2}}}(G_1) & \text{si } N_{G_1}[v] \neq V_1 \text{ et } N_{G_2}[v] = V_2, \\ \gamma_{c_{w|_{v \to 0}}}(G_1) + \gamma_{c_{w|_{v \to 0}}}(G_2) & \text{si } N_{G_1}[v] \neq V_1 \text{ et } N_{G_2}[v] \neq V_2. \end{cases}$$

Démonstration. Dans le premier cas, G est la jointure de $G_1 - v$ et $G_2 - v$. Un ensemble dominant connexe minimal est donc soit un ensemble dominant connexe de $G_1 - v$, soit un ensemble dominant connexe de $G_2 - v$, soit l'union d'un sommet de $G_1 - v$ et de $G_2 - v$.

Dans le deuxième cas, si D est un ensemble dominant connexe minimum de G alors $D \cap N_{G_2}(v) \neq \emptyset$. Si $D \cap (V_1 \setminus \{v\}) \neq \emptyset$, alors $|D \cap (V_1 \setminus \{v\})| = 1$ puisque D est minimal, et ainsi $w(D \cap (V_1 \setminus \{v\})) = a_1$. Le troisième cas est similaire au deuxième.

Dans le dernier cas, soit D un ensemble dominant connexe de G. $D \cap N_{G_1}(v) \neq \emptyset$ et $D \cap N_{G_2}(v) \neq \emptyset$. Alors $D \cap V_1 \cup \{v\}$ est un ensemble dominant connexe de G_1 et $D \cap V_2 \cup \{v\}$ est un ensemble dominant connexe de G_2. Ainsi $\gamma_{c_w}(G) \geq \gamma_{c_{w|_{v \to 0}}}(G_1) + \gamma_{c_{w|_{v \to 0}}}(G_2)$.

De l'autre côté, si D_1 est un ensemble dominant connexe de G_1 et D_2 est un ensemble dominant connexe de G_2, alors $D_1 \cup D_2 \setminus \{v\}$ est connexe, est un ensemble dominant de G et $w(D_1 \cup D_2 \setminus \{v\}) = w(D_1) + w(D_2)$. Au final on a $\gamma_{c_w}(G) = \gamma_{c_{w|_{v \to 0}}}(G_1) + \gamma_{c_{w|_{v \to 0}}}(G_2)$. □

Le même, on a le lemme suivant pour le problème de clique dominante. La preuve est très similaire à celle du lemme 3.12.

Lemme 3.13. *Soit* $a_1 = \min_{u \in V_1} w(u)$ *et* $a_2 = \min_{u \in V_2} w(u)$. *Alors on a :*

$$\gamma_{cl_w}(G) = \begin{cases} \min\left(\gamma_{cl_w}(G_1 - v), \gamma_{cl_w}(G_2 - v), a_1 + a_2\right) & \text{si } N_{G_1}[v] = V_1 \text{ et } N_{G_2}[v] = V_2, \\ \gamma_{cl_{w|_{v \to a_1}}}(G_2) & \text{si } N_{G_1}[v] = V_1 \text{ et } N_{G_2}[v] \neq V_2, \\ \gamma_{cl_{w|_{v \to a_2}}}(G_1) & \text{si } N_{G_1}[v] \neq V_1 \text{ et } N_{G_2}[v] = V_2, \\ \gamma_{cl_{w|_{v \to 0}}}(G_1) + \gamma_{cl_{w|_{v \to 0}}}(G_2) & \text{si } N_{G_1}[v] \neq V_1 \text{ et } N_{G_2}[v] \neq V_2. \end{cases}$$

Pour les problèmes ensemble stable, clique et les deux précédentes variantes d'ensemble dominant, le problème à résoudre sur les composantes était soit le même, soit la version

pondérée du problème. Pour ensemble dominant, on a besoin de définir un problème plus général.

Soit $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$ et $w : V \to \overline{\mathbb{N}}^4$. Soit $\Pi_i : \overline{\mathbb{N}}^4 \to \overline{\mathbb{N}}$ la projection de la ième coordonnée, $i \in \{1, 2, 3, 4\}$. Soit

$$\tilde{\mathrm{w}}(G, D, w, u) = \begin{cases} \Pi_1(w(u)) & \text{si } u \in D \text{ et } N_G(u) \cap D \neq \emptyset, \\ \Pi_2(w(u)) & \text{si } u \in D \text{ et } N_G(u) \cap D = \emptyset, \\ \Pi_3(w(u)) & \text{si } u \notin D \text{ et } N_G(u) \cap D \neq \emptyset, \\ \Pi_4(w(u)) & \text{si } u \notin D \text{ et } N_G(u) \cap D = \emptyset. \end{cases}$$

Le **poids généralisé** d'un ensemble D dans G est

$$\tilde{\mathrm{w}}(G, D, w) = \sum_{u \in V(G)} \tilde{\mathrm{w}}(G, D, w, u).$$

Soit **le nombre de domination généralisé** de G, noté $\gamma_{\mathrm{g}_w}(G)$, le poids généralisé minimum d'un sous-ensemble de sommets D de G. Il est facile de voir que nombre de domination généralisé est une généralisation d'ensemble dominant, puisque $\gamma_w(G) = \gamma_{\mathrm{g}_{w'}}(G)$ avec $w'(u) = (w(u), w(u), 0, +\infty)$ pour tout $u \in V$. Il s'agit aussi d'une généralisation d'ensemble dominant total, avec $w'(u) = (w(u), +\infty, 0, +\infty)$, et d'ensemble dominant indépendant, avec $w'(u) = (+\infty, w(u), 0, +\infty)$.

Lemme 3.14. *Soit*

$$\begin{aligned} a_1 &= \gamma_{g_{w|_{v \to (0,\infty,\infty,\infty)}}}(G_2), \\ a_2 &= \gamma_{g_{w|_{v \to (\infty,\infty,0,\infty)}}}(G_2), \\ a_3 &= \gamma_{g_{w|_{v \to (\infty,0,\infty,\infty)}}}(G_2), \\ a_4 &= \gamma_{g_{w|_{v \to (\infty,\infty,\infty,0)}}}(G_2). \end{aligned}$$

Alors

$$\gamma_{g_w}(G) = \gamma_{g_{w|_{v \to (a_1,a_2,a_3,a_4)}}}(G_1).$$

Démonstration. Soit $w' = w|_{v \to (a_1,a_2,a_3,a_4)}$. Soit D un sous-ensemble de V. Supposons que $D \cap N_{G_2}(v) = \emptyset$ et $D \cap N_{G_1}(v) \neq \emptyset$. Dans ce cas, on obtient :

$$\begin{aligned} \tilde{\mathrm{w}}_{w'}(G_1, D \cap V_1) &= a_3 + \sum_{u \in V_1} \tilde{\mathrm{w}}_w(G_1, D \cap V_1, u) \\ &\leq \tilde{\mathrm{w}}_{w|_{v \to (\infty,0,\infty,\infty)}}\left(G_2, (D \cap V_2) \cup \{v\}\right) + \sum_{u \in V_1} \tilde{\mathrm{w}}_w\left(G_1, D \cap V_1, u\right) \\ &\leq \sum_{u \in V_2} \tilde{\mathrm{w}}_w\left(G_2, (D \cap V_2) \cup \{v\}, u\right) + \sum_{u \in V_1} \tilde{\mathrm{w}}_w\left(G_1, D \cap V_1, u\right) \\ &\leq \tilde{\mathrm{w}}_w(G, D). \end{aligned}$$

Similairement, si $D \cap N_{G_2}(v) = \emptyset$ et $D \cap N_{G_1}(v) = \emptyset$ alors $\tilde{\mathrm{w}}_{w'}(G_1, D \cap V_1) \leq \tilde{\mathrm{w}}_w(G, D)$, et si $D \cap N_{G_2}(v) \neq \emptyset$ alors $\tilde{\mathrm{w}}_{w'}(G_1, (D \cap V_1) \cup \{v\}) \leq \tilde{\mathrm{w}}_w(G, D)$. Dans tous les cas on obtient $\gamma_{\mathrm{g}_{w'}}(G_1) \leq \tilde{\mathrm{w}}_w(G, D)$. Ainsi $\gamma_{\mathrm{g}_{w'}}(G_1) \leq \gamma_{\mathrm{g}_w}(G)$.

Soit D_i, $i \in \{1, 2, 3, 4\}$, des sous-ensembles de $V_2 \cup \{v\}$ tels que

$$\begin{aligned} a_1 &= \tilde{\mathrm{w}}_{w|_{v \to (0,\infty,\infty,\infty)}}(G_2, D_1), \\ a_2 &= \tilde{\mathrm{w}}_{w|_{v \to (\infty,\infty,0,\infty)}}(G_2, D_2), \\ a_3 &= \tilde{\mathrm{w}}_{w|_{v \to (\infty,0,\infty,\infty)}}(G_2, D_3), \\ a_4 &= \tilde{\mathrm{w}}_{w|_{v \to (\infty,\infty,\infty,0)}}(G_2, D_4). \end{aligned}$$

Soit D' un sous-ensemble de $V_1 \cup \{v\}$. Soit

$$D = \begin{cases} D' \setminus \{v\} \cup D_1 & \text{si } v \in D' \text{ et } D' \cap N_{G_1}(v) \neq \emptyset, \\ D' \setminus \{v\} \cup D_2 & \text{si } v \in D' \text{ et } D' \cap N_{G_1}(v) = \emptyset, \\ D' \cup D_3 & \text{si } v \notin D' \text{ et } D' \cap N_{G_1}(v) \neq \emptyset, \\ D' \cup D_4 & \text{si } v \notin D' \text{ et } D' \cap N_{G_1}(v) = \emptyset. \end{cases}$$

Alors $\gamma_{\mathrm{g}_w}(G) \leq \tilde{\mathrm{w}}_w(G, D) \leq \tilde{\mathrm{w}}_{w'}(G_1, D')$. Ainsi $\gamma_{\mathrm{g}_w}(G) \leq \gamma_{\mathrm{g}_{w'}}(G_1)$. □

Tout comme pour les problèmes ensemble stable et clique, ces lemmes peuvent être utilisés pour calculer le paramètre γ_{g_w} en utilisant l'arbre de décomposition en coupes.

3.5.3 Nombre chromatique et coloration

On est maintenant intéressé par le calcul du nombre chromatique d'un graphe en utilisant son arbre de décomposition en coupes. On sait déjà que dans le cas des graphes parfaits le problème se comporte bien car il est équivalent au problème CLIQUE.

Mais dans le cas général, le problème semble plus complexe, puisque si $G = G_1 * G_2$, $\chi(G)$ ne peut pas être déduit facilement des solutions des sous-problèmes sur G_1 et G_2. L'algorithme calculera le nombre chromatique pondéré des sous-graphes correspondants aux noeuds de l'arbre de décomposition en coupes enraciné (défini à la section 3.3), pour une collection de fonctions de poids. Soit G un graphe, et T son arbre de décomposition en coupes, calculé par exemple en utilisant l'algorithme linéaire de Dahlhaus [Dah00]. On choisit arbitrairement un noeud r de l'arbre qu'on prendra comme racine de T pour le reste de cette section. Notre algorithme travaillera récursivement sur l'arbre T enraciné en r.

On rappelle que pour un noeud h de T, G_h^* est le graphe caractéristique correspondant au noeud h de T. G_h est le graphe correspondant au sous-graphe enraciné en h, et si $h \neq r$, v_h est l'unique marqueur commun à G_h^* et $G_{h'}^*$, où h' est le père de h dans T. Notez que $G_r = G$.

On présente notre algorithme à la figure 3.6. Sa validité est basée sur les lemmes 3.15 et 3.16. Les notations suivantes sont utilisées tout au long de la section. Soit C une coloration w-pondérée de $G = (V, E)$ et $V' \subseteq V$. On note par $C_{V'}$ la collection de tous les ensembles

Fonction ChromaticNumber(T)
Entrée : L'arbre de décomposition en coupes T de G,
Sortie : le nombre chromatique de G
début
  pour tout noeud h de T, des feuilles à la racine, où k est le nombre de fils
      de h et $h_1, h_2, \ldots, h_k$ sont ses fils
    si $h \neq r$ **alors**
      $D_h \leftarrow \emptyset$
      pour b **de** 0 **à** n **faire**
        $D_h \leftarrow D_h \cup \{(\text{ChromSplit}(h, b, k, D_{h_1}, D_{h_2}, \ldots, D_{h_k}) - b, b)\}$
      fin pour
    sinon retourner $\text{ChromSplit}(r, 0, k, D_{h_1}, D_{h_2}, \ldots, D_{h_k})$
    fin si
  fin pour
fin {ChromaticNumber}

Fonction $\text{ChromSplit}(h, b, k, D_{h_1}, D_{h_2}, \ldots, D_{h_k})$
début
  si $k = 0$ **alors**
    soit $w^* : V_h^* \to \mathbb{N}$ tel que $\forall v \in V_h^* \setminus \{v_h\}$, $w^*(v) = 1$ et, si $h \neq r$, $w(v_h) = b$
    retourner $\chi_{w^*}(G_h^*)$
  sinon
    soit $m \leftarrow +\infty$
    pour c **de** 0 **à** $2n$ **faire**
      pour i **de** 1 **à** k **faire**
        soit $a_i \leftarrow \min\left\{a' : \exists b' \text{ tel que } (a', b') \in D_{h_i} \text{ et } a' + b' \leq c\right\}$
      fin pour
      soit $w^* : V_h^* \to \mathbb{N}$ tel que pour tout $i \in \{1, 2, \ldots, k\}$, $w^*(v_{h_i}) = a_i$,
          pour tout $v \in V_h^* \setminus \{v_h, v_{h_1}, \ldots v_{h_k}\}$, $w^*(v) = 1$ et, si $h \neq r$, $w(v_h) = b$
      $m \leftarrow \min\left(m, \max\left(c, \chi_{w^*}(G_h^*)\right)\right)$
    fin pour
    retourner m
  fin si
fin {ChromSplit}

Fig. 3.6 : – Algorithme calculant le nombre chromatique.

stables de la coloration C qui contient au moins un sommet de V', c'est à dire $C_{V'} = \langle S \in C : S \cap V' \neq \emptyset \rangle$. On note par $\mathcal{D}(G, w, V')$ l'ensemble de tous les couples $(a, b) \in \mathbb{N} \times \mathbb{N}$ tels qu'il y ait une coloration x pondérée C de G avec $a + b$ couleurs et $|C_{V'}| = a$. On voit facilement que si $(a, b) \in \mathcal{D}(G, w, V')$, alors $(a', b) \in \mathcal{D}(G, w, V')$ pour tout $a' > a$ et $(a, b') \in \mathcal{D}(G, w, V')$ pour tout $b' > b$. On dit qu'un couple $(a, b) \in \mathcal{D}(G, w, V')$ est **minimal** si $(a - 1, b) \notin \mathcal{D}(G, w, V')$ et $(a, b - 1) \notin \mathcal{D}(G, w, V')$. Notez que pour chaque couple minimal (a, b) de $\mathcal{D}(G, w, V')$, on a $a \leq \chi_w(G)$ et $b \leq \chi_w(G)$.

A partir de maintenant, pour chaque noeud $h \neq r$, on appelle l'ensemble $\mathcal{D}(h) = \mathcal{D}(G_h - v_h, w, N_{G_h}(v_h))$ le **$\mathcal{D}$-ensemble** de h, où pour tout $v \in V_h \setminus \{v_h\}$, $w(v) = 1$. Notre algorithme utilise la programmation dynamique et calcule des noeuds à la racine, pour chaque noeud $h \neq r$ de T, le $\mathcal{D}$-ensemble de h depuis les $\mathcal{D}$-ensembles de tous ses fils. Au final, il calcule $\chi(G)$ depuis les $\mathcal{D}$-ensembles des fils de la racine r.

Le lemme suivant expose l'opération à exécuter pour chaque noeud de l'arbre de décomposition pour calculer son $\mathcal{D}$-ensemble et son nombre chromatique, en utilisant les $\mathcal{D}$-ensembles de ses fils. Soit h un noeud de T. Soit $G_h^* = (V_h^*, E_h^*)$ le graphe caractéristique correspondant à h, et soit $G_h = (V_h, E_h)$ le graphe correspondant au sous-graphe enraciné en h. Soit $h_1, h_2, \ldots, h_k$ les fils de h dans T. Alors G_{h_i} est le graphe correspondant au sous-arbre enraciné en h_i et v_{h_i} est l'unique marqueur en commun à G_h^* et G_{h_i}. Pour simplifier les notations dans le lemme 3.15 et sa preuve, on notera $G_i = G_{h_i} = (V_i, E_i)$, $v_i = v_{h_i}$ et $G^* = G_h^* = (V^*, E^*)$. Comme il a été dit précédemment, $G_h = G^* * G_1 * \ldots * G_k$. Soit $w : V_h \to \mathbb{N}$ une fonction de poids.

Lemme 3.15. *Pour tout $i \in \{1, \ldots k\}$, soit $(a_i, b_i) \in \mathcal{D}(G_i - v_i, w, N_{G_i}(v_i))$. Soit $\mathfrak{C}$ l'ensemble de toutes les colorations w-pondérées C de G_h avec $|C_{V_i \setminus \{v_i\}}| = a_i + b_i$ et $|C_{N_{G_i}(v_i)}| = a_i$. Alors*

$$\min_{C \in \mathfrak{C}} |C| = \max\left(\chi_{w^*}(G^*), \max_{i \in \{1,2,\ldots,k\}} (a_i + b_i)\right)$$

où $w^* : V^* \to \mathbb{N}$ tel que pour tout $i \in \{1, 2, \ldots, k\}, w^*(v_i) = a_i$, et pour tout $v \in V^* \setminus \{v_1, v_2, \ldots, v_k\}$, $w^*(v) = w(v)$.

Démonstration. On rappelle que si C est une coloration pondérée de G, et $V' \subseteq V$, alors $C_{V'}$ est défini comme le multi-ensemble des ensembles stables de C qui ont une intersection non vide avec V'. Pour tout $i \in \{1, \ldots, k\}$, soit $A_i = N_{G_i}(v_i)$ et $B_i = V_i \setminus N_{G_i}[v_i]$.

On montre que toutes les colorations $C \in \mathfrak{C}$ contiennent au moins $\max\Big(\chi_{w^*}(G^*), \max_{i \in \{1,2,\ldots,k\}}(a_i + b_i)\Big)$ ensembles stables. Pour tout $i \in \{1, \ldots k\}$, $a_i + b_i = |C_{V_i \setminus \{v_i\}}| \leq |C|$. Il reste à prouver que $\chi_{w^*}(G^*) \leq |C|$. Pour cela, on construit une coloration pondérée C^* de (G^*, w^*) depuis $C_{V_h \setminus \bigcup_{i=1}^k B_i}$. Pour tout $S \in C_{V_h \setminus \bigcup_{i=1}^k B_i}$, soit $S^* \subseteq V^*$ l'ensemble vérifiant $S^* \cap (V^* \setminus \{v_1, \ldots v_k\}) = S \cap (V^* \setminus \{v_1, \ldots v_k\})$, et pour tout $i \in \{1, 2, \ldots, k\}$, $v_i \in S^*$ si et seulement si $S \cap A_i \neq \emptyset$. Notons que S^* est un ensemble stable de G^*. Soit C^* le multi-ensemble de tous les ensembles S^*. Clairement $|C^*| \leq |C|$. De plus C^*

est une coloration w^*-pondérée de G^* puisque pour tout i, $|C^*_{\{v_i\}}| = |C_{A_i}| = a_i$. Ainsi $\min_{C\in\mathfrak{C}} |C| \geq \max\Big(\chi_{w^*}(G^*), \max_{i\in\{1,2,\ldots,k\}}(a_i + b_i)\Big)$.

Il reste à démontrer l'égalité. Pour cela on construit une coloration w-pondérée C de G_h telle que $C \in \mathfrak{C}$ et $|C| = \max\Big(\chi_{w^*}(G^*), \max_{i\in\{1,2,\ldots,k\}}(a_i + b_i)\Big)$. Soit C^* une coloration w^*-pondérée minimum de G^*, et pour tout i, soit C^i une coloration de $G_i - v_i$ telle que $|C^i| = a_i + b_i$ et $|C^i_{A_i}| = a_i$.

On construit C de la façon suivante. Dans la première étape, on prend un ensemble stable $S^* \in C^*$ et pour tout $i \in \{1, 2, \ldots, k\}$, si $v_i \in S^*$ on prend un ensemble stable S^i dans $C^i_{A_i}$, sinon on prend S^i dans $C^i \setminus C^i_{A_i}$. S'il ne reste aucun ensemble stable, on prend $S^i = \emptyset$. On ajoute à C l'ensemble $S = S^* \setminus \{v_1, \ldots v_k\} \cup \bigcup_{i=1}^k S^i$. On note que S est un ensemble stable. On répète cette opération jusqu'à ce que C^* soit vide. Comme $w^*(v_i) = a_i = |C^i_{A_i}|$, il ne reste aucun ensemble dans $S^i \in C^i$ tel que $S_i \cap A_i \neq \emptyset$ à la fin de la première étape.

Dans la deuxième étape, tant qu'il existe un $i \in \{1, 2, \ldots, k\}$ tel que $C^i \neq \emptyset$, on prend, pour tout $i \in \{1, 2, \ldots, k\}$, un ensemble stable $S^i \in C^i$ si $C^i \neq \emptyset$, sinon on prend $S^i = \emptyset$. On ajoute à C l'ensemble $S = \bigcup_{i=1}^k S^i$. On note que S est un ensemble stable. Il n'est pas difficile de voir que C est une coloration w-pondérée de G_h, et que $C \in \mathfrak{C}$.

À la fin de la première étape, C a $\chi_{w^*}(G^*)$ ensembles stables, et pour tout $i \in \{1, \ldots, k\}$, C^i a $\max\Big(0, |C^i \setminus C^i_{A_i}| - |C^* \setminus C^*_{v_i}|\Big) = \max\big(0, b_i - \chi_{w^*}(G^*) + a_i\big)$ ensembles stables restants. Alors on ajoute dans la deuxième étape $\max\Big(0, -\chi_{w^*}(G^*) + \max_{i\in\{1,2,\ldots,k\}}(a_i + b_i)\Big)$ ensembles stables à C. Ainsi, à la fin $|C| = \max\Big(\chi_{w^*}(G^*), \max_{i\in\{1,2,\ldots,k\}}(a_i + b_i)\Big)$. □

Alors $\chi_w(G_h)$ est le minimum de $\max\Big(\chi_{w^*}(G^*_h), \max_{i\in\{1,2,\ldots,k\}}(a_i + b_i)\Big)$ parmi tous les choix possibles des k-uplet $\big((a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\big)$ dans $\mathcal{D}(h_1) \times \mathcal{D}(h_2) \times \ldots \times \mathcal{D}(h_k)$. Pour tout $c \in \mathbb{N}$, soit t_c le k-uplet tel que pour tout $i \in \{1, 2, \ldots k\}$, (a_i, b_i) est un couple de $\{(a, b) \in D(h_i) : a + b \leq c\}$ avec le plus petit a possible. On voit facilement, si t est un k-uplet $\big((a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\big)$ et $c = \max_i(a_i + b_i)$, alors le choix de t_c est au moins aussi bon que le choix de t. L'algorithme calcule $\chi_w(G_h)$ en prenant le minimum parmi tous les k-uplet t_c, $c \in \{0, 1, \ldots, 2n\}$.

Le lemme précédent montrait comment calculer le nombre chromatique pondéré de G_h depuis les $\mathcal{D}$-ensembles des fils de h. Si h est un noeud différent de r, on doit calculer le $\mathcal{D}$-ensemble de h. Le lemme suivant nous montre comment cela peut être fait.

Lemme 3.16. *Soit $G = (V, E)$ un graphe, $v \in V$ et $w : V \setminus \{v\} \to \mathbb{N}$ une fonction de poids. Soit $b \in \mathbb{N}$ et $w' : V \to \mathbb{N}$ tels que $w'(v) = w(v)$ pour tout $v \in V \setminus \{v\}$, et $w'(v) = b$. Alors $(\chi_{w'}(G) - b, b) \in \mathcal{D}(G - v, w, N(v))$, et $(\chi_{w'}(G) - b - 1, b) \notin \mathcal{D}(G - v, w, N(v))$.*

Démonstration. Soit C une coloration w'-pondérée minimum de G. Alors $C' = \langle S \setminus \{v\} : S \in C\rangle$ est une coloration w-pondérée de G, et a au moins b ensembles stables n'ayant pas de sommets en commun avec $N(v)$. Ainsi $(\chi_{w'}(G) - b, b) \in \mathcal{D}(G - v, w, N(v))$.

S'il existait une coloration w-pondérée $\widehat{C}$ de $G - v$ telle que $|\widehat{C}| < \chi_{w'}(G)$ ayant au moins b ensembles stables n'ayant pas de sommets en commun avec $N(v)$, alors le multi-ensemble

C obtenu en ajoutant v à tous les ensembles stables n'ayant pas de sommets en commun avec $N(v)$, serait une coloration w'-pondérée de G, ce qui implique une contradiction. □

Maintenant nous savons comment calculer les couples minimaux du $\mathcal{D}$-ensemble de h depuis $\chi_{w|_{v_h \to b}}(G_h)$, $b = 0, 1, \ldots, n$. Nous avons donc tout ce qui est nécessaire à la preuve de l'algorithme.

Théorème 3.17. *L'algorithme* ChromaticNumber *prend en entrée un graphe G et son arbre de décomposition en coupes T et calcule le nombre chromatique de G.*

Démonstration. La fonction ChromSplit$(h, b, \ldots)$ retourne $\chi_w(G_h)$, avec $w(v) = 1$ pour tout v dans $V_h \setminus \{v_h\}$ et, si $h \neq r$, $w(v_h) = b$, en utilisant le lemme 3.15 et la remarque suivant la preuve du lemme 3.15. Pour tout $h \neq r$, la boucle principale calcule D_h qui contient toutes les paires minimum du $\mathcal{D}$-ensemble de h, utilisant le lemme 3.16. Au final, l'algorithme calcule le nombre chromatique de $G_r = G$. □

L'algorithme calcule le nombre chromatique du graphe donné en entrée, en utilisant en sous-routine une procédure calculant le nombre chromatique pondéré des graphes caractéristiques. Il n'est pas difficile de modifier l'algorithme pour qu'il calcule en même temps une coloration de taille minimum, si on utilise en sous-routine une procédure calculant une coloration pondérée minimum des graphes caractéristiques.

Théorème 3.18. *Si l'algorithme utilisé pour calculer le nombre chromatique pondéré des graphes caractéristiques de la décomposition en coupes à un temps de $f(n, m)$, alors le temps total de l'algorithme* ChromaticNumber *est en $O(n^3 \cdot f(n, m))$.*

Démonstration. Cet algorithme exécute $O(n^2)$ fois l'algorithme pour le nombre chromatique pondéré pour tous les graphes caractéristiques dans la décomposition en coupes, et l'arbre de décomposition en coupes possède $O(n)$ noeuds. Le reste peut être fait en temps $O(n^3)$ en pré-calculant $\min\{a' : \exists b' \text{ tel que } (a', b') \in D_h \text{ et } a' + b' \leq c\}$ pour tout noeud h de T et pour tout $c \in \{0, 1, \ldots, 2n\}$. En conséquence, le temps total est en $O(n^3 \cdot f(n, m))$. □

Remarque. *Le temps peut être amélioré en $O(n^2 \cdot f(n, m))$ pour les fonctions f telles que $f(a+a', b+b') \geq f(a, b) + f(a', b')$ pour a, a', b, b' entiers strictement positifs. Dans ce cas le temps pour exécuter l'algorithme pour le nombre chromatique pondéré sur tous les graphes premiers est en $O(f(n, m))$. Cela est le cas par exemple si $f(n, m) = n + m$.*

Chapitre 4

Décomposition bi-joint

La décomposition bi-joint[1] a été introduite par Fabien de Montgolfier [Mon03]. Il s'agit d'une autre décomposition généralisant la décomposition modulaire, également basée sur les familles bi-partitives.

4.1 Bi-joints d'un graphe

Un **bi-joint** dans un graphe $G = (V, E)$ est une partition de V en deux ensembles V_1 et V_2 tels que $|V_1| \geq 1$, $|V_2| \geq 1$, $\exists W_1 \subseteq V_1$, $\exists W_2 \subseteq V_2$ et les seules arêtes dans G entre V_1 et V_2 sont toutes les arêtes entre W_1 et W_2 et toutes les arêtes entre $V_1 \setminus W_1$ et $V_2 \setminus W_2$. La figure 4.1 en donne un exemple.

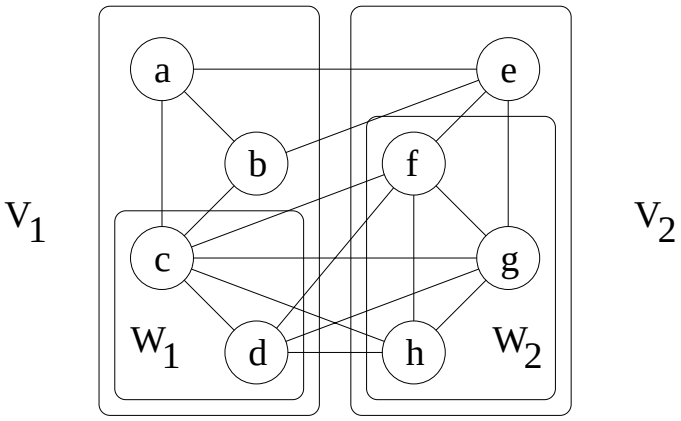


Fig. 4.1 : – *Un bi-joint $\{V_1, V_2\}$ dans un graphe.*

Un bi-joint est **trivial** si $|V_1| = 1$ ou $|V_2| = 1$. On peut noter que toutes les bi-partitions $\{V_1, V_2\}$ avec $|V_1| = 1$ sont des bi-joints. Un graphe est **premier par rapport à la décomposition bi-joint** si tous ses bi-joints sont triviaux.

[1] Originellement, Fabien de Montgolfier a nommé les bi-joints des 2-**joints**. Pour éviter des confusions avec les 2-joints du théorème des graphes parfaits, un autre nom a été ensuite utilisé.

On voit facilement que les bi-joints généralisent les modules : si X est un module, alors $\{X, V \setminus X\}$ est un bi-joint. Tout comme c'est le cas pour les coupes, X peut être un module non trivial et $\{X, V \setminus X\}$ un bi-joint trivial.

Fabien de Montgolfier a montré dans sa thèse que la famille des bi-joints est également une famille bi-partitive.

Théorème 4.1. *[Mon03] La famille de tous les bi-joints d'un graphe est un famille bi-partitive.*

4.2 Relation avec les modules

On va montrer une forte relation entre les bi-joints et les modules, qui sera faite par le biais de la transformation de graphe définie dans le paragraphe suivant. Cette relation, donnée par le théorème 4.2, nous permettra de simplifier la preuve de résultats déjà connus sur les bi-joints, comme la propriété de bi-partitivité, et nous servira en démontrer de nouvelles, en se servant des résultats déjà connus pour la décomposition modulaire.

La **permutation de Seidel** [Sei76] d'un graphe $G = (V, E)$ avec comme support un sous-ensemble $W \subseteq V$ est le graphe avec le même ensemble de sommets, et un sommet x est adjacent à un sommet y si soit $\{x, y\} \in E$ et $\{x, y\}$ ne chevauche pas W, soit $\{x, y\} \notin E$ et $\{x, y\}$ chevauche W. Le graphe obtenu sera noté $\overline{G}^W$. Soit la **permutation de voisinage** d'un graphe G au sommet v, noté $\overline{G}^v$, le graphe $\overline{G}^{N(v)} - v$. La figure 4.2 donne un exemple d'une permutation de voisinage.

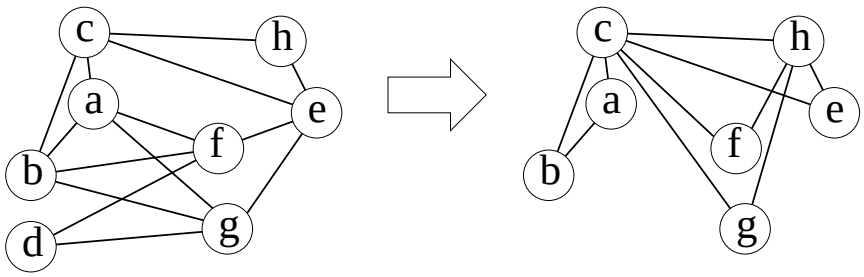


Fig. 4.2 : – *La permutation de voisinage au sommet d.*

Théorème 4.2. *Soit $\{V_1, V_2\}$ une bi-partition de V et $v \in V_1$. Alors $\{V_1, V_2\}$ est un bi-joint de G si et seulement si V_2 est un module de $\overline{G}^v$.*

Démonstration. Soit $\{V_1, V_2\}$ une bi-partition de G et soit $v \in V_1$. Alors, V_2 est un module de $\overline{G}^v$ si et seulement si pour tout $w \in V_1 \setminus \{v\}$, $V_2 \subseteq N_{\overline{G}^v}(w)$ ou $V_2 \cap N_{\overline{G}^v}(w) = \emptyset$. De plus, $\{V_1, V_2\}$ est un bi-joint de G si et seulement si pour tout $w \in V_1 \setminus \{v\}$, $V_2 \cap N_G(w) = V_2 \cap N_G(v)$ ou $V_2 \cap N_G(w) = V_2 \setminus N_G(v)$.

Supposons que v est adjacent à w. $V_2 \subseteq N_{\overline{G}^v}(w)$ si et seulement si pour tout $u \in V_2$, u est adjacent à v et w, ou u n'est adjacent à aucun de v et w. Alors, $V_2 \subseteq N_{\overline{G}^v}(w) \iff$

$V_2 \cap N_G(w) = V_2 \cap N_G(v)$. De plus, $V_2 \cap N_{\overline{G}^v}(w) = \emptyset$ si et seulement si pour tout $u \in V_2$, u est adjacent à exactement un sommet de $\{v, w\}$. Alors $V_2 \cap N_{\overline{G}^v}(w) = \emptyset \iff V_2 \cap N_G(w) = V_2 \setminus N_G(v)$.

Si v n'est pas adjacent à w, alors l'observation précédente sur $\overline{G}$ nous donne immédiatement $V_2 \subseteq N_{\overline{G}^v}(w) \iff V_2 \cap N_G(w) = V_2 \setminus N_G(v)$ et $V_2 \cap N_{\overline{G}^v}(w) = \emptyset \iff V_2 \cap N_G(w) = V_2 \cap N_G(v)$. Dans tout les cas, $V_2 \cap N_{\overline{G}^v}(w) = \emptyset$ ou $V_2 \subseteq N_{\overline{G}^v}(w)$ si et seulement si $V_2 \cap N_G(w) = V_2 \cap N_G(v)$ ou $V_2 \cap N_G(w) = V_2 \setminus N_G(v)$. □

Le théorème 4.2 nous donne facilement une preuve alternative pour montrer que la famille des bi-joints est bi-partitive.

Théorème 4.3. *La famille des bi-joints d'un graphe G est bi-partitive.*

Démonstration. Soit $\mathcal{F}$ la famille des bi-joints de $G = (V, E)$. Soit $v \in V$ et $\mathcal{F}' = \left\{V' \subseteq V : v \notin V' \text{ et } \{V, V \setminus V'\} \in \mathcal{F}\right\}$. Par le théorème 4.2, $\mathcal{F}'$ est la famille des modules de $\overline{G}^v$, et donc est une famille partitive. Donc par le lemme 1.14, $\mathcal{F}$ est une famille bi-partitive. □

On peut remarquer plus généralement qu'une permutation de Seidel ne change pas la famille des bi-joints d'un graphe.

Lemme 4.4. *Soit $G = (V, E)$ un graphe, et $W \subseteq V$. Alors $\{V_1, V_2\}$ est un bi-joint de G si et seulement si $\{V_1, V_2\}$ est un bi-joint de $\overline{G}^W$.*

Démonstration. On peut remarquer que si $W = A \cup B$ et $A \cap B = \emptyset$, alors $\overline{G}^W = \overline{\overline{G}^A}^B$. Il suffit donc de montrer que pour tout $v \in V$, $\{V_1, V_2\}$ est un bi-joint de G si et seulement si $\{V_1, V_2\}$ est un bi-joint de $\overline{G}^{\{v\}}$. Soit $W_1 \subseteq V_1$ et $W_2 \subseteq V_2$ tels que les arêtes de G entre V_1 et V_2 soient les arêtes entre W_1 et W_2, et entre $V_1 \setminus W_1$ et $V_2 \setminus W_2$. Sans perte de généralité, $v \in W_1$. On voit immédiatement que dans $\overline{G}^{\{v\}}$, les arêtes entre V_1 et V_2 sont les arêtes entre $W_1 \setminus \{v\}$ et W_2, et entre $(V_1 \setminus W_1) \cup \{v\}$ et $V_2 \setminus W_2$. Donc $\{V_1, V_2\}$ est un bi-joint de $\overline{G}^{\{v\}}$. Inversement si $\{V_1, V_2\}$ est un bi-joint de $\overline{G}^{\{v\}}$, alors $\{V_1, V_2\}$ est un bi-joint de $\overline{\overline{G}^{\{v\}}}^{\{v\}} = G$. □

On dit que deux graphes $G = (V, E)$ et $H = (V, E')$ sont équivalents par la permutation de Seidel s'il existe un $W \subseteq V$ tel que H soit isomorphe à $\overline{G}^W$. On voit facilement qu'il s'agit d'une relation d'équivalence. Quand G et H sont équivalents par la permutation de Seidel, le lemme 4.4 nous dit que la famille des bi-joints de G sera égale à la famille des bi-joints de H.

4.3 Décomposition simple

La décomposition en coupes peut être définie par l'application successive de la décomposition en coupes simple. Une décomposition simple peut aussi être définie pour le bi-joint.

Soit $\{V_1, V_2\}$ un bi-joint non trival de G, et soit $W_1 \subseteq V_1$ et $W_2 \subseteq V_2$ tels qu'il y ait une jointure entre W_1 et W_2, une jointure entre $V_1 \subseteq W_1$ et $V_2 \subseteq W_2$, et qu'il n'y ait pas d'autres arêtes entre V_1 et V_2. Une **décomposition bi-joint simple** de G par le bi-joint $\{V_1, V_2\}$ est $\{G_1, G_2\}$, où, pour $i \in \{1, 2\}$, G_i est le graphe avec pour ensemble de sommets $V_i \cup \{v\}$, et pour ensemble d'arêtes $\big(E \cap \mathcal{P}_2(V_i)\big) \cup \big\{\{v, w\} : w \in W_i\big\}$.

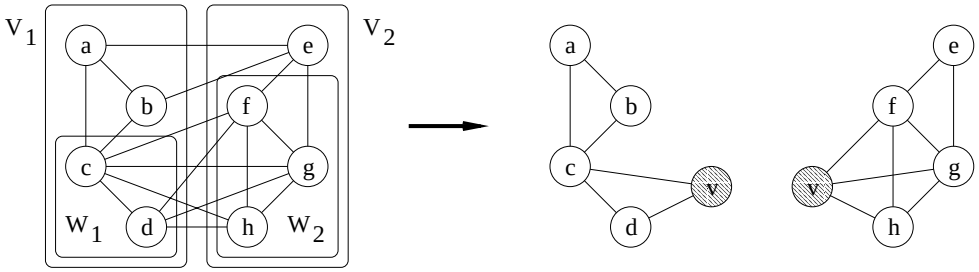


Fig. 4.3 : – Une décomposition bi-joint simple d'un graphe.

On peut remarquer que contrairement à la décomposition en coupes simple, pour un bi-joint fixé, la décomposition simple n'est pas unique. L'autre décomposition simple serait celle donnée dans par la figure 4.4. Ainsi, cette décomposition ne respecte pas la propriété F2 du cadre de définition défini par Cunningham et Edmonds [CE80]. On verra en section 4.5 comment on peut obtenir une certaine unicité.

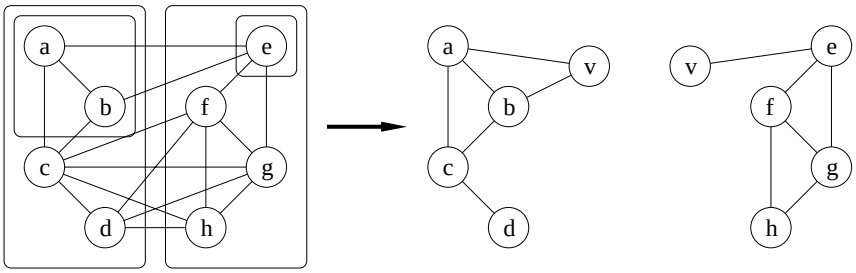


Fig. 4.4 : – Une autre décomposition bi-joint simple pour le même bi-joint.

4.4 Décomposition bi-joint

4.4.1 Arbre de décomposition

Le théorème 4.1 et le théorème 1.11 nous donne directement :

Corollaire 4.5. *Soit G un graphe. Il existe un unique arbre T, l'**arbre de décomposition bi-joint** de G, tel qu'il y ait bijection entre les feuilles de T et les sommets de G, et que les noeuds internes soient étiquetés **dégénérés** ou **premiers**. Pour tout bi-joint $\{V_1, V_2\}$ de G, il existe un noeud α de T et $I \subseteq \{1, \ldots, d(\alpha)\}$ tels que $\{V_1, V_2\} = \{\bigcup_{i \in I} C^i_\alpha, \bigcup_{i \notin I} C^i_\alpha\}$.*

De plus pour toute arête e de T, $\{C_e^1, C_e^2\}$ est un bi-joint de G, et pour tout noeud α de T et $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$, $\{\bigcup_{i \in I} C_\alpha^i, \bigcup_{i \notin I} C_\alpha^i\}$ est un bi-joint de G.

La figure 4.4.1 donne en exemple un graphe avec son arbre de décomposition bi-joint.

4.4.2 Graphes dégénérés

Lemme 4.6. *Soit $G = (V, E)$ un graphe dégénéré pour la décomposition bi-joint, c'est à dire tel que toutes les bi-partitions $\{V_1, V_2\}$ de V, avec $|V_1| \geq 1$ et $|V_2| \geq 1$, sont des bi-joints de G. Alors G est l'union disjointe de deux graphes complets (éventuellement vides), ou G est un graphe biparti complet.*

Démonstration. Soit $G = (V, E)$ un graphe avec la propriété donné, et $v \in V$. Alors $\overline{G}^v$ est un graphe tel que pour tout $V' \subseteq V \setminus \{v\}$ non vide, V' est un module de $\overline{G}^v$. $\overline{G}^v$ est donc un graphe dégénéré de la décomposition modulaire. Donc $\overline{G}^v$ est un graphe sans arêtes, ou $\overline{G}^v$ est un graphe complet. Dans le premier cas, G est un graphe biparti complet, dans le deuxième, G est l'union disjointe de deux graphes complets. □

4.4.3 Graphes caractéristiques

Tout comme pour la décomposition modulaire et la décomposition en coupes, un graphe caractéristique peut aussi être défini pour chaque noeud de l'arbre. Malheureusement, comme pour la décomposition simple, nous n'avons pas directement leurs unicité.

Soit $G = (V, E)$ un graphe, T son arbre de décomposition bi-joint et α un noeud interne de T. Pour tout $i \in \{1, \ldots, d(\alpha)\}$, on choisit un sommet $v_i \in C_\alpha^i$. On appelle alors le **graphe caractéristique** de α, noté $BJC(\alpha)$, le graphe $G[\{v_1, \ldots, v_{d(\alpha)}\}]$.

Lemme 4.7. *Si α est un noeud* `premier`*, alors $BJC(\alpha)$ sera un graphe premier par la décomposition bi-joint. Si α est un noeud* `dégénéré`*, alors $BJC(\alpha)$ sera un graphe dégénéré pour la décomposition bi-joint.*

Démonstration. Si α est un noeud `dégénéré`, alors par définition toutes les bi-partitions de $\{C_\alpha^1, \ldots C_\alpha^{d(\alpha)}\}$ induisent des bi-joints de G. Donc toutes les bi-partitions de $V(BJC(\alpha))$ induisent des bi-joints de $BJC(\alpha)$, et $BJC(\alpha)$ est dégénéré.

Soit un couple (A, B) tel que $\{A, B\}$ est une bi-partition de V. On note par $\sim_{(A,B)}$ la relation sur A telle que $u \sim_{(A,B)} v$ si $N(u) \cap B = N(v) \cap B$ ou si $\{N(u) \cap B, N(v) \cap B\}$ est une partition de B. Notons que $\sim_{(A,B)}$ est une relation d'équivalence, et que $\{A, B\}$ est un bi-joint si et seulement si $\sim_{(A,B)}$ possède une seule classe d'équivalence.

Soit α un noeud `premier`. Supposons que $BJC(\alpha)$ n'est pas premier par rapport à la décomposition bi-joint, et que $\{A, B\}$ est un bi-joint non trivial $BJC(\alpha) = G[\{v_1, \ldots, v_k\}]$. Soit $V_1 = \bigcup_{v_i \in A} C_\alpha^i$. On montre que $\{V_1, B\}$ est un bi-joint de $G[V_1 \cup B]$. Pour tout $i \in \{1, \ldots, k\}$ tel que $v_i \in A$ et pour tout $u \in C_\alpha^i$, $u \sim_{(V_1,B)} v_i$, car $\{C_\alpha^i, V \setminus C_\alpha^i\}$ est un bi-joint de G. De plus, pour tout $v_i, v_{i'} \in A$, $v_i \sim_{(V_1,B)} v_{i'}$ car $\{A, B\}$ est un bi-joint de $G[A \cup B]$. Alors $\sim_{(V_1,B)}$ a une classe d'équivalence unique et $\{V_1, B\}$ est un bi-joint

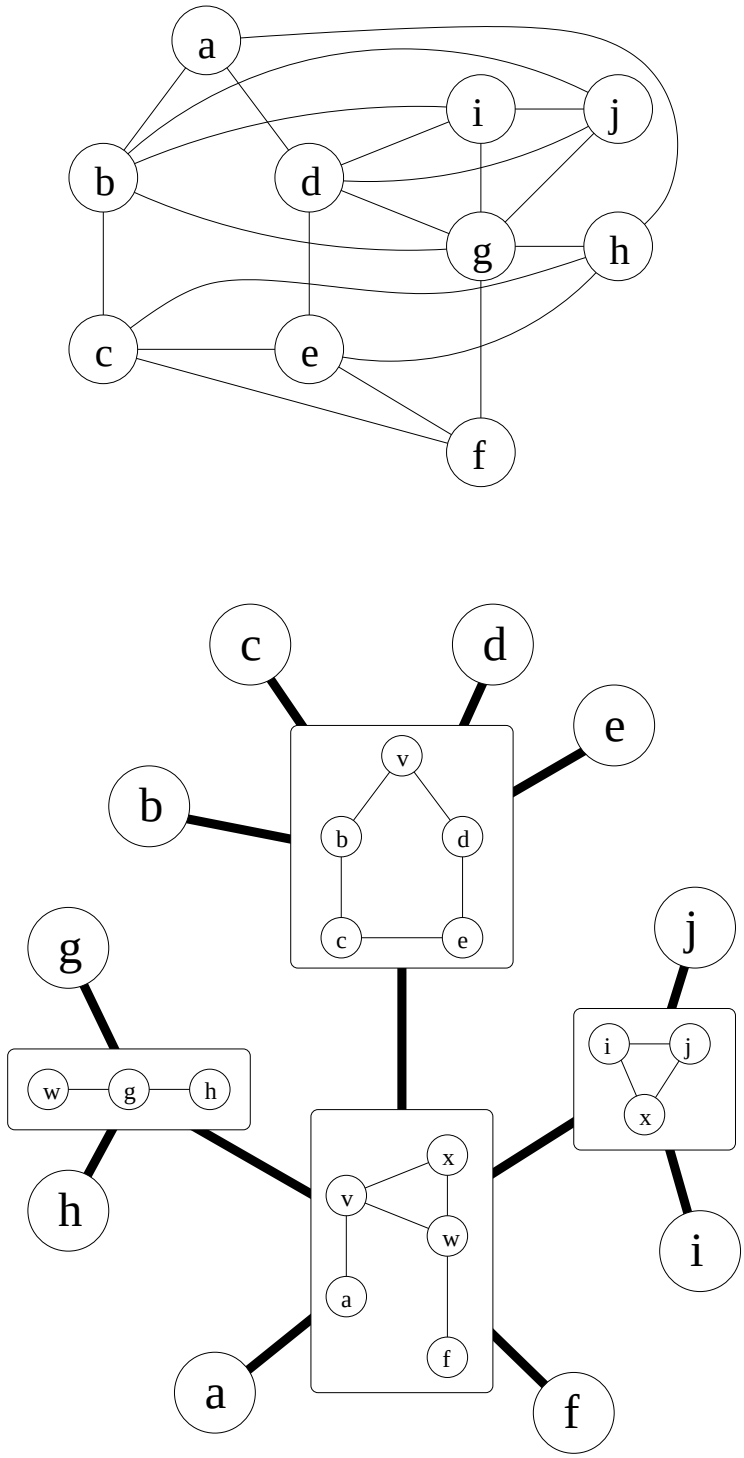


Fig. 4.5 : – Un graphe et son arbre de décomposition bi-joint. Les noeuds internes avec un graphe caractéristique possédant 5 sommets sont `premiers`, les autres sont `dégénérés`.

de $G[V_1 \cup B]$. En utilisant le même argument, le bi-joint $\{B, V_1\}$ de $G[V_1 \cup B]$ peut être étendu à un bi-joint $\{V_2, V_1\}$ de G, avec $V_2 = \bigcup_{v_i \in B} C^i_\alpha$. $\{V_1, V_2\}$ n'est pas dans l'arbre de décomposition de G et nous avons une contradiction. Ainsi $BJC(\alpha)$ est premier par rapport à la décomposition bi-joint. □

Les graphes caractéristiques des noeuds complets de l'arbre de décomposition bi-joint sont donc des graphes bipartis complets, ou des union disjointes de deux graphes complets.

4.5 Décomposition des deux-graphes

Si $\{G_1, G_2\}$ et $\{G'_1, G'_2\}$ sont les deux décompositions simples différentes de G par le bi-joint $\{V_1, V_2\}$, telles que $V(G_1) = V(G'_1) = V_1 \cup \{v\}$, où v est le marqueur, alors G_1 et G'_1 sont équivalents par la permutation de Seidel, et G_2 et G'_2 également. En effet $G'_1 = \overline{G_1}^{\{v\}}$ et $G'_2 = \overline{G_2}^{\{v\}}$. Le choix de la décomposition simple n'a donc pas d'importance sur les décompositions simples suivantes. La décomposition simple est donc unique *modulo* la permutation de Seidel.

Un **deux-graphe**[2] est un couple (V, Δ) tel que Δ est une famille de sous-ensembles de taille 3 de V, et tout sous-ensemble de taille 4 de V inclut un nombre pair d'ensembles de Δ. Seidel [Sei76] montre qu'il y a une bijection entre les deux-graphes sur l'ensemble V et les classes d'équivalences par permutation de Seidel des graphes d'ensembles de sommets V. On dira donc qu'une bi-partition $\{V_1, V_2\}$ de V est un **bi-joint** d'un deux-graphe si elle est un bi-joint d'un graphe de la classe d'équivalence correspondant au deux-graphe. La décomposition simple de H par un bi-joint $\{V_1, V_2\}$ sera la décomposition de H en H_1 et H_2, telle que H_i, $i \in \{1, 2\}$, est le deux-graphe correspondant à G_i, $\{G_1, G_2\}$ est une décomposition simple de G par le bi-joint $\{V_1, V_2\}$, et G est un graphe de la classe d'équivalence correspondant à H. D'après les observations précédentes, cette décomposition simple est unique.

Soit $\mathcal{T}$ la famille des deux-graphes, et V la fonction qui à un deux-graphe H associe son ensemble de sommets. Soit $\rightarrow$ la relation qui à un deux-graphe H associe une décomposition simple $\{H_1, H_2\}$ par un bi-joint $\{V_1, V_2\}$ de H.

Lemme 4.8. *$(\mathcal{T}, V, \rightarrow)$ est un cadre de décomposition, et est symétrique et transitif.*

Démonstration. Les propriétés (F1) et (F4) sont respectées par définition de la décomposition simple. (F3), la propriété de symétrie et la propriété de transitivité proviennent immédiatement du fait que la famille des bi-joints du deux-graphe est une famille bi-partitive. Enfin la propriété (F2) provient du fait que la décomposition simple est unique. □

On dispose donc d'une décomposition des deux-graphes. Pour cette décomposition, les deux-graphes caractéristiques sont uniques.

[2] de l'anglais *two-graph*.

4.6 Algorithmes de décomposition

Fabien de Montgolfier donne un algorithme en $O(n^2m^2)$ pour trouver, s'il existe, un bi-joint non trivial dans un graphe. Il donne ensuite un algorithme en $O(n^3m^2)$ pour calculer la décomposition bi-joint d'un graphe, en utilisant la technique utilisée par Cunningham pour son algorithme de décomposition en coupes : on décompose tant que cela est possible les graphes par la décomposition bi-joint simple, puis on fusionne, tant que possible, les noeuds dégénérés qui peuvent être fusionnés. Au final, l'arbre obtenu est l'arbre de décomposition bi-joint.

En fait le calcul de la décomposition bi-joint peut se faire en temps linéaire, en utilisant le théorème 4.2. Si on prend un sommet v de degré minimum, alors le graphe $\overline{G}^v$ aura au plus trois fois le nombre d'arêtes de G. On peut donc utiliser un algorithme linéaire pour calculer la décomposition modulaire de $\overline{G}^v$, pour obtenir en temps linéaire l'arbre de décomposition bi-joint de G.

Théorème 4.9. *La décomposition bi-joint peut être calculée en temps linéaire.*

4.7 Graphes sans C_5, taureau, gem et co-gem induit

Pour la décomposition modulaire et la décomposition en coupes, nous connaissons les classes de graphes entièrement décomposables, c'est à dire les classes des graphes telles que tous les noeuds dans l'arbre de décomposition sont dégénérés. Dans cette section, on caractérisera la classe des graphes entièrement décomposables par la décomposition bi-joint. Cette classe peut être définie de deux façons. Le lemme suivant montre que ces deux conditions sont équivalentes.

Lemme 4.10. *Soit G un graphe. Les deux conditions suivantes sont équivalentes :*
(1) tout sous-graphe induit de G avec au moins 4 sommets a un bi-joint non trivial,
(2) tout noeud dans l'arbre de décomposition bi-joint de G est complet.

Démonstration. Supposons que tous les noeud dans l'arbre de décomposition bi-joint de G sont complets. Soit H un sous-graphe induit de G avec au moins 4 sommets. Alors tout noeud dans l'arbre T' de décomposition bi-joint de H est complet. Soit α un noeud de T'. Si le degré de α est au moins 4, alors la bi-partition induite par deux composantes de $T' - \alpha$ et de son complémentaire est un bi-joint, et n'est pas triviale. Sinon, il existe une composante C^i_α de taille au moins 2, car H a plus de 4 sommets. Alors $\{C^i_\alpha, V \setminus C^i_\alpha\}$ est un bi-joint de H, et n'est pas trivial.

Supposons qu'il y ait un noeud α dans l'arbre de décomposition bi-joint de G qui est premier. Alors, par le lemme 4.7, $BJC(\alpha)$ est premier, et est un sous-graphe induit de G. □

Le théorème suivant donne différentes caractérisations de la classe des graphes entièrement décomposables par la décomposition bi-joint.

Théorème 4.11. *Soit $G = (V, E)$ un graphe. Alors les conditions suivantes sont équivalentes :*

(1) G est complètement décomposable par la décomposition bi-joint.

(2) G est sans C_5, taureau, gem et co-gem induit.

(3) G peut être obtenu depuis un sommet par une séquence d'extensions de sommets par un jumeau et anti-jumeau.

(4) $\forall v \in V$, $\overline{G}^v$ est un cographe.

Démonstration. (1) $\iff$ (4) : Par le lemme 1.14 et le théorème 4.2, chaque noeud dans l'arbre de décomposition bi-joint de G est `dégénéré` si et seulement si chaque noeud dans l'arbre de décomposition modulaire de $\overline{G}^v$ est `dégénéré`. Ainsi G est complètement décomposable par la décomposition bi-joint si et seulement si $\overline{G}^v$ est un cographe.

(1) $\Rightarrow$ (3) : Par induction sur le nombre de sommets du graphe. Cela est immédiat si $|V| \leq 3$. Sinon, il existe un noeud α dans l'arbre de décomposition bi-joint T adjacent aux feuilles u et v. Ce noeud est `dégénéré`, ainsi $\big\{\{u, v\}, V \setminus \{u, v\}\big\}$ est un bi-joint de G. Les sommets u et v sont donc soit des jumeaux, soit des anti-jumeaux.

(3) $\Rightarrow$ (4) : Par induction sur le nombre de sommets du graphe. Soit G un graphe obtenu depuis un sommet isolé par une séquence d'extensions de jumeaux/anti-jumeaux, soit $u \in V(G)$, et G' le graphe obtenu depuis G par l'ajout d'un jumeau ou d'un anti-jumeau w à u. Soit $v \in V(G)$; par induction, $\overline{G}^v$ est un cographe. Si $u \neq v$, w est un vrai jumeau de u dans $\overline{G'}^v$, ainsi $\overline{G'}^v$ est un cographe par le théorème 2.8. Si $u = v$ et w est un vrai jumeau ou un faux anti-jumeau de u, w est un sommet dominant de $\overline{G'}^v$ (un sommet adjacent à tous les autres sommets du graphe). Si $u = v$ et w est un faux jumeau ou un vrai anti-jumeau de u, w est un sommet isolé de $\overline{G'}^v$ (un sommet de degré zéro). Dans tous les cas, $\overline{G'}^v$ est un cographe.

(2) $\iff$ (4) : Soit H un graphe et $v \in V(H)$. Alors $\overline{H}^v$ est un P_4 si et seulement si H est un C_5, un taureau, un gem ou un co-gem.

Soit G un graphe tel que pour tout $v \in V$, $\overline{G}^v$ est un cographe. Soit H un sous-graphe induit de G, et soit $v \in V(H)$. Alors $\overline{H}^v = \overline{G}^v[V(H) \setminus \{v\}]$ n'est pas un P_4, donc H n'est pas un C_5, un taureau, un gem ou un co-gem. Inversement, soit G un graphe sans C_5, taureau, gem et co-gem induit, et soit $v \in V$. Soit H' un sous-graphe de $\overline{G}^v$. $G[V(H) \cup \{v\}]$ n'est pas un C_5, un taureau, un gem ou un co-gem, donc $H' = \overline{G[V(H) \cup \{v\}]}^v$ n'est pas un P_4. □

La classe des graphes sans C_5, taureau, gem et co-gem apparaît déjà dans l'article de Alain Hertz [Her99]. Il y montre que cette classe de graphes est exactement la classe des graphes pour lesquels chaque permutation de Seidel est un graphe parfait. Il donne également une preuve pour l'équivalence entre les propriétés (2) et (4) du théorème 4.11, et un algorithme de reconnaissance de temps $O(n^2)$, utilisant l'algorithme de reconnaissance des cographes donné dans [CPS85]. En fait, cette classe peut être reconnue en temps linéaire. Il n'est pas nécessaire de calculer tout son arbre de décomposition bi-joint pour cela, il suffit d'utiliser un algorithme de reconnaissance des cographes de temps linéaire [BCH03, CPS85, HP05], sur le graphe $\overline{G}^v$, où v est un sommet de degré minimum.

Les deux-graphes complètement décomposables par la décomposition bi-joint sont exactement les deux-graphes correspondant à la classe de permutation de Seidel des graphes parfaits.

4.8 Relation avec la largeur de clique

Comme pour la décomposition modulaire et la décomposition en coupes, la largeur de clique d'un graphe est bornée si et seulement si la largeur de clique de chaque graphe caractéristique de la décomposition bi-joint est bornée.

Théorème 4.12. *Soit G un graphe. Alors* $\mathrm{cwd}(G) \leq 8 \times k$*, où k est la largeur de clique maximum des graphes caractéristiques des noeuds internes de la décomposition bi-joint de G.*

Il ne sera pas démontré immédiatement. Il s'agit d'un corollaire des théorèmes 6.11 et 6.6 du chapitre 6 qui parlera entre autre de la largeur modulaire.

Dans le cas particulier des graphes sans C_5, taureau, gem et co-gem induits, la largeur de clique est au plus 4. Cela sera également une conséquence immédiate d'observations du chapitre 6. En effet les graphes sans C_5, taureau, gem et co-gem induits ont une largeur modulaire d'au plus 2, et la largeur de clique est au plus 2 fois la largeur modulaire.

4.9 Graphes sans gem et co-gem induit

Nous allons voir dans cette section un théorème de décomposition par bi-joints des graphes sans gem et co-gem induit. Le théorème est similaire à ceux utilisant la décomposition modulaire ou la décomposition en coupes. On montre que tout graphe sans gem et co-gem induit, premier par la décomposition bi-joint, appartient à une certaine classe de graphes. (Dans notre cas, cette classe est finie.) Cela implique que tous les graphes caractéristiques dans l'arbre de décomposition bi-joint d'un graphe sans gem et co-gem induit est soit dégénéré, soit dans la classe définie.

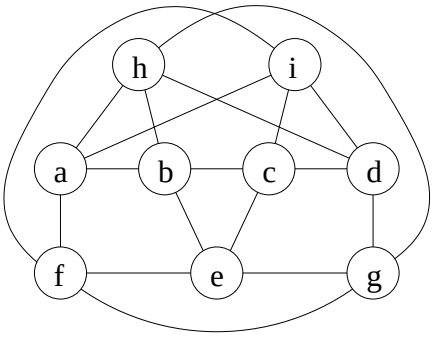


Fig. 4.6 : – Extension d'un taureau dans un graphe sans C_5, gem et co-gem induit.

Lemme 4.13. *Soit G un graphe sans C_5, gem et co-gem induit, premier par la décomposition bi-joint. Alors soit G est sans C_5, taureau, gem et co-gem induit, soit G est un sous-graphe induit du graphe donné en figure 4.6.*

Démonstration. Supposons que G n'est pas sans taureau induit. Soit a, b, c, d, e un taureau de G, tel que a, b, c, d soit un P_4 et e soit adjacent à b et à c. Pour $V' \subseteq \{a, b, c, d, e\}$, soit M'_V l'ensemble $\{v \in V \setminus \{a, b, c, d, e\} : N_G(v) \cap \{a, b, c, d, e\} = V'\}$. Alors $M_{\{b\}}$, $M_{\{a,b\}}$, $M_{\{c\}}$, $M_{\{b,c\}}$, $M_{\{a,b,d\}}$, $M_{\{c,d\}}$, $M_{\{a,c,d\}}$, $M_{\{a,e\}}$, $M_{\{a,c,e\}}$, $M_{\{b,c,e\}}$, $M_{\{a,b,c,e\}}$, $M_{\{d,e\}}$, $M_{\{b,d,e\}}$ et $M_{\{b,c,d,e\}}\}$ forment une partition de $V \setminus \{a, b, c, d, e\}$. Le tableau de la figure 4.7 présente les adjacences entre les classes de cette partition. Ces adjacences proviennent directement du fait que le graphe est sans C_5, gem et co-gem induit. Dans ce tableau, `0` signifie que les deux classes ne sont pas adjacentes, `1` signifie qu'elles sont adjacentes et $*$ signifie qu'elles doivent être à la fois adjacentes et non adjacentes, c'est à dire que l'une d'elle doit être vide.

	$\{a\}$	$M_{\{b\}}$	$M_{\{a,b\}}$	$\{d\}$	$M_{\{c\}}$	$M_{\{c,d\}}$	$\{e\}$	$M_{\{b,c\}}$	$M_{\{b,c,e\}}$	$\{b\}$	$M_{\{a,c,e\}}$	$M_{\{a,b,c,e\}}$	$\{c\}$	$M_{\{b,d,e\}}$	$M_{\{b,c,d,e\}}$	$M_{\{a,e\}}$	$M_{\{d,e\}}$	$M_{\{a,b,d\}}$	$M_{\{a,c,d\}}$
$\{a\}$		0	1	0	0	0	0	0	0	1	1	1	0	0	0	1	0	1	1
$M_{\{b\}}$	0		*	0	0	0	0	0	0	1	1	1	0	0	0	1	0	1	1
$M_{\{a,b\}}$	1	*		0	0	0	0	0	0	1	1	1	0	0	0	1	0	1	1
$\{d\}$	0	0	0		0	1	0	0	0	0	0	0	1	1	1	0	1	1	1
$M_{\{c\}}$	0	0	0	0		*	0	0	0	0	0	0	1	1	1	0	1	1	1
$M_{\{c,d\}}$	0	0	0	1	*		0	0	0	0	0	0	1	1	1	0	1	1	1
$\{e\}$	0	0	0	0	0	0		0	1	1	1	1	1	1	1	1	1	0	0
$M_{\{b,c\}}$	0	0	0	0	0	0	0		*	1	1	1	1	1	1	1	1	0	0
$M_{\{b,c,e\}}$	0	0	0	0	0	0	1	*		1	1	1	1	1	1	1	1	0	0
$\{b\}$	1	1	1	0	0	0	1	1	1		0	1	1	1	1	0	0	1	0
$M_{\{a,c,e\}}$	1	1	1	0	0	0	1	1	1	0		*	1	1	1	0	0	1	0
$M_{\{a,b,c,e\}}$	1	1	1	0	0	0	1	1	1	1	*		1	1	1	0	0	1	0
$\{c\}$	0	0	0	1	1	1	1	1	1	1	1	1		0	1	0	0	0	1
$M_{\{b,d,e\}}$	0	0	0	1	1	1	1	1	1	1	1	1	0		*	0	0	0	1
$M_{\{b,c,d,e\}}$	0	0	0	1	1	1	1	1	1	1	1	1	1	*		0	0	0	1
$M_{\{a,e\}}$	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0		1	0	1
$M_{\{d,e\}}$	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1		1	0
$M_{\{a,b,d\}}$	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	1		0
$M_{\{a,c,d\}}$	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	

Fig. 4.7 : – Adjacence entre les différentes classes de la partition dans le cas d'un graphe sans C_5, gem et co-gem possédant un taureau.

$M_{\{b\}} \cup M_{\{a,b\}} \cup \{a\}$ est un module de G. Comme G est premier par rapport à la décomposition bi-joint, tous les modules de taille au plus $n-2$ sont triviaux, et donc $M_{\{b\}} = \emptyset$ et $M_{\{a,b\}} = \emptyset$. Avec le même argument, $M_{\{c\}} = \emptyset$, $M_{\{c,d\}} = \emptyset$, $M_{\{b,c\}} = \emptyset$, $M_{\{b,c,e\}} = \emptyset$, $M_{\{a,c,e\}} = \emptyset$, $M_{\{a,b,c,e\}} = \emptyset$, $M_{\{b,d,e\}} = \emptyset$ et $M_{\{b,c,d,e\}} = \emptyset$. $M_{\{a,e\}}$, $M_{\{d,e\}}$, $M_{\{a,b,d\}}$ et $M_{\{a,c,d\}}$ sont des modules de G, alors $|M_{\{a,e\}}| \leq 1$, $|M_{\{d,e\}}| \leq 1$, $|M_{\{a,b,d\}}| \leq 1$ et $|M_{\{a,c,d\}}| \leq 1$. Ainsi, G est un graphe sous-graphe induit du graphe de la figure 4.6. □

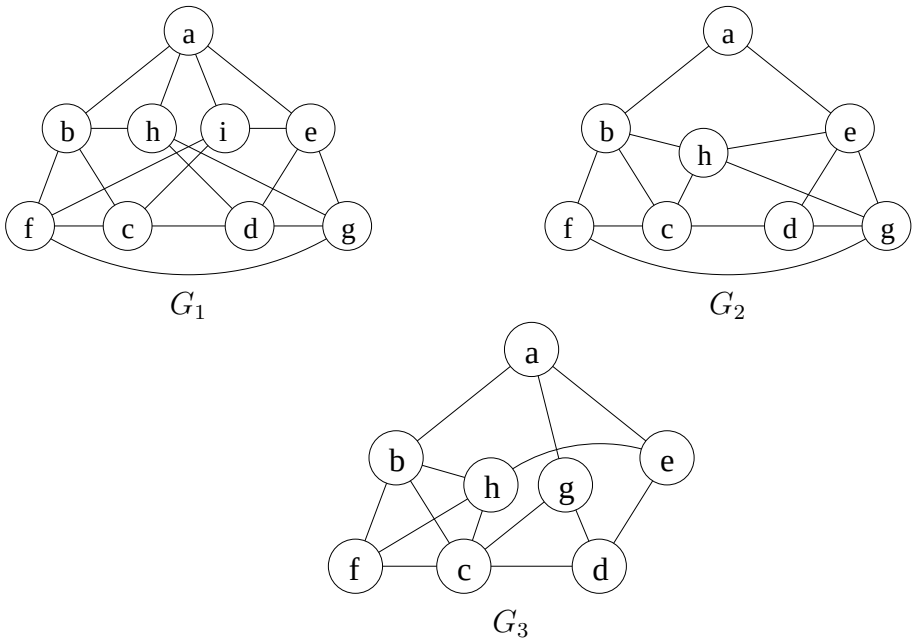


Fig. 4.8 : – Extensions d'un C_5 dans un graphe sans gem et co-gem induit.

Lemme 4.14. *Soit G un graphe sans gem et co-gem induit, premier par la décomposition bi-joint. Alors G est sans C_5 induit, ou G est le sous-graphe induit d'un graphe donné en figure 4.8.*

Démonstration. Supposons que G n'est pas sans C_5 induit. Soit v_0, v_1, v_2, v_3, v_4 un C_5 de G. Tous les indices sont données modulo 5.

Pour tout $i \in \{0, 1, 2, 3, 4\}$, soit :
- $X_i = \{v \in V \setminus C : N(v) \cap C = \{v_{i-2}, v_{i+2}\}\}$,
- $Y_i = \{v \in V \setminus C : N(v) \cap C = \{v_{i-2}, v_i, v_{i+2}\}\}$,
- $Z_i = \{v \in V \setminus C : N(v) \cap C = \{v_{i-1}, v_{i+1}\}\}$,
- $T_i = \{v \in V \setminus C : N(v) \cap C = \{v_{i-1}, v_i, v_{i+1}\}\}$.

On a $V = \bigcup_{i=0}^{4}(X_i \cup Y_i \cup Z_i \cup T_i \cup \{v_i\})$.

Le tableau de la figure 4.9 montre les adjacences entre les ensembles définis précédemment. Elles proviennent directement du fait que G est sans gem et co-gem induit.

On peut faire quelques observations simples. Les quatre premières observations sont une retranscription des précédentes adjacences. Les autres concernent trois ensembles ou plus.

(t1) Si $X_i \neq \emptyset$, alors $X_{i+1} = X_{i-1} = \emptyset$.

(t2) Si $Y_i \neq \emptyset$, alors $Y_{i+2} = Y_{i-2} = \emptyset$.

(t3) Si $X_i = Y_i = \emptyset$, alors $Z_i \cup T_i \cup \{v_i\}$ est un module de G.

(t4) S'il existe un i tel que pour tout $j \neq i$, $X_j = Y_j = \emptyset$, alors $X_i \cup Y_i \cup Z_i \cup T_i \cup \{v_i\}$ est un bi-joint de G.

	Z_{i+1}	T_{i+1}	X_{i+1}	Y_{i+1}	Z_{i+2}	T_{i+2}	X_{i+2}	Y_{i+2}	Z_{i+3}	T_{i+3}	X_{i+3}	Y_{i+3}	Z_{i+4}	T_{i+4}	X_{i+4}	Y_{i+4}
Z_i	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0
T_i	1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0
X_i	0	0	X	0	1	1	1	1	1	1	1	1	0	0	X	0
Y_i	0	0	0	0	1	1	1	X	1	1	1	X	0	0	0	0

Fig. 4.9 : – Adjacence entre les différentes classes de la partition dans le cas d'un graphe sans gem et co-gem possédant un C_5.

(i) Si $X_i \neq \emptyset$ et $X_{i+2} \neq \emptyset$, alors X_i est non adjacent à Z_i, T_i et Y_i, et X_{i+2} est non adjacent à Z_{i+2}, T_{i+2} et T_{i+2}.

(ii) Si $Y_i \neq \emptyset$ et $Y_{i+1} \neq \emptyset$, alors Y_i est adjacent à Z_i, T_i et X_i, et Y_{i+1} est adjacent à Z_{i+1}, T_{i+1} et X_{i+1}.

(iii) Si $X_i \neq \emptyset$ et $X_{i+2} \neq \emptyset$, alors $Y_{i+1} = \emptyset$.

(iv) Si $Y_i \neq \emptyset$ et $Y_{i+1} \neq \emptyset$, alors $X_{i-2} = \emptyset$.

(v) Si $X_i \neq \emptyset$ alors Y_{i+1} est adjacent à Z_{i+1} et T_{i+1}, et Y_{i-1} est adjacent à Z_{i-1} et T_{i-1}.

(vi) Si $Y_i \neq \emptyset$ alors X_{i+2} est non adjacent à Z_{i+2} et T_{i+2}, et Y_{i-2} est non adjacent à Z_{i-2} et T_{i-2}.

(vii) Si $Y_i \neq \emptyset$ alors X_{i+1} est non adjacent à Z_{i+1} et T_{i+1}, et X_{i-1} est non adjacent à Z_{i-1} et T_{i-1}.

(viii) Si $X_i \neq \emptyset$ alors Y_{i+2} est adjacent à Z_{i+2} et T_{i+2}, et Y_{i-2} est adjacent à Z_{i-2} et T_{i-2}.

(ix) Si $Y_i \neq \emptyset$ alors X_{i+1} est adjacent à Y_{i+1}, et X_{i-1} est adjacent à Y_{i-1}.

(x) Si $X_i \neq \emptyset$ alors Y_{i+2} est non adjacent à X_{i+2}, et Y_{i-2} est non adjacent à X_{i-2}.

Puisque G est premier par rapport à la décomposition bi-joint, on a :

(xi) Si $X_i = Y_i = \emptyset$, alors $Z_i = T_i = \emptyset$ (de l'observation t3).

(xii) S'il existe i tel que pour tout $j \neq i$, $X_j = Y_j = \emptyset$, alors G est un C_5 (de l'observation t4).

(xiii) Si $X_i \neq \emptyset$ et $X_{i+2} \neq \emptyset$, alors $|X_i| = |X_{i+2}| = 1$, car par l'observation i, X_i et X_{i+2} sont des modules.

(xiv) Si $Y_i \neq \emptyset$ et $Y_{i+1} \neq \emptyset$, alors $|Y_i| = |Y_{i+1}| = 1$, car par l'observation ii, Y_i et Y_{i+1} sont des modules.

On rappelle qu'il ne peut y avoir au plus que deux indices différents i et j tels que $X_i \neq \emptyset$ et $X_j \neq \emptyset$, et dans ce cas $j = i + 2$ ou $j = i - 2$. De même, il ne peut y avoir au plus que deux indices différents i et j tels que $Y_i \neq \emptyset$ et $Y_j \neq \emptyset$, et dans ce cas $j = i + 1$ ou $j = i - 1$. On étudie maintenant les différents cas.

Cas 1 : pour tout i, $X_i = \emptyset$.

Cas 1.1 : pour tout j, $Y_j = \emptyset$. Alors G est un C_5 (observation xii).

Cas 1.2 : $Y_j \neq \emptyset$ pour un unique j. Alors G possède un bi-joint non trivial (observation t4). Contradiction.

Cas 1.3 : $Y_j \neq \emptyset$ et $Y_{j+1} \neq \emptyset$. Alors, par l'observation xiv, Y_j, Y_{j+1} sont des modules. Pour tout $k \notin \{i, i+1\}$, $Z_k = T_k = \emptyset$ (observation xi), et pour tout $k \in \{i, i+1\}$, $Z_k \cup T_k \cup \{v_k\}$ est un module (observation i). Ainsi, G est le sous-graphe de G_1 induit par $\{a, b, c, d, e, h, i\}$.

Cas 2 : $X_i \neq \emptyset$ pour un unique i.

Cas 2.1 : $Y_j = \emptyset$ pour tout j. Alors G possède un bi-joint non trivial (observation t4). Contradiction.

Cas 2.2 : $Y_j \neq \emptyset$ pour un unique j.

Cas 2.2.1 : $i = j$. Alors G possède un bi-joint non trivial (observation t4). Contradiction.

Cas 2.2.2 : $i \in \{j+1, j-1\}$, par l'observation v, Y_j est adjacent à Z_j et T_j, et par l'observation vii, X_i est non adjacent à Z_i et T_j. Alors X_i, Y_j, et pour tout k, $Z_k \cup T_k \cup \{v_k\}$ sont des modules. Ainsi, G est le sous-graphe de G_1 induit par $\{a, b, c, d, e, f, h\}$.

Cas 2.2.3 : $i \in \{j+2, j-2\}$. Complémentaire du cas 2.2.2. G est le sous-graphe de G_1 induit par $\{a, b, c, d, e, f, i\}$.

Cas 2.3 : $Y_j \neq \emptyset$ et $Y_{j+1} \neq \emptyset$. Alors, par l'observation iv, $i \neq j-2$. Si $i \in \{j, j+1\}$, sans perte de généralité on assume $i = j$. Par les observations ii, v et xi, G est le graphe G_3.

Si $i \in \{j-1, j+2\}$, sans perte de généralité on assume $i = j-1$. Par l'observation iv, X_i et $Z_i \cup T_i \cup \{v_i\}$ sont des modules. Par les observations v et viii, Y_j, Y_{j+1}, $Z_j \cup T_j \cup \{v_j\}$ et $Z_{j+1} \cup T_{j+1} \cup \{v_{j+1}\}$ sont des modules. Donc G est le sous-graphe de G_1 induit par $\{a, b, c, d, e, f, h, i\}$.

Cas 3 : $X_i \neq \emptyset$ et $X_{i+2} \neq \emptyset$.

Cas 3.1 : $Y_j = \emptyset$ pour tout j. Complémentaire du cas 1.3. G est le sous-graphe de G_1 induit par $\{a, b, c, d, e, f, g\}$.

Cas 3.2 : $Y_j \neq \emptyset$ pour un unique j. Complémentaire du cas 2.3. G est soit le graphe G_2, soit le sous-graphe de G_1 induit par $\{a, b, c, d, e, f, g, h\}$.

Cas 3.3 : $Y_j \neq \emptyset$ et $Y_{j+1} \neq \emptyset$. Par les observations iii et iv, $j = i+3$. Par les observations v et vi, X_i X_{i+2}, Y_{i+3} et Y_{i+4} sont des modules. De plus pour tout k, $Z_k \cup T_k \cup \{v_k\}$ sont des modules. Donc G est le graphe G_1. □

Théorème 4.15. *Soit G un graphe sans gem et co-gem induit, et T son arbre de décomposition bi-joint. Alors les graphes caractéristiques des noeuds* `premiers` *de T sont des sous-graphes induits des graphes donnés en figures 4.6 et 4.8 (et donc ont au plus 9 sommets).*

Il est connu de [BLM04] que la largeur de clique des graphes sans gem et co-gem induit est bornée par 16. Le théorème de décomposition précédent nous donne une preuve alternative et plus courte pour affirmer que cette classe a une largeur de clique bornée.

Lemme 4.16. *La largeur de clique des graphes donnés en figure 4.6 et 4.8 est majoré par* 6.

Démonstration. L'expression suivante est une 6-expression pour le graphe de la figure 4.6.

$$t_0 = \eta_{4,5}\eta_{4,6}\eta_{5,6}\eta_{4,1}\eta_{5,3}\eta_{6,2}((a)_4 \oplus (b)_5 \oplus (h)_6 \oplus \rho_{6\to1}\rho_{5\to3}\rho_{4\to2}(t'_0))$$

où

$$t'_0 = \eta_{1,2}\eta_{1,2}\eta_{2,3}\eta_{2,4}\eta_{3,5}\eta_{4,5}\eta_{5,6}\eta_{4,6}((i)_1 \oplus (d)_2 \oplus (c)_3 \oplus (g)_4 \oplus (e)_5 \oplus (f)_6).$$

Les expression t_1, t_2 et t_3 sont des expressions pour les graphes G_1, G_2 et G_3 de la figure 4.8.

$$\begin{aligned} t_1 = &\quad \eta_{4,5}\eta_{4,6}\eta_{5,6}\eta_{4,1}\eta_{5,2}\eta_{6,3}((a)_4 \oplus (i)_5 \oplus (e)_6 \oplus \rho_{6\to 3}\rho_{5\to 2}\rho_{4\to 1}(t'_0)) \\ t_2 = &\quad \eta_{6,3}\eta_{6,4}\eta_{6,5}((g)_6 \oplus \eta_{3,2}\rho_{6\to 3}\eta_{6,5}((h)_6 \oplus (f)_3 \oplus \rho_{3\to 2}(t_{C_5}))) \\ t_3 = &\quad \eta_{6,3}\eta_{6,2}\eta_{6,1}(g)_6 \oplus \eta_{4,3}\eta_{4,2}\rho_{6\to 4}\eta_{4,6}\eta_{6,5}((f)_4 \oplus (h)_6 \oplus \rho_{4\to 1}(t_{C_5})) \end{aligned}$$

où

$$t'_1 = \eta_{3,6}\eta_{3,4}\eta_{3,5}\eta_{2,6}\eta_{2,5}\eta_{1,5}\eta_{1,4}\eta_{1,2}((b)_1 \oplus (f)_2 \oplus (d)_3 \oplus (h)_4 \oplus (c)_5 \oplus (g)_6)$$

et

$$t_{C_5} = \eta_{5,1}\eta_{4,5}\eta_{3,4}\eta_{2,3}\eta_{1,2}((a)_1 \oplus (b)_2 \oplus (c)_3 \oplus (d)_4 \oplus (e)_5).$$

□

Le lemme 4.16 et le théorème 4.12 nous donne immédiatement :

Corollaire 4.17. *La largeur de clique des graphes sans gem et co-gem induit est bornée par* 48.

On obtient une meilleure borne si on passe directement par la largeur modulaire. La largeur modulaire des graphes en figures 4.6 et 4.8 est bornée par 3. En utilisant le théorème 6.11, on voit que la largeur modulaire des graphes sans gem et co-gem induits est bornée par 12, et donc, par le théorème 6.6, leur largeur de clique est bornée par 24.

4.10 Conclusion

Ainsi s'achève notre étude de cette nouvelle décomposition. La décomposition bi-joint est donc une décomposition dans la lignée de la décomposition modulaire et de la décomposition en coupes. On a l'unicité de l'arbre représentant les bi-joints, et presque l'unicité des graphes caractéristiques (on a l'unicité *modulo* la permutation de Seidel). On dispose également d'un algorithme linéaire pour calculer cette décomposition, et plusieurs caractérisations des graphes complètement décomposables. Cette décomposition peut être généralisée aux deux-graphes, et dans ce cas on obtient l'unicité des deux-graphes caractéristiques.

Une question ouverte est de savoir si on peut généraliser cette décomposition à d'autres structures, comme les graphes orientés, les k-structures ou les hypergraphes.

En application, on donne un théorème de décomposition des graphes sans gem et co-gem induit, ce qui nous donne une preuve alternative que leur largeur de clique est bornée. Une autre question serait de trouver d'autres classes joliment décomposables par cette décomposition, et de trouver d'autres problèmes naturels qui se comportent bien par la décomposition bi-joint.

Chapitre 5

Décomposition en cliques

De nombreux problèmes ont été montrés polynomiaux sur les classes de graphes de largeur arborescente bornée. C'est le cas de tous les problèmes qui peuvent s'écrire en logique monadique du second ordre [ALS91]. Les problèmes ENSEMBLE STABLE, ENSEMBLE DOMINANT, CIRCUIT HAMILTONIEN en font partie. Une autre classe de problèmes de partitionnement de l'ensemble des sommets est également polynomiale sur les graphes de largeur arborescente bornée [TP97]. On peut noter que cela est également le cas pour le problème ISOMORPHISME DE GRAPHE [Bod90], pour décider si un graphe a une largeur arborescente bornée par un k fixé [Bod96] et pour décider si un graphe a une largeur de clique d'au plus k [EGW03]. Une vue générale des utilisations de la largeur arborescente peut être trouvée dans [Bod93].

Tous les problèmes qui peuvent s'écrire en logique monadique du second ordre avec quantification sur les sommets peuvent être résolus en temps linéaire sur les graphes de largeur de clique bornée [CMR00]. Tous les problèmes d'une classe de problèmes de partitionnement (une sous-classe de celle donnée dans [TP97]) peut être résolus en temps polynomial [GC03], et [EGW01] donne une liste de problèmes polynomiaux.

Dans ce chapitre, on montrera qu'une nouvelle classe de problèmes de partitionnement de l'ensemble des sommets est polynomiale sur les graphes de largeur de clique bornée. On en déduira qu'une autre classe de problèmes, plus grande, est polynomiale sur les graphes de largeur arborescente bornée.

5.1 Coloration sans H induit

Espelage, Gurski et Wanke [EGW01] donnent une liste de problèmes qui peuvent être résolus en temps polynomial sur les graphes de largeur de clique bornée. On peut synthétiser leur travail de la façon suivante : on recherche un ensemble de propriétés $\mathcal{P}$ tel que les propriétés dans $\mathcal{P}$ de $\eta(G)$, $\rho(G)$ et $G \oplus H$ peuvent être décidées depuis les propriétés dans $\mathcal{P}$ de G et H.

Le problème suivant peut être résolu en temps polynomial par cette méthode.

Coloration sans H induit (H fixé)
INSTANCE : un graphe $G = (V, E)$.
SORTIE : le plus petit $r \in \mathbb{N}$ tel que V soit partitionnable en r ensembles induisant des graphes sans H induit ?

Théorème 5.1. *Soit H un graphe fixé, et $k \in \mathbb{N}$ fixé. Alors il existe un algorithme polynomial pour le problème* COLORATION SANS H INDUIT *sur les graphes de largeur de clique au plus k (si la k-expression est donnée avec le graphe).*

Démonstration. L'algorithme utilise la méthode de [EGW01].

Soit G' un graphe étiqueté. On note par $\mathcal{B}^q(G')$ l'ensemble des sous-graphes induits de G' (éventuellement vides) avec au plus q sommets.

Soit $q = |V(H)|$. L'algorithme calcule récursivement, pour chaque sous-graphe apparaissant dans la décomposition du graphe, l'ensemble de multi-ensembles suivant :

$$\mathcal{S}(G) = \left\{ \left\langle \mathcal{B}^q(G[V_1]), \ldots, \mathcal{B}^q(G[V_r]) \right\rangle : V_1, \ldots V_r \text{ est une partition de V(G)} \right\}.$$

Les classes de la partition $V_1, \ldots, V_r$ sont éventuellement vides.

Au final, V est partitionnable en r sous-ensembles induisant un graphe sans H induit si et seulement si il existe un multi-ensemble $\langle B_1, \ldots, B_r \rangle$ de $\mathcal{S}(G)$ tel que pour tout $i \in \{1, \ldots r\}$, B_i n'est pas l'ensemble composé uniquement du graphe vide, et pour tout $H' \in B_i$, unlab(H') n'est pas isomorphe à H.

On peut remarquer qu'il y a un nombre fini de graphes étiquetés d'au plus q sommets. Soit $l(q, k)$ le nombre de graphes k-étiquetés d'au plus q sommets. Il y a au plus $2^{l(q,k)}$ ensembles possibles pour $\mathcal{B}^q(G')$, et donc $(r+1)^{2^{l(q,k)}}$ multi-ensembles possibles (pour chaque ensemble, la multiplicité est comprise entre 0 et r). Dans tour les cas $r \leq |V(G)|$. On voit facilement que $l(q, k)$ est borné par $k^q \times 2^{\frac{q \times (q-1)}{2}}$.

Les ensembles $\mathcal{S}$ peuvent être calculés récursivement. Pour les opérations $\cdot$, η et ρ, on a immédiatement :

- $\mathcal{S}(\cdot_i) = \left\{ \langle \{\emptyset, \cdot_i\}, \{\emptyset\}, \ldots, \{\emptyset\} \rangle \right\}$ (où $\emptyset$ représente le graphe étiqueté vide). En effet il n'existe qu'une partition des sommets du graphe $\cdot_i$.
- $\mathcal{S}(\rho_{i \to j}(G)) = \left\{ \left\langle \left\{ \rho_{i \to j}(H') : H' \in B_i \right\} \right\rangle_{i \in \{1,\ldots,r\}} : \left\langle B_i \right\rangle_{i \in \{1,\ldots,r\}} \in \mathcal{S}(G) \right\}$.
- $\mathcal{S}(\eta_{i,j}(G)) = \left\{ \left\langle \left\{ \eta_{i,j}(H') : H' \in B_i \right\} \right\rangle_{i \in \{1,\ldots,r\}} : \left\langle B_i \right\rangle_{i \in \{1,\ldots,r\}} \in \mathcal{S}(G) \right\}$.

Pour le dernier cas, étant donné $\mathcal{S}(G_1)$ et $\mathcal{S}(G_2)$ on doit calculer $\mathcal{S}(G_1 \oplus G_2)$. Toutes les partitions $\mathcal{P} = \{V_1, \ldots V_r\}$ de $V(G_1 \oplus G_2)$ proviennent d'une partition $\{V \cap V(G_1) : V \in \mathcal{P}_1\}$ de $V(G_1)$ et d'une partition $\{V \cap V(G_2) : V \in \mathcal{P}_2\}$ de $V(G_2)$.

Il peut y avoir un nombre exponentiel de façons de combiner une partition $\mathcal{P}_1$ de $V(G_1)$ et $\mathcal{P}_2$ de $V(G_2)$, mais il nous suffit de connaître les ensembles $\mathcal{B}^q(G_i[V'])$, avec $i \in \{1, 2\}$, pour les différentes classes V' des partitions $\mathcal{P}_i$ pour pouvoir calculer les ensembles $\mathcal{B}^q(G[V'])$ pour les classes V' de la partition de $V(G)$.

En effet, on voit facilement que

$$\mathcal{B}^q(G_1 \oplus G_2) = \{H_1 \oplus H_2 : H_1 \in \mathcal{B}^q(G_1), H_2 \in \mathcal{B}^q(G_2) \text{ et } |V(H_1)| + |V(H_2)| \leq q\}.$$

Soit B_1 et B_2 deux ensembles de graphes étiquetés d'au plus q sommets. On définit par $B_1 \oplus B_2$ l'ensemble $\{H_1 \oplus H_2 : H_1 \in B_1, H_2 \in B_2 \text{ et } |V(H_1)| + |V(H_2)| \leq q\}$. On a donc $\mathcal{B}^q(G_1 \oplus G_2) = \mathcal{B}^q(G_1) \oplus \mathcal{B}^q(G_2)$.

$\mathcal{S}(G_1 \oplus G_2)$ peut être calculé de la manière suivante : on commence avec l'ensemble $\mathcal{D} = \{\langle\rangle\} \times \mathcal{S}(G_1) \times \mathcal{S}(G_2)$, puis on étend $\mathcal{D}$ à l'ensemble de tous les triplets qui peuvent être obtenus d'un triplet $(\mathcal{M}, \mathcal{M}_1, \mathcal{M}_2) \in \mathcal{D}$ en supprimant un ensemble B_1 de $\mathcal{M}_1$ et un ensemble B_2 de $\mathcal{M}_2$, et en rajoutant $B_1 \oplus B_2$ à $\mathcal{M}$. On voit que quand $\mathcal{D}$ ne pourra plus être étendu, on aura $\mathcal{S}(G_1 \oplus G_2) = \{\mathcal{M} : (\mathcal{M}, \langle\rangle, \langle\rangle) \in \mathcal{D}$. Comme $\mathcal{M}$, $\mathcal{M}_1$ et $\mathcal{M}_2$ appartiennent à un univers d'au plus $(r+1)^{2^{l(q,k)}}$ éléments, $\mathcal{D}$ a au plus $(r+1)^{3\times 2^{l(q,k)}}$ éléments.

Une k-expression a $O(n)$ opérations $\oplus$, chaque opération prends un temps $O((n+1)^{3\times 2^{l(q,k)}})$, étant donné que l'opération $\oplus$ sur les ensembles se fait en temps constant, quand q et k sont fixés. L'opération pour $\cdot$ se fait en temps constant et celles pour η et ρ se font en temps $O((n+1)^{2^{l(q,k)}})$. Au final, le temps de l'algorithme est polynomial. □

On voit que le problème demandant de partitionner l'ensemble des sommets en $V_1, \ldots, V_r$ tel que chaque V_i induit un graphe sans $H_1, \ldots H_k$ induit est également polynomial, la preuve n'est qu'une légère modification de la preuve précédente. Cette classe de problèmes est exactement la classe des problèmes tels qu'il existe une formule φ sans quantificateur et de variables libres $x_1, \ldots x_l$, et où l'on demande de partitionner V en $V_1, \ldots, V_r$ tel que pour tout $i \in \{1, \ldots, r\}$ on ait

$$\langle G, V_i\rangle \models \forall x_1, \ldots, x_l \in V_i, \varphi(x_1, \ldots, x_l).$$

On verra que l'on peut étendre ce résultat pour n'importe quelle formule en logique monadique du second ordre.

5.2 Problèmes MSOL

Certaines définitions sont données formellement en annexe de ce chapitre. Plus de détails sur la théorie des modèles finis peuvent être trouvé dans [EF95].

5.2.1 Vocabulaire et structures

Un **vocabulaire** est un ensemble fini de **symboles de relations**[1]. Chaque symbole de relation possède une **arité**, un entier naturel non nul. Une **relation** d'arité $k \in \mathbb{N}^*$ sur un ensemble A (appelée également **relation** k**-aire** sur A), est un sous-ensemble de A^k.

[1] D'autres définitions autorisent des constantes et des fonctions dans les vocabulaires, mais elles peuvent être représentés par des relations.

Soit τ le vocabulaire avec comme symboles des relations $R_1, \ldots, R_k$, d'arités $a_1, \ldots, a_k$ respectivement. Une **structure** $\mathcal{A}$ du vocabulaire τ (appelé également une **τ-structure**) est constituée d'un ensemble A, le **domaine**, et d'une relation a_i-aire $R_i^{\mathcal{A}}$ sur A pour chaque $i \in \{1, \ldots, k\}$. Elle sera notée $\langle A, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}} \rangle$.

Quand il n'y aura pas d'ambiguïté sur la structure $\mathcal{A}$, on écrira parfois, par abus de notation, R à la place de $R^{\mathcal{A}}$. Par la suite, toutes les structures seront supposées finies (c'est-à-dire de domaine fini).

Deux possibilités peuvent être envisagées pour représenter un graphe. Dans la première, les éléments sont les sommets du graphe, et on dispose d'une relation binaire qui est vraie si les deux sommets sont adjacents. Dans la seconde, les éléments sont les sommets et les arêtes du graphe. On doit donc ajouter un prédicat pour savoir si un élément est un sommet, ou une arête. On a toujours une relation binaire, mais cette fois ci la relation d'incidence entre les sommets et les arêtes.

On désigne par τ_1 le vocabulaire possédant un seul symbole de relation Adj d'arité 2. On désigne par τ_2 le vocabulaire possédant un symbole de relation Adj_2 d'arité 2 et un symbole de relation P_V d'arité 1.

Soit $G = (V, E)$ un graphe. On notera $\tau_1(G)$ la τ_1-structure $\langle V, \mathrm{Adj} \rangle$, où Adj désigne l'adjacence entre les sommets, et $\tau_2(G)$ la τ_2-structure $\langle V \cup E, P_V, \mathrm{Adj}_2 \rangle$ où P_V est un prédicat unaire vrai si l'élément est dans V, et Adj_2 est une relation binaire vraie si un élément est un sommet, l'autre est une arête et le sommet est incident à l'arête. On note que la structure $\tau_2(G)$ restreinte de P_V est isomorphe à $\tau_1(G)$, où $I(G)$ est le graphe d'incidence de G.

D'autres objets mathématiques peuvent être représentés par des structures, comme les graphes orientés (la relation n'est pas symétrique) et les hypergraphes (par leur graphe d'incidence).

5.2.2 Logique monadique du second ordre

En logique du premier ordre, les quantificateurs portent sur les éléments du domaine. En logique du second ordre, on peut quantifier également sur les relations. La logique monadique du second ordre (MSOL pour *monadic second order logic*) est une restriction de la logique du second ordre, permettant de quantifier uniquement sur les relations unaires. Dans la suite, sans indications supplémentaires, les formules seront supposées être monadiques du second ordre.

Une formule est **atomique** si elle est de la forme $x_1 = x_2$ ou $R(x_1, \ldots, x_a)$, où $x_1, \ldots, x_a$ sont des variables du premier ordre, et R est un symbole de relation d'arité a du vocabulaire, ou une variable du second ordre.

Une **variable libre** d'une formule est une variable qui n'est pas dans la portée d'un quantificateur. Une formule est **close** si elle n'a aucune variable libre.

Une **assignation** pour un ensemble de variables $\mathcal{X}$ et une structure $\mathcal{A}$ est une fonction qui à chaque variable libre du premier ordre de $\mathcal{X}$ associe un élément du domaine de $\mathcal{A}$, et à chaque variable libre du deuxième ordre de $\mathcal{X}$ associe un sous-ensemble du domaine

de $\mathcal{A}$. Une τ-structure $\mathcal{A}$ avec une assignation z des variables $X_1, \ldots, X_l$ est assimilable à une $(\tau \cup \{X_1, \ldots X_l\})$-structure.

Soit φ une formule. Une τ-structure $\mathcal{A}$ avec une assignation des variables libres de φ est un **modèle** de φ si φ est satisfaite dans la structure $\mathcal{A}$ avec l'assignation z des variables (la définition formelle est donnée en annexe). On le notera $\langle \mathcal{A}, z \rangle \models \varphi$, ou $\mathcal{A} \models \varphi$ si φ est close.

La **profondeur de quantification** d'une formule φ, notée $\mathrm{qr}(\varphi)$, est définie récursivement. Si φ n'a pas de quantificateur, alors $\mathrm{qr}(\varphi) = 0$. Si $\varphi = \neg\psi$, alors $\mathrm{qr}(\varphi) = \mathrm{qr}(\psi)$. Si $\varphi = \psi_1 \wedge \psi_2$, ou $\varphi = \psi_1 \vee \psi_2$, alors $\mathrm{qr}(\varphi) = \max(\mathrm{qr}(\psi_1), \mathrm{qr}(\psi_2))$. Enfin si $\varphi = \forall X \psi$ ou $\varphi = \exists X \psi$ (où X est une variable du premier ou du second ordre), $\mathrm{qr}(\varphi) = 1 + \mathrm{qr}(\psi)$.

Soit τ un vocabulaire. On note par $\mathrm{MSOL}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ l'ensemble des formules monadiques du second ordre du vocabulaire τ, de profondeur de quantification au plus q, avec des variables du premier ordre dans $\mathfrak{X}$ et les variables du second ordre dans $\mathfrak{Y}$. On note par $\mathrm{MSOL}^q(\tau)$ l'ensemble des formules monadiques du second ordre closes du vocabulaire (c'est-à-dire $\mathrm{MSOL}^q(\tau, \emptyset, \emptyset)$).

La **théorie** de $\langle \mathcal{A}, z \rangle$, noté $\mathrm{Th}(\langle \mathcal{A}, z \rangle)$, est l'ensemble des formules dont $\langle \mathcal{A}, z \rangle$ est un modèle, c'est-à-dire :

$$\mathrm{Th}(\langle \mathcal{A}, z \rangle) = \{\varphi : \varphi \in \mathrm{MSOL}(\tau) \text{ et } \langle \mathcal{A}, z \rangle \models \varphi\}.$$

De même

$$\mathrm{Th}(\mathcal{A}) = \{\varphi : \varphi \in \mathrm{MSOL}(\tau) \text{ et } \mathcal{A} \models \varphi\}.$$

On notera $\mathrm{Th}^q(\langle \mathcal{A}, z \rangle)$, où $k \in \mathbb{N}$, l'ensemble des formules de profondeur de quantification au plus q dont $\langle \mathcal{A}, z \rangle$ est un modèle, c'est-à-dire $\mathrm{Th}(\langle \mathcal{A}, z \rangle) \cap \mathrm{MSOL}^q(\tau)$. De même $\mathrm{Th}^q(\mathcal{A}) = \mathrm{Th}(\mathcal{A}) \cap \mathrm{MSOL}^q(\tau)$.

Certains algorithmes manipulent des sous-ensembles de $\mathrm{MSOL}^q(\tau)$. Pour cela, il faudrait que cet ensemble soit fini, ce qui n'est pas le cas directement, mais qui l'est modulo l'équivalence des formules.

Malheureusement, l'équivalence entre les formules (même du premier ordre) est indécidable. Par contre, on peut trouver une relation d'équivalence (c'est-à-dire réflexive, symétrique et transitive), notée $\approx$, entre les formules telle que si $\varphi \approx \psi$, alors φ est équivalente à ψ, et pour un $q \in \mathbb{N}$ fixé, il y ait un nombre fini de formules non équivalentes par $\approx$, de profondeur de quantification au plus q. La définition de cette relation, ainsi que la borne supérieure du nombre de classes d'équivalence pour les formules de profondeur de quantification au plus q sont données en annexe.

Dans la suite, on manipulera les formules modulo $\approx$, et donc on supposera que les ensembles $\mathrm{MSOL}^q(\tau)$ sont finis, pour q et τ fixés.

5.2.3 Propriétés MSOL

Pour simplifier les notations, si X est une relation unaire, on écrira $x \in X$ pour $X(x)$. On écrira $\forall x \in X, \varphi$ pour $\forall x, x \in X \Rightarrow \varphi$ et $\exists x \in X, \varphi$ pour $\exists x, x \in X \wedge \varphi$.

Définition 5.2.1. Une propriété sur les τ-structures P est une **propriété** MSOL s'il existe une formule monadique du second ordre sur le vocabulaire φ telle que P est vraie sur la structure $\mathcal{A}$ si et seulement si $\mathcal{A} \models \varphi$

On peut appliquer la définition sur les structures τ_1 et τ_2 des graphes. On obtient :

Définition 5.2.2. Une propriété sur les graphes est MSOL_1 s'il existe une formule en logique monadique du second ordre φ sur le langage τ_1 telle qu'un graphe ait la propriété si et seulement si $\tau_1(G) \models \varphi$.

Une propriété sur les graphes est MSOL_2 s'il existe une formule monadique du second ordre φ sur le langage τ_2 telle qu'un graphe ait la propriété si et seulement si $\tau_2(G) \models \varphi$.

Par exemple, la propriété « G est connexe » est MSOL_1, avec

$$\varphi \triangleq \forall X, (\forall x \in X, \forall y, \mathrm{Adj}(x, y) \Rightarrow X(y)) \Rightarrow \forall x, x \in X.$$

La propriété « G peut être colorié avec 3 couleurs » l'est également :

$$\varphi \triangleq \exists X \exists Y \exists Z, \mathrm{Stable}(X) \wedge \mathrm{Stable}(Y) \wedge \mathrm{Stable}(Z) \wedge \mathrm{Partition}(X, Y, Z)$$

avec

$$\begin{aligned}\mathrm{Partition}(X, Y, Z) \triangleq &\forall x, (x \in X \vee x \in Y \vee x \in Z) \wedge \neg(x \in X \wedge x \in Y) \wedge \\ &\neg(x \in X \wedge x \in Z) \wedge \neg(x \in Y \wedge x \in Z)\end{aligned}$$

et

$$\mathrm{Stable}(X) \triangleq \forall x \in X, \forall y \in X, x \neq y \Rightarrow \neg\, \mathrm{Adj}(x, y).$$

La propriété « G possède un circuit Hamiltonien » est MSOL_2. On cherche un sous-ensemble d'arêtes X tel que (V, X) soit 2-régulier et connexe.

$$\begin{aligned}\varphi \triangleq\ & \exists X \subseteq V, \\ & [[\forall x \in V, \exists e, f \in X, (e \neq f \wedge \mathrm{Adj}_2(x, e) \wedge \mathrm{Adj}_2(x, f) \wedge \\ & \qquad \forall g \in X, \mathrm{Adj}_2(x, g) \Rightarrow (g = e \vee g = f))] \wedge \\ & [\forall V' \subseteq V, [\forall u \in V', \forall v \in V, (\exists e \in X, (\mathrm{Adj}_2(e, u) \wedge \mathrm{Adj}_2(e, v))) \Rightarrow y \in V') \\ & \qquad \Rightarrow V \subseteq V]]].\end{aligned}$$

On voit facilement qu'une propriété MSOL_1 peut être traduite en une propriété MSOL_2. Le contraire n'est pas toujours vrai : la propriété « G possède un circuit Hamiltonien » n'est pas MSOL_1 [Cou94].

Certains problèmes NP-complets sont donc MSOL_1 ou MSOL_2. Par exemple, les problèmes k-COLORATION et PARTITION EN k ENSEMBLES DOMINANTS (pour k fixé) sont MSOL_1. Les problèmes CIRCUIT HAMILTONIEN et SOUS-GRAPHE CUBIQUE sont MSOL_2.

Les problèmes LinEMSOL sont un équivalent des problèmes MSOL pour les problèmes d'optimisations.

Définition 5.2.3. [CMR00] Un problème d'optimisation sur la classe des τ-structures est un problème LinEMSOL s'il peut être exprimé sous la forme suivante :

Étant donné une τ-structure $\mathcal{A}$, et m fonctions de poids $f_1, \ldots, f_m$ définies sur A, trouver un assignement z des variables libres de ϕ tel que

$$\sum_{\substack{1\leq i\leq i\\ 1\leq j\leq m}} a_{ij}|z(X_i)|_j = \text{opt}\left\{\sum_{\substack{1\leq i\leq i\\ 1\leq j\leq m}} a_{ij}|z'(X_i)|_j : \langle \mathcal{A}, z'\rangle \models \varphi(X_1, \ldots, X_l)\right\}$$

où φ est une formule ayant comme variables d'ensembles libres $X_1, \ldots X_l$, opt est soit min ou max, a_{ij} est un entier pour tout $1 \leq i \leq l$ et $1 \leq j \leq m$, et $|z(X_i)|_j = \sum_{a\in z(X_i)} f_j(a)$.

On dira qu'un problème sur les graphes est un problème LinEMSOL_1 s'il s'agit d'un problème LinEMSOL sur la structure τ_1 du graphe, et qu'il s'agit d'un problème LinEMSOL_1 s'il s'agit d'un problème LinEMSOL sur la structure τ_2 du graphe. Les problèmes ENSEMBLE STABLE, ENSEMBLE DOMINANT, FEEDBACK VERTEX SET sont des problèmes LinEMSOL_1. Par contre le problème NOMBRE CHROMATIQUE ne l'est pas [Lau93].

Théorème 5.2. *[ALS91, Cou92] Soit $k \in \mathbb{N}$ et Π un problème* LinEMSOL_2 *fixé. Alors Π peut être résolu en temps linéaire sur les graphes de largeur arborescente bornée par k (si la décomposition arborescente est donnée).*

Théorème 5.3. *[CMR00] Soit $k \in \mathbb{N}$ et Π un problème* LinEMSOL_1 *fixé. Alors Π peut être résolu en temps linéaire sur les graphes de largeur de clique bornée par k (si la k-expression du graphe est donnée).*

5.3 Partition MSOL

Dans cette section, on parlera d'une nouvelle classe de problèmes. Il s'agit des problèmes de partition des sommets d'un graphe (ou plus généralement le domaine d'une structure) en $\{V_1, \ldots, V_r\}$ tel que chaque classe de la partition ait une propriété MSOL.

Définition 5.3.1. Un problème Π est un **problème de partitionnement** MSOL s'il existe une formule τ monadique du second ordre φ avec une variable d'ensemble libre X telle que Π peut être exprimé sous la forme suivante :

Étant donné une τ-structure $\mathcal{A}$ et un entier r, le domaine de $\mathcal{A}$ peut-il être partitionné en $\{A_1, \ldots A_r\}$ de façon que

$$\forall i \in \{1, 2, \ldots r\}, \langle \mathcal{A}, A_i\rangle \models \varphi(A_i)\,?$$

On dit qu'un problème de graphes est un problème de partitionnement MSOL_1 s'il s'agit d'un problème de partitionnement MSOL sur la structure τ_1 du graphe, et un problème de partitionnement MSOL_2 s'il s'agit d'un problème de partitionnement MSOL sur la structure τ_2 du graphe.

Par exemple, les problèmes COLORATION, COLORATION SANS H INDUIT, NOMBRE DOMATIQUE, PARTITION EN GRAPHE AVEC UNE PROPRIÉTÉ Π (où Π est une propriété MSOL) et PARTITION EN GRAPHE PARFAITS sont des problèmes de partitionnement MSOL_1. Les problèmes COLORATION DES ARÊTES et PARTITIONNEMENT EN GRAPHES HAMILTONIENS sont des problèmes de partitionnement MSOL_2.

On peut remarquer que si on fixe r, le problème est un problème MSOL, avec comme formule $\psi = \mathrm{Partition}(X_1, \ldots, X_r) \wedge \varphi(X_1) \wedge \ldots \wedge \varphi(X_r)$.

Pour résoudre en temps polynomial un problème de partitionnement, il ne suffit pas d'utiliser l'algorithme de Courcelle, Makowski et Rotics pour les problèmes MSOL pour r entre 1 et $|V|$, car la constante du temps d'exécution de l'algorithme dépend de la formule. Par exemple, k-ACHROMATIC NUMBER, pour k fixé, est un problème MSOL_1, mais ACHROMATIQUE NUMBER est NP-complet sur les cographes [Bod89].

5.3.1 Largeur de clique d'une structure

La largeur de clique peut tout naturellement être étendue à toutes les structures. Pour cela on a besoin d'une opération η^R pour chaque symbole de relation R. On a toujours une opération ρ pour le re-étiquetage, une opération $\oplus$ pour l'union, et $\cdot_i$ représente la structure d'un élément étiqueté i. La définition est donnée en annexe.

Il est facile à voir que la largeur de clique d'un graphe G est égale à la largeur de clique de la τ_1-structure $\tau_1(G)$. En effet, comme $\mathrm{Adj}(v, v)$ est faux pour tout v du domaine, on ne peut pas avoir l'opération $\eta^{\mathrm{Adj}}_{(i,i)}$ (pour un i dans $\{1, \ldots, k\}$) dans la construction de la structure $\tau_1(G)$. De plus comme la relation Adj est symétrique, on peut toujours trouver une expression pour $\tau_1(G)$ telle que l'opération $\eta^{\mathrm{Adj}}_{(i,j)}$ soit suivie de l'opération $\eta^{\mathrm{Adj}}_{(j,i)}$. Une k-expression pour G proviendrait directement de la k-expression de $\tau_1(G)$.

5.3.2 Théorème de Feferman-Vaught

Le théorème a initialement été montré en logique du premier ordre, pour les unions de structures et les produits de structures, par Feferman et Vaught [FV59]. Puis le résultat pour les unions de structures a été étendu à la logique monadique du second ordre [Läu68, She75]. Une preuve constructive peut être trouvée dans [Mak04].

Théorème 5.4. *[FV59, Läu68, She75] Soit τ un vocabulaire et soit $\mathcal{A}$ et $\mathcal{A}'$ des τ-structures. Soit $q \in \mathbb{N}$ et $\varphi \in \mathrm{MSOL}^q(\tau)$. Alors il existe $l \in \mathbb{N}$, une formule booléenne $B : \{0,1\}^{2l} \to \{0,1\}$, et pour tout $i \in \{1, \ldots, l\}$, $\varphi_i, \varphi'_i \in \mathrm{MSOL}^q(\tau)$ tels que*

$$\mathcal{A} \bigsqcup \mathcal{A}' \models \varphi \quad \textit{si et seulement si} \quad B(a_1, \ldots, a_l, a'_1, \ldots, a'_l) = 1$$

où $a_i = 1$ si et seulement si $\mathcal{A} \models \varphi_i$ et $a'_i = 1$ si et seulement si $\mathcal{A}' \models \varphi'_i$, pour tout $i \in \{1, \ldots, l\}$.

En d'autres termes, le théorème 5.4 dit que $\mathrm{Th}^q(\mathcal{A} \bigsqcup \mathcal{A}')$ est uniquement déterminé par $\mathrm{Th}^q(\mathcal{A})$ et $\mathrm{Th}^q(\mathcal{A}')$.

5.3.3 Schéma de translation

Le schéma de translation sans quantificateur est un cas particulier du schéma de translation monadique du second ordre [CMR00, EF95].

Soit $\tau = \{R_1, \ldots, R_k\}$ un vocabulaire et pour $i \in \{1, \ldots, k\}$, soit a_i l'arité de R_i. Un **schéma de translation sans quantificateur** dans τ est $\Phi = \langle \psi_1, \ldots, \psi_k \rangle$ où pour tout $i \in \{1, \ldots, k\}$, $\psi_i \in \mathrm{MSOL}^0(\tau)$ (c'est-à-dire est une formule monadique du vocabulaire τ sans quantificateur), a a_i variables libres du premier ordre, et n'a pas de variable libre du second ordre.

Soit $\mathcal{A}$ une τ-structure. Alors $\Phi^*(\mathcal{A})$ est la τ-structure de domaine A et pour tout $i \in \{1, \ldots, k\}$, $R_i^{\Phi^*(\mathcal{A})} = \{(x_1, \ldots, x_{a_i}) : x_1, \ldots, x_{a_i} \in A \text{ et } \mathcal{A} \models \psi_i(x_1, \ldots, x_k)\}$, où a_i désigne l'arité de R_i.

Soit φ une formule sur le vocabulaire τ. Alors $\Phi^\sharp(\varphi)$ est la formule φ dans laquelle on a remplacé les occurrences de $R_i(x_1, \ldots, x_a)$ par $\psi_i(x_1, \ldots, x_a)$. Plus formellement, $\Phi^\sharp(\varphi)$ peut être définie récursivement :

- $\Phi^\sharp(\forall X, \varphi') \triangleq \forall X, \Phi^\sharp(\varphi)$ et $\Phi^\sharp(\exists X, \varphi') \triangleq \exists X, \Phi^\sharp(\varphi)$.
- $\Phi^\sharp(\varphi_1 \wedge \varphi_2) \triangleq \Phi^\sharp(\varphi_1) \wedge \Phi^\sharp(\varphi_2)$ et $\Phi^\sharp(\varphi_1 \vee \varphi_2) \triangleq \Phi^\sharp(\varphi_1) \vee \Phi^\sharp(\varphi_2)$.
- $\Phi^\sharp(\neg\varphi') \triangleq \neg\Phi^\sharp(\varphi')$.
- $\Phi^\sharp(R_i(x_1, \ldots, x_{a_i})) \triangleq \psi_i(x_1, \ldots, x_{a_i})$ pour tout $i \in \{1, \ldots, k\}$.
- $\Phi^\sharp(U(x)) \triangleq U(x)$ si U est une variable du second ordre.
- $\Phi^\sharp(x = y) \triangleq x = y$.

On note que si Φ est un schéma de translation sans quantificateur, alors $\mathrm{qr}(\Phi^\sharp(\varphi)) = \mathrm{qr}(\varphi)$.

Théorème 5.5. *[EF95] Soit τ un vocabulaire, $\mathcal{A}$ une τ-structure, Φ un schéma de translation sans quantificateur et φ une formule sur le vocabulaire τ. Alors*

$$\mathcal{A} \models \Phi^\sharp(\varphi) \iff \Phi^*(\mathcal{A}) \models \varphi.$$

5.3.4 Partition MSOL sur les structures de largeur de clique bornée

Théorème 5.6. *Soit un entier k, un vocabulaire τ^k, et $\varphi \in \mathrm{MSOL}(\tau, \emptyset, \{X\})$ fixés. Il existe un algorithme polynomial pour le problème suivant :*
Entrée : Une τ^k-structure $\mathcal{A}$ de largeur de clique au plus k, et une k-expression pour $\mathcal{A}$
Sortie : $C \subseteq \{1, \ldots, |V|\}$ tel que $r \in C$ si et seulement si il existe une partition $\{A_1, \ldots, A_r\}$ de A telle que :

$$\forall i \in \{1, \ldots, r\}, \langle \mathcal{A}, A_i \rangle \models \varphi(A_i).$$

Démonstration. Soit $q = \mathrm{qr}(\varphi)$.

L'idée de la preuve est de calculer récursivement tous les multi-ensembles des théories de toutes structures apparaissant dans la décomposition de la structure $\mathcal{A}$.

Plus formellement, on calcule récursivement sur l'arbre de décomposition l'ensemble

$$\mathcal{S}(\mathcal{A}) = \left\{ \langle \mathrm{Th}^q(\langle \mathcal{A}, A_i \rangle) \rangle_{i \in \{1, \ldots, r\}} : \{A_1, \ldots, A_r\} \text{ est une partition de } A \right\}.$$

L'ensemble $\mathcal{S}$ pour les structures $\cdot_i$ est le suivant :

$$\mathcal{S}(\cdot_i) = \left\{\langle \mathrm{Th}^q\left(\langle \mathcal{A}, \{v\}\rangle\right), \mathrm{Th}^q\left(\langle \mathcal{A}, \emptyset\rangle\right), \ldots, \mathrm{Th}^q\left(\langle \mathcal{A}, \emptyset\rangle\right)\rangle\right\}$$

car il n'y a qu'une seule partition possible du domaine de $\cdot_i$. Les théories peuvent être calculées en temps constant, car les structures sont de tailles fixées et il y a un nombre fixé de formules à tester.

Pour tout $R \in \tau$ et $i_1, \ldots, i_a \in \{1, \ldots, k\}$ il existe un schéma de translation sans quantificateur pour $\eta^R_{(i_1,\ldots,i_a)} : \Phi_{\eta^R_{(i_1,\ldots,i_a)}} = \langle \psi_{R'} : R' \in \tau^k \rangle$ où $\psi_{R'} = R'(x_1, \ldots x_{a'})$ si $R' \in \tau^k \setminus \{R\}$ (où a' est l'arité de R'), et $\psi_R = R(x_1, \ldots x_a) \vee (P_{i_1}(x_1) \wedge P_{i_2}(x_2) \ldots \wedge P_{i_a}(x_a))$.

De même, pour tout $i, j \in \{1, \ldots, k\}$, il existe un schéma de translation sans quantificateur pour $\rho_{i\to j}$, qui est le suivant : $\Phi_{\rho_{i,j}} = \langle \psi_{R'} : R \in \tau^k \rangle$ où $\psi_{R'} = R'(x_1, \ldots x_{a'})$ si $R' \in \tau^k \setminus \{P_i, P_j\}$ (où a' est l'arité de R'), $\psi_{P_i} = \text{Faux}$ et $\psi_{P_j} = P_i(x_1) \vee P_j(x_1)$.

Soit f une opération $\rho_{i\to j}$ ou $\eta^R_{(i_1,\ldots,i_a)}$, et Φ son schéma de translation correspondant. Alors :

$$\mathcal{S}(f(\mathcal{A})) = \{\langle \Phi^\sharp(T_i)\rangle_{i\in\{1,\ldots,r\}} : \langle T_i\rangle_{i\in\{1,\ldots,r\}} \in \mathcal{S}(\mathcal{A})\}$$

avec $\Phi^\sharp(T) = \{\Phi^\sharp(\varphi) : \varphi \in T\}$.

Comme les multi-ensembles appartiennent à un univers d'au plus $(r+1)^{|\mathrm{MSOL}^q(\tau,\emptyset,\{X\})|}$ éléments, $\mathcal{S}(f(\mathcal{A}))$ peut être calculé en temps $(r+1)^{|\mathrm{MSOL}^q(\tau,\emptyset,\{X\})|+1}$.

Pour l'opération $\oplus$, on utilise le théorème de Feferman-Vaught. Soit $\mathcal{A}$ et $\mathcal{A}'$ deux τ^k-structures, et soit $A_1 \subseteq A$ et $A'_1 \subseteq A'$. Le théorème nous dit qu'il est possible de calculer $\mathrm{Th}^q(\langle \mathcal{A} \bigsqcup \mathcal{A}', A_1 \cup A'_1\rangle)$ depuis $\mathrm{Th}^q(\langle \mathcal{A}, A_1\rangle)$ et $\mathrm{Th}^q(\langle \mathcal{A}', A'_1\rangle)$. De plus, comme la taille de ces ensembles est bornée par une constante fixée (dépendante de τ^k et de q), cela se fait en temps constant.

Soit $\mathcal{M}_a \in \mathcal{S}(\mathcal{A})$ et $\mathcal{M}_b \in \mathcal{S}(\mathcal{A}')$. Chaque ensemble T_a de $\mathcal{M}_a$ correspond à $\mathrm{Th}^q(\langle \mathcal{A}, A_1\rangle)$ pour un $A_1 \subseteq A$ et chaque ensemble T_b de $\mathcal{M}_b$ correspond à $\mathrm{Th}^q(\langle \mathcal{A}', A'_1\rangle)$ pour un $A'_1 \subseteq A'$. On peut donc calculer $\mathrm{Th}^q(\langle \mathcal{A} \bigsqcup \mathcal{A}', A_1 \cup A'_1\rangle)$, que l'on notera $T_a \boxplus T_b$.

On commence avec l'ensemble $\mathcal{D} = \{\langle\rangle\} \times \mathcal{S}(\mathcal{A}) \times \mathcal{S}(\mathcal{A}')$, puis on étend $\mathcal{D}$ par tous les triplets qui peuvent être obtenus d'un triplet $(\mathcal{M}, \mathcal{M}_a, \mathcal{M}_b) \in \mathcal{D}$ en supprimant un ensemble T_a de $\mathcal{M}_a$ et un ensemble T_b de $\mathcal{M}_b$, et en rajoutant $T_a \boxplus T_b$ à $\mathcal{M}$. On voit que quand $\mathcal{D}$ ne pourra plus être étendu, on aura $\mathcal{S}(\mathcal{A} \oplus \mathcal{A}') = \{\mathcal{M} : (\mathcal{M}, \langle\rangle, \langle\rangle) \in \mathcal{D}\}$.

Les opérations $\boxplus$ se font en temps constant. Il y a au plus n^2 façons de combiner un élément de $\mathcal{M}_1$ avec un élément de $\mathcal{M}_2$. Comme $\mathcal{M}$, $\mathcal{M}_1$ et $\mathcal{M}_2$ appartiennent à un univers d'au plus $(r+1)^{|\mathrm{MSOL}^q(\tau,\emptyset,\{X\})|}$ éléments, $\mathcal{D}$ a au plus $(r+1)^{3\times|\mathrm{MSOL}^q(\tau,\emptyset,\{X\})|}$ éléments, et peut donc se calculer en temps $(r+1)^{3\times|\mathrm{MSOL}^q(\tau,\emptyset,\{X\})|+2}$ □

Le théorème 5.6 appliqué sur la structure τ_1 d'un graphe nous donne directement :

Corollaire 5.7. *Tout problème de partitionnement* MSOL_1 *peut être résolu en temps polynomial sur les graphes de largeur de clique bornée, si la k-expression est donnée avec le graphe.*

Remarque. *Il n'est pas nécessaire de calculer* $\mathrm{Th}_q(\mathcal{A})$. *Il suffit de trouver* $\Psi \subseteq \mathrm{MSOL}^q(\tau)$ *tel que pour tout* $\psi \in \Psi$ *;*

- *pour tout schéma de translation* Φ *(correspondant aux opérations* ρ *et* η*),* $\Phi^\sharp(\psi) \in \Psi$ *et*
- *pour toutes les formules* ψ_i *apparaissant dans le théorème de Feferman-Vaught pour la formule* ψ*,* $\psi_i \in \Psi$*.*

Cet ensemble de formules peut être calculé en commençant par l'ensemble $\{\varphi\}$ *(où* φ *est la propriété* MSOL *à vérifier), et en agrandissant tant que possible l'ensemble, en ajoutant* $\Phi^\sharp(\psi)$ *et les* ψ_i *pour tout* $\psi \in \Psi$*. Quand l'ensemble ne peut plus être agrandi, alors il a la propriété demandée.*

5.3.5 Application à la largeur arborescente

La largeur de clique de la structure $\tau_2(G)$ est égale à la largeur de clique de $I(G)$, car $\tau_2(G)$ est la structure $\tau_1(G)$, augmentée d'une relation unaire P_V (qui ne change pas la largeur de clique). Donc si la largeur arborescente de G est bornée par k, la largeur arborescente de $I(G)$ est au plus k [LR04], et la largeur de clique est au plus $3\times 2^{k-1}$ [CR05].

Corollaire 5.8. *Tout problème de partitionnement* MSOL_2 *peut être résolu en temps polynomial sur les graphes de largeur arborescente bornée, si la décomposition est donnée avec le graphe.*

Dans ce dernier cas, si la décomposition n'est pas donnée avec le graphe, on peut utiliser l'algorithme donné dans [Bod96] pour en trouver une en temps linéaire.

5.4 Application

On voit facilement que l'on peut résoudre directement le problème de coloration sur les graphes de largeur de clique au plus k en testant les propriétés suivantes :

- p est la propriété « X est un ensemble stable »,
- pour tout $i \in \{1, \ldots, k\}$, p_i est la propriété « X possède un sommet étiqueté i ».

Il suffit donc de $1 + k$ propriétés. On voit facilement que ces propriétés sont des propriétés MSOL_1 ; p peut s'écrire $\forall u, v \in X, \neg\,\mathrm{Adj}(u, v)$, et p_i peut s'écrire $\exists v \in X, P_i(v)$.

La liste des propriétés à tester générée suivant la remarque de la fin de la section 5.3.4, pour $k = 2$ et $\varphi = \forall u, v(\neg\,\mathrm{Adj}(u, v) \vee u \notin X \vee v \notin X)$ (c'est-à-dire « X est un ensemble stable »), est la suivante :

$$\varphi_1 = \forall u, \forall v, (\neg \mathrm{Adj}(u,v) \vee u \notin X \vee v \notin X)$$
$$\varphi_2 = \forall u, \forall v, (\neg(\mathrm{Adj}(u,v) \vee (u \in P_1 \wedge v \in P_2) \vee (v \in P_1 \wedge u \in P_2)) \vee u \notin X \vee v \notin X))$$
$$\varphi_3 = \forall u, \forall v, (u \notin P_1 \vee v \notin P_2 \vee u \notin X \vee v \notin X)$$
$$\varphi_4 = \forall u, \forall v, (v \notin P_1 \vee u \notin P_2 \vee u \notin X \vee v \notin X)$$
$$\varphi_5 = \forall u, (u \notin P_1 \vee u \notin X)$$
$$\varphi_6 = \forall u, (u \notin P_2 \vee u \notin X)$$
$$\varphi_7 = \forall v, (\neg(v \in P_1 \vee v \in P_2) \vee v \notin X)$$

On remarque que $\varphi_1 \equiv \varphi \equiv p$, que $\varphi_5 \equiv \neg p_1$ et que $\varphi_6 \equiv \neg p_2$.

Pour la propriété « ensemble stable », pour $k = 2$, la méthode génère donc 7 propriétés. Pour $k = 3$, elle génère 27 propriétés, et pour $k = 4$, elle génère 129 propriétés. En pratique, on obtient donc beaucoup mieux que la borne supérieure théorique de $|\mathrm{MSOL}^q(\tau_{1,k} \cup \{X\})|$ (qui est une tour d'exponentielle en la profondeur de quantification).

On peut remarquer que l'ensemble des propriétés généré pour le problème PARTITION EN ENSEMBLE AYANT LA PROPRIÉTÉ φ est exactement l'ensemble des propriétés pour trouver le plus grand ensemble ayant la propriété φ, pour l'algorithme de [CMR00]. Pour le problème ENSEMBLE STABLE, sur les graphes de petite largeur de clique, l'algorithme de [CMR00] semble donc utilisable en pratique.

5.5 Conclusion

On a montré qu'une nouvelle classe de problèmes est polynomiale sur les classes de largeur de clique bornée, ce qui rajoute des problèmes à la déjà grande liste de problèmes polynomiaux pour les graphes de largeur de clique bornée. Vu la relation entre la largeur arborescente et la largeur de clique du graphe d'incidence, on a une nouvelle classe de problèmes polynomiaux pour les graphes de largeur arborescente bornée.

Néanmoins, on ne connaît encore pas la complexité de certains problèmes naturels, dont on peut citer INDEX CHROMATIQUE (problème où l'on veut colorier les arêtes avec un nombre minimum de couleurs).

On ne sait non plus pas si la largeur de clique peut aider pour décider en temps polynomial si un certain paramètre (largeur de clique, largeur NLC...) est borné par un k. Combiné avec les algorithmes pour la largeur de rang [OS06], cela nous donnerait un algorithme polynomial pour décider si la largeur de clique est bornée par k, k fixé. Un autre problème ouvert intéressant est le problème ISOMORPHISME DE GRAPHE.

5.6 Annexe

5.6.1 Logique monadique du second ordre

Soit τ un vocabulaire. Les **formules atomiques** sont les formules : Vrai, Faux, $x_1 = x_2$ et $R(x_1, \ldots, x_a)$, où $x_1, \ldots, x_a$ sont des variables, et R est un symbole de relation du vocabulaire.

Les **formules du premier ordre** sont les formules construites récursivement en utilisant les formules atomiques, et les connecteurs $\vee$ (ou), $\wedge$ (et), $\neg$ (non), $\forall$ (pour tout) et $\exists$ (il existe).

- Si φ est une formule atomique, alors φ est une formule du premier ordre,
- Si φ et ψ sont des formules du premier ordre, alors $\varphi \vee \psi$, $\varphi \wedge \psi$ et $\neg\varphi$ sont des formules du premier ordre,
- Si φ est une formule du premier ordre et x une variable, alors $\forall x, \varphi$ et $\exists x, \varphi$ sont des formules du premier ordre.

En **logique du second ordre**, la quantification est possible sur les relations. Les formules atomiques peuvent donc aussi être de la forme $X(x_1, \ldots, x_a)$, où X est une variable du second ordre, a son arité et $x_1, \ldots x_a$ des variables du premier ordre. En **logique monadique du second ordre**, la quantification n'est possible que sur les variables et les relations unaires (qui peuvent être vues comme des sous-ensembles du domaine).

5.6.2 Variables libres

L'ensemble des variables libres peut être calculé récursivement :

- si φ est une formule atomique, alors libre(φ) est l'ensemble des variables apparaissant dans φ,
- libre($\varphi \wedge \psi$) $\triangleq$ libre(φ) $\cup$ libre(ψ),
- libre($\varphi \vee \psi$) $\triangleq$ libre(φ) $\cup$ libre(ψ),
- libre($\neg\varphi$) $\triangleq$ libre(φ),
- libre($\forall X, \varphi$) $\triangleq$ libre(φ) $\setminus \{X\}$, où X est une variable du premier ou du second ordre,
- libre($\exists X, \varphi$) $\triangleq$ libre(φ) $\setminus \{X\}$.

5.6.3 Modèles

La relation $\models$ est définie de la façon suivante :

- $\langle \mathcal{A}, z\rangle \models \varphi = R(x_1, \ldots, x_r)$ si $(z'(x_1), \ldots, z'(x_r)) \in z'(R)$, où z' est la fonction qui étend z aux symboles de τ (c'est-à-dire $z'(R) = R^{\mathcal{A}}$ pour toute relation R de τ, et $z'(X) = z(X)$ pour toute variable dans le domaine de z).
- $\langle \mathcal{A}, z\rangle \models \varphi_1 \wedge \varphi_2$ si $\langle \mathcal{A}, z\rangle \models \varphi_1$ et $\langle \mathcal{A}, z\rangle \models \varphi_2$,
- $\langle \mathcal{A}, z\rangle \models \varphi_1 \vee \varphi_2$ si $\langle \mathcal{A}, z\rangle \models \varphi_1$ ou $\langle \mathcal{A}, z\rangle \models \varphi_2$,
- $\langle \mathcal{A}, z\rangle \models \neg\varphi_1$ si on n'a pas $\langle \mathcal{A}, z\rangle \models \varphi_1$,
- $\langle \mathcal{A}, z\rangle \models \exists X, \varphi_1$ s'il existe $C \subseteq A$ ($C \in A$ si X est du premier ordre), $\langle \mathcal{A}, z[\frac{C}{X}]\rangle \models \varphi_1$,

– $\langle \mathcal{A}, z\rangle \models \forall X, \varphi_1$ si pour tout $C \subseteq A$ ($C \in A$ si X est du premier ordre), $\langle \mathcal{A}, z[\frac{C}{X}]\rangle \models \varphi_1$.

Si $\langle \mathcal{A}, z\rangle \models \varphi$, on dit que $\langle \mathcal{A}, z\rangle$ est un **modèle** de φ. Si le domaine de z est nul, on écrira $\mathcal{A} \models \varphi$.

5.6.4 Équivalence des formules

On dit que deux formules φ et ψ sont **équivalentes**, que l'on notera par $\varphi \equiv \psi$, si pour toute structure $\mathcal{A}$ et assignation z, $\langle \mathcal{A}, z\rangle \models \varphi$ si et seulement si $\langle \mathcal{A}, z\rangle \models \psi$. Si on fixe $q \in \mathbb{N}$, il y a un nombre non fini de formules de profondeur de quantification au plus q, mais il n'y a qu'un nombre fini de formules non équivalentes. Malheureusement, l'équivalence entre les formules (même du premier ordre) est indécidable.

Par contre, on peut trouver une relation d'équivalence $\approx$ entre les formules telle que si $\varphi \approx \psi$, alors $\varphi \equiv \psi$, et pour un $q \in \mathbb{N}$ fixé, il y ait un nombre fini de formules non équivalentes par $\approx$, de profondeur de quantification au plus q.

Cette équivalence utilise l'équivalence des formules booléennes, qui est décidable.

Une **formule booléenne** est une formule construite à partir des connecteurs $\wedge$, $\vee$ et $\neg$, et de **variables booléennes**. Une **assignation** pour une formule booléenne est une fonction qui à chaque variable booléenne assigne vrai ou faux. Deux formules booléennes f et g sont **équivalentes** si pour toutes les assignations, f est vraie si et seulement si g est vraie. L'équivalence entre les formules booléennes est décidable. Il suffit par exemple de comparer les tableaux de vérité des deux formules.

Soit ${\mathrm{MSOL}_*}^0(\tau, \mathfrak{X}, \mathfrak{Y})$ l'ensemble des formules atomiques de $\mathrm{MSOL}^0(\tau, \mathfrak{X}, \mathfrak{Y})$, et pour tout $q > 0$, ${\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ est l'ensemble des formules de $\mathrm{MSOL}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ de la forme QX, ψ, où Q est un quantificateur et $X \notin \mathfrak{X} \cup \mathfrak{Y}$ une variable du premier ou du second ordre. Toute formule $\varphi \in \mathrm{MSOL}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ peut s'écrire $B(\varphi_1, \ldots, \varphi_m)$, où B est une formule booléenne et pour tout $i \in \{1, \ldots, m\}$, $\varphi_i \in {\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$.

Soit $q \in \mathbb{N}$ et $\varphi \in {\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$. Par définition de la profondeur de quantification, φ peut s'écrire $QX, B(\psi_1, \ldots, \psi_m)$, où B est une formule booléenne et pour tout $i \in \{1, \ldots, m\}$, $\psi_i \in {\mathrm{MSOL}_*}^{q-1}(\tau, \mathfrak{X}, \mathfrak{Y})$.

On fixe un vocabulaire τ avec u_i symboles de relation d'arité i, pour tout $i \in \mathbb{N}^*$. Soit $\mathfrak{X}$ un ensemble fini de variables du premier ordre et $\mathfrak{Y}$ un ensemble fini de variables du second ordre. Les formules atomiques avec les variables du premier ordre dans $\mathfrak{X}$ et les variables du second ordre sont de la forme $x_1 = x_2$, $U(x_1)$ ou $R(x_1, \ldots x_i)$ où R est une relation d'arité i, $U \in \mathfrak{Y}$ est une variable du second ordre et pour tout j, $x_j \in \mathfrak{X}$. Donc $|{\mathrm{MSOL}_*}^0(\tau, \mathfrak{X}, \mathfrak{Y})| \leq |\mathfrak{X}|^2 + |\mathfrak{Y}| \times |\mathfrak{X}| + \sum_{i>0} u_i |\mathfrak{X}|^i$.

L'équivalence $\approx$ est définie récursivement. Soit $q \in \mathbb{N}$, et $\varphi, \psi \in {\mathrm{MSOL}_*}^0(\tau, \mathfrak{X}, \mathfrak{Y})$. Si $q = 0$, alors $\varphi \approx \psi$ si $\varphi = \psi$. Sinon, φ est de la forme QX, φ' et ψ de la forme $Q'X', \psi'$. Si X est une variable du premier ordre, alors $\varphi \in \mathrm{MSOL}^{q-1}(\tau, \mathfrak{X} \cup \{X\}, \mathfrak{Y})$, sinon $\varphi \in \mathrm{MSOL}^{q-1}(\tau, \mathfrak{X}, \mathfrak{Y} \cup \{X\})$. De même, si X' est une variable du premier ordre, alors $\psi \in \mathrm{MSOL}^{q-1}(\tau, \mathfrak{X} \cup \{X'\}, \mathfrak{Y})$, sinon $\psi \in \mathrm{MSOL}^{q-1}(\tau, \mathfrak{X}, \mathfrak{Y} \cup \{X'\})$. Alors $\varphi \approx \psi$ si $Q = Q'$, X et X' sont des variables du même ordre, et $\varphi \approx \psi' \frac{X'}{X}$. Par récurrence, il y a un nombre fini de

classes d'équivalence de $\approx$ dans $\mathrm{MSOL}^{q-1}(\tau, \mathfrak{X}, \mathfrak{Y})$. Donc le nombre de classes d'équivalence de $\approx$ dans ${\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ est fini et sera d'au plus $2 \times (|\,\mathrm{MSOL}^{q-1}(\tau, \mathfrak{X} \cup \{X\}, \mathfrak{Y})| + |\,\mathrm{MSOL}^{q-1}(\tau, \mathfrak{X}, \mathfrak{Y} \cup \{X\})|)$.

Soit $\varphi, \psi \in \mathrm{MSOL}^q(\tau, \mathfrak{X}, \mathfrak{Y})$. φ et ψ peuvent s'écrire respectivement $B(\varphi_1, \ldots, \varphi_m)$ et $B'(\psi_1, \ldots, \psi_{m'})$, avec $\varphi_i, \psi_i \in {\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$. Par récurrence, il y a un nombre fini de classes d'équivalence de $\approx$ dans ${\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})$. Le nombre de classes d'équivalence de $\approx$ dans $\mathrm{MSOL}^q(\tau, \mathfrak{X}, \mathfrak{Y})$ est fini et sera d'au plus $2^{2^{|{\mathrm{MSOL}_*}^q(\tau, \mathfrak{X}, \mathfrak{Y})|}}$.

5.6.5 Largeur de clique d'une structure

Soit τ un vocabulaire. On définit τ^k le vocabulaire contenant toute les relations de τ, avec k relations unaires $P_i, \ldots P_k$.

Soit $\mathcal{A}$ et $\mathcal{A}'$ deux τ-structures, de domaines respectifs A et A'. On suppose que $A \cap A' = \emptyset$ (sinon on prend une copie disjointe de $\mathcal{A}'$). L'union disjointe de $\mathcal{A}$ et $\mathcal{A}'$ est la τ-structure, notée $\mathcal{A} \bigsqcup \mathcal{A}'$, de domaine $A \cup A'$, et telle que pour chaque relation R, $R^{\mathcal{A} \bigsqcup \mathcal{A}'} = R^{\mathcal{A}} \cup R^{\mathcal{A}'}$. Dans les cas des graphes, cette opération correspond à l'union disjointe.

- $\cdot_i$ représente la τ^k-structure, avec comme domaine $\{v\}$, telle que $P_i = \{v\}$ et telle que toutes les autres relations sont vides.
- $\rho_{i \to j}(\mathcal{A})$ est la structure de domaine A, telle que $P_i^{\rho_{i \to j}(\mathcal{A})} = \emptyset$, $P_j^{\rho_{i \to j}(\mathcal{A})} = P_i^{\mathcal{A}} \cup P_j^{\mathcal{A}}$, et que pour toutes les autres relations $R^{\rho_{i \to j}(\mathcal{A})} = R^{\mathcal{A}}$.
- Soit R un symbole de relation autre que P_i, $i \in \{1, \ldots, k\}$, et soit a son arité. Alors $\eta^R_{(i_1, \ldots, i_a)}(\mathcal{A})$ est la structure de domaine A, avec $R^{\eta^R_{(i_1, \ldots, i_a)}(\mathcal{A})} = R^{\mathcal{A}} \cup P_{i_1}^{\mathcal{A}} \times \ldots \times P_{i_a}^{\mathcal{A}}$ et pour tout symbole de relation Q autre que R, $Q^{\eta^R_{(i_1, \ldots, i_a)}(\mathcal{A})} = Q^{\mathcal{A}}$.
- $\mathcal{A} \oplus \mathcal{A}'$ est l'union disjointe des structures $\mathcal{A}$ et $\mathcal{A}'$, c'est-à-dire $\mathcal{A} \bigsqcup \mathcal{A}'$.

La largeur de clique d'une τ-structure $\mathcal{A}$ est le plus petit k tel qu'il est possible de construire la τ^k-structure $\mathcal{A}'$ avec les opérations ci dessus, et tel que $\mathcal{A}$ est la structure $\mathcal{A}'$ restreinte des prédicats unaires P_i, $i \in \{1, \ldots, k\}$.

Chapitre 6

Généralisations

La décomposition en coupes ainsi que la décomposition bi-joint sont deux généralisations de la décomposition modulaire, généralisation dans le sens qu'un module d'un graphe est toujours une coupe et un bi-joint. De ce fait un graphe complètement décomposable par la décomposition modulaire sera complètement décomposable par la décomposition en coupes et la décomposition bi-joint.

Comme autre généralisation de la décomposition modulaire, on peut ajouter la décomposition en coupes du graphe complémentaire. Une bi-partition des sommets d'un graphe est appelée une **co-coupe** si cette bi-partition est une coupe du graphe complémentaire. Les bi-joints du complémentaire d'un graphe sont exactement les bi-joints du graphe, et donc n'induisent pas de nouvelle décomposition.

Dans ce chapitre, on étudie des généralisations de ces décompositions.

6.1 k-modules et largeur modulaire

Jean-Marc Lanlignel, dans la conclusion de sa thèse [Lan01], parle informellement de ce que pourrait être la largeur modulaire d'un graphe, une sorte de généralisation de la décomposition modulaire. Il dit que ce paramètre pourrait être une alternative à la largeur de clique et à la largeur NLC.

Dans cette section, on donne une définition formelle d'une largeur modulaire, ainsi que des relations de cette largeur avec la largeur de clique et la largeur NLC. On verra qu'il s'agit de paramètres similaires, et que le rapport entre ces largeurs est au plus 2. On donnera aussi une nouvelle algèbre pour construire un graphe de largeur modulaire k avec au plus k couleurs différentes.

6.1.1 Les k-modules

Définition 6.1.1. Soit $k \in \mathbb{N}^*$, $G = (V, E)$ un graphe et $V' \subseteq V$. On dit que V' est un **k-module** s'il existe une partition $\{V'_1, \ldots, V'_k\}$ de V' telle que pour tout $i \in \{1, \ldots, k\}$, et pour tout $u, v \in V'_i$, $N(u) \setminus V' = N(v) \setminus V'$.

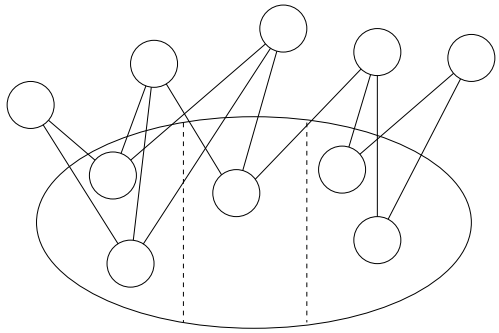


Fig. 6.1 : – Un 3-module dans un graphe avec sa partition.

On voit facilement qu'un 1-module correspond à un module. La figure 6.1 donne en exemple un 3-module dans un graphe.

On verra dans la section 6.3 comment on peut décider en temps polynomial si un graphe possède un k-module non trivial, pour k fixé. On verra également qu'une structure représentant tous les k-modules du graphe peut être calculée en temps polynomial, pour k fixé.

Une **classe** dans G d'un sous-ensemble V' de V est un sous-ensemble maximal C de sommets de V' tel que pour tout $u \in V(G) \setminus V'$, soit u est adjacent à tous les sommets de C, soit à aucun. On notera par $\mathrm{Part}_G(V')$ l'ensemble des classes de V' dans G. On voit que V' est un k-module si et seulement si $|\mathrm{Part}_G(V')| \leq k$. Les deux observations suivantes sont immédiates.

Lemme 6.1. *Si V' est un k-module de G, alors V' est un k-module de $\overline{G}$.*

Lemme 6.2. *Si $V'' \subseteq V' \subseteq V$ et V'' est un k-module de G, alors V'' est un k-module de $G[V']$.*

Si $\{V_1, V_2\}$ est une coupe, une co-coupe ou un bi-joint, alors V_1 et V_2 sont des 2-modules. Donc si V' est un module, $V \setminus V'$ est un 2-module. Plus généralement on peut démontrer le lemme suivant.

Lemme 6.3. *Si V' est un k-module, alors $V \setminus V'$ est un 2^k-module.*

Démonstration. Soit $u \notin V'$. Alors pour chaque classe C de V', u est soit adjacent à tous les sommets de C, soit à aucun. Comme il y a au plus k classes dans V', il y a au plus 2^k adjacences possibles pour les sommets de $V \setminus V'$ dans V'. Donc $|\mathrm{Part}_G(V \setminus V')| \leq 2^k$. □

Fabien de Montgolfier a consacré un chapitre de sa thèse à l'étude des 2-modules. Il montre en particulier qu'ils ne forment pas, en général, une famille partitive.

Cela est déjà un « mauvais point » pour la définition de ce que pourrait être une décomposition k-modulaire ; la propriété de partitivité permettait d'avoir l'unicité de la décomposition. Elle garantit aussi qu'une décomposition récursive par un module non trivial aboutit toujours à la décomposition.

On peut néanmoins suggérer à quoi correspondrait une décomposition k-modulaire d'un graphe.

Définition 6.1.2. Soit G un graphe. Une **décomposition k-modulaire** de G est une famille $\mathcal{F}$ d'ensembles de sommets du graphe telle que :
- pour tout sommet v du graphe, $\{v\} \in \mathcal{F}$,
- $V \in \mathcal{F}$,
- pour tout $X \in \mathcal{F}$ et $Y \in \mathcal{F}$, X ne chevauche pas Y,
- pour tout $X \in \mathcal{F}$, X induit un k-module de G.

Dans le cas des 1-modules, cette famille correspond aux modules forts du graphe. Par définition, une décomposition k-modulaire d'un graphe est une famille arborescente. Elle demande en plus que chaque membre de la famille induise un k-module. Une décomposition k-modulaire d'un graphe correspond donc à un arbre, tel qu'il y ait une bijection entre les sommets du graphe et les feuilles de l'arbre, et pour tout noeud de l'arbre, l'ensemble des feuilles du sous-arbre enraciné au noeud induit un k-module de G. On appelle cet arbre **l'arbre de la décomposition**.

On va s'intéresser à partir de maintenant aux graphes complètement décomposables par cette décomposition k-modulaire.

6.1.2 Décomposition par k-modules et largeur modulaire

Définition 6.1.3. Soit $G = (V, E)$ un graphe. On dit que G est **décomposable par k-modules** s'il existe un arbre binaire enraciné T tel que il y ait une bijection entre les feuilles de l'arbre et les sommets du graphe, et que pour chaque noeud a de l'arbre, l'ensemble des sommets correspondant aux feuilles du sous-arbre enraciné en a est un k-module.

On voit bien qu'il s'agit d'un cas particulier de la décomposition k-modulaire, tel que chaque noeud de l'arbre de décomposition a au plus deux fils. À partir de maintenant, quand on parlera de décomposition k-modulaire, on sous entendra que chaque noeud dans l'arbre de décomposition a au plus 2 fils. Les graphes décomposables par 1-modules sont exactement les cographes. On retombe donc bien sur la classe des graphes entièrement décomposables par la décomposition modulaire.

On peut définir un paramètre de graphe correspondant à cette décomposition :

Définition 6.1.4. Soit $G = (V, E)$ un graphe. La **largeur modulaire** de G est le plus petit $k \in \mathbb{N}^*$ tel que G soit décomposable par k-modules. Il est noté $\text{mod-wd}(G)$.

Les graphes distance héréditaire, leur complément et les graphes sans C_5, taureau, gem et co-gem induit ont une largeur modulaire d'au plus 2. En effet, ces graphes sont complètement décomposables par la décomposition en coupes, la décomposition bi-joint ou la décomposition en co-coupes. Si l'on enracine l'arbre de décomposition, et on divise tous les noeuds internes en noeuds de degré 3, l'arbre obtenu induira une décomposition 2-modulaire. Mais la classe des graphes de largeur modulaire au plus 2 est strictement plus grande que l'union de ces trois classes de graphes, car elle contient, par exemple, le C_5.

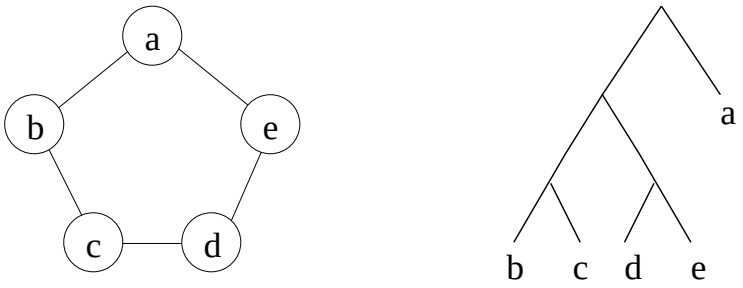


Fig. 6.2: – Une décomposition 2-modulaire du C_5.

Lemme 6.4. *Soit G un graphe. Alors* mod-wd(G) = mod-wd$(\overline{G})$.

Démonstration. Immédiat, car, par le lemme 6.1, une décomposition k-modulaire de G est une décomposition k-modulaire de $\overline{G}$. □

Lemme 6.5. *Soit $G = (V, E)$ un graphe $V' \subseteq V$. Alors* mod-wd$(G[V'])$ ≤ mod-wd(G).

Démonstration. Soit $\mathcal{F}$ une décomposition k-modulaire de G telle que l'arbre de décomposition soit binaire. La famille $\mathcal{F}' = \{X \cap V' : X \in \mathcal{F} \text{ et } X \cap V' \neq \emptyset\}$ est une famille arborescente, dont l'arbre d'inclusion de la famille est binaire. De plus par le lemme 6.2, chaque membre de $\mathcal{F}'$ est un k-module de $G[V']$. □

Ce paramètre est en fait similaire aux paramètres déjà rencontrés.

Théorème 6.6. *Soit G un graphe. Alors* mod-wd(G) ≤ NLC-wd(G) ≤ cwd(G) ≤ 2 × mod-wd(G).

Démonstration. Il est déjà connu que NLC-wd(G) ≤ cwd(G) [Joh98].

Soit k la largeur NLC de G et t une k-expression NLC de G. Soit t' une sous-expression de t, et V' l'ensemble des sommets apparaissant dans t'. On voit facilement que V' est un k-module de G. En effet, tous les sommets de même étiquette dans le graphe exprimé par t' a le même voisinage dans $V \setminus V'$. Donc T induit un arbre binaire tel que l'ensemble de toutes les feuilles d'un sous-arbre est un module : mod-wd(G) ≤ NLC-wd(G).

Soit T un arbre binaire tel que pour tout sous-arbre, l'ensemble des feuilles est un k-module. Soit a un noeud de T, on note V_a l'ensemble des feuilles du sous-arbre enraciné en a. On construit récursivement une $2k$-expression t_a de $G[V_a]$ pour tout noeud a de l'arbre, telle que pour tout $u \in V_a$, $lab_{G(t_a)}(u) \in \{1, \ldots, k\}$ et pour tout $u, v \in V_a$ telle que $N(u) \setminus V_a \neq N(v) \setminus V_a$, $lab_{G(t_a)}(u) \neq lab_{G(t_a)}(v)$. On suppose sans perte de généralité que les noeuds internes ont toujours deux fils.

Si a est une feuille, alors l'expression est triviale. Sinon, soit b et c ses fils. Alors $V_a = V_b \cup V_c$. Soit t_b et t_c l'expression pour $G[V_b]$ et $G[V_c]$, respectivement. Soit $t''_a = t_b + \circ_{i \in \{1,\ldots,k\}} \rho_{i \to i+k}(t_c))$ (où $\circ$ représente la composition). Comme V_b et V_c sont des k-modules, on peut construire une expression t'_a depuis t''_a en ajoutant des arêtes entre les étiquettes de $\{1, \ldots k\}$ et des étiquettes de $\{k+1, \ldots, 2k\}$. Comme on n'ajoute que des

arêtes entre des étiquettes de $\{1, \dots k\}$ et des étiquettes de $\{k+1, \dots, 2k\}$, on n'ajoute pas d'arêtes entre deux sommets de V_b ou entre deux sommets de V_c. t'_a est donc une expression pour $G[V_a]$. Comme V_a est un k-module, il y a au plus k voisinages possibles pour les sommets de $V \setminus V_a$ dans V_a. On peut associer les étiquettes de $\{1, \dots, 2k\}$ en k classes, car si $u, v \in V_b$ ne sont pas dans la même classe de V_a, alors u et v ne sont pas dans la même classe de V_b (de même pour V_c). On construit l'expression t_a depuis t'_a telle que pour tout $u \in V_a$, $lab_{G(t_a)}(u) \in \{1, \dots, k\}$, et les différentes classes d'étiquettes correspondent aux différentes classes du k-module V_a. □

Remarque. *On ne peut pas prouver des égalités dans le théorème précédent; le P_{10} a une largeur modulaire de 2 et une largeur de clique de 3 (car il s'agit un graphe distance héréditaire et ce n'est pas un cographe), et sa largeur NLC est de 3.*

6.1.3 Expression pour les graphes décomposables par k-modules

L'arbre induisant les k-modules d'un graphe de largeur modulaire k ne nous donne pas directement un moyen de reconstruire le graphe. Il serait intéressant d'avoir une expression définissant le graphe, comme pour la largeur de clique et la largeur NLC. Pour cela, on peut introduire une nouvelle opération sur les graphes étiquetés. Cette opération fait l'union, ajoute les arêtes entre les 2 composantes, et re-étiquette les 2 composantes, en appliquant deux fonctions différentes pour chacune des composantes.

Définition 6.1.5. Soit $k \in \mathbb{N}^*$. Soit $G_1 = (V_1, E_1, lab_1)$ et $G_2 = (V_2, E_2, lab_2)$ deux graphes étiquetés, $S \subseteq \{1, \dots, k\}^2$, et pour $i \in \{1, 2\}$, $R_i : \{1, \dots, k\} \to \{1, \dots, k\}$. Alors $G_1 \times_{S,R_1,R_2} G_2 = (V, E, lab)$ avec $V = V_1 \cup V_2$,

$$E = E_1 \cup E_2 \cup \big\{\{u, v\} : u \in V_1, v \in V_2, (lab_1(u), lab_2(v)) \in S\big\}$$

et

$$lab(u) = \begin{cases} R_1(lab_1(u)) & \text{si } u \in V_1, \\ R_2(lab_2(u)) & \text{si } u \in V_2. \end{cases}$$

Théorème 6.7. *Soit $k \in \mathbb{N}^*$. Soit MOD_k la classe de graphes définie récursivement :*
- *pour tout $i \in \{1, \dots, k\}$, $\cdot_i \in MOD_k$*
- *soit $G \in MOD_k$ et $H \in MOD_k$. Alors, pour tout $S \subseteq \{1, \dots, k\}^2$, $R_1 : \{1, \dots, k\} \to \{1, \dots, k\}$ et $R_2 : \{1, \dots, k\} \to \{1, \dots, k\}$, alors $G \times_{S,R_1,R_2} H \in MOD_k$*

Soit G un graphe. Alors mod-wd$(G) \leq k$ *si et seulement si il existe un graphe étiqueté $G' \in MOD_k$ tel que G est isomorphe à G' sans ses étiquettes.*

Démonstration. Soit $G' \in MOD_k$, et si t est une k-expression pour le graphe G', alors pour chaque sous-expression t' de t, l'ensemble des sommets présents dans t' induit un k-module. Donc la famille des ensembles des noeuds présents dans les sous-expressions de t induit une décomposition k-modulaire de G', et mod-wd$(G) \leq k$.

Soit G de largeur modulaire k, et $\mathcal{F}$ une décomposition k-modulaire de G. On construit récursivement, pour tout $V_a \in \mathcal{F}$, une k-expression t_a pour $G[V_a]$ telle que si $u, v \in V_a$ ne sont pas dans la même classe de V_a, alors l'étiquette de u est différente de l'étiquette de v dans $G(t_a)$. Si $|V_a| = 1$, $t_a = \cdot_1$. Sinon soit $V_b, V_c \in \mathcal{F}$ tels que $V_b \cap V_c = \emptyset$ et $V_b \cup V_c = V_a$, et soit t_b et t_c les expressions pour $G[V_b]$ et $G[V_c]$. Soit $S \subseteq \{1, \ldots, k\}^2$ tel que $(i, j) \in S$ si et seulement si les sommets étiquetés i dans V_b sont adjacents aux sommets étiquetés j dans V_c. Comme V_a est un k-module, et que si $u, v \in V_a$ sont dans deux classes différentes, alors ils ont des étiquettes différentes, on peut trouver $R_b : \{1, \ldots, k\} \to \{1, \ldots, k\}$ et $R_c : \{1, \ldots, k\} \to \{1, \ldots, k\}$ tels que pour tout $u \in V_b$ étiqueté i et $v \in V_c$ étiqueté j, on a $R_b(i) = R_c(j)$ si et seulement si u et v sont dans la même classe de V_a. Alors $t_a = t_b \times_{S,R_b,R_c} t_c$ est bien l'expression de $G[V_a]$ avec les propriétés demandées. □

Comme exemple, l'expression suivante est une 2-expression pour le C_5. Les fonctions sont représentées par leur ensemble de couples valeurs–images, et R_I représente la fonction identité.

$$t = \cdot_1 \times_{\{(1,1)\},R_I,R_I} \left((\cdot_1 \times_{\{(1,2)\},R_I,R_I} \cdot_2) \times_{\{(1,2)\},\{(1,2),(2,1)\},R_I} (\cdot_1 \times_{\{(1,2)\},R_I,R_I} \cdot_2) \right)$$

6.1.4 Relation avec la décomposition modulaire et bi-joint

La largeur modulaire a le même comportement face à la décomposition modulaire que la largeur de clique et la largeur NLC : elle est égale au maximum de largeur modulaire parmi tous les graphes caractéristiques apparaissant dans la décomposition modulaire.

Lemme 6.8. *Soit $G = (V_G, E_G)$ et $H = (V_H, E_H)$ deux graphes. Soit $k \in \mathbb{N}^*$ et $V' \subseteq V_G$ un k-module. Soit $v \in V'$. Alors $(V' \setminus \{v\}) \cup V_H$ est un k-module $G\,{}^{H}_{v}$.*

Lemme 6.9. *Soit $G = (V_G, E_G)$ et $H = (V_H, E_H)$ deux graphes, et $v \in V_G$. Alors* $\text{mod-wd}(G\,{}^{H}_{v}) \geq \max(\text{mod-wd}(G), \text{mod-wd}(H))$.

Démonstration. Soit $k = \max(\text{mod-wd}(G), \text{mod-wd}(H))$. Clairement, $\text{mod-wd}(G\,{}^{H}_{v}) \geq k$, car G et H sont des sous-graphes de $G\,{}^{H}_{v}$.

Soient T_G et T_H des arbres induisant des décompositions par k-modules de G et H, respectivement. Par le lemme 6.8, l'arbre T obtenu en substituant v par T_H dans T_G est un arbre induisant une décomposition par k-modules de $G\,{}^{H}_{v}$. □

On obtient donc une égalité pour la largeur modulaire, similaire à celle donnée au théorème 2.20 pour la largeur de clique.

Corollaire 6.10. *Soit G un graphe. Alors* $\text{mod-wd}(G) = \max\{\text{mod-wd}(H) : H$ *est un graphe représentatif d'un noeud interne dans la décomposition modulaire de G }.

De plus, comme la largeur modulaire est un paramètre similaire à la largeur de clique, la largeur modulaire est bornée si et seulement si chaque graphe caractéristique dans la

décomposition en coupes est de largeur modulaire bornée. Il en est de même pour la décomposition bi-joint. On peut montrer plus précisément que dans le cas du bi-joint on a la relation suivante.

Théorème 6.11. *Soit G un graphe. Alors $cwd(G) \leq 4 \times \max\{cwd(H) : H$ est le graphe caractéristique d'un noeud interne de la décomposition bi-joint de $G\}$.*

Démonstration. On montre que mod-wd$(G) \leq 2 \times$ mod-wd$(\overline{G}^v)$ et mod-wd$(\overline{G}^v) \leq 2 \times$ mod-wd(G).

Proposition. *Soit $G = (V, E)$ un graphe et $W \subseteq V$. Soit V' un k-module de G. Alors V' est un $2k$-module de $\overline{G}^W$*

Démonstration. Soit C une classe de V' dans G. Alors $C \cap W$ est inclus dans une classe de V' dans $\overline{G}^W$, et $C \setminus W$ est inclus dans une classe de $\overline{G}^W$. □

On obtient donc immédiatement :

Proposition. *Soit $G = (V, E)$ un graphe et $W \subseteq V$. Si $\mathcal{F}$ est une décomposition k-modulaire de G, alors $\mathcal{F}$ est une décomposition $2k$-modulaire de $\overline{G}^W$.*

Proposition. *Soit $G = (V, E)$ un graphe et $v \in V$. Alors* mod-wd$(G) \leq 2 \times$ mod-wd$(\overline{G}^v)$.

Démonstration. Soit k la largeur modulaire de $\overline{G}^v = (V \setminus \{v\}, E')$ et $\mathcal{F}$ une décomposition k-modulaire de $\overline{G}^v$. Soit $\mathcal{F}' = \mathcal{F} \cup \{V\}$. Alors $\mathcal{F}'$ est une décomposition k-modulaire de $G' = (V, E')$ et donc une décomposition $2k$-modulaire de $G = \overline{G'}^{N_G(v)}$. □

Proposition. *Soit $G = (V, E)$ un graphe et $v \in V$. Alors* mod-wd$(\overline{G}^v) \leq 2 \times$ mod-wd(G).

Démonstration. Soit k la largeur modulaire k de G et $\mathcal{F}$ une décomposition k-modulaire de G. Soit $\mathcal{F}' = \{X \setminus \{v\} : X \in \mathcal{F}\}$. Alors F' est une décomposition k-modulaire de G, et donc une décomposition $2k$-modulaire de $\overline{G}^v = \overline{G - v}^{N_G(v)}$. □

La largeur modulaire de G est donc au plus deux fois la largeur modulaire de $\overline{G}^v$, et donc au plus deux fois le maximum de la largeur modulaire des graphes caractéristiques dans la décomposition modulaire de $\overline{G}^v$. De plus, si H est un graphe caractéristique de la décomposition modulaire de $\overline{G}^v$, alors il existe un graphe caractéristique H' de la décomposition bi-joint de G, et un $w \in V(H')$ tels que $H = \overline{H'}^w$. Donc la largeur modulaire est au plus 4 fois le maximum des largeurs modulaires des graphes caractéristiques dans la décomposition bi-joint de G. □

6.2 Bimodules et largeur bimodulaire

La décomposition k-modulaire, comme la décomposition en cliques et la décomposition NLC, sont basées sur des arbres enracinés. Le paramètre similaire introduit récemment, la largeur de rang, est quand à lui basé sur un arbre non enraciné.

On introduira dans cette section la décomposition k-bimodulaire, qui est une généralisation non enraciné de la décomposition k-modulaire.

6.2.1 Bimodules

Définition 6.2.1. Soit $k \in \mathbb{N}$ et $G = (V, E)$ un graphe. Une bi-partition $\{V_1, V_2\}$ de V est un **k-bimodule**[1] de G si V_1 et V_2 sont des k-modules de G.

Les 1-bimodules sont les jointures et les co-jointures, et les 2-bimodules sont des coupes, des co-coupes ou des bi-joint.

On verra dans la section 6.3 comment on peut décider en temps polynomial si un graphe possède un k-bimodule non trivial, pour k fixé. On verra également qu'une structure représentant tous les k-bimodules du graphe peut être calculée en temps polynomial, pour k fixé.

L'arbre d'inclusion des ensembles forts d'une famille partitive nous donne un moyen de stocker en taille linéaire tous les ensembles de la famille. Ainsi l'arbre de décomposition modulaire est une représentation compacte de tous les modules d'un graphe, alors qu'il peut il y en avoir un nombre exponentiel (par exemple pour un graphe complet, tous les ensembles sont des modules). L'arbre de décomposition en coupes et bi-joint donne les mêmes résultats pour les coupes et les bi-joint.

La famille des k-bimodules n'est pas bi-partitive, mais il est quand même possible de stocker tous les k-bimodules en espace polynomial.

6.2.2 Largeur bimodulaire

Définition 6.2.2. Une **décomposition k-bimodulaire** de $G = (V, E)$ est une famille arborée $\mathcal{F}$ sur V de k-bimodules, telle que dans l'arbre représentatif de $\mathcal{F}$, chaque noeud soit de degré au plus 3.

Définition 6.2.3. La **largeur bimodulaire** de G est le plus petit k tel qu'il existe une décomposition k-bimodulaire de G.

On sait que ce nombre est toujours défini, car une partition $\{V_1, V_2\}$ est toujours un n-bimodule, donc tout graphe est de largeur bimodulaire au plus n. On notera bimod-wd(G) la largeur bimodulaire de G.

Un graphe sera de largeur bimodulaire k si et seulement si il existe un arbre non enraciné T tel qu'il y ait une bijection entre V et les feuilles de T, que chaque noeud interne soit de degré au plus 3, et que pour chaque arête e de T, $\{C_e^1, C_e^2\}$ induise un k-bimodule.

[1] Attention à ne pas confondre avec les bimodules de [FHM04, Mon03] pour la décomposition des graphes bipartis.

Lemme 6.12. *Soit G un graphe. Alors la largeur bimodulaire de $\overline{G}$ est la largeur bimodulaire de G.*

Démonstration. L'arbre induisant la k-décomposition de G induit également une k-décomposition de $\overline{G}$. Donc bimod-wd$(\overline{G}) \leq$ bimod-wd$(G) \leq$ bimod-wd$(\overline{G})$. □

Théorème 6.13. *Soit G un graphe. Alors G a une largeur bimodulaire de 1 si et seulement si G est un graphe complet ou un graphe sans arêtes.*

Démonstration. Supposons le contraire. Alors dans l'arbre de décomposition, une arête au moins induit une jointure (sinon le graphe n'aurait pas d'arête), et au moins une co-jointure (sinon il serait complet). Donc le graphe n'est pas connexe et son complémentaire n'est pas connexe, contradiction. □

On voit que l'arbre de décomposition en coupes d'un graphe distance héréditaire nous donne directement une décomposition induisant une largeur bimodulaire de 2. Il en est de même pour les co-coupes et les bi-joint.

Théorème 6.14. *Soit G un graphe. G a une largeur bimodulaire de 2 si et seulement si G est un graphe distance héréditaire, le complément d'un graphe distance héréditaire ou un graphe sans C_5, taureau, gem et co-gem induit.*

Démonstration. Si G est un graphe distance héréditaire, le complément d'un graphe distance héréditaire ou un graphe sans C_5, taureau, gem et co-gem induit, alors G est complètement décomposable par une des décompositions suivantes : décomposition en coupes, décomposition en co-coupes, ou décomposition bi-joint. Dans tous les cas, soit T l'arbre de décomposition. On construit un arbre T' en ajoutant des arêtes à l'arbre T pour séparer les voisins des noeuds de degré supérieur à 3, tant qu'il en existe. Toutes les arêtes introduites induisent une coupe, co-coupe ou bi-joint, car tous les noeuds dans l'arbre T sont dégénérés. Donc l'arbre T' induit une 2-décomposition de G.

Remarquons que si $\{V_1, V_2\}$ est une 2 partition de G et que V_1 et V_2 ne sont pas des modules, alors $\{V_1, V_2\}$ est exclusivement soit une coupe, soit une co-coupe, soit un bi-joint. On appellera une coupe propre une coupe qui n'est ni une co-coupe, ni un bi-joint. De même une co-coupe propre sera une co-coupe qui n'est ni une coupe, ni un bi-joint, et un bi-joint propre est un bi-joint qui n'est ni une coupe, ni une co-coupe.

Supposons que G est un graphe de largeur bimodulaire 2 qui n'est pas dans une des classes citées.

Cas 1 : Il existe une arête e de T qui induit un bi-joint propre. Comme G n'est pas un graphe sans C_5, taureau, gem et co-gem induit, alors il existe une arête f qui n'induit pas un bi-joint.

Cas 1.1 : f induit une coupe propre. Sans perte de généralité, $C_e^1 \subseteq C_f^1$. Soit $V_1 = C_e^1$, $V_2 = C_e^2 \setminus C_f^2$ et $V_3 = C_f^2$. Alors $\{V_1 \cup V_2, V_3\}$ est une coupe propre, donc il existe $v \in V_3$ tel que $N(v) \subseteq V_3$, et $\{V_1, V_2 \cup V_3\}$ est un bi-joint propre, donc tout sommet de $V_1 \cup V_2$ a au moins un voisin dans V_3. Contradiction.

Cas 1.2 : f induit une co-coupe propre. Complémentaire du cas 1.1.

Cas 2 : Il n'existe une arête e de T qui induit un bi-joint propre. Comme G n'est ni un graphe distance héréditaire, ni le complément d'un graphe héréditaire, il existe deux arêtes e et f tel que e induit une coupe propre et f induise une co-coupe propre. Sans perte de généralité, $C_e^1 \subseteq C_f^1$. Soit $V_1 = C_e^1$, $V_2 = C_e^2 \setminus C_f^2$ et $V_3 = C_f^2$. Alors $\{V_1, V_2 \cup V_3\}$ est une coupe propre, donc il existe $u \in V_1$ tel que $N(u) \subseteq V_1$, et $\{V_1 \cup V_2, V_3\}$ est une co-coupe propre, donc il existe $v \in V_3$ tel que $V_1 \cup V_2 \subseteq N(v)$. Contradiction.

□

Contrairement à la largeur modulaire, on a donc une caractérisation des graphes de largeur bimodulaire au plus 2. On peut remarquer que le C_5 n'en fait pas partie.

6.2.3 Relation avec la largeur modulaire et la largeur de rang

Théorème 6.15. *Soit G un graphe. Alors* $\text{mod-wd}(G) \leq \text{bimod-wd}(G) \leq 2^{\text{mod-wd}(G)}$.

Démonstration. Soit $\mathcal{F}'$ une décomposition k-bimodulaire de G, et $v \in V$. Alors $\mathcal{F} = \{V' : v \notin V' \text{ et } \{V, V \setminus V'\} \in \mathcal{F}\} \cup \{V, \{v\}\}$ est une décomposition k-modulaire de G.

Soit $\mathcal{F}$ une décomposition k-modulaire de G. Alors $\mathcal{F}' = \{\{V', V \setminus V'\} : V \in \mathcal{F}\}$ est une décomposition 2^k-bimodulaire de G par le lemme 6.3. □

Corollaire 6.16. *Soit G un graphe. Alors* $\text{NLC-wd}(G) \leq \text{cwd}(G) \leq 2 \times \text{bimod-wd}(G)$, *et* $\text{bimod-wd}(G) \leq 2^{\text{NLC-wd}(G)} \leq 2^{\text{cwd}(G)}$.

Théorème 6.17. *Soit G un graphe. Alors* $\text{rwd}(G) \leq \text{bimod-wd}(G) \leq 2^{\text{rwd}(G)}$.

Démonstration. L'arbre induisant la k-décomposition de G induit également une largeur de rang de k, car un k-bimodule a un rang d'au plus k.

Soit une bi-partition $\{V_1, V_2\}$ de rang l. Soit $V_2 = \{v_1, \ldots, v_{|V_2|}\}$. Alors il existe l vecteurs de $U_1, \ldots, U_l$ de taille $|V_2|$ tels que pour tout $u \in V_1$, le vecteur $(a_1, \ldots, a_{|V_2|})$ est une combinaison linéaire des U_i, où $a_i = 1$ si $v_i \in N(u)$, 0 sinon. Il y a au plus 2^l combinaisons linéaires des U_i tel que toutes les projections des combinaisons soient dans $\{0, 1\}$. Il y a donc au plus 2^l adjacences possibles d'un sommet de V_1 dans V_2.

De même, il y a au plus 2^l adjacences possibles de V_2 dans V_1. La partition $\{V_1, V_2\}$ est donc une 2^l partition. Un arbre induisant une décomposition de rang l induit une 2^l-décomposition. □

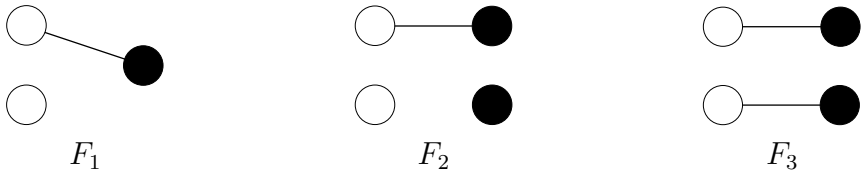
6.3 F-partitions

On présente dans cette section un moyen de trouver tous les k-modules et k-bimodules d'un graphe en temps polynomial, pour un k fixé. De plus, on verra que tous les k-modules et k-bimodules peuvent être stockés en espace polynomial (pour k fixé).

6.3.1 Définition

Définition 6.3.1. Soit $F = (X_1, X_2, E_F)$ un graphe biparti et $G = (V, E)$ un graphe. Un couple (V_1, V_2) tel que $\{V_1, V_2\}$ soit une bi-partition de V est une **F-partition** de G s'il existe une fonction $f : V \to X_1 \cup X_2$ telle que $\forall i \in \{1, 2\}$, $\forall v \in V_i$, $f(v) \in X_i$, et $\forall (u, v) \in (V_1, V_2)$, $\{u, v\} \in E \iff \{f(u), f(v)\} \in E_F$.

On voit facilement qu'il s'agit d'une généralisation du module (pour $F = F_1$), de la coupe (pour $F = F_2$), et du bi-joint (pour $F = F_3$).



F_1 F_2 F_3

On voit que $\{V_1, V_2\}$ est un k-bimodule si et seulement si il existe un graphe biparti $F = (X, Y, E)$ avec $|X| \leq k$ et $|Y| \leq k$, tel que (V_1, V_2) soit une F-partition. De même, V' est un k-module si et seulement si il existe un graphe biparti $F = (X, Y, E)$ avec $|X| \leq k$ et $|Y| \leq 2^k$, tel que $(V', V \setminus V')$ soit une F-partition.

6.3.2 Algorithme pour trouver une F-partition

On présente un algorithme polynomial pour trouver les F-partitions d'un graphe G, avec F-fixé. Le nombre des F-partitions peut être exponentiel en la taille de G, mais on verra que toutes les F-partitions peuvent être stockées dans une structure de taille polynomiale.

Proposition 6.18. *Soit $F = (X, Y, E_F)$ tel qu'il existe $x \in X$ et $x' \in X$, $x \neq x'$, et $N(x) = N(x')$. Alors (V_1, V_2) est une F-partition de G si et seulement si elle est une $(F - \{x\})$-partition de G.*

Donc, sans perte de généralité, on se limitera aux graphes bipartis tels que le voisinage de deux sommets différents appartenant à la même classe soit toujours différent.

Définition 6.3.2. Soit $F = (X, Y, E_F)$. On dit que (V_1, V_2) est une **F-partition propre** de G si (V_1, V_2) est une F-partition de G et pour tout $u \in X \cup Y$, $\{V_1, V_2\}$ n'est pas une $F - \{x\}$.

Si (V_1, V_2) est une F-partition, alors il existe toujours un sous-graphe F' de F tel que (V_1, V_2) soit une F'-partition propre.

On fixe $F = (X, Y, E_F)$ un graphe biparti, tel que pour tout $x, x' \in X$, $N(x) \neq N(x')$ et pour tout $y, y' \in Y$, $N(y) \neq N(y')$. Soit $X = \{x_1, \ldots, x_k\}$ et $Y = \{y_1, \ldots, y_l\}$, avec $k = |X|$ et $l = |Y|$.

A chaque sommet de X et de Y correspond une classe de V_1 et V_2 pour laquelle on va choisir un premier élément, une « graine ». L'algorithme cherchera toutes les F-partitions propres que l'on peut obtenir avec les graines fixées.

Soit $x'_1, \ldots, x'_k, y'_1, \ldots, y'_l \in V$, différents deux à deux, tels que $\forall i \in \{1, \ldots, k\}, \forall j \in \{1, \ldots, l\}, \{x'_i, y'_j\} \in E \iff \{x_i, y_j\} \in E_F$.

Pour tout $i \in \{1, \ldots, k\}$, soit $V_1^i = \{x'_i\}$ et pour tout $i \in \{1, \ldots, l\}$, soit $V_2^i = \{y'_i\}$. Soit $V' = V \setminus (\bigcup_{i\in\{1,\ldots,k\}} V_1^i \cup \bigcup_{i\in\{1,\ldots,l\}} V_2^i)$.

L'algorithme cherche à agrandir les classes V_1^i et V_2^i tel que $(\bigcup_{i\in\{1,\ldots,k\}} V_1^i, \bigcup_{i\in\{1,\ldots,l\}} V_2^i)$ soit toujours une F-partition de $G - V'$.

Soit $u \in V'$. Si on veut rajouter u à une des classes, on a au plus deux possibilités : une parmi les classes de V_1 et une parmi les classes de V_2.

S'il existe un $i \in \{1, \ldots, l\}$ tel que u distingue deux sommets de V_2^i, alors u ne peut pas être ajouté dans une classe de V_1. De même, s'il existe une classe de V_1 telle que u en distingue deux sommets, on ne peut pas le placer dans une classe de V_2.

S'il existe un sommet de u, qui ne peut être placé dans V_1 ni dans V_2, alors l'algorithme s'arrête : il n'existe pas de F-partition de G avec la graine choisie. Si u ne peut être placé que dans V_1, on le supprime de V' et on l'ajoute dans l'unique classe de V_1 dans laquelle il peut être placé. On fait de même s'il ne peut être placé que dans V_2.

Tant qu'il existe des sommets de V' qui peuvent être placé uniquement dans V_1 ou V_2, on les supprime de V' et on les ajoute dans les classes.

Au final, tous les sommets de V' peuvent être placés soit dans V_1, soit dans V_2. On sait donc qu'il existe au moins une F-partition avec la graine choisie (et au moins deux si $V' \neq \emptyset$) ; $(V_1 \cup V', V_2)$ et $(V_1, V_2 \cup V')$ sont des F-partitions.

Pour trouver toutes les F-partitions propres de G, il suffit d'essayer toutes les graines possibles ; il y en a $\Theta(n^{k+l})$.

Théorème 6.19. *Soit $F = (X, Y, E_f)$ un graphe biparti avec $|X| = k$ et $|Y| = l$. On peut décider si un graphe G possède une F-partition propre en temps $O(n^{k+l+2})$.*

Soit $k \geq 2$ fixé. On sait que tout graphe possède un k-bimodule : pour tout $v \in V$, les $\big\{\{v\}, V \setminus \{v\}\big\}$ sont des k-bimodules. Soit r un entier fixé. L'algorithme pour trouver une F-partition propre nous donne directement un algorithme polynomial pour trouver un k-bimodule tel que chaque classe soit de taille au moins r ; il suffit d'essayer pour tous les graphes bipartis avec au plus $\max(k, r)$ sommets dans une classe.

Corollaire 6.20. *Soit $k, r \in \mathbb{N}^*$, fixés. On peut décider en temps $O(n^{2\times\max(k,r)+2})$ si un graphe possède un k-bimodule tel que chaque classe soit de taille au moins r.*

De même, on obtient :

Corollaire 6.21. *Soit $k, r, l \in \mathbb{N}^*$, fixés. On peut décider en temps $O(n^{\max(k,r)+\max(2^k,l)+2})$ si un graphe possède un k-module de taille au moins r et de taille au plus $n - l$.*

On s'intéresse maintenant aux propriétés des F-partitions d'un graphe.

Pour tout $u \in V'$, soit $f_1(u)$ l'unique i tel que u peut être placé dans V_1^i et $f_2(u)$ l'unique i tel que u peut être placé dans V_2^i. On se ramène à un problème 2-SAT : pour chaque

sommet de u de V', on introduit une variable u. La variable correspond à « le sommet u est placé dans V_1 ». On définit l'ensemble $\mathcal{C}$ des clauses du problème :

$$\mathcal{C} = \left\{ u \Rightarrow v : u, v \in V' \text{ et } \neg(\{x_{f_1(u)}, y_{f_2(v)}\} \in E_F \iff \{u, v\} \in E) \right\}.$$

Au final on a un problème 2-SAT particulier où toutes les clauses sont du type $u \Rightarrow v$, avec u et v des littéraux positifs.

Lemme 6.22. *À chaque instanciation des variables satisfaisant $\mathcal{C}$ correspond une F-partition de G.*

Démonstration. Soit z une instanciation des variables telle que $V_1 = V_1' \cup \{u \in V' : z(u) \text{ est vrai }\}$ et $V_2 = V_2' \cup \{u \in V' : z(u) \text{ est faux }\}$.

Si l'instanciation ne satisfait pas $\mathcal{C}$, alors on a une clause $u \Rightarrow v$ dans $\mathcal{C}$ telle que u soit instancié à vrai et v à faux. Alors $u \in V_1$, $v \in V_2$, et l'adjacence entre u et v ne respecte pas F, donc (V_1, V_2) n'est pas une F-partition avec la graine fixée.

Si l'instanciation satisfait $\mathcal{C}$, alors pour tous les $u \in V_1$ et $v \in V_2$, $\{u, v\} \in E \iff \{x_{f_1(u)}, y_{f_2(v)}\} \in E(F)$. Donc (V_1, V_2) est une F-partition. □

Satisfaire $\mathcal{C}$ revient à trouver un ensemble de sommets V'' du graphe orienté $G' = (V', A)$, avec $A = \{(u, v) : u \Rightarrow v \in \mathcal{C}\}$, tel que $\forall u \in V'', \forall v \in V' \setminus V'', (u, v) \notin A$. Soit $\mathcal{F}$ la famille des sous-ensembles de V' satisfaisant la propriété précédente.

Lemme 6.23. *La famille $\mathcal{F}$ a les propriétés suivantes :*

1. $\emptyset \in \mathcal{F}$, $V' \in \mathcal{F}$.
2. *Si $X, Y \in \mathcal{F}$, alors $X \cup Y \in \mathcal{F}$ et $X \cap Y \in \mathcal{F}$.*

Démonstration. On voit immédiatement que les ensembles $\emptyset$ et V' respectent la propriété. Si X et Y respectent la propriété, alors il n'y a pas d'arc entre X et $V' \setminus X$, ni entre Y et $V' \setminus Y$. Il n'y a donc pas d'arc entre $X \cup Y$ et $V' \setminus (X \cup Y)$. Il n'y a non plus pas d'arc entre $X \cap Y$ et $V' \cup (X \setminus Y) \cup (Y \setminus X)$. Donc $X \cup Y \in \mathcal{F}$ et $X \cap Y \in \mathcal{F}$. □

Soit G' le graphe obtenu en calculant la fermeture transitive G, et en contractant tous les sommets équivalents. Ce graphe, qui est un graphe dirigé acyclique, est une représentation de toutes les F-partitions du graphe avec la graine choisie : chaque sous-ensemble V' de sommets de G' tel qu'il n'y ait aucune arête sortante de V' dans G' correspond à une F-partition différente.

On peut calculer $f_1(u)$ et $f_2(u)$ pour tous les $u \in V'$ en temps $O(n^2)$. Puis le graphe G' peut être construit en temps $O(n^2)$. Pour trouver toutes les F-partitions propres de G, il suffit d'essayer toutes les graines possibles; il y en a $O(n^{k+l})$. Pour trouver toutes les F-partitions de G, il suffit de trouver toutes les F'-partitions propres, pour F' sous-graphe de F. Pour un F fixé, il y en a un nombre fixé.

Théorème 6.24. *Soit $F = (X, Y, E_f)$ un graphe biparti. Une structure représentant toutes les F-partitions de G peut être calculée en temps $O(n^{|X|+|Y|+2})$.*

6.4 Classes de graphes définies par des extensions de sommets

Pour la décomposition modulaire, la décomposition en coupes et la décomposition bi-joint, les classes de graphes entièrement décomposables étaient de largeur de clique bornée, et avaient une caractérisation par extensions de sommets. Les graphes décomposables par les décompositions k-modulaire et k-bimodulaires sont également de largeur de clique bornée.

Dans cette section, on étudiera les classes de graphes définies par certaines extensions de sommets. On verra qu'il suffit de peu d'extensions pour que la classe de graphes devienne de largeur de clique non bornée.

Jean-Marc Lanlignel parle d'une généralisation de la décomposition en coupes qu'il appelle la c-décomposition. Il part de l'observation qu'un graphe premier par la décomposition en coupes n'est pas forcément premier par la décomposition modulaire : il peut s'agir d'un graphe premier pour la décomposition modulaire auquel on a ajouté un sommet universel (ce qui est le cas par exemple du gem). La c-décomposition consiste à faire récursivement une décomposition modulaire du graphe, puis une décomposition en coupes sur les graphes premiers de la décomposition modulaire, ainsi de suite jusqu'à ce que tous les graphes obtenus soient premiers par la décomposition modulaire et la décomposition en coupes.

Nous allons voir que la classe des graphes entièrement décomposables par cette décomposition n'est pas de largeur de clique bornée.

On s'intéresse à des classes de graphes définies par certaines extensions de sommets simples. Soit u un sommet d'un graphe $G = (V, E)$. On cherche à ajouter un sommet v au graphe tel que v ne distingue pas deux sommets qui n'étaient pas distingués par u. Pour ajouter v il suffit donc de connaître le voisinage de u. Il y a 8 possibilités pour cela, selon que v soit adjacent ou non à $\{u\}$, $N_G(u)$ ou $V \setminus N_G[u]$.

Les cas déjà rencontrés sont : l'ajout d'un vrai jumeau, l'ajout d'un faux jumeau, l'ajout d'un sommet pendant, l'ajout d'un vrai anti-jumeau et l'ajout d'un faux anti-jumeau. L'ajout d'un sommet anti-pendant est l'ajout d'un sommet adjacent à tous les sommets sauf u. Enfin, les deux derniers cas, les cas dégénérés puisque l'ajout est indépendant du u choisi, sont l'ajout d'un sommet isolé, et l'ajout d'un sommet universel.

La classe des cographes, des graphes distances héréditaires, et des graphes sans C_5, taureau, gem et co-gem induits peuvent être définies de cette manière.

On notera par TT l'extension ajout d'un vrai jumeau, FT l'ajout d'un faux jumeau, PV l'ajout d'un sommet pendant, TA pour l'ajout d'un vrai anti-jumeau, FA pour l'ajout d'un faux anti-jumeau, CV pour l'ajout d'un sommet anti-pendant, DV l'ajout d'un sommet dominant et enfin par IV l'ajout d'un sommet isolé.

Soit $\mathcal{O} \subseteq \{\mathrm{TT}, \mathrm{FT}, \mathrm{PV}, \mathrm{TA}, \mathrm{FA}, \mathrm{CV}, \mathrm{IV}, \mathrm{DV}\}$. On définit par $\mathcal{G}_{\{\mathcal{O}\}}$ la classe des graphes qui peuvent être obtenus à partir d'un sommet isolé, avec des extensions de sommets dans $\mathcal{O}$.

On rappelle que :

– $\mathcal{G}_{\{\mathrm{TT,FT}\}}$ est la classe des cographes,
– $\mathcal{G}_{\{\mathrm{TT,FT,PV}\}}$ est la classe de graphes distances héréditaires,
– $\mathcal{G}_{\{\mathrm{TT,FT,CV}\}}$ est la classe des graphes complémentaires des distances héréditaires,
– $\mathcal{G}_{\{\mathrm{TT,FT,TA,FA}\}}$ est la classe des graphes sans C_5, taureau, gem et co-gem induit,
et donc sont de largeur de clique bornée.

Théorème 6.25. *La classe obtenue par les extensions « ajout d'un sommet dominant » et « ajout d'un sommet pendant » (c'est à dire la classe $\mathcal{G}_{\{\mathrm{PV,DV}\}}$) est de largeur modulaire non bornée (et est de largeur de clique non bornée).*

Démonstration. Supposons l'inverse. Soit k la largeur modulaire de la classe.

Soit $s > 2k$ et soit $G_s = (V, E)$ le graphe suivant. V peut être partitionné en une clique $U = \{u_1, \ldots, u_s\}$, et n ensembles $P_i = \{v_{i,1}, \ldots, v_{i,s}\}$ induisant un chemin $(v_{i,1}, \ldots, v_{i,s})$. Chaque u_l est adjacent à tous les $v_{i,j}$ avec $i \in \{1, \ldots, s\}$ et $j \in \{1, \ldots, l\}$. G_s n'a pas d'autres arêtes.

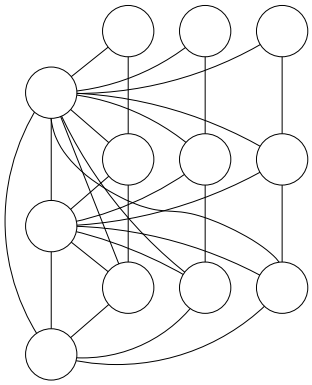


Fig. 6.3 : – Le graphe G_3.

Pour tout n, G_s est dans la classe $\mathcal{G}_{\{\mathrm{PV,DV}\}}$. Il peut être construit à partir d'un ensemble stable de taille s, en ajoutant successivement un sommet dominant et n sommets pendants.

Soit $\mathcal{F}$ une décomposition k-modulaire de G_s. Soit $V_a, V_b, V_c \in \mathcal{F}$ avec $V_a = V_b \cup V_c$ et $V_b \cap V_c = \emptyset$, tels que V_a contienne un P_l, et que V_b et V_c ne contiennent pas P_l.

Proposition. *$|U \setminus V_a| \leq k$, et $|U \cap V_a| > k$.*

Démonstration. Si deux sommets $v_{l,i}$ et $v_{l,j}$ de P_l sont dans la même classe de V_a, alors tous les sommets u_m, avec $m \in \{i, \ldots, j-1\}$, sont dans V_a, car ils distinguent $v_{l,i}$ et $v_{l,j}$.

Soit $a \in \{1, \ldots, s\}$ tel que $u_i \notin V_a$. Alors il n'y a pas de classe de V_a telle que $V_a \cap \{u_{l,1}, \ldots, u_{l,a}\} \neq \emptyset$ et $V_a \cap \{u_{l,a+1}, \ldots, u_{l,s}\} \neq \emptyset$.

Supposons que $|U \setminus V_a| > k$, et soit $a_1 < a_2 < \ldots < a_{k+1}$ tels que pour tout $i \in \{1, \ldots, k+1\}$, $u_{a_i} \notin V_a$. Soit $a_0 = 0$ et $a_{k+2} = s$. Alors pour tout $i, j \in \{0, 1, \ldots, k+1\}$, $i \neq j$, aucun sommet de $\{u_{a_i+1}, u_{a_i+2}, \ldots, u_{a_{i+1}}\}$ n'est dans la même classe de V_a que $\{u_{a_i}, u_{a_i+1}, \ldots, u_{a_{i+1}}\}$. Comme pour tout $i \in \{0, 1, \ldots, k\}$, $a_{i+1} > a_i$ (notez que a_{k+1} peut être égal à a_{k+2}), V_a aurait strictement plus de k classes. Contradiction. □

Proposition. *Pour chaque* $r \in \{1, \ldots, s\}$, $P_r \cap V_a \neq \emptyset$.

Démonstration. Supposons qu'il existe r tel que $P_r \cap V_a = \emptyset$. Alors chaque sommet dans $U \cap V_a$ serait dans une classe différente. Contradiction car il y aurait au moins $k+1$ classes dans V_a. Donc pour chaque $r \in \{1, \ldots, s\}$, $P_r \cap V_a \neq \emptyset$. □

Proposition. *Il y a au moins* $k+1$ P_i *qui sont inclus dans* V_a.

Démonstration. Supposons maintenant qu'il y ait plus de $k+1$ P_i qui ne sont pas entièrement inclus dans V_a. Il y aurait donc au moins une classe dans V_a pour chacun de ces P_i, pour les sommets adjacents a un sommet dans $P_i \setminus V_a$. Contradiction car V_a aurait plus de k classes. Il y a au plus k P_i qui ne sont pas entièrement inclus dans V_a, et donc il y a au moins $k+1$ P_i qui sont inclus dans V_a. □

Nous arrivons maintenant à une contradiction. Par notre choix de V_a, V_b et V_c, aucun P_i n'est entièrement inclus dans V_b ou V_c. Il y a donc plus de $k+1$ P_i qui peuvent être partitionnées en $\{P_i \cap V_b, P_i \cap V_c\}$. Il y a au moins une classe dans V_b pour chacun de ces P_i, pour les sommets adjacents à un sommet dans $P_i \cap V_c$. Donc V_b a au moins $k+1$ classe. Contradiction. □

La c-décomposition décompose complètement la classe $\mathcal{G}_{\{\text{PV,DV}\}}$, donc la classe des graphes complètement décomposables par cette décomposition a une largeur de clique non bornée. Ceci est particulièrement intéressant, car ce n'est le cas d'aucune des autre décompositions étudiées dans ce mémoire.

Soit $\mathcal{O} \subseteq \{\text{TT}, \text{FT}, \text{PV}, \text{TA}, \text{FA}, \text{CV}, \text{IV}, \text{DV}\}$. Soit $\mathcal{G}'_{\mathcal{O}}$ la fermeture héréditaire de $\mathcal{G}_{\mathcal{O}}$ (c'est à dire l'ensemble des graphes qui sont des sous-graphe induit d'un graphe dans $\mathcal{G}_{\mathcal{O}}$). La largeur de clique de la classe $\mathcal{G}'_{\mathcal{O}}$ est la largeur de clique de $\mathcal{G}_{\mathcal{O}}$.

Soit $o \in \{\text{TT}, \text{FT}, \text{PV}, \text{TA}, \text{FA}, \text{CV}, \text{IV}, \text{DV}\}$. On dit qu'une classe de graphes est stable par o si toute les extensions o d'un graphe dans la classe est dans la classe.

On peut faire les observations suivantes :

Observation.

1. $\mathcal{G}_{\{\text{PV,TT,FT,IV}\}} = \mathcal{G}_{\{\text{PV,TT FT}\}}$.
2. $\mathcal{G}_{\{\text{TT,FT,TA,FA,IV,DV}\}} = \mathcal{G}_{\{\text{TT FT,TA,FA}\}}$.

Ce qui veut dire que si on ajoute un sommet isolé à un graphe distance héréditaire, on a toujours un graphe distance héréditaire, et si on ajoute un sommet isolé ou un sommet dominant à un cographe, on obtient un cographe.

Observation. *Soit* $\mathcal{O} \subseteq \{\text{TT}, \text{FT}, \text{PV}, \text{TA}, \text{FA}, \text{CV}, \text{IV}, \text{DV}\}$.

1. *Si* PV $\in \mathcal{O}$ *et* TA $\in \mathcal{O}$, *alors* $\mathcal{G}'_{\mathcal{O}}$ *est stable par* DV.
2. *Si* PV $\in \mathcal{O}$ *et* FA $\in \mathcal{O}$, *alors* $\mathcal{G}'_{\mathcal{O}}$ *est stable par* DV.
3. *Si* PV $\in \mathcal{O}$ *et* CV $\in \mathcal{O}$, *alors* $\mathcal{G}'_{\mathcal{O}}$ *est stable par* DV.

Si on ajoute un sommet u pendant à un sommet du graphe, puis un sommet v pendant à v, et enfin un vrai/faux anti-jumeau w à v, alors w sera adjacent à tous les sommets d'origine (de même si w est un sommet anti-pendant). Comme la classe $\mathcal{G}'_{\mathcal{O}}$ est héréditaire, elle est donc stable par l'opération DV.

Le fait d'autoriser en plus l'extension TA, FA ou CV en plus des extensions TT, FT et PV nous donne donc une classe de largeur de clique non bornée.

Observation. *Soit* $\mathcal{O} \subseteq \{\mathrm{TT}, \mathrm{FT}, \mathrm{PV}, \mathrm{TA}, \mathrm{FA}, \mathrm{CV}, \mathrm{IV}, \mathrm{DV}\}$ *tel que* $\{\mathrm{TT}, \mathrm{FT}, \mathrm{PV}\} \subseteq \mathcal{O}$ *et* $\mathcal{G}_{\mathcal{O}}$ *soit une super-classe propre des graphes distances héréditaires. Alors* $\mathcal{G}_{\mathcal{O}}$ *est de largeur de clique non bornée.*

De même on a :

Observation. *Soit* $\mathcal{O} \subseteq \{\mathrm{TT}, \mathrm{FT}, \mathrm{PV}, \mathrm{TA}, \mathrm{FA}, \mathrm{CV}, \mathrm{IV}, \mathrm{DV}\}$ *tel que* $\{\mathrm{TT}, \mathrm{FT}, \mathrm{CV}\} \subseteq \mathcal{O}$ *et* $\mathcal{G}_{\mathcal{O}}$ *soit une super-classe propre de la classe des graphes complémentaires des distances héréditaires. Alors* $\mathcal{G}_{\mathcal{O}}$ *est de largeur de clique non bornée.*

Observation. *Soit* $\mathcal{O} \subseteq \{\mathrm{TT}, \mathrm{FT}, \mathrm{PV}, \mathrm{TA}, \mathrm{FA}, \mathrm{CV}, \mathrm{IV}, \mathrm{DV}\}$ *tel que* $\{\mathrm{TT}, \mathrm{FT}, \mathrm{TA}, \mathrm{FA}\} \subseteq \mathcal{O}$ *et* $\mathcal{G}_{\mathcal{O}}$ *soit une super-classe propre des graphes sans* C_5*, taureau, gem et co-gem induit. Alors* $\mathcal{G}_{\mathcal{O}}$ *est de largeur de clique non bornée.*

Les seules classes de graphes de largeur de clique bornée, définies par des extensions de sommets données, sont donc des sous-classes des graphes distances héréditaires, de leurs compléments, et des graphes sans C_5, taureau, gem et co-gem induit.

Conclusion

Le travail présenté partait de l'idée d'utiliser une décomposition pour résoudre efficacement un problème. Il était initialement porté sur la décomposition modulaire, puis sur différentes généralisations et enfin sur la décomposition en cliques.

Nous avons montré comment la décomposition modulaire peut servir pour résoudre certains problèmes tels que DOMINATION, FEEDBACK VERTEX SET, NOMBRE DE SÉPARATEURS MINIMAUX... Nous nous sommes servi de cette décomposition pour obtenir des algorithmes linéaires pour les problèmes ENSEMBLE STABLE, CLIQUE, COLORATION et PARTITION EN CLIQUES sur les graphes sans P_5 et gem induit. Nous avons ensuite fait le travail sur la décomposition en coupes, domaine qui n'a pas souvent été traité jusqu'à présent. Cela a abouti en particulier à un algorithme pour calculer une coloration minimum d'un graphe en utilisant sa décomposition en coupes. Enfin nous avons montré qu'une nouvelle classe de problèmes de partitionnement, définissables en logique monadique du second ordre, peuvent être résolus en temps polynomial sur les graphes de largeur de clique bornée.

Du point de vue théorique, nous avons avancé dans l'étude de la décomposition bi-joint, une nouvelle généralisation de la décomposition modulaire. Nous avons vu comment cette décomposition peut servir dans le cas des graphes sans gem et co-gem induit. Enfin, nous avons défini deux nouvelles généralisations, les décompositions k-modulaire et k-bimodulaire, et montré une similitude avec la décomposition en cliques.

On présente maintenant quelques directions de recherche, ayant pour objectif d'apporter de nouvelles connaissances dans le sujet.

Il semble que les généralisations de la décomposition modulaire se rapprochent de la décomposition en cliques. Cependant une différence fondamentale de la décomposition en cliques est qu'elle n'est pas unique, alors que les décomposition modulaire, en coupes et bi-joint le sont. C'est aussi le cas pour la décomposition NLC, k-modulaire, k-bimodulaire et celle de la largeur de rang. Une perspective serait de chercher d'autres généralisations de la décomposition modulaire qui gardent une certaine unicité (comme la décomposition en coupes et la décomposition bi-joint). On pourrait aussi étudier des décompositions dont les graphes complètement décomposables ne soient pas de largeur de clique bornée, comme la c-décomposition.

On a vu que les décompositions nous permettent d'obtenir des algorithmes de reconnaissance et des algorithmes polynomiaux pour des problèmes NP-complets en général. Mais pour cela il faut préalablement trouver des classes de graphes qui se prêtent bien à

une certaine décomposition. Un autre axe de recherche, d'un point de vue théorique, serait d'exhiber d'autres classes.

Les modules et la décomposition modulaire peuvent être généralisés à d'autres structures (graphes orientés, hypergraphes et k-structures), et c'est également le cas de la décomposition en cliques. Une question ouverte est de savoir s'il est possible de définir les décompositions en coupes et bi-joint sur d'autres structures, et notamment s'il est possible de définir une décomposition bi-joint sur les graphes orientés.

Enfin, on dispose d'un ensemble de paramètres similaires à la largeur de clique : la largeur NLC, la largeur de rang, la largeur modulaire et bi-modulaire. Il existe un algorithme polynomial pour décider si la largeur de rang est bornée par un k fixé. Malheureusement la largeur de rang ne nous donne pas une bonne approximation de la largeur de clique. Ceci est gênant pour l'utilisation pratique des algorithmes sur les graphes de largeur de clique bornée. D'un autre côté, on sait que la largeur de clique est au plus deux fois la largeur NLC ou la largeur modulaire. Une question ouverte est de savoir si la décision pour un autre paramètre plus « proche » de la largeur de clique, comme la largeur NLC ou modulaire, serait plus facile que la décision pour la largeur de clique.

Index

Liste des symboles

$\mathcal{P}(V)$	Ensembles des sous-ensembles de V
$\mathcal{P}_k(V)$	Ensembles des sous-ensembles de taille k de V
$X \Delta Y$	Différence symétrique : $(X \setminus Y) \cup (Y \setminus X)$
$\langle \ldots \rangle$	Multi-ensemble
$f_{\|x \to y}$	Redéfinition de la fonction f en x
V	Ensemble des sommets du graphe
E	Ensemble des arêtes du graphe
n	Nombre de sommets dans le graphe
m	Nombre d'arêtes dans le graphe
$N(v)$	Ensembles des sommets voisins de v
$N[v]$	$N(v) \cup \{v\}$
$d(v)$	Degré du sommet v
$G[X]$	Sous graphe de G induit par X
$G - v$	$G[V \setminus \{v\}]$
$\overline{G}$	Complémentaire de G
$I(G)$	Graphe d'incidence de G
$\alpha(G)$	Taille d'un plus grand ensemble stable de G
$\alpha_w(G)$	Poids maximum d'un ensemble stable de G
$\chi(G)$	Nombre chromatique de G
$\chi_w(G)$	Nombre chromatique pondéré de G
$\omega(G)$	Taille d'une plus grande clique de G
$\omega_w(G)$	Poids maximum d'une clique de G
$\gamma(G)$	Taille d'un plus petit ensemble dominant de G

$\gamma_{\mathrm{c}}(G)$	Taille d'un plus petit ensemble dominant connexe de G
$\gamma_{\mathrm{cl}}(G)$	Taille d'un plus petite clique dominante de G
$\gamma_{\mathrm{i}}(G)$	Taille d'un plus petit ensemble dominant indépendant de G
$\gamma_{\mathrm{t}}(G)$	Taille d'un plus petit ensemble dominant total de G
$s(G)$	Nombre de séparateurs minimaux de G
$\mathrm{twd}(G)$	Largeur arborescente de G
$\mathrm{cwd}(G)$	Largeur de clique de G
NLC-$\mathrm{wd}(G)$	Largeur NLC de G
$\mathrm{rwd}(G)$	Largeur de rang de G
$\oplus$	Union disjointe de graphes
$\cdot_i$	Graphe d'un sommet d'étiquette i
$\eta_{i,j}$, $\rho_{i \to j}$	Opérations de la largeur de clique
$\times_S$, ρ_R	Opérations de la largeur NLC
$\langle A, R_1, \ldots, R_k \rangle$	Structure de domaine A avec les relations $R_1, \ldots, R_k$
$\tau_1(G)$	Structure τ_1 de G
$\tau_2(G)$	Structure τ_2 de G

Bibliographie

[ACP87] S. Arnborg, D. G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM Journal of Algebraic and Discrete Methods* 8 (1987) 277–284

[ALS91] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, *Journal of Algorithms* 12 (1991) 308–340

[APT79] B. Aspvall, M. F. Plass, R. E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* 8 (1979) 121–123

[BM84] H.-J. Bandelt, H. M. Mulder, Distance-hereditary graphs, *Journal of Combinatorial Theory Series B* 41 (1986) 182–208

[Ber61] C. Berge, Sur une conjecture relative au probleme des codes optimaux, *Comm. 13eme Assemble Generale de l'URSI*, Tokyo (1961)

[Bix84] R. Bixby, A composition for perfect graphs, *Annals of Discrete Mathematics* 21 (1984) 221–224

[Bod89] H. L. Bodlaender, Achromatic number is NP-complete for cographs and interval graphs, *Information Processing Letters*, 31 (1989) 135–138.

[Bod90] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *Journal of Algorithms* 11 (1990) 631–643

[Bod93] H. L. Bodlaender, A tourist guide through treewidth, *Acta Cybernetica* 11 (1993) 1–23

[Bod96] H. L. Bodlaender, A Linear Time Algorithm for Finding Tree-decompositions of Small Treewidth, *SIAM Journal on Computing* 25 (1996) 1305–1317

[BBK03] H. L. Bodlaender, A. Brandstädt, D. Kratsch, M. Rao, J. Spinrad, Linear time algorithms for some NP-complete problems on (P_5,gem)-free graphs, *Theoretical Computer Science* 349 (2005) 2–21

[BBF02] H.L. Bodlaender, H.J. Broersma, F.V. Fomin, A.V. Pyatkin, G.J. Woeginger, Radio labeling with pre-assigned frequencies, *Proceedings of the 10th European Symposium on Algorithms (ESA'2002)*, LNCS 2461 (2002) 211-222

[BJ00] H.L. Bodlaender, K. Jansen, On the complexity of the maximum cut problem, *Nordic Journal of Computing* 7 (2000) 14–31.

[BT03] H. L. Bodlaender, U. Rotics, Computing the treewidth and the minimum fill-in with the modular decomposition, *Algorithmica* 36 (2003) 375–408.

[Bou87] A. Bouchet, Reducing prime graphs and recognizing circle graphs, *Combinatorica* 7 (1987) 243–254

[BK02] A. Brandstädt, D. Kratsch, On the structure of (P_5,gem)-free graphs, *Discrete Applied Mathematics* 145 (2005) 155–166.

[BLM04] A. Brandstädt, H.-O. Le, R. Mosca, Gem And Co-Gem-Free Graphs Have Bounded Clique-Width, *International Journal of Foundations of Computer Science* 15 (2004) 163–185

[BLM05] A. Brandstädt, H. -O. Le, R. Mosca, Chordal co-gem-free and (P_5, gem)-free graphs have bounded clique-width, *Discrete Applied Mathematics* 145 (2005) 232–241

[BLS99] A. Brandstädt, V.B. Le, J. Spinrad, Graph Classes : A Survey, *SIAM Monographs on Discrete Mathematics and Applications* 3, SIAM, Philadelphia (1999)

[BCH03] A. Bretscher, D. G. Corneil, M. Habib, C. Paul, A Simple Linear Time LexBFS Cograph Recognition Algorithm. *Proceedings of WG 2003*, LNCS 2880 (2003) 119–130

[Bun71] P. Buneman, The recovery of trees from measures of dissimilarity Dans : *Mathematics in Archaeological and Historical Sciences*, Editeurs : F.H. Hodson, D.G. Kendall, P. Tautu, Edimburg University Press, (1971) 387–395

[BU84] M. Burlet, J.-P. Uhry, Parity graphs, *Annals of Discrete Mathematics* 21 (1984) 253–277

[CHM02] C. Capelle, M. Habib, F. de Montgolfier, Graph Decompositions and Factorizing Permutations, *Discrete Mathematics & Theoretical Computer Science* 5 (2002) 55–70

[CHM81] M. Chein, M. Habib, M. C. Maurer, Partitive hypergraphs, *Discrete Mathematics* 37 (1981) 35–50

[CRS02] M. Chudnovsky, N. Robertson, P.D.Seymour, R.Thomas, The strong perfect graph theorem, *Annals of Mathematics*, à paraître

[Chv75] V. Chvátal, On certain polytopes associated with graphs, *Journal of Combinatorial Theory Series B* 18 (1975) 138–154

[CS99] S. Cicerone, D. Di Stefano, On the extension of bipartite graphs to parity graphs, *Discrete Applied Mathematics* 95 (1999) 181–195

[Coo71] S. A. Cook, The complexity of theorem proving procedures, *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing*, ACM, New York (1971) 151–158

[CHL00] D. G. Corneil, M. Habib, J.-M. Lanlignel, B. A. Reed, U. Rotics, Polynomial Time Recognition of Clique-Width ≤ 3 Graphs (Extended Abstract), *Proceedings LATIN 2000*, LNCS 1776 (2000) 126–134

[CLS81] D. G. Corneil, H. Lerchs, L. Stewart Burlingham, Complement reducible graphs, *Discrete Applied Mathematics* 3 (1981) 163–174

[CPS84] D.G. Corneil, Y. Perl, L.K. Stewart, Cographs : recognition, applications, and algorithms, *Congressus Numerantium* 43 (1984) 249–258

[CPS85] D. G. Corneil, Y. Perl, L. Stewart Burlingham, A linear recognition algorithm for cographs, *SIAM Journal on Computing* 14 (1985) 926–934

[CR05] D. G. Corneil, U. Rotics, On the Relationship Between Clique-Width and Treewidth, *SIAM Journal on Computing*, 34 (2005) 825–847

[Cou92] B. Courcelle, The monadic second-order logic of graphs III : tree-decompositions, minor and complexity issues, *Informatique Théorique et Applications* 26 (1992) 257–286

[Cou94] B. Courcelle, The monadic second-order logic of graphs VI : On several representations of graphs by relational structures, *Discrete Applied Mathematics* 54 (1994) 117–149

[CER93] B. Courcelle, J. Engelfriet, G. Rozenberg, Handle-rewriting hypergraph languages, *Journal of Computer and System Sciences* 46 (1993) 218-270

[CMR00] B. Courcelle, J. A. Makowsky, U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, *Theory of Computing Systems* 33 (2000) 125–150

[CO00] B. Courcelle, S. Olariu, Upper bounds to the clique width of graphs, *Discrete Applied Mathematics* 101 (2000) 77–114

[CO04] B. Courcelle, S. Oum, Vertex-Minors, Monadic Second-Order Logic, and a Conjecture by Seese, *Journal of Combinatorial Theory Series B*, à paraître *Manuscript 2004*

[CH94] A. Cournier, M. Habib, A new linear algorithm for modular decomposition, *Trees in Algebra and Programming - CAAP '94*, LNCS 787 (1994) 68–84

[CI98] A. Cournier, P. Ille, Minimal indecomposable graphs, *Discrete Mathematics* 183 (1998) 61–80

[CJS72] D. D. Cowan, L. O. James, R. G. Stanton, Graph decomposition for undirected graphs, *3rd S-E Conf. Combinatorics, Graph Theory and computing, Utilitas Math*, F. Hoffman, R. B. Levow eds (1972) 281–290

[Cun82] W. H. Cunningham, Decomposition of directed graphs, *SIAM Journal on Algebraic and Discrete Methods* 3 (1982) 214–228

[Cun83] W. H. Cunningham, Decomposition of submodular functions, *Combinatorica* 3 (1983) 53–68

[CE80] W. H. Cunningham, J. Edmonds, A combinatorial decomposition theory, *Canad. J. Math* 32 (1980) 734–765

[Dah00] E. Dahlhaus, Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition, *Journal of Algorithms* 36 (2000) 205–240

[DGM96] E. Dahlhaus, J. Gustedt, R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, *Journal of Algorithms* 41 (2001) 360–387

[EF95] H.-D. Ebbinghaus, J. Flum, Finite model theory, Springer–Verlag, Berlin (1995)

[EM94] A. Ehrenfeucht, R. M. McConnell, A k-structure generalization of the theory of two-structures, *Theoretical Computer Science*, 132 (1994) 209–227

[ER90] A. Ehrenfeucht, G. Rozenberg, Primitivity is Hereditary for 2-Structures, *Theoretical Computer Science*, 70 (1990) 343–358

[EGW01] W. Espelage, F. Gurski, E. Wanke, How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time, *Proceedings of WG 2001*, LNCS 2204, 117–128, Springer-Verlag, Berlin, 2001

[EGW03] W. Espelage, F. Gurski, E. Wanke, Deciding clique-width for graphs of bounded tree-width, *Journal of Graph Algorithms and Applications* 7 (2003) 141–180

[EPL72] S. Even, A. Pnueli, A. Lempel, Permutation graphs and transitive graphs, *Journal of the Association for Computing Machinery* 19 (1972) 400–410

[FM71] M. J. Fischer, A. R. Meyer, Boolean matrix multiplication and transitive closure, *In Conference Record* (1971)

[FHM04] J.-L. Fouquet, M. Habib, F. de Montgolfier, J. -M. Vanherpe, Bimodular Decomposition of Bipartite Graphs, *Proceedings of WG 2004*, LNCS 3353 (2004) 117–1280

[Fra84] R. Fraïssé, L'intervalle en théorie des relations, ses généralisations, filtre intervallaire et clôture d'une relation, *Annals of Discrete Mathematics* 23 (1984) 313–341

[Fra76] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, *Proceedings of the Fifth British Combinatorial Conference (Univ. Aberdeen, Aberdeen*, (1975) 211–226, *Congressus Numerantium* No. XV, Utilitas Math., Winnipeg, Man. (1976)

[FV59] S. Feferman, R. Vaught, The first order properties of algebraic systems, *Fundamenta Mathematicae* 47 (1959) 57–103

[FRR06] M. R. Fellows, F. A. Rosamond, U. Rotics, S. Szeider, Clique-width Minimization is NP-hard, *Proceedings of STOC 2006*, 38th ACM Symposium on Theory of Computing, Seattle, Washington, USA, à paraître

[FG65] D. R. Fulkerson, O. A. Gross., Incidence matrices and interval graphs, *Pacific Journal of Math.* 15 (1965) 835–855

[GHS89] C. P. Gabor, W. L. Hsu, K. J. Supowit, Recognizing circle graphs in polynomial time, *Journal of the ACM* 36 (1989) 435–473

[Gal67] T. Gallai, Transitiv orientierbare Graphen, *Acta Mathematica Academiae Scientiarum Hungaricae* 18 (1967) 25–66

[GJ78] M. R. Garey, D. S. Johnson, Computers and intractability : a guide to the theory of NP-completeness, W.H. Freeman and Company (1978)

[GC03] M. U. Gerber, D. Kobler, Algorithms for vertex-partitioning problems on graphs with fixed clique-width, *Theoretical Computer Science* 299 (2003) 719–734

[GR97] V. Giakoumakis, I. Rusu, Weighted parameters in $(P_5, \overline{P_5})$-free graphs, *Discrete Applied Mathematics* 80 (1997) 255–261

[GV97] V. Giakoumakis, J.-M. Vanherpe, On Extended P_4-Reducible and Extended P_4-Sparse Graphs, *Theoretical Computer Science* 180 (1997) 269–186

[Gol78] M.C. Golumbic, Trivially perfect graphs, *Discrete Mathematics* 24 (1978) 105–107

[Gol80] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, *Academic Press, New York* (1980)

[GLS84] M. Grötschel, L. Lovász, A. Schrijver, Polynomial algorithms for perfect graphs, *Annals of Discrete Mathematics* 21 (1984) 325–356

[GW00] Frank Gurski, Egon Wanke, The Tree-Width of Clique-Width Bounded Graphs Without $K_{n,n}$, *Proceedings of WG 2000*, LNCS 1938 (2000) 196–205

[GW05] Frank Gurski, Egon Wanke, Minimizing NLC-width is NP-complete, *Proceedings of WG 2005*, LNCS 3787 (2005) 69–80

[Hab81] M. Habib, Substitution des Structures Combinatoires, Théorie et Algorithmes, Thèse d'État, Université Pierre et Marie Curie, (1981)

[HP05] M. Habib, C. Paul, A simple linear time algorithm for cograph recognition, *Discrete Applied Mathematics* 145(2) (2005) 183–197

[HM79] M. Habib, M. C. Maurer, On the X-join decomposition for undirected graphs, *Discrete Applied Mathematics* 1 (1979) 201–207

[HMP04] M. Habib, F. de Montgolfier, C. Paul, A Simple Linear-Time Modular Decomposition Algorithm for Graphs, Using Order Extension, *Proceedings of SWAT 2004*, LNCS 3111 (2004) 187–198

[HM90] P. Hammer, F. Maffray, Completely separable graphs, *Discrete Applied Mathematics* 27 (1990) 85–99

[Her99] A. Hertz, On perfect switching classes, *Discrete Applied Mathematics* 94 (1999) 3–7

[Hoa83] C. T. Hoàng, A class of perfect graphs, M.Sc. Thesis, School of Computer Science, McGill University, Montreal (1983)

[Hoa94] C. T. Hoàng, Efficient algorithms for minimum weighted colouring of some classes of perfect graphs, *Discrete Applied Mathematics* 55 (1994) 133–143

[JO92] B. Jamison , S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Applied Mathematics* 35 (1992) 115–129

[JS97] K. Jansen, P. Scheffler, Generalized coloring for tree-like graphs, *Discrete Applied Mathematics* 75 (1997) 135–155

[Joh98] Ö. JOHANSSON, Clique-decomposition, NLC-decomposition, and modular decomposition - relationships and results for random graphs, *Congressus Numerantium* 132 (1998) 39–60

[Joh00] Ö. JOHANSSON, NLC2-Decomposition in Polynomial Time, *International Journal of Foundations of Computer Science* 11 (2000) 373–395

[Joh01] Ö. JOHANSSON, $\log n$-Approximative NLCk-Decomposition in $O(n^{2k+1})$ Time, *Proceedings of WG 2001*, LNCS 2204 (2001) 229–240

[KR01] D. KOBLER, U. ROTICS, Edge dominating set and colorings on graphs with fixed clique-width, *Discrete Applied Mathematics* 126 (2003) 197–221

[Lan01] J.-M. LANLIGNEL, Autour de la décomposition en coupes, Thèse, Université de Montpellier II, LIRMM (2001)

[Läu68] H. LÄUCHLI, A decision procedure for the weak second order theory of linear order, *Logic Colloquium '66*, North Holland, Amsterdam, (1968) 189–197

[Lau93] C. LAUTEMANN, Logical definability of NP-optimization problems with monadic auxiliary predicates, LNCS 702 (1993) 327–339

[Lov72] L. LOVÁSZ, Normal hypergraphs and the perfect graph conjecture, *Discrete Mathematics* 2 (1972) 253–267

[LR04] V. V. LOZIN, D. RAUTENBACH, The Tree- and Clique-width of Bipartite Graphs in Special Classes, *Australasian Journal of Combinatorics* 34 57–68

[Lub87] A. LUBIW, Doubly Lexical orderings of matrices, *SIAM Journal on Computing* 16 (1987) 854–879

[Mak04] J. MAKOWSKY, Algorithmic uses of the Feferman-Vaught theorem, *Annals of Pure and Applied Logic* 126 (2004) 159–213

[MS94] T.-H. MA, J. SPINRAD, An $O(n^2)$ algorithm for undirected split decomposition, *Journal of Algorithms* 16 (1994) 154–160

[MP01] F. MAFFRAY, M. PREISSMANN, A translation of Gallai's paper : 'Transiv orientierbare Graphen', dans *Perfect graphs*, J.L. Ramirez Alfonsin, B.A. Reed, (eds.), Wiley (2001) 25–66

[McC95] R. M. MCCONNELL, An $O(n^2)$ incremental algorithm for modular decomposition of graphs and two-structures, *Algorithmica* 14 (1995) 209–227

[MM05] R. M. MCCONNELL, F. DE MONTGOLFIER, Linear-time modular decomposition of directed graphs, *Discrete Applied Mathematics* 145 (2005) 189–209

[MS99] R. M. MCCONNELL, J. SPINRAD, Modular decomposition and transitive orientation, *Discrete Mathematics* 201 (1999) 189–241

[MR00] C. MCDIARMID, B. A. REED, Channel assignment and weighted coloring, *Networks* 36 (2000) 114-117

[MR84] R. H. MÖHRING, F. J. RADERMACHER, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics* 19 (1984) 257–356

[Mon03] F. de Montgolfier, Décomposition modulaire des graphes : théorie, extensions et algorithmes, Thèse, Université de Montpellier II, LIRMM (2003)

[MR05] F. de Montgolfier, M. Rao, The bi-join decomposition, *Proceedings of ICGT 2005*, Electronic Notes in Discrete Mathematics 22 (2005) 173–177

[MS89] J. H. Muller, J. Spinrad, Incremental Modular Decomposition, *Journal of the ACM*, 36 (1989) 1–19

[Mur71] I. Munro, Efficient determination of the transitive closure of a directed graph, *Information Processing Letters* 1 (1971) 56–58

[NP06] S. D. Nikolopoulos, L. Palios, Minimal Separators in $P4$-sparse Graphs, *Discrete Mathematics* 306 (2006) 381–392

[Oum05] S. Oum, Approximating Rank-width and Clique-width Quickly, *Proceedings of WG 2005*, LNCS 3787 (2005) 49–58

[OS06] S. Oum, P. Seymour, Approximating Clique-width and Branch-width, *Journal of Combinatorial Theory Series B*, à paraître (2006)

[PLE71] A. Pnueli, A. Lempel, S. Even, Transitive orientation of graphs and identification of permutation graphs, *Canadian Journal of Mathematics* 23 (1971) 160–175

[Rao02] M. Rao, Décomposition modulaire de graphes et problèmes NP-complets, Rapport de DEA, Université de Metz (2002)

[Rao04] M. Rao, Coloring a Graph Using Split Decomposition, *Proceedings of WG 2004*, LNCS 3353 (2004) 129–141

[RS00] T. Raschle, K. Simon, On the P_4-components of graphs, *Discrete Applied Mathematics* 100 (2000) 215–235

[RS86] N. Robertson, P. Seymour, Graph minors. II. Algorithmic aspects of treewidth, *Journal of Algorithms* 7 (1986) 309–322

[RTL76] D. Rose, R. Tarjan, G. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM Journal on Computing* 5 (1976) 266–283

[Sab59] G. Sabidussi, The composition of graphs, *Duke Math. J.* 26 (1959) 693–696

[Sch86] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester (1986)

[Sei76] J. J. Seidel, A survey of two-graphs, *Intern. Coll. Teorie Combinatorie (Roma, 1973)*, I, Accad. Naz. Lincei, Rome (1976)

[Sha56] C. E. Shannon, The Zero Error Capacity of a Noisy Channel, *IEEE Transactions on Information Theory*, 2 (1956) 8–19

[She75] S. Shelah, The monadic theory of order, *Annals of Mathematics* 102 (1975) 379–419

[Spi92] J. Spinrad, P_4-trees and substitution decomposition, *Discrete Applied Mathematics* 39 (1992) 263–291

[Spi94] J. Spinrad, Recognition of circle graphs, *Journal of Algorithms* 16 (1994) 264–282

[Sum73] D. P. Sumner, Graphs indecomposable with respect to the X-join, *Discrete Mathematics* 6 (1973) 281–298

[Tar85] R. E. Tarjan, Decomposition by clique separators, *Discrete Mathematics* 55 (1985) 221–232

[TP97] J. A. Telle, A. Proskurowski, Algorithms for Vertex Partitioning Problems on Partial k-Trees, *SIAM Journal on Discrete Mathematics* 10 (1997) 529–550

[Van99] J.-M. Vanherpe, Décomposition et algorithmes efficaces sur les graphes, Thèse, Université de Picardie, LaRIA (1999)

[Viz64] V. G. Vizing, On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.* 3 (1964) 25–30

[Wan94] E. Wanke, k-NLC graphs and polynomial algorithms, *Discrete Applied Mathematics* 54 (1994) 251–266

Décompositions de graphes et algorithmes efficaces

Résumé : Ce mémoire traite de la décomposition modulaire ainsi que différentes de ses généralisations.
Dans un premier temps, on explique comment se servir de décompositions pour résoudre efficacement certains problèmes sur les graphes. En particulier, en utilisant la décomposition modulaire, on obtient des algorithmes linéaires pour les problèmes ENSEMBLE STABLE, CLIQUE, NOMBRE CHROMATIQUE et PARTITION EN CLIQUES sur les graphes sans P_5 et gem induit. On étudie également comment la décomposition en coupes peut servir pour calculer le nombre chromatique, et on exhibe une nouvelle classe de problèmes de partitionnements pour lesquels on peut obtenir des algorithmes polynomiaux sur les graphes de largeur de clique bornée.
Dans un second temps, on s'intéresse à généraliser la décomposition modulaire. On étudiera une nouvelle décomposition appelée décomposition bi-joint. On donne notamment différentes caractérisations des graphes complètement décomposables par cette décomposition, et un algorithme linéaire pour la calculer. On donne également des généralisations paramétrées de la décomposition modulaire, qui s'avèrent relativement proches de la largeur de clique.

Mots clés : Graphe, algorithme, décomposition, module, coupe, bi-joint, largeur de clique.

Graph decompositions and efficient algorithms

Abstract : This thesis deals with the modular decomposition and several of its generalizations.
In a first time we show how graph decompositions can be used to solve efficiently some problems on graphs. We show how the modular decomposition can be used to obtain linear algorithms for INDEPENDENT SET, CLIQUE, CHROMATIC NUMBER and PARTITION INTO CLIQUES on (P_5,gem)-free graphs. We also show how the split decomposition can be used to compute the chromatic number, and we give a new class of vertex partitioning problems which can be solved in polynomial time on graphs of bounded clique width.
In a second time, we are interested to generalize the modular decomposition. We study a new decomposition called the bi-join decomposition. We give in particular several characterizations of completely decomposable graphs, and a linear time decomposition algorithm. We introduce some parametrized generalization of the modular decomposition, and we show that theses generalizations are relatively close to the clique width.

Keywords : Graph, algorithm, decomposition, module, split, bi-join, clique width.