



Représentations dynamiques de graphes

Christophe Crespelle

► To cite this version:

Christophe Crespelle. Représentations dynamiques de graphes. Informatique [cs]. Université Montpellier II - Sciences et Techniques du Languedoc, 2007. Français. NNT : . tel-00402838

HAL Id: tel-00402838

<https://theses.hal.science/tel-00402838>

Submitted on 8 Jul 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ MONTPELLIER II
— SCIENCES ET TECHNIQUES DU LANGUEDOC —

Thèse

présentée au Laboratoire d'Informatique de Robotique
et de Microélectronique de Montpellier

SPÉCIALITÉ : Informatique
Formation Doctorale : Informatique
École Doctorale : Information, Structures, Systèmes

Représentations dynamiques de graphes

par

Christophe Crespelle

Soutenue le 28 septembre 2007, devant le jury composé de :

M. Bruno Courcelle, Professeur, Université de Bordeaux 1 examinateur
M. Michel Habib, Professeur, Université de Paris 7 co-directeur de thèse
M. Dieter Kratsch, Professeur, Université de Metz rapporteur
M. Frédéric Mafray, Directeur de Recherches CNRS, G-SCOP, Grenoble rapporteur
M. Christophe Paul, Chargé de Recherches CNRS, LIRMM, Montpellier ... directeur de thèse
M. Stéphan Thomassé, Professeur, Université de Montpellier 2 examinateur

Remerciements

Mes premiers remerciements vont naturellement à mes encadrants dont l'aide a été essentielle à l'aboutissement de ce mémoire. Merci à Christophe Paul de son investissement constant dans mon travail et de m'avoir accompagné pas à pas dans ma découverte de l'activité de recherche. Je lui dois en grande partie ce que j'en connais aujourd'hui. Que lui soit témoignée ici ma gratitude pour ce qu'il m'a transmis.

Merci à Michel Habib de m'avoir accueilli à Montpellier bien avant que je n'y débute ma thèse. Je lui suis très reconnaissant du soutien qu'il m'a apporté depuis cette époque jusqu'à aujourd'hui. J'ai beaucoup apprécié ses conseils scientifiques et extra-scientifiques chaque fois que j'en avais besoin et je voudrais l'en remercier chaleureusement ici.

J'adresse mes plus vifs remerciements à Dieter Kratsch et Frédéric Maffray qui ont accepté de relire et d'écrire un rapport sur mon manuscrit. Je suis honoré de la considération qu'ils ont ainsi témoigné pour ce travail. Je les remercie de leurs remarques, tant pour l'aide qu'elles m'ont fourni pour améliorer la rédaction de ce document que pour les ouvertures qu'elles m'ont proposées.

Un grand merci aussi à Bruno Courcelle et Stéphan Thommassé d'avoir accepter de prendre part au jury de soutenance. Cette marque d'intérêt m'a touché et je voudrais leur dire l'honneur et le plaisir que cela a été pour moi d'exposer devant eux.

Merci à l'équipe enseignante de l'IUT de Montpellier de m'avoir accueilli si chaleureusement. Un merci particulier à Olivier Cogis pour la bienveillance avec laquelle il a accompagné mes premiers pas d'enseignant. Je garde un excellent souvenir de nos nombreuses discussions, qui m'ont beaucoup appris sur la pédagogie et les rapports humains en général. Merci encore à Jean-Claude Bajard pour sa bonne humeur communicative qui m'a rendu très agréable le travail avec lui.

Je remercie tous les membres de l'équipe VAG du lirmm dont le contact a été très enrichissant pour moi. Merci aux deux Stéphan(e) pour l'engouement et la légèreté qu'ils ont apportés par leur arrivée dans cette équipe. Les côtoyer au laboratoire comme en dehors a été un grand plaisir.

J'adresse également mes remerciements aux membres de l'équipe administrative du lirmm pour leur aide et leur gentillesse.

Au moment de conclure cette thèse, je ne peux m'empêcher de penser à tous mes copains de fac de Montpellier et je voudrais ici les remercier très chaleureusement pour tous les bons moments passés avec eux. Merci à Pascal mon compère de maîtrise, à Lyliia pour sa cuisine réconfortante et ses attentions maternelles, à Jonathan pour son goût du partage et

des rencontres, à Nicolas pour sa convivialité et ses convictions, à Anh-Tuan la forte tête, à Isabelle pour son sens de l'hospitalité et son humeur riante, merci à Joanna et à Marie pour leur folie et leur sens de la fête, à Aurélien pour son humour, son sens de l'amitié et pour m'avoir appris qu'il ne faut pas se leurrer, merci à Fon pour sa gentillesse et son optimisme, merci à Venus et Djouher de m'avoir fait connaître "la maison du bonheur", à Teresa pour la joie permanente qu'elle distille et son sens inné de la fête, merci à Mathias pour les moments de bonheur à tâter du caillou, à Guta pour son réconfort pendant la difficile épreuve de la rédaction, à Mehdi pour m'avoir fait jouer au tapis quand j'étais à terre, à Luc pour avoir ramé sa part dans notre galère des finisseurs de thèses à n'en plus finir. Merci enfin à mes deux jumeaux de thèse, Clément et John avec qui j'ai partagé de manière privilégiée cette période. Merci à Clément pour l'enthousiasme que suscite son caractère passionné et communicatif, pour le renouveau qu'apporte son goût pour l'action et pour son sens de l'accueil et du partage. A John, je veux dire merci pour son ouverture d'esprit, pour son amour du désordre et du non-conformisme, qui m'ont souvent fait l'effet d'une bouffée d'air pur, ainsi que pour les moments de folie qu'on a partagé, ils sont gravés dans ma mémoire.

En pensant à tous ceux dont la rencontre a marqué ma vie Montpelliéraine, je pense inévitablement à Leïla qui me manque beaucoup.

Merci à tous ceux-là de leur amitié et du beau souvenir qu'ils me laissent de ces années.

Merci enfin à ma famille et mes amis d'avoir toujours été là pour moi et de l'avoir été une fois de plus le jour de la soutenance. Le fait de les avoir à mes côtés m'a beaucoup touché et a apporté une grande dimension affective à cet événement important pour moi. Qu'ils sachent que cela a été un immense plaisir d'exposer mon travail devant eux et que leur présence a beaucoup contribué à me laisser de cette journée un superbe souvenir.

Table des matières

1	Représentation des graphes	25
1.1	Performances d'une représentation	25
1.2	Décompositions de graphes	28
1.2.1	Décomposition modulaire	28
1.2.2	Décomposition en coupes	40
1.2.3	Décomposition en composantes triconnexes	54
1.2.4	Décomposition par séparateurs complets	62
1.2.5	Conclusion	64
1.3	Modèles d'intersection	66
1.3.1	Graphes d'intervalles	67
1.3.2	Graphes de permutation	69
1.3.3	Graphes de cordes	71
1.3.4	Graphes d'arcs de cercle	72
1.3.5	Graphes triangulés	73
1.3.6	Graphes planaires	74
1.4	Représentations arborescentes par degrés de liberté	75
1.4.1	PQ -arbres et graphes d'intervalles	75
1.4.2	PC -arbres et graphes d'arcs de cercle de Helly	80
1.4.3	Décomposition modulaire et graphes de permutation	82
1.4.4	Décomposition en coupes et graphes de cordes	85
1.4.5	Décomposition en composantes triconnexes et graphes planaires . .	87
1.5	Conclusion	90
2	Algorithmes dynamiques	93
2.1	Problématique de l'algorithmique dynamique	93
2.2	Représentation et reconnaissance dynamique	97
2.3	Remarques sur la complexité des algorithmes dynamiques	98
2.4	Des listes d'adjacence dynamiques	99
3	Dynamicité de modèles géométriques de graphes	103
3.1	Les graphes de permutation	103
3.2	Les graphes d'intervalles	110
3.2.1	Insertion d'un sommet dans un modèle d'intervalles	111

3.2.2	Algorithme et complexité	113
3.3	L'exemple des graphes planaires	115
4	Décomposition modulaire dynamique	117
4.1	Le cas des graphes quelconques	117
4.2	Les cographes	120
4.2.1	Performance du coarbre	120
4.2.2	Entretien dynamique	122
4.3	Les cographes orientés	125
4.3.1	Définition et premières propriétés	126
4.3.2	Structure de données	129
4.3.3	Opérations dynamiques sur les sommets	130
4.3.4	Opérations dynamiques sur les arcs	147
4.4	Les graphes P_4 -sparse	151
4.5	Les graphes de permutation	155
4.5.1	Suppression de sommet	156
4.5.2	Insertion de sommet	159
4.6	Conclusion	175
4.7	Perspectives	177
4.7.1	Les graphes d'intervalles unitaires	178
4.7.2	Graphes distance héréditaires	181
4.7.3	Graphes triangulés	182
5	Graphes d'intervalles dynamiques	185
5.1	Décomposition modulaire d'un graphe d'intervalles	187
5.1.1	Comportement des graphes d'intervalles vis à vis du quotient et de la substitution d'un graphe à un sommet	187
5.1.2	MD -représentation d'un graphe d'intervalles	188
5.2	PQ -représentation d'un graphe d'intervalles	190
5.2.1	Définition de la PQ -représentation	191
5.2.2	Structure des pointeurs de la PQ -représentation	194
5.2.3	Intervalles des arrangements consécutifs et connexité	197
5.2.4	Restriction, quotient et composition de PQ -représentations	199
5.2.5	Retrait d'un sommet dans un ordre consécutif	206
5.3	Trois représentations des graphes d'intervalles	207
5.3.1	Implémentations	207
5.3.2	Modèle d'intervalles minimal et PQ -représentation	208
5.3.3	PQ -représentation et MD -représentation	209
5.4	Maintien entièrement dynamique	216
5.4.1	Opérations dynamiques sur les arêtes	216
5.4.2	Suppression de sommet	216
5.5	Insertion de sommet	221
5.5.1	Grappes minces et cliques maximales additionnelles	221

5.5.2	Noeuds propres et noeud d'insertion	222
5.5.3	Caractérisation dynamique des graphes d'intervalles	226
5.5.4	Déterminer $PQ(G + x)$	244
5.5.5	Algorithme et complexité	254

Introduction

Représentations de graphes

Les graphes ont démontré leur intérêt pour la modélisation dans beaucoup de domaines. En informatique théorique, ils sont étudiés tant parce qu'ils constituent de bonnes structures de données que parce qu'ils sont un objet théorique servant à raisonner en vue de la résolution d'un problème. Le besoin de représenter les graphes vient de ces deux utilisations.

Pour stocker un graphe en machine, il faut le représenter par une suite de 0 et de 1. L'enjeu du choix de la représentation est alors la performance. On peut évaluer une représentation selon une immense variété de critères. Le premier auquel on peut penser est la taille de stockage, en nombre de bits, occupée par la représentation. Ce critère est primordial par exemple pour l'archivage ou l'envoi de données. Si le graphe constitue la donnée d'un problème à résoudre par des moyens informatiques, le critère prépondérant sera le temps de résolution que permet d'atteindre la représentation choisie. Lorsqu'il s'agit de les manipuler en machine, le but recherché par la représentation des graphes est l'efficacité.

Un autre but essentiel à la thématique est l'intelligibilité. Un graphe est un objet qui n'est pas facile à cerner pour un humain dès qu'il a plus de quelques sommets et arêtes. Les représentations de graphes ont aussi pour but de rendre les graphes plus compréhensibles en en dégagant une structure. Lorsqu'il est décrit par une liste de sommets et d'arêtes, un graphe n'offre aucune prise à l'esprit humain. Heureusement, il existe une multitude de représentations qui permettent d'avoir une vue du graphe selon un certain angle : la décomposition en composantes biconnexes, un dessin planaire, une extension linéaire d'un ordre partiel, par exemple, donnent une certaine vision du graphe ainsi représenté. L'intelligibilité apportée par les représentations de graphes permet de mieux saisir certaines propriétés des graphes qu'on étudie, de raisonner et de démontrer de nouvelles propriétés. Elles jouent souvent un rôle essentiel dans la démonstration de nombreux résultats théoriques. Le meilleur exemple étant sans doute la démonstration récente du théorème fort des graphes parfait qui est entièrement basée sur des décompositions de graphes.

Les buts d'intelligibilité et d'efficacité ne sont pas déconnectés l'un de l'autre, ils se rejoignent : les représentations permettant de bien comprendre la structure des graphes sont souvent celles qui permettent de les traiter efficacement.

Le chapitre 1 fournit plusieurs exemples de trois types de représentations de graphes : les décompositions, les modèles géométriques et les représentations arborescentes par degrés de liberté.

Décompositions

Les décompositions donnent souvent une bonne vision de la structure d'un graphe. Leur but est de représenter le graphe par un ensemble de graphes, qui sont souvent des sous-graphes induits du premier ou en sont très proches. Cette approche vise à "découper" le graphe en morceaux qui sont plus intelligibles car étant plus petits et ayant plus de propriétés. La plupart des décompositions (dont toutes celles présentées dans ce mémoire) sont arborescentes. C'est à dire que les relations entre les graphes issus de la décomposition forment un arbre. Cette propriété améliore grandement la lisibilité du résultat et joue pour beaucoup dans l'intérêt de ces décompositions. Une décomposition se définit par un procédé à double sens de découpage/reconstruction. Le procédé de découpage exploite la présence d'une configuration donnée dans le graphe : un module pour la décomposition modulaire, une coupe pour la décomposition en coupes, une paire séparante pour la décomposition en composante triconnexes. Lorsqu'une telle configuration est trouvée, elle sert de base à une étape de découpage, puis on cherche à nouveau une telle configuration dans les morceaux produits et ainsi de suite. Dans les décompositions arborescentes, chaque arête de l'arbre correspond à une opération de découpage et chaque noeud à un des graphes issus de la décomposition. En appliquant l'opération de reconstruction selon toutes les arêtes de l'arbre, on retrouve le graphe de départ.

Une des questions délicates dans les décompositions est l'unicité de la représentation obtenue. Le problème vient du fait qu'à une étape donnée peuvent s'offrir plusieurs choix différents de décomposition. Certains de ces choix s'ils sont faits d'abord interdiront d'autres choix par la suite. Cela a pour effet d'aboutir à des décompositions différentes selon les choix qui sont faits. Dans les trois décompositions citées ci-dessus, l'unicité peut être obtenue en ne décomposant que selon des configurations qui ne sont gênées par aucune autre. Ces configurations sont dites fortes, vocabulaire qui provient de la décomposition modulaire. Dans ce mémoire (section 1.2), il a été apporté un grand soin à la façon de définir une décomposition canonique. Cela a conduit à présenter les notions classiques sous un angle parfois légèrement différent et à mettre en évidence des liens entre les différentes approches classiques de ces décompositions.

Modèles géométriques

Les classes de graphes définies par modèles géométriques apparaissent naturellement dans la modélisation de certains problèmes pratiques susceptibles d'être traités de manière informatisée. Par exemple, en biologie, le problème de reconstruction du génome fait jouer un rôle de première importance aux graphes d'intervalles. En phylogénie, l'étude du réarrangement des séquences génétiques au cours de l'évolution fait intervenir les graphes de permutation.

Les graphes d'intervalles sont ceux que l'on peut représenter en attribuant à chaque sommet un intervalle de la droite réelle de sorte que deux intervalles s'intersectent si et seulement si leurs sommets correspondants sont adjacents. De manière similaire, les graphes de permutation sont ceux que l'on peut représenter par intersections de segments

joignant deux lignes droites parallèles données. Le fait que ces graphes admettent de telles représentations leur confère des propriétés fortes.

Non seulement ces classes de graphes ont de bonnes aptitudes de modélisation de problèmes concrets, mais elles sont aussi fascinantes d'un point de vue théorique par les correspondances qu'elles établissent entre leurs définitions géométriques et les caractérisations qu'elles possèdent en termes de théorie des graphes. Parmi les plus beaux exemples se trouvent celui des graphes d'intervalles : ce sont les graphes de co-comparabilité sans C_4 ; et ce sont aussi exactement les graphes triangulés sans triplés astéroïdaux.

En termes d'efficacité de représentation, les modèles d'intersections permettent dans beaucoup de cas d'avoir une structure de donnée de taille $O(n)$ qui permet de tester l'adjacence entre deux sommets en temps constant. Cela vient du fait que l'objet géométrique correspondant à un sommet est représentable en espace constant. Ces représentations offrent d'excellentes performances algorithmiques. Par exemple, le nombre chromatique peut être calculé en $O(n)$ pour les graphes d'intervalles et en $O(n \log n)$ pour les graphes de permutation. Sur ces classes de graphes, on arrive souvent à mettre au point des algorithmes qui ne parcourent pas les arêtes du graphe mais seulement ses sommets, en faisant quelques tests d'adjacence. C'est pourquoi le nombre d'arêtes est souvent absent des complexités.

La problématique principale de ce mémoire est l'entretien dynamique de représentations géométriques. La question posée est la sensibilité de ces représentations aux modifications du graphe. Plus précisément, on se demande quel est le temps de calcul nécessaire pour recalculer un modèle géométrique du nouveau graphe à partir du modèle de l'ancien. Pour ce faire, on peut rarement se contenter d'examiner un modèle particulier du graphe d'origine (avant la modification). On est souvent amené à les examiner tous. C'est le rôle des représentations arborescentes par degrés de liberté.

Représentations arborescentes par degrés de liberté

Le paradigme des représentations arborescentes par degrés de liberté est le suivant. Elles sont basées sur un arbre. A chaque noeud de l'arbre est associé un sous-ensemble d'objets mathématiques du même type. Ces objets sont munis d'une opération de composition qui étant donné deux objets en donne un autre du même type. Si on choisit pour chaque noeud un objet parmi l'ensemble de ceux qui lui sont associés, et si on applique l'opération de composition selon toutes les arêtes de l'arbre, on obtient un nouvel objet. L'arbre représente précisément l'ensemble des objets que l'on peut obtenir de cette manière à partir d'un choix quelconque d'objets pour les noeuds de l'arbre. La qualification "par degrés de liberté" vient de ce que dans les applications, on peut souvent distinguer deux types de noeuds avec comme critère la taille de l'ensembles d'objets qui peut leur être associé : certains noeuds n'offrent qu'un choix très restreint, un nombre constant d'objets, alors que d'autres offrent beaucoup plus de liberté dans le choix, des ensembles arbitrairement grand peuvent leur être associés.

Ce paradigme permet de représenter l'ensemble des modèles géométriques de certaines classes de graphes. La section 1.4 fournit plusieurs exemples de telles classes. Ces représentations proviennent souvent de décompositions de graphes. Dans ce cas, l'arbre de

la représentation n'est autre que l'arbre de décomposition du graphe, les objets associés aux noeuds de l'arbre sont des modèles géométriques de la classe en question et l'opération de composition sur ces objets est une adaptation aux modèles géométriques de l'opération de composition sur les graphes qui définit la décomposition utilisée. Les noeuds qui offre un choix restreint de possibilités sont les noeuds premiers de la décomposition et ceux qui offrent un choix vaste sont les noeuds dégénérés. Toutes ces notions sont détaillées dans le chapitre 1.

Les représentations arborescentes par degrés de liberté jouent un rôle déterminant dans l'étude dynamique des modèles géométriques et dans les problèmes d'optimisation sur ces objets.

Algorithmique dynamique

La mention "dynamique" pour un algorithme s'oppose à celle de "statique". Un algorithme statique de graphe possède une donnée pour laquelle est posée une question à laquelle l'algorithme doit répondre par un calcul. En algorithmique dynamique, la problématique est différente. On considère qu'on a déjà effectué le calcul demandé sur le graphe (par un algorithme statique par exemple). Ce qui nous intéresse alors c'est d'actualiser ce résultat lorsque le graphe subit une légère modification. Est-il possible de ne pas refaire tout le calcul et de déduire le nouveau résultat à moindre frais? Les problèmes peuvent avoir des comportements dynamiques très différents, certains souffrent bien les modifications alors que d'autres y sont intolérants et exigent un recalcul complet. Prenons deux exemples pratiques.

Pour un anniversaire, un nombre n d'enfants se partagent équitablement un gâteau au chocolat. Après le partage, un des enfants avoue qu'il n'aime pas le chocolat : on découpe sa part en $n - 1$ parts égales et on distribue un des morceaux à chacun des autres enfants. Ce problème a une bonne dynamique car il est facile de refaire un partage équitable lors du retrait d'un enfant¹.

Lors d'un déménagement, on a emballé les affaires de la maison en un nombre optimal de caisses (d'une taille fixée assez grande). Au moment de partir, on s'aperçoit qu'une des paires de boucles d'oreilles qui ont été rangées dans les cartons appartenait à la voisine et que son mari n'avait pas restitué les boules de pétanques qu'on lui avait prêtées. Dans ce cas, on sent que trouver une nouvelle solution optimale à partir de l'existante, dans le cas où les caisses étaient remplies à ras bord, peut nécessiter presque autant de travail que tout refaire en repartant de zéro.

L'intérêt pratique de l'algorithmique dynamique de graphes est évident. Dès lors que les graphes modélisent des systèmes amenés à se modifier dans le temps, le besoin de concevoir des algorithmes dynamiques pour entretenir les structures utilisées se fait ressentir. En effet, pour les problèmes ayant une forte dynamique, c'est à dire subissant des modifications à une fréquence élevée, il serait beaucoup trop coûteux de refaire les calculs à chaque modifications. Dans ces cas, les algorithmes utilisés doivent avoir une complexité d'autant plus basse que la fréquence des modifications est élevée. Des systèmes dynamiques peuvent se rencontrer par exemple dans les réseaux de communications tels que les réseaux ad'hoc, où le graphe physique des connections est fortement dynamique, sur internet où les groupes se créent et se modifient, où les connexions s'établissent et se perdent. Les structures de données à l'intérieur des programmes ont souvent une dynamique forte et de bons algorithmes sont nécessaires pour modifier ces structures sans trop pénaliser le temps d'exécution des programmes.

Les domaines de représentation des graphes et de l'algorithmique dynamique ont fait beaucoup l'un pour l'autre. Les représentations de graphes fournissent nombre de problèmes à traiter par des algorithmes dynamiques. En retour, l'algorithmique dynamique étant très

¹Le lecteur avisé aura remarqué qu'en pratique le problème devient plus délicat s'il faut en plus trouver un substitut alimentaire pour l'enfant n'aimant pas le chocolat.

exigeantes en structures de données, elle a engendré de nouvelles représentations de graphes qui ont un intérêt pour elles mêmes.

L'importance qu'accorde l'algorithmique dynamique aux structures de données vient du fait que dans un algorithme dynamique, l'essentiel de la donnée est au choix du concepteur de l'algorithme ! Cela s'explique par les enjeux de la problématique. Le point de départ des algorithmes statiques est imposé : on dispose de telle information et on veut calculer telle information. Pour les algorithmes on peut disposer de ce qu'on veut comme donnée de départ, simplement si on veut une structure particulière, il faudra aussi l'entretenir lors d'une modification en plus de répondre à la question que l'algorithme cherche à résoudre. On peut donc inclure dans la donnée de départ toute structure utile qui ne pénalise pas, par son coût d'entretien, le temps de réponse de l'algorithme. C'est pourquoi l'algorithmique dynamique est particulièrement exigeante en structures de données : il faut emporter toute l'information nécessaire mais pas de superflu. Cette recherche de performance a permis de mieux comprendre et d'améliorer certaines représentations de graphes.

Contributions du mémoire

Ce mémoire s'intéresse particulièrement au problème de la représentation et reconnaissance dynamique d'une classe de graphes. Après chaque modification du graphe, on veut savoir si le graphe modifié est toujours dans la classe et en entretenir une certaine représentation si c'est le cas. Sa contribution essentielle est la conception de trois algorithmes entièrement dynamiques de représentation et reconnaissance de graphes : pour les cographes orientés, pour les graphes de permutation et pour les graphes d'intervalles. Chacun des trois algorithmes traite les quatre opérations élémentaires suivantes : ajout et retrait d'un sommet, ajout et retrait d'une arête.

Cographe orientés

L'algorithme pour les cographe orientés étend un résultat de [CPS85, SS04] sur les cographe. La complexité de cet algorithme est optimale : les modifications de sommet sont traitées en $O(d)$, où d est le degré du sommet modifié, et les modifications d'arête en $O(1)$. De plus, lorsqu'une modification du graphe ne résulte pas en un cographe, l'algorithme retourne dans la même complexité un certificat montrant que le graphe modifié n'est pas un cographe orienté. Ce certificat est un sous-graphe induit exclus pour la famille des cographe, dont le nombre de sommets est borné par une constante.

Cet algorithme est basé sur l'entretien du coarbre orienté, qui est l'arbre de décomposition modulaire du cographe orienté. Par rapport au coarbre non orienté, un type de noeud supplémentaire est introduit : les noeuds ordres, qui encapsulent les arcs non symétriques du graphe. Ce que montre ce résultat est que la difficulté ajoutée par l'orientation des arcs et la présence d'un nouveau noeud dans l'arbre est technique mais ne menace pas la complexité optimale déjà atteinte sur les cographe. Ce travail a été publié en version courte dans [CP04] et en version complète dans [CP06].

Graphes de permutation

Le deuxième algorithme concerne la classe des graphes de permutation, une version courte en a été publiée dans [CP05]. Pour cette classe, toutes les modifications d'arête et de sommet sont traitées en temps $O(n)$. Cette complexité est intéressante car nettement inférieure au temps $O(n + m)$ nécessaire pour la reconnaissance statique de la classe. De plus, il existe des ajouts et des suppressions d'arête et de sommet qui induisent $\Omega(n)$ changements dans la représentation choisie, qui est basée sur l'arbre de décomposition modulaire. Ce qui constitue un bon argument pour dire que la complexité atteinte, en ayant fait le choix de cette structure de donnée, est bonne.

L'enseignement tiré de cette approche est que la bonne complexité atteinte repose entièrement sur la représentation des quotients des noeuds premiers de l'arbre de décomposition modulaire. Dans le cas des graphes de permutations, ces quotients sont eux-mêmes des graphes de permutation et ils admettent une représentation par modèle d'intersection. C'est cette représentation qui permet d'atteindre la complexité de $O(n)$. On peut imaginer

d'autres classes de graphes ayant des représentations efficaces des quotients des noeuds premiers de leur arbre de décomposition modulaire. L'algorithme pour les graphes de permutation a été élaboré dans l'optique de fournir un cadre générique pour l'entretien dynamique de ces classes de graphes. En effet, il est présenté une caractérisation mathématique de la modification de l'arbre de décomposition modulaire lors d'un ajout de sommet qui n'est pas spécifique aux graphes de permutation mais commune à tous les graphes. Cette caractérisation est une traduction mathématique de l'algorithme de [MS89]. De plus, la partie de l'algorithme traitant la modification de l'arbre est complètement déconnectée de la partie de reconnaissance spécifique à la classe.

Enfin, l'entretien dynamique de l'arbre de décomposition modulaire des graphes de permutation est une étape intermédiaire intéressante dans l'optique de traiter l'entretien cet arbre pour les graphes quelconques. En effet, la classe des graphes de permutation a la particularité de ne présenter aucune contrainte sur son arbre de décomposition modulaire : tout arbre de décomposition modulaire (sans y inclure les quotients des noeuds premiers) est l'arbre de décomposition modulaire d'un graphe de permutation. Cela nous permettra de mener une réflexion sur les conditions que doit satisfaire la représentation des noeuds premiers pour permettre une complexité de $O(n)$ sur toute modification élémentaire du graphe.

Graphes d'intervalles

Un algorithme entièrement dynamique de reconnaissance des graphes d'intervalles est présenté. Il maintient trois représentations des graphes d'intervalles : un modèle d'intervalles minimal, le PQ -arbre des cliques maximales du graphe et sa décomposition modulaire. Les quatre opérations élémentaires sont traitées en $O(n)$. Cette complexité est assez satisfaisante dans le sens où chacun des quatre types de modification peut induire $\Omega(n)$ changements dans le PQ -arbre comme dans l'arbre de décomposition modulaire.

Plusieurs algorithmes avaient déjà traité le problème de reconnaissance dynamique des graphes d'intervalles [BL76, Kor87, KM89, Hsu96, Iba01], mais celui présenté ici est le premier, à ma connaissance, à traiter la suppression d'un sommet. De plus, on améliore la complexité des opérations d'arêtes de [Iba01], qui est de $O(n \log n)$. Enfin, notre algorithme est purement dynamique, contrairement à ceux de [BL76, KM89] qui sont faussement incrémentaux car ils nécessitent un calcul préliminaire statique impliquant la connaissance préalable du graphe dans sa totalité.

L'opération déterminante est l'insertion de sommet. Cette opération a été traitée de manière purement incrémentale dans la thèse de Norbert Korte [Kor87]. L'algorithme de [Kor87] est décrit comme étant assez complexe par [KM89]. L'algorithme présenté ici reprend l'approche de [Kor87] en y apportant des éléments nouveaux de compréhension et de simplification.

L'élément essentiel est apporté par la correspondance mathématique que nous établissons entre le PQ -arbre des cliques maximales d'un graphe d'intervalles et sa décomposition modulaire. Il était couramment admis dans la communauté que ces deux structures avaient un lien sans qu'il n'ait jamais été formellement établi. La section 5.3 montre que bien que

ces deux arbres puissent être assez différents, ils ont un lien mathématique très fort et sont linéairement équivalents d'un point de vue algorithmique, c'est à dire qu'on peut passer de l'un à l'autre en temps linéaire. Ce rapprochement éclaire d'un jour nouveau l'algorithme de [Kor87] et le place dans la lignée du travail sur les graphes de permutation présenté dans ce mémoire et de celui sur les graphes en général présenté dans [MS89]. Notamment, grâce au travail sur les graphes de permutation et au théorème 5.6, qui montre comment lire les modules sur le PQ -arbre, on est en mesure de déduire la forme du PQ -arbre après insertion d'un sommet sans travail supplémentaire. Néanmoins, la partie du travail qui construit l'ordre des fils du nouveau noeud premier créé lors de l'insertion reste technique et demanderait à être simplifiée davantage.

Relations entre décompositions et modèles géométriques

Le chapitre 1 ne présente pas de résultats nouveaux mais essaye de donner une présentation mettant en évidence des parallèles entre plusieurs approches classiques.

En ce qui concerne les décompositions de graphes, un effort particulier a été fait pour les présenter dans un cadre commun. Notamment en ce qui concerne la définition et la caractérisation d'une décomposition canonique pour la décomposition modulaire, la décomposition en coupes et la décomposition en composantes triconnexes. En rapprochant les visions de Gallai pour la décomposition modulaire [Gal67] et de Cunningham [Cun82] pour la décomposition en coupes, on obtient une vision intéressante de la problématique générale de décomposition et d'une manière de résoudre le problème de l'unicité d'une décomposition. Cette vision est aussi mise en oeuvre sur la décomposition en composantes triconnexes et résulte, me semble-t-il, en une présentation plus simple de cette décomposition que celle de Tutte [Tut66].

L'état de l'art met également en avant des relations fortes entre certaines décompositions de graphes et certaines classes de graphes définies par modèles géométriques. Ces relations sont faites par les représentations arborescentes par degrés de liberté. Ces objets représentent tous les modèles géométrique d'un graphe donné. Ils sont souvent fournis par des décompositions de graphes. Le PQ -arbre des cliques maximales semble déroger à cette règle car sa définition n'est issue d'aucune décomposition. Pourtant, nous montrons au chapitre 5 que cet objet est fortement lié à la décomposition modulaire. Ce mémoire, en rassemblant une collection de décompositions et de classes de graphes définies géométriquement qui ont des comportements relatifs très similaires, veut montrer qu'il ne s'agit pas là d'un phénomène fortuit mais d'un schéma général qui s'applique sûrement à d'autres exemples encore.

Pré-requis et notations communes à tout le mémoire

Cette section répertorie toutes les notions et notations de mathématiques et théorie des graphes nécessaires à la lecture de ce mémoire. La plupart sont classiques. C'est pourquoi le lecteur préférera sans-doute ne pas inclure ce passage dans sa lecture mais plutôt s'y référer lorsque le besoin s'en fera ressentir. C'est dans cet esprit que cette section a été rédigée.

Notions ensemblistes

La lecture de ce mémoire ne suppose que la connaissance des premières notions ensemblistes. Je précise ici les notations qui y sont employées.

Soit X un ensemble, l'ensemble de ses parties est noté $\mathcal{P}(X)$, l'ensemble de ses parties à deux éléments étant noté $\mathcal{P}_2(X)$. On appelle **singleton** un ensemble à un élément. $\emptyset$ est l'ensemble vide. L'inclusion ensembliste est notée $\subseteq$, et l'inclusion stricte $\subsetneq$. Pour deux ensembles X et Y , on note $X \times Y$ leur produit cartésien. Le produit $X \times X$ est noté X^2 , $X \times X \times X$ est noté X^3 , et ainsi de suite pour tous les entiers naturels supérieurs à 4. Dans la suite, j'utilise presque toujours l'abus de notation $x_1, x_2, \dots, x_k \in X$ pour $(x_1, x_2, \dots, x_k) \in X^k$, où k est un entier naturel supérieur à deux. L'union de deux ensembles est notée $X \cup Y$, et leur intersection $X \cap Y$. On note parfois $X \sqcup Y$ l'union de X et Y lorsque X et Y sont disjoints, c'est à dire $X \cap Y = \emptyset$. Si cette notation est employée avec plus de deux ensembles, cela signifie que les ensembles impliqués dans la notation sont deux à deux disjoints.

La différence ensembliste de X et Y est notée $X \setminus Y$, et leur différence symétrique est notée $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$

Définition 0.1 Deux ensembles X et Y se **chevauchent** ssi $X \cap Y \neq \emptyset$ et $X \setminus Y \neq \emptyset$ et $Y \setminus X \neq \emptyset$. On note $X \oslash Y$.

Soit E un ensemble de référence dans lequel on travaille. Soit I un ensemble d'indices et soit $(E_i)_{i \in I}$ une famille de sous-ensembles de E indicés par des éléments de I . L'union des E_i pour $i \in I$ est noté $\bigcup_{i \in I} E_i$, et l'intersection des E_i pour $i \in I$ est notée $\bigcap_{i \in I} E_i$. Par convention, lorsque $I = \emptyset$, $\bigcup_{i \in I} E_i = \emptyset$ et $\bigcap_{i \in I} E_i = E$.

Une partition d'un ensemble E est un ensemble P de parties de E tel que :

- $\forall A \in P, A \neq \emptyset$, et
- $\forall A, B \in P, A \cap B = \emptyset$, et
- $\bigcup_{A \in P} A = E$.

Si on enlève la contrainte qui veut que chaque partie de la partition soit non vide, on parlera de **partition au sens large**. Une **bipartition** est une partition en deux sous-ensembles.

On note $\mathbb{N}$ l'ensemble des entiers naturels et $\mathbb{R}$ celui des nombres réels. La plupart des ensembles considérés dans ce mémoire sont finis. Le cardinal d'un ensemble fini X est noté $|X|$.

Pour une fonction f d'un ensemble A vers un ensemble B , on note :

- pour $X \subseteq A$, $f\langle X \rangle = \{f(x) \mid x \in X\}$;
- pour $b \in B$, $f^{-1}(b) = \{a \in A \mid f(a) = b\}$;
- pour $B \subseteq Y$, $f^{-1}(B) = \bigcup_{b \in B} f^{-1}(b)$.

Pour deux applications f et g de $\mathbb{N}$ dans $\mathbb{R}$, on note $f = O(g)$ ou $f(n) = O(g(n))$ ssi $\exists k \in \mathbb{R}, \forall n \in \mathbb{N}, f(n) \leq k * g(n)$. Et on note $f(n) = \Omega(g(n))$ ssi $\exists k \in \mathbb{R} \setminus \{0\}, \forall n \in \mathbb{N}, f(n) \geq k * g(n)$.

Les relations binaires. Une relation binaire $\mathcal{R}$ sur un ensemble X est une partie de $X \times X$. Pour $(x, y) \in X^2$, si $(x, y) \in \mathcal{R}$, on note $x\mathcal{R}y$ et on dit que x est en relation avec y . Dans le cas contraire, on note $x \not\mathcal{R}y$ pour $(x, y) \notin \mathcal{R}$.

Une relation binaire est dite :

- réflexive si $\forall x \in X, x\mathcal{R}x$.
- antiréflexive si $\forall x \in X, x \not\mathcal{R}x$.
- symétrique si $\forall (x, y) \in X^2, x\mathcal{R}y \Rightarrow y\mathcal{R}x$.
- antisymétrique si $\forall (x, y) \in X^2, (x\mathcal{R}y \text{ et } x \neq y) \Rightarrow y \not\mathcal{R}x$.
- transitive si $\forall (x, y, z) \in X^3, (x\mathcal{R}y \text{ et } y\mathcal{R}z) \Rightarrow x\mathcal{R}z$.

Les relations d'ordre. Une **relation d'ordre partiel**, ou tout simplement **relation d'ordre**, est une relation binaire réflexive, antisymétrique et transitive. Un ensemble muni d'une relation d'ordre est appelé **ensemble ordonné**, on dit que c'est l'ensemble **sous-jacent** à la relation d'ordre. Deux éléments en relation par une relation d'ordre sont dits **comparables**. Lorsque tout couple d'éléments est comparable, la relation d'ordre est appelée **relation d'ordre total**. On parle souvent d'**ordre partiel**, ou d'**ordre total** pour désigner le couple $(X, \mathcal{R})$ formé par une relation d'ordre $\mathcal{R}$ et l'ensemble sous-jacent X sur lequel elle est définie. Mais en pratique, on confond souvent ce couple et la relation d'ordre elle-même. Un ordre partiel est souvent appelé simplement un ordre, et un ordre total est aussi appelé **ordre linéaire**. Les relations d'ordre sont en général notées $\leq$. Ainsi, pour deux éléments $x, y \in X$, on note $x \leq y$. Lorsque $x \leq y$ et $x \neq y$, on note $x < y$.

La restriction d'un ordre $(X, \mathcal{R})$ à un sous-ensemble $Y \subseteq X$ est l'ordre $(Y, \mathcal{R}|_Y)$, où $\mathcal{R}|_Y = \{(a, b) \in \mathcal{R} \mid a, b \in Y\}$.

Un **intervalle** I d'un ensemble fini X muni de la relation d'ordre $\leq$ est un sous-ensemble de X tel que $\exists a, b \in X, I = \{x \in X \mid a \leq x \leq b\}$. L'ensemble des entiers naturels compris entre les entiers i et j est noté $\llbracket i, j \rrbracket$, on l'appelle aussi intervalle. Lorsque $i > j$, par convention, on a $\llbracket i, j \rrbracket = \emptyset$.

Définition 0.2 On dit que G est un *graphe de co-comparabilité* ssi $\overline{G}$ est un *graphe de comparabilité*.

Notation 0.1 Pour un intervalle I d'un ordre total, on note $f(I)$ et $l(I)$ respectivement le premier et dernier élément de I .

Dans un ordre total σ , le prédécesseur immédiat d'un élément x est noté $pred(x)$ et son successeur immédiat $succ(x)$. Ces notations sont indéfinies respectivement lorsque x est le minimum ou le maximum de σ .

Notions de théorie des graphes

Les graphes considérés dans ce mémoire sont finis, simples et sans boucle. Un graphe non orienté G est un couple (V, E) où V est un ensemble fini et E un sous-ensemble de $\mathcal{P}_2(V)$. V est appelé l'ensemble des **sommets** de G , et E l'ensemble des **arêtes** de G . Un graphe orienté G est un couple (V, A) où V est un ensemble fini et A une relation binaire antiréflexive sur V . V est encore appelé l'ensemble des sommets de G , et A l'ensemble des **arcs** de G . Pour un graphe orienté ou non, on note $V(G)$ l'ensemble de ses sommets et $n = |V(G)|$ leur nombre. Enfin, m désigne le nombre d'arcs ou d'arêtes de G .

Graphes non orientés. Dans un graphe non orienté $G = (V, E)$, lorsque $\{x, y\} \in E$ on dit que les sommets x et y sont **adjacents**, ou encore que x **voit** y (et vice versa). Pour une arête $e \in E$ telle que $e = \{x, y\}$ on dit que e est **incidente** à x et y , et on appelle x et y les extrémités de e . Dans ce manuscrit, l'arête $\{x, y\}$ est presque toujours notée xy (ou de manière équivalente yx). On dit que x est voisin de y . Le **voisinage** d'un sommet $x \in V$ est l'ensemble des sommets de G voisins de x , on le note $N(x) = \{y \in V \mid xy \in E\}$. Le degrés $d(x)$ d'un sommet x est son nombre de voisins dans G : $d(x) = |N(x)|$. On utilise parfois aussi la notion de **voisinage fermé**, noté $N[x]$ et défini par $N[x] = N(x) \cup \{x\}$. Inversement, si $x \neq y$ et $xy \notin E$, on dit que xy est une **non arête** de G . Le complémentaire de G est le graphe $\bar{G} = (V, E')$ ayant le même ensemble de sommets que G et dont les arêtes sont les non arêtes de G : $E' = \{xy \in V^2 \mid x \neq y \text{ et } xy \notin E\}$. Le complémentaire du voisinage de x dans G noté $\bar{N}(x)$ est aussi le voisinage de x dans le complémentaire $\bar{G}$ de G , on l'appelle également le non voisinage de x dans G .

Pour un sous-ensemble de sommets $S \subseteq V$, on appelle voisinage, ou **voisinage extérieur**, de S , l'ensemble noté $N(S)$ et défini par $N(S) = \{x \in V \setminus S \mid \exists y \in S, x \in N(y)\}$. Deux sous-ensembles de sommets disjoints $A \subseteq V$ et $B \subseteq V$ sont dits **entièrement adjacents** si $A \subseteq N(B)$.

On éprouvera parfois le besoin de préciser dans les notations quel est le graphe dans lequel la notion de voisinage est appliquée, il sera alors placé en indice. On notera $N_G(x)$ pour un sommet et $N_G(S)$ pour un ensemble de sommets.

Etant donné un ensemble $S \subseteq V$ de sommets de G , le **sous-graphe de G induit par S** , noté $G[S]$, est défini par $G[S] = (S, E_S)$ où $E_S = \{xy \in S^2 \mid xy \in E\}$. De manière analogue, pour un ensemble d'arêtes $F \subseteq E$, on définit le **sous-graphe de G induit par F** , noté $G(F)$, par $G(F) = (V', F)$ où $V' = \{x \in V \mid \exists e \in F, x \in e\}$ est l'ensemble des extrémités des arêtes de F .

L'ensemble E des arêtes d'un graphe non orienté $G = (V, E)$ définit une relation binaire antiréflexive et symétrique sur V appelée **relation d'adjacence**. Sa fermeture réflexive et transitive est donc une relation d'équivalence. Les classes d'équivalence de cette relation sont appelées les **composantes connexes** de G . Si G n'a qu'une composante connexe, qui est alors l'ensemble V de ses sommets, on dit que G est **connexe**.

Un **chemin simple** d'un graphe non orienté $G = (V, E)$ est une séquence $x_1, \dots, x_k$ de sommets de G , avec $k \geq 2$ un entier naturel, telle que les sommets $x_1, \dots, x_k$ sont deux à deux distincts et $\forall i \in \llbracket 1, k-1 \rrbracket, x_i x_{i+1} \in E$. La longueur du chemin $x_1, \dots, x_k$ est $k-1$. Le

graphe à n sommets qui est un chemin simple sera noté P_n . Un **cycle simple** d'un graphe non orienté $G = (V, E)$ est une séquence $x_1, \dots, x_k$ de sommets de G , avec $k \geq 3$ un entier naturel, telle que $x_1, \dots, x_{k-1}$ est un chemin simple de G et $x_{k-1}x_1 \in E$. La longueur du cycle $x_1, \dots, x_k$ est k . Le graphe à n sommets qui est un cycle simple sera noté C_n . Si un graphe G ne possède aucun cycle, on dit que G est **acyclique**.

$G = (V, E)$ est un graphe **complet** si $E = \mathcal{P}_2(V)$. Le graphe complet à n sommets sera noté K_n . Une **clique** d'un graphe est un sous-ensemble de sommets qui induit un sous-graphe complet. L'ensemble des cliques d'un graphe G sera noté $\text{cliques}(G)$. Un **stable** d'un graphe G est un sous-ensemble de sommets qui induit un sous-graphe complet dans $\overline{G}$. Autrement dit, un stable est un sous-ensemble de sommets induisant un graphe sans arêtes dans G .

Un graphe G est **biparti** s'il existe une bipartition $\{A, B\}$ de l'ensemble de ses sommets telle que A et B sont des stables. Un graphe G est un **biparti complet** s'il existe une bipartition $\{A, B\}$ de l'ensemble de ses sommets telle que l'ensemble des arêtes de G est $\{ab \mid a \in A \text{ et } b \in B\}$. Le graphe biparti complet tel que $|A| = p$ et $|B| = q$ est noté $K_{p,q}$.

Au cours de ce mémoire, nous serons amenés à retirer ou ajouter des sommets et des arêtes dans les graphes que nous considérerons. Nous adoptons les notations suivantes. Si x est un sommet d'un graphe $G = (V, E)$, on note $G - x = G[V \setminus \{x\}]$. De même, pour un sous-ensemble de sommets $S \subseteq V$, on note $G - S = G[V \setminus S]$. Pour un sommet $y \notin V$ et un sous ensemble de sommets $Y \subseteq V$, on note $G + y = (V \cup \{y\}, E \cup \{yz \mid z \in Y\})$. Si xy est une arête d'un graphe $G = (V, E)$, on note $G - xy = (V, E \setminus \{xy\})$. De même pour un ensemble A d'arêtes de G , on note $G - A = (V, E \setminus A)$.

Graphes orientés. La plupart des notions définies pour les graphes orientés sont très similaires ou renvoient directement à celles des graphes non orientés. Cependant, pour plus de clarté, nous définissons explicitement les notions se rapportant aux graphes orientés. La relation d'adjacence d'un graphe orienté n'est pas nécessairement symétrique. Lorsque $(x, y) \in A$ ou $(y, x) \in A$, on dit que les sommets x et y sont adjacents. Les arcs d'un graphe orienté seront notés de la même façon que les arêtes d'un graphe non orienté : xy désigne l'arc (x, y) . Le lecteur devra prendre garde au fait que dans le cas orienté, les arcs xy et yx sont différents. Ce conflit de notation n'amènera pas de confusion car les parties du manuscrit traitant de graphes non orientés et celles traitant de graphes orientés sont disjointes. Comme dans le cas non orienté, un arc est dit incident à ses sommets extrémités. Le voisinage d'un sommet $x \in V$ est l'ensemble des sommets de G qui lui sont adjacents, on le note $N(x) = \{y \in V \mid xy \in A \text{ ou } yx \in A\}$. Le degré $d(x)$ d'un sommet x est son nombre de voisins dans G : $d(x) = |N(x)|$. L'orientation des arcs implique que l'on distingue, pour un sommet x , son voisinage entrant $N^-(x) = \{y \in V \mid yx \in A\}$ de son voisinage sortant $N^+(x) = \{y \in V \mid xy \in A\}$. Ainsi, $N(x) = N^-(x) \cup N^+(x)$. De même, on parle de degrés entrant $d^-(x) = |N^-(x)|$ et de degrés sortant $d^+(x) = |N^+(x)|$. Le complémentaire du voisinage de x dans G , ou non voisinage, est noté $\overline{N}(x)$. Dans le cas orienté, ce n'est pas le voisinage dans le graphe complémentaire car x et y peuvent être adjacents à la fois dans G et $\overline{G}$, que l'on définit comme suit.

Si $x \neq y$ et $xy \notin A$, on dit que xy est un **non arc** de G . Le complémentaire de G est le graphe $\overline{G} = (V, A')$ ayant le même ensemble de sommets que G et dont les arcs sont les non arcs de G : $A' = \{xy \in V^2 \mid x \neq y \text{ et } xy \notin A\}$.

Les notions de sous-graphes induits par un ensemble de sommets ou d'arcs sont similaires aux graphes non orientés. Etant donné un ensemble $S \subseteq V$ de sommets de G , le sous-graphe de G induit par S , noté $G[S]$, est défini par $G[S] = (S, A_S)$ où $A_S = \{xy \in S^2 \mid xy \in A\}$. Pour un ensemble d'arcs $B \subseteq A$, on définit le sous-graphe de G induit par B , noté $G(B)$, par $G(B) = (V', B)$ où $V' = \{x \in V \mid \exists a \in B, \exists y \in V, a = (x, y) \text{ ou } a = (y, x)\}$ est l'ensemble des extrémités des arêtes de B .

Le **graphe non orienté sous-jacent** à un graphe orienté $G = (V, A)$ est défini comme le graphe $H = (V, E)$ où $E = \{\{x, y\} \in \mathcal{P}_2(V) \mid (x, y) \in A \text{ ou } (y, x) \in A\}$. Pour un graphe orienté, les notions de connexité et de composantes connexes du graphe se rapportent au graphe non orienté sous-jacent.

Comme pour les graphes non orientés, on adopte les notations suivantes. Si x est un sommet d'un graphe $G = (V, E)$, on note $G - x = G[V \setminus \{x\}]$. De même, pour un sous-ensemble de sommets $S \subseteq V$, on note $G - S = G[V \setminus S]$. Pour un sommet $y \notin V$ et deux sous-ensembles de sommets $Y_1 \subseteq V$ et $Y_2 \subseteq V$, on note $G + y = (V \cup \{y\}, E \cup \{yz \mid z \in Y_1\} \cup \{zy \mid z \in Y_2\})$. Si xy est un arc d'un graphe $G = (V, E)$, on note $G - xy = (V, E \setminus \{xy\})$. De même pour un ensemble A d'arcs de G , on note $G - A = (V, E \setminus A)$.

Notations particulières aux arbres

Un **arbre** est un graphe connexe et acyclique. Un arbre à n sommets possède exactement $n - 1$ arêtes. Il est connu que dans un arbre $G = (V, E)$, il existe un unique chemin simple entre deux sommets distincts.

Un **arbre enraciné** est un arbre dans lequel on a distingué un sommet r que l'on appelle la **racine**. Grâce à ce sommet r , on définit une relation de parenté. Un sommet p est le **père** d'un sommet $f \neq r$ si il est le premier sommet rencontré après f sur l'unique chemin de f à r . La racine r n'a pas de père et tous les autres sommets de l'arbre en ont un unique. Si x et y sont deux sommets adjacents dans un arbre G alors x est le père de y ou y est le père de x . Ainsi, les sommets adjacents à un sommet x mais qui ne sont pas son père sont appelés les **fils** de x . Un arbre enraciné est parfois appelé une **arborescence**.

Les arbres se voient souvent attribués un vocabulaire spécifiques, c'est le cas dans ce mémoire. On les nommera le plus souvent par la lettre T . Leurs sommets sont appelés **noeud**. On distingue les **noeuds internes**, dont le degré est au moins deux, des **feuilles** qui sont de degré un. L'ensemble des fils d'un noeud p d'un arbre T est noté $\mathcal{C}_T(p)$. Deux noeuds d'un arbre sont **frères** si ils ont le même père. La fermeture réflexive et transitive de la relation «est père de» est la relation «est ancêtre de». L'ensemble des **ancêtres** d'un noeud p dans un arbre T est noté $Anc_T(p)$. La fermeture réflexive et transitive de la relation «est fils de» est la relation «est descendant de». L'ensemble des **descendants** d'un noeud p dans un arbre T est noté $Desc_T(p)$. On omettra l'arbre dans lequel s'applique ces notations lorsqu'il n'y a pas de confusion possible, et on notera $\mathcal{C}(p)$, $Anc(p)$ et $Desc(p)$.

Pour un noeud p d'un arbre T , on note T_p le **sous-arbre de T enraciné en p** qui

est le sous-graphe de T induit par les descendants de p . On confondra souvent, de manière parfaitement abusive, T et l'ensemble de ses noeuds en notant $p \in T$ à la place de $p \in V(T)$, pour un noeud P de T .

Chapitre 1

Représentation des graphes

La représentation des graphes est une problématique cruciale dans de nombreux champs de la théorie des graphes. Ses buts sont multiples. En informatique, la nécessité de représenter les graphes est premièrement imposée par la machine. La représentation est alors une réification dans la mémoire du calculateur de l'objet mathématique qu'est un graphe. Dans ce cas, le but recherché est la simple connaissance du graphe : quels sont ses sommets ? quelles sont ses arêtes ? comment les écrire dans un alphabet binaire ?

Mais un graphe ne se résume pas à une liste de sommets et d'arêtes, il a une structure, ou plutôt des structures, selon le point de vue de celui qui le regarde. C'est un des buts de la représentation que de saisir la structure d'un graphe selon un point de vue donné. Poursuivre cet objectif amène d'ailleurs parfois à des représentations partielles du graphe, c'est à dire qui ne permettent pas de retrouver la liste de tous les sommets et arêtes du graphe. Ces représentations sont des vues du graphe et en donnent un ou plusieurs aspects sans permettre pour autant sa connaissance complète. Dans d'autres représentations, l'objet représenté n'est même pas le graphe lui même mais un ou des objets mathématiques qui en sont dérivés ou qui lui sont liés de quelque manière que ce soit. Ces représentations, que l'on pourrait qualifier de métonymiques, peuvent permettre ou non la connaissance complète des sommets et arêtes du graphe. Toutes ces représentations de graphes qui s'attachent à la structure permettent à un humain de mieux connaître et comprendre le graphe représenté, ou servent de base à un outil de calcul formel pour la résolution de problèmes.

Des exemples de telles représentations sont étudiés dans les sections 1.2, 1.3 et 1.4. La section 1.1 présente différents critères d'efficacité d'une représentation de graphe en machine et évalue les performances des listes d'adjacence et de la matrice d'adjacence au regard de ces critères.

1.1 Performances d'une représentation

Lorsqu'une représentation d'un graphe a pour but d'être stockée en machine, on s'intéresse naturellement à son efficacité. Celle-ci peut être évaluée selon plusieurs critères. J'en ai choisis trois qui sont de premier intérêt dans ce mémoire : la place que la représentation

prend en mémoire, sa capacité à répondre rapidement aux requêtes et le temps nécessaire à sa mise à jour lors d'une légère modification du graphe. Les représentations par listes d'adjacence et par matrice d'adjacence sont évaluées selon ces critères. Les performances des autres représentations de graphe présentes dans ce mémoire, souvent propres à des classes de graphes restreintes, seront fréquemment comparées aux performances de ces deux représentations classiques et générales à tous les graphes.

Espace

Le premier¹ des critères d'efficacité que nous considérerons pour une représentation est la place qu'elle prends en mémoire. Cette place n'est pas toujours la même selon la représentation choisie. Par exemple, stocker un graphe sous forme de listes d'adjacence nécessite $\Omega(n+m)$ entiers alors que le stocker sous forme de matrice d'adjacence nécessite n^2 bits. En utilisant un formalisme de représentation propre à une classe de graphes restreinte, on peut obtenir des encombrements en mémoire encore plus réduits : par exemple $O(n)$ entiers ($O(n \log n)$ bits) pour les graphes de permutation et les graphes d'intervalles.

Temps

Un autre critère important d'efficacité est le temps de réponse aux deux requêtes élémentaires suivantes :

- la requête d'adjacence : deux sommets x et y sont ils adjacents ?
- la requête de voisinage : donner la liste des voisins d'un sommet x .

Le choix de ces deux requêtes est arbitraire : on pourrait prendre en compte le temps de réponse à n'importe quelle requête, c'est à dire considérer n'importe quel problème de graphe. Ce choix se justifie par le fait que ces deux requêtes constituent les opérations de base de la plupart des algorithmes de graphes.

Rappelons quelles sont les performances respectives de la représentation par listes d'adjacence et de celle par matrice d'adjacence vis à vis du temps de réponse aux requêtes d'adjacence et de voisinage.

Nous présentons deux façons connues d'implémenter les listes d'adjacences (voir [Spi03]), d'autres variantes seront envisagées dans la section 2.4. La plus commune des implémentations utilise un tableau dont les cases sont indexées par les numéros des sommets et contiennent la liste chaînée des voisins du sommet qui les indexe. Dans la seconde implémentation, le tableau est remplacé par une liste chaînée. Il en résulte des performances différentes en ce qui concerne les temps de réponse et la dynamicité, qui est le troisième critère d'efficacité abordé dans cette section. La première implémentation réponds à la requête d'adjacence en temps $O(\min(d(x), d(y)))$ et à la requête de voisinage en $O(d(x))$. Dans ce cas, le temps de réponse à la requête de voisinage est optimal puisque la liste des voisins est demandée. Du fait de la nécessité d'une recherche dans la liste chaînée des sommets, la seconde implémentation réponds aux deux requêtes en temps $O(n)$, ce qui est nettement

¹Cela ne signifie pas qu'il ait une importance prépondérante dans le travail présenté ici.

moins bon. Si cette version des listes d'adjacence est mentionnée c'est pour l'intérêt de son comportement dynamique différent, dont nous donnons un exemple ci-après.

La matrice d'adjacence est une structure de donnée conçue pour le test d'adjacence. C'est un tableau de dimension deux dont les lignes et les colonnes sont indexées par les numéros des sommets. La case indexée par les sommets x et y contient un bit qui vaut un si et seulement si x et y sont adjacents. Elle permet ainsi de répondre en temps constant à la requête d'adjacence, grâce à la double indexation des cases de la matrice. En revanche, pour répondre à la requête de voisinage, un parcours de toute la ligne du sommet est nécessaire et prends donc un temps $O(n)$.

De manière beaucoup plus générale, on peut aussi évaluer les performances en temps d'une représentation de graphe par rapport à n'importe quel problème autre que celui de l'adjacence et du voisinage. Cela est légitime car toutes les représentations d'un graphe n'ont pas le même comportement par rapport à un problème donné. Cette évaluation par rapport à un problème précis peut paraître subjective, pourtant, elle est pertinente du fait qu'elle évalue les possibilités algorithmiques offertes par la représentation considérée. Il y aurait alors autant de critères d'évaluation du temps de réponse d'une représentation qu'il y a de questions à poser sur un graphe. D'où l'intérêt de se limiter à deux requêtes simples servant d'opérations élémentaires pour la conception d'un grand nombre d'algorithmes.

Dynamacité

Le troisième critère selon lequel nous évaluerons les représentations est plus particulier à la démarche adoptée ici. Il s'agit de leur comportement dynamique. Le domaine de l'algorithmique dynamique avec les notions qui lui sont propres sont présentés au chapitre 2. Apprécier le comportement dynamique d'une représentation consiste à évaluer les temps de mise à jour de cette représentation sous les quatre modifications élémentaires du graphe suivantes : ajout et retrait d'une arête, ajout et retrait d'un sommet avec les arêtes définissant son voisinage.

Donnons l'exemple du retrait d'une arête xy . Dans l'implémentation des listes d'adjacence utilisant un tableau, on peut trouver x et y en temps constant et chercher dans la liste de leurs voisins pour en retirer respectivement y et x . Cela prends un temps total de $O(\text{Max}(d(x), d(y)))$. Le cas de la seconde implémentation des listes d'adjacence est similaire mis à part la recherche de x et y dans la liste des sommets du graphe qui prends un temps $O(n)$, qui est donc aussi la complexité totale de l'opération. La matrice d'adjacence permet de traiter la mise à jour en temps constant. Il suffit de changer la valeur du bit correspondant à l'arête xy , auquel on accède en temps constant grâce à la double indexation. Ainsi, on s'aperçoit que la complexité nécessaire au maintien des structures les plus simples pour représenter un graphe n'est pas sans enjeu. Le comportement dynamique des listes et matrice d'adjacence sous les quatre modifications élémentaires est entièrement discuté dans la section 2.4. Sont également présentées dans cette section de nouvelles structures pour les listes d'adjacences qui visent à optimiser les temps de mise à jour.

1.2 Décompositions de graphes

Les décompositions de graphes sont une forme de représentation qui s'est énormément répandue en théorie des graphes durant ces quarante dernières années. On compte notamment la décomposition en composantes triconnexes [Tut66], la décomposition modulaire [Gal67], la décomposition en coupes [Cun82], la décomposition par séparateurs complets [Tar85], la décomposition par partitions de biais, la décomposition par 2-joints, la décomposition par M -joints, la décomposition en 2-modules [CS87], la décomposition en bimodules [dM03].

L'idée commune aux différentes décompositions de graphes est de décrire un graphe comme étant construit grâce à des opérations de composition à partir de graphes de bases. Une décomposition de graphe est définie par les opérations de composition qu'elle autorise et les graphes de bases qu'elle se donne. Par exemple, la décomposition modulaire utilise comme seule opération la substitution d'un graphe à un sommet d'un autre graphe, et ses graphes de bases sont les premiers, les cliques et les stables (voir la section 1.2.1 pour une présentation complète de la décomposition modulaire).

Ce procédé, qui peut être vu comme une construction ou une dé-construction du graphe selon le côté par lequel on l'aborde, a pour but de se ramener à des graphes indécomposables ayant plus de propriétés. L'intérêt de la démarche est notamment de permettre de résoudre des problèmes algorithmiques de manière plus efficace. Lorsque cela fonctionne, le succès repose sur le bon comportement du problème considéré par rapport aux opérations de composition employées et sur la possibilité de résoudre ce problème efficacement sur les graphes indécomposables. Un bon exemple est le problème de l'orientation transitive et la décomposition modulaire. C'est d'ailleurs dans ce contexte que cette décomposition a vu le jour [Gal67] (même si elle a eu de nombreuses naissances ou renaissances dans différents domaines).

D'autre part, les décompositions de graphes sont des outils théoriques puissants qui permettent de mieux comprendre et démontrer certaines propriétés des graphes. L'exemple le plus éloquent en est probablement la démonstration récente du théorème fort des graphes parfaits [CSRT02] qui fait jouer un rôle central à trois décompositions de graphes : la décomposition par 2-joint, par M -joint et par partition de biais équilibrée. Dans ce chapitre, nous présentons en détail trois décompositions dont nous nous servirons par la suite.

1.2.1 Décomposition modulaire

La décomposition modulaire a reçu une grande attention et la quantité de travaux qui lui sont liés est astronomique. Pour les applications de la décomposition modulaire et des références, le lecteur est invité à se reporter à [MR84, Möh85] qui sont d'excellents surveys sur la question. Ces applications dépassent largement le cadre des graphes : la décomposition modulaire peut être définie sur d'autres structures discrètes telles que les relations en général, les familles de sous-ensembles d'un ensemble et les fonctions booléennes.

Cette décomposition, comme toutes celles présentées dans ce document, est de forme arborescente. Cette propriété a été souvent exploitée pour développer des algorithmes avec

l'approche "diviser pour régner" : en partant de la racine de l'arbre, on calcule des solutions récursivement pour chaque fils de la racine et ainsi de suite en descendant dans l'arbre, puis on remonte en prenant soin, pour chaque noeud, de recoller les solutions obtenues sur ses fils pour obtenir une solution pour le noeud lui-même. En arrivant à la racine, on résout le problème initial. L'opération de recollement peut être exponentielle en le nombre de fils du noeud considéré, mais pour beaucoup de classe de graphes cette opération peut être traitée efficacement et cette approche débouche sur des algorithmes performants.

La décomposition modulaire a été redécouverte dans de nombreux contextes. Elle a été introduite pour la première fois par [Gal67] pour le problème de l'orientation transitive des graphes (voir [MP01] pour une traduction en anglais de [Gal67]). Dans ce problème, il s'agit d'affecter, si possible, une orientation à chaque arête d'un graphe de sorte que la relation binaire obtenue soit transitive. Ainsi, la décomposition modulaire est à la base des meilleurs algorithmes de reconnaissance des graphes de comparabilité (qui sont les graphes qui admettent une orientation transitive), des ordres partiels de dimension 2 et des graphes de permutation (qui sont les graphes de comparabilité dont le complémentaire est aussi de comparabilité). La décomposition modulaire sera d'ailleurs la base de l'algorithme de reconnaissance dynamique des graphes de permutation présenté ici (section 4.5). Une des contributions de ce manuscrit est aussi de montrer de quelle manière la décomposition modulaire est impliquée dans la structure des graphes d'intervalles (chapitre 5).

L'intérêt de la décomposition modulaire n'est pas seulement algorithmique et pratique. Cette décomposition a aussi permis de mieux comprendre et étudier la structure des graphes. Elle joue un rôle dans la preuve de [Lov72] du théorème faible des graphes parfaits. On compte également beaucoup d'études sur la structure des graphes premiers [ER90a, III97, CI98].

Il a été produit une grande quantité d'algorithmes pour calculer la décomposition modulaire des graphes en général ou de classes restreintes. Se référer à [dM03] pour avoir l'éventail complet de ces algorithmes. Trois algorithmes linéaires ont été mis au point dans les années 90 [MS99, CH94, DGM01].

L'idée directrice de la décomposition modulaire des graphes est de factoriser la relation d'adjacence en cherchant des groupes de sommets qui ont le même voisinage extérieur. Ces sous-ensembles de sommets sont appelés les **modules**. Lorsqu'un graphe possède un module, on peut le remplacer par un unique de ses sommets, puisque tous ont les mêmes relations avec les sommets extérieurs au module, et décrire les relations internes au module séparément. Cette opération est appelée le **quotient** et les modules sont les ensembles de sommets pour lesquels on peut passer au quotient. Une des questions critiques pour une décomposition est celle de l'unicité de la décomposition d'un graphe. Elle est rarement acquise par définition et il faut souvent définir une décomposition canonique. C'est le cas pour la décomposition modulaire. Deux modules A et B peuvent se **chevaucher**, c'est à dire que $A \cap B \neq \emptyset$ et $A \setminus B \neq \emptyset$ et $B \setminus A \neq \emptyset$. Dans ce cas, passer au quotient sur un des deux modules interdit de le faire sur l'autre. L'unicité de la décomposition est donc menacée. Heureusement, Les modules ont beaucoup de bonnes propriétés et il en existe toujours qui ne chevauchent pas les autres. On peut alors quotienter selon les plus gros modules ne chevauchant pas les autres et continuer le procédé récursivement à l'intérieur

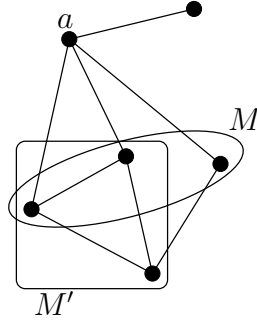


FIG. 1.1 – M est un module. M' n'en est pas un car M' n'est pas uniforme relativement à a .

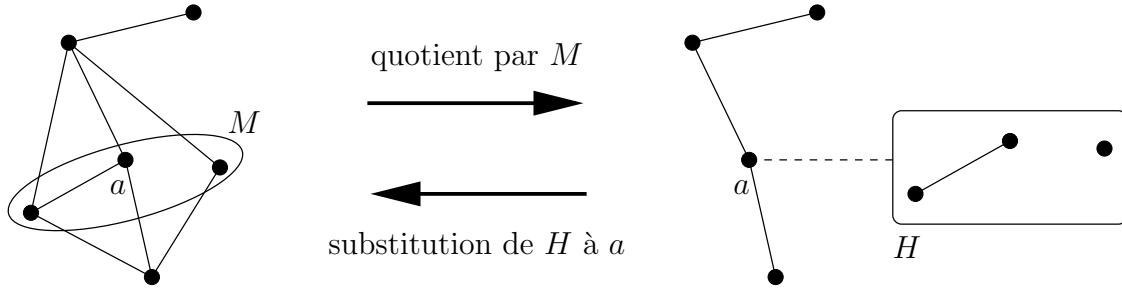


FIG. 1.2 – Le quotient par un module et la substitution d'un graphe à un sommet sont des opérations réciproques l'une de l'autre.

de ces modules. On obtient ainsi un arbre de décomposition unique.

Module, quotient et substitution d'un graphe à un sommet

Définition 1.1 Un ensemble de sommets S est **uniforme** relativement à un sommet $x \notin S$ ssi $S \subseteq N(x)$ ou $S \subseteq \overline{N}(x)$.

Définition 1.2 (voir figure 1.1) Un **module** d'un graphe $G = (V, E)$ est un sous-ensemble non vide $M \subseteq V$ de sommets de G qui est uniforme relativement à tout sommet $x \in V \setminus M$.

Si M est un module d'un graphe G , on définit le **graphe quotient** G/M de G par M comme le graphe $G[(V \setminus M) \cup \{x_M\}]$, où $x_M \in M$. Par définition d'un module, le graphe $G[(V \setminus M) \cup \{x_M\}]$ ne dépend pas du sommet x_M choisi dans M . x_M est appelé le **sommet représentatif** de M . Remarquons que le graphe quotient G/M est un sous graphe induit de G .

Lorsqu'un graphe G possède un module M on peut le décomposer en deux graphes : G/M et $G[M]$ (voir figure 1.2. La donnée de G/M , de $G[M]$ et du sommet de G/M représentatif de M suffit à reconstruire G . Cette opération de reconstruction, qui est la réciproque de l'opération de quotient, est la **substitution d'un graphe à un sommet**, aussi appelée **composition par substitution** ou simplement **composition de graphes**.

Soient $G = (V_G, E_G)$ et $H = (V_H, E_H)$ deux graphes et x un sommet de G , le graphe composé $G_{x \leftarrow H}$ de G et H par substitution de H à x est défini par $G_{x \leftarrow H} = (V, E)$ avec $V = (V_G \setminus \{x\}) \cup V_H$ et

$$E = (E_G \setminus \{e \mid e \in E_G, x \in e\}) \cup \{yz \mid y \in V_G \setminus \{x\}, z \in V_H, xy \in E_G\} \cup E_H$$

Remarquons que par définition V_H est un module de $G_{x \leftarrow H}$.

Ainsi, pour un module M de G dont le sommet représentatif dans G/M est x_M , on a $G = (G/M)_{x_M \leftarrow G[M]}$.

De manière plus générale on peut définir le quotient d'un graphe par une **partition de congruence**. Une partition $\mathcal{P} = \{M_1, \dots, M_k\}$ de l'ensemble des sommets d'un graphe G est une partition de congruence ssi toute partie M_i , avec $1 \leq i \leq k$, est un module. Le graphe quotient $G/\mathcal{P}$ d'un graphe G par une partition de congruence $\mathcal{P}$ est défini comme le sous graphe induit $G[S]$ avec $S = \{x_1, \dots, x_k\} \subseteq V$ et $\forall i \in \llbracket 1, k \rrbracket, x_i \in M_i$. De la définition d'un module, il découle que deux parties distinctes M_i et M_j d'une partition de congruence sont soit entièrement adjacentes soit entièrement non adjacentes. Donc, la définition du graphe quotient est insensible au choix des sommets représentatifs des ensembles de la partition de congruence.

Si $\mathcal{F}$ est une famille de modules disjoints mais ne forme pas nécessairement une partition de l'ensemble des sommets, nous utilisons la notation abusive $G/\mathcal{F}$ pour désigner $G/\mathcal{P}$ avec $\mathcal{P} = \mathcal{F} \cup \{\{x\} \mid x \in V \setminus \cup_{M \in \mathcal{F}} M\}$.

Premières propriétés des modules.

Le théorème suivant, de démonstration très simple, exprime quelques propriétés ensemblistes de stabilité de la famille des modules.

Théorème 1.1 *Si A et B sont deux modules d'un graphe G et se chevauchent alors :*

1. $A \cap B$ est un module de G
2. $A \cup B$ est un module de G
3. $A \Delta B$ est un module de G

Corollaire 1.1 *Si A et B sont deux modules d'un graphe G et se chevauchent alors $A \setminus B$ et $B \setminus A$ sont des modules de G .*

Preuve : $A \setminus B = A \cap (A \Delta B)$ et $A \Delta B$ chevauche A . Le théorème 1.1 conclut que $A \setminus B$ est un module de G . Par symétrie on en déduit le résultat pour $B \setminus A$. ■

Décompositions d'un graphe en modules

De la définition de module, il découle que, pour tout graphe $G = (V, E)$, l'ensemble des sommets V de G ainsi que les singletons $\{x\}$, $x \in V$ sont des modules. Ils sont appelés

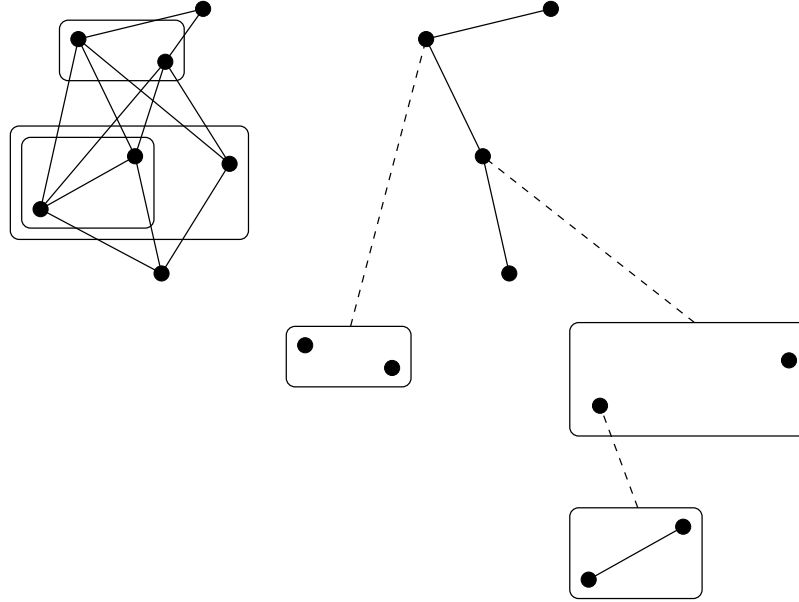


FIG. 1.3 – Un exemple de décomposition en modules d'un graphe avec l'arbre associé.

les **modules triviaux**. Les modules triviaux d'un graphe G sont exactement les modules M pour lesquels un des deux graphes G/M et $G[M]$ résultant de la décomposition de G par M est le graphe G lui même, l'autre graphe de la décomposition étant un graphe à un sommet. Dans la suite, nous ne considérons pas ces cas dégénérés de décomposition, mais uniquement les décompositions d'un graphe par un module non trivial.

Comme Cunningham pour la décomposition en coupes [Cun82], on peut définir ce qu'est une décomposition en modules d'un graphe G . Pour un module non trivial M d'un graphe G , on appelle $\{G/M, G[M]\}$ la décomposition simple de G par M . Une **décomposition en modules** D d'un graphe G est un ensemble de graphes défini inductivement de la manière suivante :

- $D = \{G\}$ est une décomposition de G ; et
- si D_1 est une décomposition de G , si $G_1 \in D_1$ et si $\{G_2, G_3\}$ est une décomposition simple de G_1 , alors $D = (D_1 \setminus \{G_1\}) \cup \{G_2, G_3\}$ est une décomposition de G .

La deuxième opération qui permet d'obtenir D à partir d'une décomposition simple d'un élément de D_1 est appelé **dérivation simple**. Lorsqu'une décomposition D' est obtenue à partir d'une décomposition D en appliquant une suite de dérivations simples, on dit que D' **dérive** de D .

Comme Cunningham le fait pour la décomposition en coupe, il est possible d'associer de manière univoque un arbre à chaque décomposition en modules (voir figure 1.3). Les sommets de cet arbre sont les graphes de la décomposition. Mais dans le cas de la décomposition en modules cet arbre est naturellement enraciné. Cela vient du fait que l'opération de composition par substitution ne fait pas jouer un rôle symétrique aux deux graphes qu'elle compose, elle a un sens : un des deux graphes vient se substituer à un sommet de l'autre et pas l'inverse. Nous verrons que la situation est différente pour la

décomposition en coupes. Ainsi, dans l'arbre associé à une décomposition, on rend un graphe G_2 fils d'un autre graphe G_1 ssi G_2 se substitue à un sommet de G_1 . La racine étant le seul graphe qui ne se substitue à aucun sommet d'un autre. Cette définition est intuitive. Nous nous en tiendrons à cette intuition car la définition rigoureuse de cet arbre est plus pénible que celle donnée par Cunningham pour la décomposition en coupe. L'arbre associé à une décomposition permet de retrouver le graphe G en appliquant la composition par substitution sur chaque arête de l'arbre.

Cette représentation arborescente d'une décomposition en modules D n'est pas celle habituellement utilisée, bien qu'elle en soit très proche. Pour obtenir la représentation classique, il suffit d'ajouter à l'ensemble des noeuds de l'arbre l'ensemble des sommets du graphe. Ces nouveaux noeuds deviennent les feuilles de l'arbre et les anciens noeuds deviennent les noeuds internes. La feuille correspondant au sommet x de G a pour père l'unique graphe de D dans lequel x n'est pas un sommet représentatif. A partir d'un tel arbre de décomposition, on peut lire la relation d'adjacence entre deux sommets x et y . Pour cela il suffit de trouver le plus petit ancêtre commun G_a (*ppca* dans la suite) de x et y , ainsi que ses deux fils G_x et G_y qui sont respectivement ancêtres de x et y . x est adjacent à y ssi les sommets représentant G_x et G_y sont adjacents dans G_a . Nous donnons plus loin la définition rigoureuse de l'arbre associé à une décomposition particulière que nous appellerons canonique.

Représentation des modules d'un graphe

Le cardinal de la famille des modules d'un graphe à n sommets peut atteindre 2^n . C'est le cas des cliques et des stables. Heureusement, cette famille est si bien structurée qu'elle admet une représentation ne nécessitant qu'un espace $O(n)$. Plus généralement, c'est le cas des **familles partitives**, introduites par [CHM81]. Ici, nous présentons très brièvement les familles partitives avec la vision adoptée dans la thèse de Fabien de Montgolfier [dM03]. Pour les démonstrations des résultats énoncés ci dessous, le lecteur se reportera aux deux documents précités.

Définition 1.3 Une famille $\mathcal{F}$ de sous-ensembles d'un ensemble V est *partitive* ssi

1. $V \in \mathcal{F}$, $\emptyset \notin \mathcal{F}$ et $\forall v \in V, \{v\} \in \mathcal{F}$; et
2. pour tous sous-ensembles $A, B \in \mathcal{F}$ tels que A chevauche B , les trois propriétés suivantes sont vraies :
 - (a) $A \cap B \in \mathcal{F}$
 - (b) $A \cup B \in \mathcal{F}$
 - (c) $A \Delta B \in \mathcal{F}$

Les **parties fortes** de $\mathcal{F}$ sont les parties de $\mathcal{F}$ qui n'en chevauchent aucune autre. La réduction transitive de l'ordre d'inclusion des parties fortes de $\mathcal{F}$ est un arbre T^F . Les feuilles de T^F sont les singletons $\{v\}, v \in V$ et l'élément de $\mathcal{F}$ représenté par un noeud p de T^F est précisément l'ensemble des feuilles du sous-arbre T_p^F de T^F enraciné en p . On a le résultat suivant.

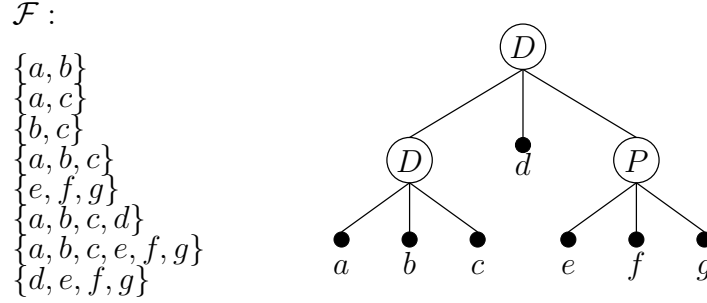


FIG. 1.4 – Une famille partitionnée $\mathcal{F}$ et son arbre T^F . Dans la définition de $\mathcal{F}$, on a omis les singletons et l'ensemble tout entier.

Théorème 1.2 [CHM81] Soit $\mathcal{F}$ une famille partitionnée et T^F l'arbre d'inclusion de ses parties fortes. Il existe une unique façon d'étiqueter les noeuds internes de T^F par **premier** ou **dégénéré** de telle sorte que les éléments de $\mathcal{F}$ soient exactement les ensembles représentés par les noeuds de T^F et les unions d'un sous-ensemble quelconque des fils d'un noeud dégénéré de T^F .

Un exemple de famille partitionnée $\mathcal{F}$ accompagnée de son arbre T^F est donné sur la figure 1.4.

Les noeuds internes de T^F ont au moins 2 fils. Comme le nombre de feuilles de T^F est n (le nombre de sommets du graphe), le nombre de noeuds internes est au plus $n - 1$. Ainsi, d'après le théorème 1.2, T^F muni des étiquettes adéquates sur ses noeuds internes est une représentation de taille $O(n)$ d'une famille partitionnée, alors que le nombre d'éléments d'une telle famille peut atteindre 2^n .

D'après le théorème 1.1, l'ensemble des modules d'un graphe est une famille partitionnée. L'ensemble des modules d'un graphe peut donc être représenté par l'arbre d'inclusion des modules forts. Il se trouve que cet arbre est celui d'une décomposition en modules particulière, que nous dirons canonique, est qui présentée plus loin.

Les graphes indécomposables et les graphes entièrement décomposables

Un graphe à au plus deux sommets n'a nécessairement que des modules triviaux.

Définition 1.4 Un graphe est dit *premier* ssi il a au moins trois sommets et tous ses modules sont triviaux.

La figure 1.5 donne en exemple deux graphes premiers très connus : le P_4 et le taureau.

Les graphes indécomposables pour la décomposition modulaire, c'est à dire les graphes G n'admettant pour décomposition que la décomposition triviale $\{G\}$, sont donc les graphes premiers et ceux ayant au plus deux sommets.

A l'opposé, on définit les graphes **friables** (par analogie avec les "brittle" de la décomposition en coupes [Cun82]) comme les graphes pour lesquels tout sous-ensemble de sommets

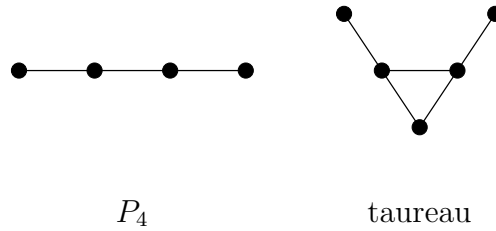


FIG. 1.5 – Deux graphes premiers.

est un module. Cette classe de graphe est évidemment très restreinte : il s'agit des graphes complets et des graphes vides.

Enfin, on définit les graphes entièrement décomposables par la décomposition modulaire comme les graphes dont tout sous graphe induit sur au moins trois sommets admet un module non-trivial. De manière équivalente, ces graphes sont ceux qui ne contiennent aucun sous graphe induit premier. Il sera vu (théorème 1.4) que ce sont exactement les graphes sans P_4 induits. Ces graphes sont bien connus sous le nom de cographes (voir 4.2).

Décomposition canonique

Il n'y a pas en général unicité de la décomposition en modules d'un graphe : celle-ci dépend des modules choisis pour décomposer le graphe. En particulier, lorsque deux modules M_1 et M_2 se chevauchent, si on décompose le graphe selon M_1 puis chacun des graphes obtenus selon M_2 (ou plutôt selon l'intersection de M_2 avec l'ensemble des sommets du graphe considéré), on obtient pas les mêmes graphes que si on quotiente d'abord par M_2 puis par M_1 (voir figure 1.6). On peut palier à ce problème en définissant une décomposition canonique. Il existe au moins deux façons de définir la décomposition canonique, qui aboutissent toutes deux à la même décomposition, mais qui donnent chacune une compréhension complémentaire de ce qu'est une décomposition plus satisfaisante que les autres.

La non unicité de la décomposition en modules vient de la possibilité de choisir de quotienter d'abord par un module ou par un autre qui le chevauche, ce qui ne conduit pas au même résultat. La première vision, adoptée par Cunningham pour la décomposition en coupes, a pour idée directrice de concentrer cet indéterminisme dans le choix des modules qui se chevauchent dans certains graphes de la décomposition. Ces graphes sont les graphes friables, ceux dont tout sous-ensemble de sommets est un module. Une décomposition bien formée est alors définie comme une décomposition dans laquelle les graphes sont soit indécomposables soit friables. Il existe encore beaucoup de décompositions de cette sorte. Ne serait-ce que parce qu'une décomposition simple d'un graphe friable donne deux graphes friables. Intuitivement, une décomposition satisfaisante serait une décomposition bien formée dans laquelle on a jamais décomposé de graphes friables. Ce qu'a montré Cunningham pour la décomposition en coupes, c'est que parmi les décompositions bien formées il y en a une unique dont dérive toutes les autres. Cela répond parfaitement à l'attente formulée précédemment. Ce résultat est vrai aussi pour la décomposition modulaire.

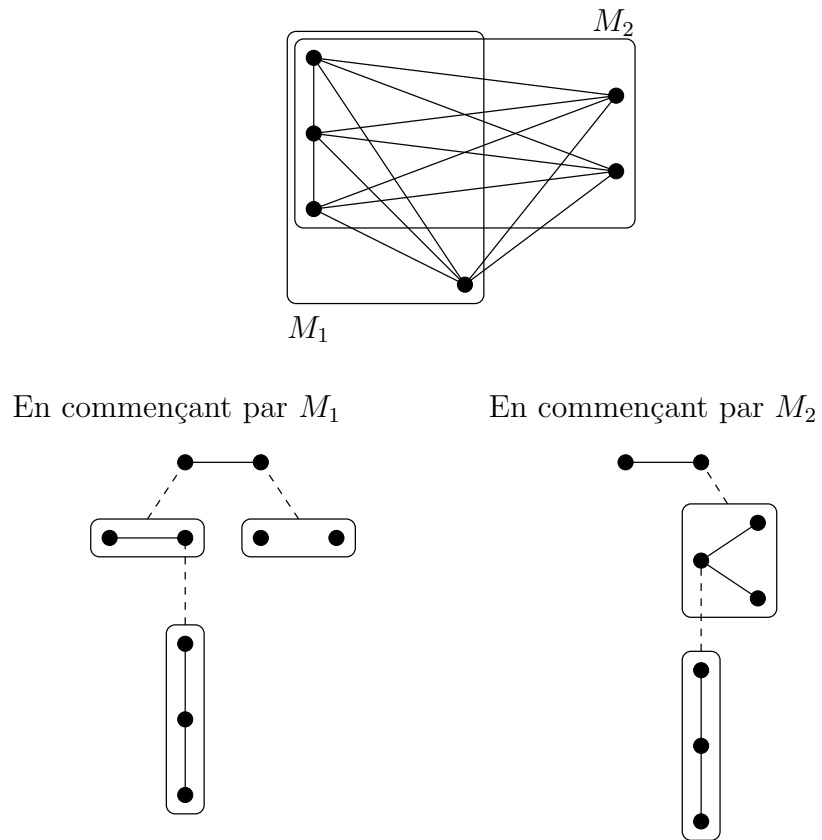


FIG. 1.6 – Deux décompositions en modules utilisant M_1 et M_2 : on obtient pas le même résultat en commençant par décomposer selon M_1 ou selon M_2 .

Ce n'est cependant pas la façon dont on introduit en général la décomposition modulaire. On utilise plutôt la notion de module fort, comme dans le cadre général des familles partitives. En fait, l'arbre de décomposition modulaire canonique peut être défini comme l'arbre d'inclusion des modules forts, exactement comme l'arbre T^F représentant une famille partitive. Ce qu'il manque à T^F c'est de représenter la structure du graphe, il ne tient compte que des relations ensemblistes entre les modules. Ce manque est comblé par le théorème 1.3 dû à Gallai. Nous détaillons un peu cette approche dans ce qui suit. Dans cette vision, le problème du choix entre les modules qui se chevauchent est réglé de manière radicale : on ne quotientte que par les modules qui n'en chevauchent aucun autre.

Le lien avec la vision de Cunningham est assez simple à établir. On décompose G uniquement par des modules forts, pour ne pas décomposer les sous graphes friables, et on décompose par tous les modules forts pour que les graphes non friables restant dans la décomposition soient indécomposables. Ce qui suit donne une présentation plus rigoureuse de la décomposition modulaire (décomposition en modules canonique) d'un graphe.

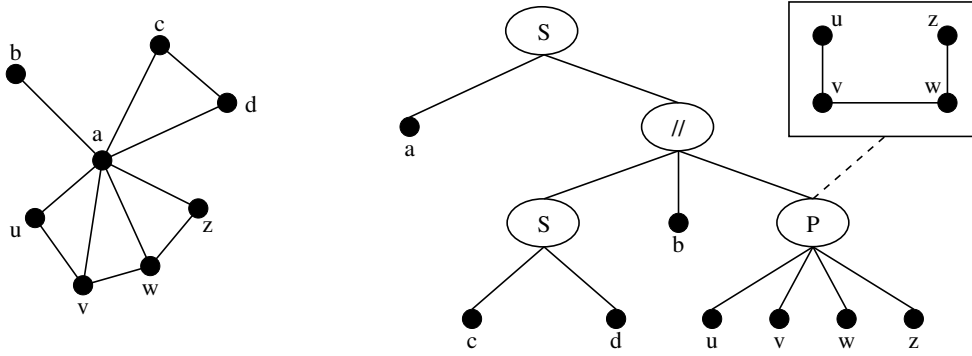
Définition 1.5 *Un module M est dit **fort** ssi il ne chevauche aucun autre module M' , c'est à dire que $M \cap M' = \emptyset$ ou $M \subseteq M'$ ou $M' \subseteq M$. On appelle les **modules forts maximaux** d'un graphe $G = (V, E)$, dont l'ensemble est noté $\mathcal{MFM}(G)$, les modules forts de G différents de V maximaux pour l'inclusion.*

On définit l'arbre de décomposition modulaire, noté $T^m(G)$, comme la réduction transitive de l'ordre d'inclusion des modules forts. il y a donc identité entre l'arbre de décomposition modulaire et celui permettant de représenter la famille partitive des modules. Une question reste en suspend : quelles étiquettes (premier ou dégénéré) attribuer aux noeuds de l'arbre ? Le théorème 1.3 répond à la question.

En définissant l'arbre de décomposition modulaire comme nous venons de le faire, nous avons complètement écarté la vision décomposition en modules. Où sont les graphes issus de la décomposition ? Dans l'arbre associé à une décomposition en modules, les graphes de la décomposition correspondaient aux noeuds de l'arbre. Dans l'arbre de décomposition modulaire, on associe un graphe à chaque noeud p . Le sous-ensemble de sommets de G représenté par p est noté $M(p)$ ou plus simplement P . On note $\{p_i\}_{1 \leq i \leq k}$ l'ensemble des fils de p . Par définition de la réduction transitive, $\mathcal{MFM}(G[P]) = \{P_i\}_{1 \leq i \leq k}$. Cela justifie que l'on associe à chaque noeud p de l'arbre de décomposition modulaire le graphe quotient $G_p = G[P]/\mathcal{MFM}(G[P])$. Pour vérifier que G_p est bien défini, il suffit de remarquer que $\mathcal{MFM}(G[P])$ est une partition de congruence de $G[P]$. Le théorème suivant est le théorème majeur de la décomposition modulaire.

Théorème 1.3 [Gal67] *Soit G un graphe ayant strictement plus d'un sommet. Un et un seul des cas suivants est vrai :*

- $G/\mathcal{MFM}(G)$ est un stable ; ou
- $G/\mathcal{MFM}(G)$ est une clique ; ou
- $G/\mathcal{MFM}(G)$ est premier.

FIG. 1.7 – Un graphe et sa MD -représentation.

Idée de la preuve. Si G est non-connexe, ses modules forts maximaux sont ses composantes connexes et $G/\mathcal{MFM}(G)$ est un stable. Si $\overline{G}$ est non-connexe, les modules forts maximaux de G sont les composantes connexes de $\overline{G}$ et $G/\mathcal{MFM}(G)$ est une clique. Enfin, si G et $\overline{G}$ sont connexes, les modules maximaux (ceux maximaux pour l'inclusion parmi les modules qui ne sont pas l'ensemble de tous les sommets lui-même) de G sont forts. Ainsi, $G/\mathcal{MFM}(G)$ n'a que des modules triviaux et est donc premier. $\square$

Il s'ensuit que les quotients associés aux noeuds de la décomposition modulaire sont soit des stables, soit des cliques, soit des graphes indécomposables. Les cliques et les stables sont les graphes dont tout sous-ensemble de sommets est un module, ce sont les quotients associés aux noeuds dégénérés de l'arbre T^F représentant la famille partitive des modules. Les graphes premiers sont ceux dont les seuls modules sont les modules triviaux, ce sont les quotients associés aux noeuds premiers de T^F . Ainsi, le théorème 1.2 donne la façon de lire les modules d'un graphe G sur son arbre de décomposition modulaire.

Il est inutile de stocker les quotients qui sont des cliques ou des stables. On préfère associer une étiquette à chaque noeud : **parallèle** si le quotient associé est un stable, **série** si c'est une clique et **premier** si c'est un graphe premier. Les seuls quotients qu'on ait besoin de stocker sont donc ceux associés aux noeuds premiers. Les noeuds séries et parallèles sont aussi appelés noeuds **dégénérés**, comme le légitime la remarque précédente. Dans la suite, l'arbre de décomposition modulaire d'un graphe G dans lequel on a étiqueté les noeuds par série, parallèle et premier et dans le quel on a associé à chaque noeud premier son graphe quotient sera appelé la **MD -représentation** de G , notée $MD(G)$. La figure 1.7 donne un graphe avec sa MD -représentation.

Propriétés de la décomposition modulaire des graphes

L'arbre de décomposition modulaire apporte plusieurs informations sur le graphe. En premier lieu, il décrit la structure des modules. Mais il permet aussi de reconstruire le graphe entier, de lire l'adjacence entre deux sommets et de lire les jumeaux (défini ci-dessous).

Pour reconstruire le graphe à partir de l'arbre de décomposition modulaire et des quotients de ses noeuds premiers, on peut appliquer la composition par substitution de bas en haut dans l'arbre. Si on souhaite seulement connaître la relation d'adjacence entre deux sommets x et y de G , on peut utiliser la méthode de l'ancêtre qui consiste à trouver le plus petit ancêtre commun (noté *ppca* dans la suite) des deux feuilles représentant x et y . Si leur *ppca* est un noeud série, alors x et y sont adjacents, si c'est un noeud parallèle, il ne le sont pas. Si le *ppca* de x et y est premier, ils ont la même relation d'adjacence que les sommets du quotient qui correspondent aux fils du *ppca* qui sont respectivement l'ancêtre de x et de y .

Définition 1.6 *Deux sommets x, y d'un graphe G sont **jumeaux** si $N(x) \setminus \{y\} = N(y) \setminus \{x\}$. Si x et y sont jumeaux et non adjacents on dit qu'ils sont faux jumeaux, et vrais jumeaux s'ils sont adjacents.*

Il est facile de trouver les jumeaux d'un sommet x sur l'arbre de décomposition modulaire. x a un jumeau ssi son père est un noeud dégénéré et a des fils feuilles. Les jumeaux de x sont précisément les sommets correspondant à ces feuilles. Si $\text{parent}(x)$ est parallèle, ce sont de faux jumeaux, si $\text{parent}(x)$ est série, ce sont de vrais jumeaux.

La notion de permutation factorisante, introduite dans [Cap97], sera de premier intérêt dans ce mémoire, elle correspond à la notion de frontière définie sur les *PQ*-arbres et les *PC*-arbres (voir section 1.4).

Définition 1.7 [Cap97] *Une permutation factorisante d'un graphe G est une permutation τ des sommets de G telle que tout module fort de G soit un facteur de τ .*

L'ordre de rencontre des feuilles de T dans un parcours en profondeur de T depuis la racine est une permutation factorisante. Et réciproquement, toute permutation factorisante peut être obtenue ainsi.

On dresse ci-après un éventail de propriétés des modules qui nous seront utiles dans les preuves de ce manuscrit.

Cette première propriété exprime que les modules sont conservés par restriction du graphe à un sous-graphe induit par un sous-ensemble de sommets.

Lemme 1.1 *Soit M un module de $G = (V, E)$ et $S \subseteq V$ un sous-ensemble de sommets. $M \cap S$ est un module de $G[S]$.*

Lemme 1.2 *Un module M d'un graphe G est aussi un module de $\overline{G}$.*

Il découle du lemme 1.2 que le complémentaire d'un graphe premier est premier. L'arbre de décomposition modulaire du complémentaire $\overline{G}$ de G est le même que celui de G , seuls changent les quotients associés aux noeuds qui sont les complémentaires de ceux associés aux noeuds de $T^m(G)$.

Comme le montre le théorème suivant, il existe un unique graphe premier minimal au sens des sous-graphes induits, qui est le chemin à 4 sommets noté P_4 .

Théorème 1.4 [Gal67, Sei74] *Si G est un graphe premier alors G contient un P_4 comme sous-graphe induit.*

Il est connu que par tout sommet d'un graphe premier sauf éventuellement un passe un P_4 [ER90a]. Cette propriété n'est pas aisée à établir, par contre la suivante, qui dit que tout sommet d'un graphe premier est l'extrémité d'un P_3 est très simple. Elle nous servira plus tard.

Lemme 1.3 *Soit x un sommet d'un graphe premier G . Il existe $y \in N(x)$ et $z \in \overline{N}(x)$ tels que yz est une arête de G .*

Preuve : Dans un graphe premier $G = (V, E)$, tout sommet x a au moins un voisin et un non voisin, sinon $V \setminus x$ serait un module. Si tous les voisins de x sont non adjacents aux non voisins de x , alors $N(x) \cup \{x\}$ est un module de G . ■

La propriété suivante établit que le quotient selon une partition de congruence respecte les modules forts.

Lemme 1.4 *Soient $\mathcal{P}$ une partition de congruence de G et $\mathcal{X} \subseteq \mathcal{P}$. On a l'équivalence : $\mathcal{X}$ est un module fort non trivial de $G/\mathcal{P}$ ssi $X = \cup_{M \in \mathcal{X}} M$ est un module fort non trivial de G .*

Preuve : Il est prouvé dans [MR84] que $\mathcal{X} \subseteq \mathcal{P}$ est un module de $G/\mathcal{P}$ ssi $X = \cup_{M \in \mathcal{X}} M$ est un module de G . Soit $\mathcal{X}$ un module non trivial de $G/\mathcal{P}$.

$\Leftarrow$ Supposons que $\mathcal{X}$ n'est pas un module fort de $G/\mathcal{P}$. Alors, il existe un module $\mathcal{Y} \subseteq \mathcal{P}$ de $G/\mathcal{P}$ tel que $\mathcal{X} \otimes \mathcal{Y}$. De ce qui précède, on déduit que $Y = \cup_{M \in \mathcal{Y}} M$ est un module de G . Par conséquent, $Y \otimes X$ et X n'est pas un module fort de G .

$\Rightarrow$ Supposons que X n'est pas un module fort de G . Alors, c'est l'union de sous-modules forts maximaux d'un module fort dégénéré Z de G . Puisque $\mathcal{P}$ est une partition de congruence, il existe $\mathcal{Y} \subseteq \mathcal{P}$ tel que $Y = \cup_{M \in \mathcal{Y}} M = Z \setminus X$. Comme Z est dégénéré, Y est un module de G et $\mathcal{Y}$ est un module de $G/\mathcal{P}$. Par hypothèse, $\mathcal{X}$ est non trivial. Il s'ensuit qu'il existe $\mathcal{S} \subsetneq \mathcal{X}$ tel que $S = \cup_{M \in \mathcal{S}} M$ est l'union de certains sous-modules forts maximaux de Z . Comme Z est dégénéré, $Y \cup S$ est un module de G . En d'autres termes $\mathcal{X}$ et $\mathcal{Y} \cup \mathcal{S}$ sont aussi des modules de $G/\mathcal{P}$. Donc $\mathcal{X}$ n'est pas un module fort de $G/\mathcal{P}$, car par définition de $\mathcal{S}$, $\mathcal{X} \otimes \mathcal{Y} \cup \mathcal{S}$. ■

1.2.2 Décomposition en coupes

La décomposition en coupes est fortement liée à la décomposition modulaire. [Lan01] constitue une très bonne introduction sur le sujet et présente de nombreuses relations entre modules et coupes d'un graphe. La décomposition en coupes a reçu nettement moins d'attention que la décomposition modulaire, mais s'est pourtant montrée très utile

dans de nombreux contextes. Cette décomposition a été introduite en 1982 par Cunningham [Cun82] qui la définit sur les graphes orientés. Nous l'étudierons ici seulement dans le cas particulier des graphes non orientés, qui sont équivalents aux graphes orientés symétriques.

La décomposition en coupes s'est révélé très utile pour l'étude de certaines classes de graphes. C'est le cas des graphes distances héréditaires qui sont les graphes entièrement décomposables pour la décomposition en coupes [HM90], comme les cographes pour la décomposition modulaire. [GP07] utilise cette propriété pour développer un algorithme de reconnaissance de la classe entièrement dynamique sur les arêtes de complexité optimale. La décomposition en coupes joue, pour les graphes de cordes, le rôle joué par la décomposition modulaire pour les graphes de permutation : elle permet d'en représenter tous les modèles géométriques [Bou87, GSH89] (voir section 1.3.3). Cette représentation est aussi à la base de l'algorithme de reconnaissance de [Spi94].

La décomposition en coupes a des relations avec la largeur de rang [iO05b] ("rank width" en anglais), voir [iO05a] pour une introduction à la largeur de rang. Les graphes distance héréditaires sont exactement les graphes de largeur de rang au plus 2. La décomposition en coupes joue un rôle central dans l'algorithme de reconnaissance des graphes de largeur de clique au plus 3 de [Lan01]. Voir [CO00] pour une présentation de la largeur de clique.

Enfin, Courcelle [Cou06] montre que la décomposition en coupes, comme la décomposition modulaire et la décomposition en composantes triconnexes, est descriptible en logique monadique du second ordre. Cette approche consistant à décrire ces trois décompositions dans un cadre commun est celle qui est poursuivi dans ce manuscrit.

Bien que nombre de liens entre la décomposition modulaire et la décomposition en coupes soient déjà connus [Cun82, Lan01, dM03], ces deux décompositions ne sont en général pas présentées de la même façon. La raison en est peut-être que cela n'est pas complètement possible. Je m'efforce ici de rapprocher le plus possible les définitions des deux décompositions. Les travaux cités ci-dessus proposent déjà des cadres communs pour présenter certains aspects des deux théories, essayons de pousser l'unification un cran plus loin.

Dans la thèse de doctorat de Fabien de Montgolfier, un parallèle fort est mis en exergue entre les deux décompositions grâce à la théorie des **familles partitives** pour la décomposition modulaire et des **familles bipartitives** pour la décomposition en coupe. Ce parallèle apparaît aussi dans [HM03] en utilisant le point de vue des **arbres à degrés de liberté** (définis en section 1.4). La décomposition modulaire a été présentée sans vraiment utiliser les résultats généraux sur les familles partitives. Ce choix a été fait afin d'essayer de présenter la décomposition sous un angle le plus intuitif possible. Afin de mieux percevoir les similitudes entre décomposition modulaire et décomposition en coupe nous définissons succinctement et donnons les propriétés essentielles des familles partitives et bipartitives.

Dans ce qui suit, nous rapprochons l'arbre représentant les coupes d'un graphe avec les arbres de décomposition obtenus par [Cun82, Lan01]. Nous proposons, à l'instar de ce qui existe pour la décomposition modulaire, un arbre qui représente à la fois le graphe considéré et l'ensemble de ses coupes. Comme [Cun82], nous nous restreignons aux graphes connexes.

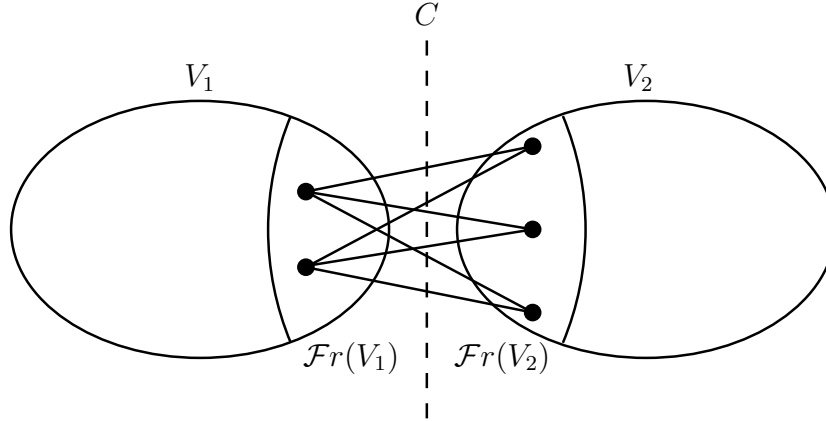


FIG. 1.8 – V_1 et V_2 sont des demi-coupes complémentaires l'une de l'autre. $C = \{V_1, V_2\}$ est une coupe.

Nous prenons ici le parti de présenter la théorie de la décomposition en coupes en favorisant la notion de demi-coupe plutôt que celle de coupe. Ce point de vue est celui adopté dans [HM03]. L'intérêt d'utiliser des ensembles plutôt que des bipartitions est de se rapprocher du cadre de la décomposition modulaire où les modules sont des ensembles. Cela permet de comparer plus directement les deux décompositions. Lorsqu'on utilise les bipartitions comme base de la décomposition en coupes, les deux décompositions apparaissent comme manipulant des objets différents ayant des rapports entre eux plus ou moins facile à saisir. En prenant les demi-coupes comme base de la théorie, les deux décompositions apparaissent comme manipulant des objets de mêmes types (les modules et les demi-coupes sont des ensembles de sommets) ayant des propriétés algébriques différentes ; la différence principale étant que la famille des demi-coupes est fermée par complémentation alors que celle des module ne l'est pas.

Dans tout le reste de la section sur la décomposition en coupe, bien que nous ne le précisions pas, tous les graphes considérés sont connexes.

Demi-coupes, décomposition simple suivant une demi-coupe et composition *.

Définition 1.8 (voir figure 1.8) Une demi-coupe d'un graphe $G = (V, E)$ est un sous-ensemble de sommets $M \subseteq V$ tel que $M \neq \emptyset$ et $M \neq V$ et $\forall x, y \in V \setminus M$, si $N(x) \cap M \neq \emptyset$ et $N(y) \cap M \neq \emptyset$ alors $N(x) \cap M = N(y) \cap M$.

Dans ce formalisme, un module peut être défini comme un sous-ensemble de sommets $M \subseteq V$ tel que $\forall x \in V \setminus M$, si $N(x) \cap M \neq \emptyset$ alors $M \subseteq N(x)$. Par conséquent, tout module est une demi-coupe.

Lemme 1.5 [HM03] Le complémentaire $V \setminus M$ d'une demi-coupe M est une demi-coupe. Et les arêtes entre M et $V \setminus M$ induisent un biparti complet.

Preuve : Soit M une demi-coupe de $G = (V, E)$. On pose $F_1 = \{y \in M \mid N(y) \cap (V \setminus M) \neq \emptyset\}$ et $F_2 = \{x \in V \setminus M \mid N(x) \cap M \neq \emptyset\}$. Pour montrer que $V \setminus M$ est une demi-coupe, il faut montrer que tous les sommets de F_1 ont le même voisinage dans $V \setminus M$. Soit donc $y \in F_1$, par définition de F_2 , $N(y) \cap (V \setminus M) \subseteq F_2$ et comme M est une demi-coupe et qu'il existe $x \in V \setminus M$ tel que x et y sont adjacents, alors $\forall z \in F_2, y \in N(z)$. Ce qui implique que $F_2 \subseteq N(y)$. Donc, $N(y) \cap (V \setminus M) = F_2$. Par symétrie, on a $\forall y \in F_2, N(y) \cap M = F_1$. Donc, les arêtes entre M et $V \setminus M$ induisent un biparti complet. ■

Définition 1.9 Une **coupe** C d'un graphe $G = (V, E)$ est une bipartition $\{V_1, V_2\}$ de ses sommets telle que V_1 est une demi-coupe.

D'après le lemme 1.5, si une partie d'une bipartition est une demi-coupe, l'autre aussi. La définition classique d'une coupe donnée dans [Cun82] est très proche de celle qui suit. La seule différence est qu'ici on s'autorise les coupes triviales, c'est à dire les coupes telles qu'une ou les deux parties de la bipartition ne contiennent qu'un sommet.

Définition 1.10 [Cun82] Une **coupe** C d'un graphe $G = (V, E)$ est une bipartition $\{V_1, V_2\}$ de ses sommets telle que les arêtes entre V_1 et V_2 induisent un biparti complet.

Ce qui fournit la caractérisation équivalente qui suit.

Lemme 1.6 [Cun82] Une bipartition $\{V_1, V_2\}$ de V est une coupe de $G = (V, E)$ ssi $\forall x_1, y_1 \in V_1, \forall x_2, y_2 \in V_2, x_1x_2 \in E$ et $y_1y_2 \in E \Rightarrow x_1y_2 \in E$.

L'ensemble des sommets incidents aux arêtes traversant la bipartition est appelé la **frontière** de C et noté $\mathcal{Fr}(C)$. Pour une demi-coupe V_1 , on note $\mathcal{Fr}(V_1) = \mathcal{Fr}(C) \cap V_1$, où $C = \{V_1, V \setminus V_1\}$. On a donc $\mathcal{Fr}(C) = \mathcal{Fr}(V_1) \sqcup \mathcal{Fr}(V_2)$.

Remarque 1.1 Dans les définitions qui précèdent, la notion de bipartition est à prendre au sens strict, c'est à dire qu'aucune partie ne peut être vide. Comme de plus nous nous restreignons à décomposer des graphes connexes, il découle que pour une coupe $C = \{V_1, V_2\}$, $\mathcal{Fr}(V_1) \neq \emptyset$ et $\mathcal{Fr}(V_2) \neq \emptyset$.

Remarquons également que d'après la définition de coupe et le lemme 1.5, il est parfaitement équivalent de donner une coupe ou une demi-coupe d'un graphe.

Définition 1.11 Pour une demi-coupe V_1 d'un graphe $G = (V, E)$, on appelle le **quotient de G selon la demi-coupe V_1** le graphe noté $G//V_1$ et défini par $G//V_1 = G[(V \setminus V_1) \cup \{\tilde{v}\}]$, avec $\tilde{v} \in \mathcal{Fr}(V \setminus V_1)$.

Remarque 1.2 L'utilisation du mot quotient dans ce contexte est abusive et ne doit pas tromper le lecteur. Il ne suffit pas, comme c'est le cas dans la décomposition modulaire, de choisir un sommet représentatif quelconque pour représenter une classe. Dans le cas de la décomposition en coupes, le sommet représentatif doit être choisi dans la frontière de la demi-coupe.

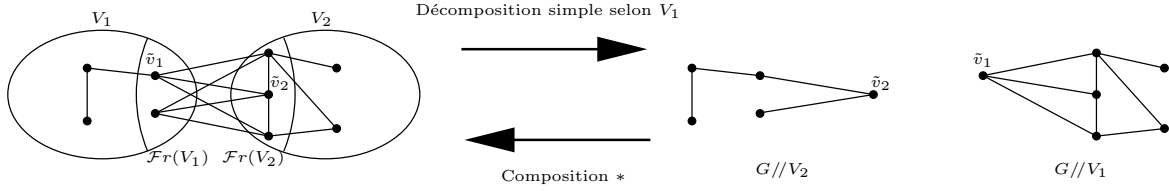


FIG. 1.9 – La décomposition simple selon une demi-coupe et la composition $*$ sont des opérations réciproques l’une de l’autre.

Définition 1.12 (voir figure 1.9) pour un graphe $G = (V, E)$ et une demi-coupe V_1 de G , on appelle **décomposition simple de G selon la demi-coupe V_1** , ou la coupe $\{V_1, V_2\}$ avec $V_2 = V \setminus V_1$, la paire $\{G//V_1, G//V_2\}$.

Remarque 1.3 Pour un graphe connexe G et une décomposition simple $\{G_1, G_2\}$ de G , G_1 et G_2 sont connexes.

Cela tient au fait que l’on laisse dans G_1 et G_2 un sommet représentatif de la frontière de l’autre partie de la bi-partition. Remarquons que ce fait permet d’appliquer la décomposition en coupes récursivement sur les graphes connexes.

A la place des deux sommets $\tilde{v}_1$ et $\tilde{v}_2$, [Cun82] introduit un sommet spécial unique. Les deux choix ont leur utilité. L’avantage de choisir deux sommets du graphe plutôt qu’un sommet spécial dupliqué dans chacune des deux parties est que cela renforce le parallèle avec le quotient de la décomposition modulaire, qui fait usage de sommets représentatifs des modules, et cela fait ressortir une propriété essentielle commune aux deux décompositions : les graphes obtenus par décomposition sont des sous-graphes induits de celui qu’on décompose. Cependant, la vision de Cunningham utilisant un sommet spécial se révèle très pratique pour définir l’arbre associé à une décomposition, nous ne nous priverons pas de l’utiliser.

L’opération de composition réciproque est appelée composition $*$ par [Cun82]. Elle est définie ainsi.

Définition 1.13 (voir figure 1.9) Soient $G_1 = (V_1 \cup \{\tilde{v}_2\}, E_1)$ et $G_2 = (V_2 \cup \{\tilde{v}_1\}, E_2)$ deux graphes, où V_1, V_2 est une bipartition d’un ensemble V ne contenant ni $\tilde{v}_1$ ni $\tilde{v}_2$. Le graphe $G = G_1 * G_2$ a pour ensemble de sommets V et pour ensemble d’arêtes $\{xy \in \mathcal{P}_2(V) \mid xy \in E_1 \cup E_2\} \cup \{xy \mid x, y \in V \text{ et } x \in V_1 \text{ et } y \in V_2 \text{ et } x\tilde{v}_2 \in E_1 \text{ et } y\tilde{v}_1 \in E_2\}$.

Remarquons que V_1 et V_2 sont des demi-coupes complémentaires de $G_1 * G_2$.

De manière plus générale on peut définir le quotient d’un graphe selon un ensemble $\mathcal{S} = \{M_1, \dots, M_k\}$ de demi-coupes deux à deux disjointes (voir figure 1.10). Le graphe quotient $G//\mathcal{S}$ est défini comme le sous graphe induit $G[R \cup S]$ avec $R = V \setminus \bigcup_{1 \leq i \leq k} M_i$ et $S = \{x_1, \dots, x_k\} \subseteq V$ et $\forall i \in \llbracket 1, k \rrbracket, x_i \in \mathcal{Fr}(M_i)$. De la définition d’une demi-coupe, il découle que la définition de $G//\mathcal{S}$ est insensible au choix des sommets représentatifs S .

Premières propriétés des demi-coupes.

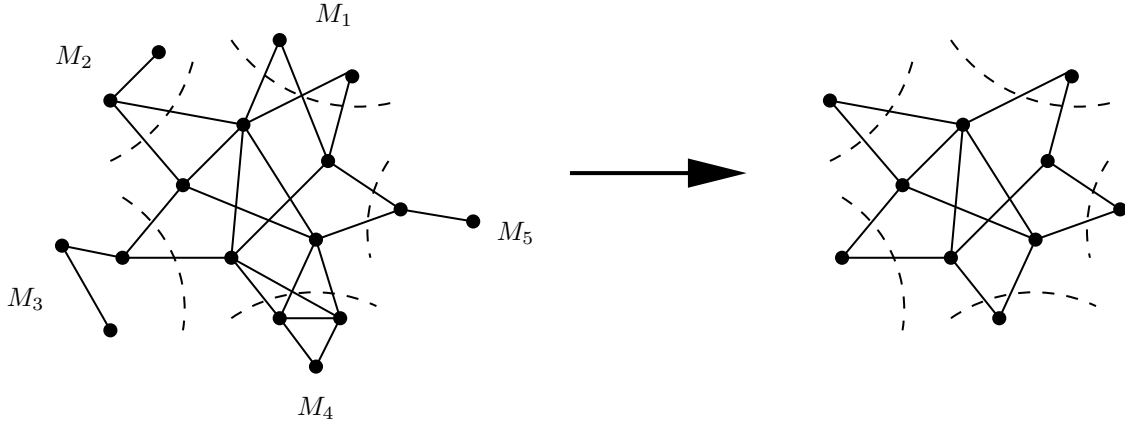
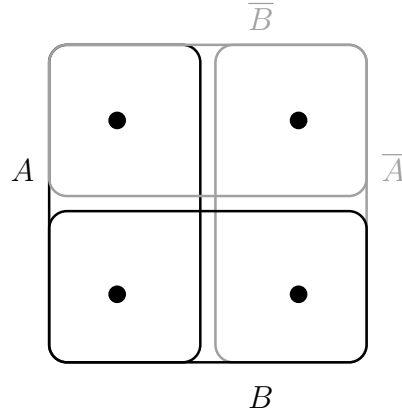


FIG. 1.10 – Quotient selon un ensemble de demi-coupes deux à deux disjointes.

FIG. 1.11 – Les sous-ensembles A et B se croisent.

Nous avons vu que l'ensemble des demi-coupes d'un graphe est clos par complémentation. C'est une propriété algébrique importante que ne possède pas l'ensemble des modules. Cette propriété des demi-coupes explique que certaines notions qui leur sont relatives aient besoin d'être définies par symétrie sur leur complémentaire. C'est le cas de la notion de chevauchement, qui dans le cas des demi-coupes sera appelé **croisement**.

Définition 1.14 (voir figure 1.11) Deux sous-ensembles $M_1, M_2 \subseteq V$ se croisent si M_1 et M_2 se chevauchent et leurs complémentaires font de même.

Comme nous le verrons, la notion de sous-ensembles qui se croisent revêt une importance majeure pour les demi-coupes. Dans la suite, lorsque l'ensemble de référence est clair, on note $\bar{A}$ le complémentaire de A dans cet ensemble.

Remarque 1.4 M_1 et M_2 se croisent ssi les quatre ensembles suivants sont non vide : $M_1 \cap M_2$, $M_1 \cap \bar{M}_2$, $\bar{M}_1 \cap M_2$ et $\bar{M}_1 \cap \bar{M}_2$. C'est comme ça que l'on définit classiquement le fait que deux coupes $\{M_1, \bar{M}_1\}$ et $\{M_2, \bar{M}_2\}$ se croisent.

A la lumière de la précédente, les trois remarques suivantes se passent de démonstration.

Remarque 1.5 Si deux demi-coupes M_1, M_2 se croisent, alors M_1 et $\overline{M_2}$ aussi.

Remarque 1.6 Deux demi-coupes M_1, M_2 se croisent ssi M_1 et M_2 se chevauchent et leurs complémentaires sont d'intersection non vide.

Remarque 1.7 Si on préfère raisonner sur les coupes, on remarquera que deux coupes se croisent ssi leurs parties qui ne contiennent pas un sommet donné x se chevauchent.

Théorème 1.5 [Cun82, dM03] *Si A et B sont deux demi-coupes d'un graphe G et se croisent alors :*

1. $A \cap B$ est une demi-coupe de G
2. $A \Delta B$ est une demi-coupe de G

Corollaire 1.2 *Si A et B sont deux demi-coupes de G et se croisent alors :*

- $A \cup B$ est une demi-coupe de G ; et
- $A \setminus B$ et $B \setminus A$ sont des demi-coupes de G .

Preuve : D'après la remarque 1.5, $\overline{B}$ chevauche $\overline{A}$ et A . Comme $A \cup B = \overline{\overline{A} \cap \overline{B}}$, $A \cup B$ est une demi-coupe. Et comme $A \setminus B = A \cap \overline{B}$, $A \setminus B$ est une demi-coupe ; et par symétrie, $B \setminus A$ également. ■

Lemme 1.7 *Soit $G = (V, E)$ un graphe et A une demi-coupe de G . Soit $S \subseteq V$ tel que $A \cap S \neq \emptyset$ et $S \setminus A \neq \emptyset$ et $G[S]$ est connexe. $A \cap S$ est une demi-coupe de $G[S]$.*

Preuve : Il est évident que les arêtes entre $A \cap S$ et $S \setminus A$ induisent un biparti complet. ■

Décompositions en coupes d'un graphe

De la définition de demi-coupe, il découle que, pour tout graphe $G = (V, E)$, les singletons $\{x\}$, $x \in V$ sont des demi-coupes, leurs complémentaires aussi. On les appelle les demi-coupes triviales.

Définition 1.15 *Une **demi-coupe triviale** M d'un graphe G est une demi-coupe telle que M ou $\overline{M}$ est réduit à un singleton.*

Les demi-coupes triviales d'un graphe G sont exactement les demi-coupes M pour lesquelles un des deux graphes G_1 et G_2 résultant de la décomposition simple de G selon M est le graphe G lui même. Dans la suite, nous ne considérons pas ces cas dégénérés de décomposition, mais uniquement les décompositions d'un graphe selon une demi-coupe non triviale.

Une décomposition en coupes d'un graphe G est un ensemble de graphes défini inductivement de la manière suivante :

- $D = \{G\}$ est une décomposition de G ; et
- si D_1 est une décomposition de G , si $G_1 \in D_1$ et si $\{G_2, G_3\}$ est une décomposition simple de G_1 , alors $D = (D_1 \setminus \{G_1\}) \cup \{G_2, G_3\}$ est une décomposition de G .

Cette définition, due à Cunningham, est rigoureusement identique à celle que nous avons utilisé pour la décomposition en modules. Nous reprenons le vocabulaire introduit alors, qui est une traduction (bien que peu fidèle) de celui utilisé dans [Cun82]. La deuxième opération qui permet d'obtenir D à partir d'une décomposition simple d'un élément de D_1 est appelé **dérivation simple**. Lorsqu'une décomposition D' est obtenue à partir d'une décomposition D en appliquant une suite de dérivations simples, on dit que D' **dérive** de D .

Dans une décomposition D , on prend garde à ce que

- les sommets spéciaux utilisés dans les dérivations simples permettant d'obtenir D soient distincts des sommets du graphe et à ce que
- à deux dérivations différentes soient associés deux sommets spéciaux distincts.

Ainsi, à toute décomposition D d'un graphe G on peut associer un arbre T défini de manière univoque comme suit. Les sommet de T sont les membres de D et il y a une arête entre deux graphes de D si ils partagent un même sommet spécial. Clairement, avec la précaution précédente, un sommet spécial n'engendre qu'une arête. Le lecteur peut vérifier que le graphe ainsi défini est simple et sans cycle.

[Lan01] représente l'arbre de décomposition d'une manière équivalente. Lorsqu'une décomposition simple d'un graphe G en G_1 et G_2 est opérée, le graphe décomposé est représenté par l'union disjointe de G_1 et G_2 dans laquelle on rajoute une arête, dite spéciale, entre le sommet spécial de G_1 et celui de G_2 . En réitérant le procédé, on obtient une représentation de n'importe quelle décomposition de G . Si on retire les arêtes spéciales, les composantes connexes sont exactement les graphes de la décomposition au sens de [Cun82]. Si on contracte chaque composante connexe en un sommet unique et si on remplace les arêtes spéciales entre les deux sommets représentant les composantes connexes qu'elles reliaient, on obtient l'arbre de décomposition de [Cun82].

On peut reprocher à ces représentations de ne pas faire apparaître les sommets du graphe comme feuilles de l'arbre de décomposition. Pourtant, lorsqu'on le fait, on obtient une représentation des demi-coupes utilisées pour décomposer le graphe, comme nous le verrons plus tard sur la décomposition en coupes canonique. La représentation que nous adoptons pour une décomposition en coupes est basée sur un arbre noté T^s . Les feuilles de T^s sont les sommets de G et les noeuds internes de T^s représentent les graphes de D . Le graphe associé à un noeud p de T^s est noté G_p . Les voisins de p sont d'une part les noeuds internes q dont le graphe associé G_q partage un sommet spécial avec G_p , et d'autre part les feuilles de T^s correspondant aux sommets non-spéciaux de G_p . Comme un sommet non spécial appartient exactement à un graphe de D , cela définit bien un arbre. Il est à noter que cette représentation est aussi celle adoptée dans [GP07] pour la classe des graphes distance héréditaires.

Ainsi, T^s restreint à ses sommets internes est exactement l'arbre de décomposition défini par Cunningham, et les graphes associés aux noeuds internes de T^s sont les graphes de la décomposition, que l'on retrouve aussi dans la représentation de [Lan01]. Le fait

de faire apparaître les sommets de G comme feuilles de T^s mime la représentation de la décomposition modulaire et permet en définissant une décomposition canonique de retrouver un arbre représentant toutes les coupes de G au sens des familles bi-partitives [dM03]. Cet aspect sera détaillé plus loin.

La figure 1.12 fait une comparaison entre la représentation de [Cun82], celle de [Lan01] et celle que nous adoptons dans ce mémoire.

L'arbre T^s avec les graphes associés à ses noeuds internes est une représentation complète du graphe. Pour retrouver G , il suffit d'appliquer la composition $*$ le long de toutes les arêtes de T^s reliant deux noeuds internes ; la composition s'appliquant entre les graphes associés aux noeuds incidents à l'arête considérée. On peut également lire l'adjacence entre deux sommets x et y sans reconstruire le graphe. Il suffit de vérifier que sur l'unique chemin C de T^s reliant la feuille de x à celle de y , tous les noeuds internes p du chemin sont tels que les deux sommets de G_p associés aux deux voisins de p dans C sont adjacents (voir [Lan01] qui montre ce résultat dans le formalisme de sa représentation).

Familles bi-partitives

[dM03, HM03] montrent que l'on peut obtenir des représentations arborescente très efficaces pour des familles de bi-partition qui vérifie des axiomes ensemblistes comparables à ceux vérifiés par les familles partitives. Ces familles sont appelées bi-partitives par [dM03]. [HM03] étudie les mêmes familles, mais du point de vue des demi-bi-partition. Ici, nous suivons essentiellement le cheminement de [dM03], en adoptant la vision de [HM03].

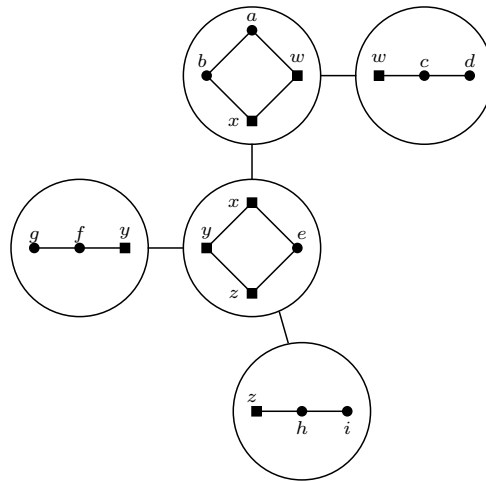
Définition 1.16 *Une famille $\mathcal{F}$ de sous-ensembles d'un ensemble V est bi-partitive ssi*

1. $\emptyset \notin \mathcal{F}$ et $\forall v \in V, \{v\} \in \mathcal{F}$; et
2. $\forall A \in \mathcal{F}, \overline{A} \in \mathcal{F}$; et
3. *pour tous sous-ensembles $A, B \in \mathcal{F}$ tels que A croise B , les deux propriétés suivantes sont vraies :*
 - (a) $A \cap B \in \mathcal{F}$
 - (b) $A \Delta B \in \mathcal{F}$

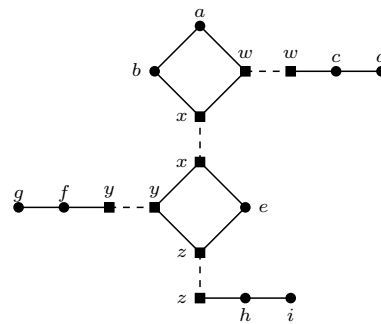
Les **parties fortes** de $\mathcal{F}$ sont les parties de $\mathcal{F}$ qui n'en croisent aucune autre. A l'évidence, les singletons sont des parties fortes.

La sous-famille $\mathcal{F}'$ des parties fortes d'une famille bi-partitive $\mathcal{F}$ forme une **famille arborée** au sens de [dM03]. Ainsi, on peut représenter cette sous-famille $\mathcal{F}'$ par un arbre non-enraciné T^B dont les feuilles sont les éléments de V et tel que les éléments de $\mathcal{F}'$ sont exactement les demi-ensembles par arête des feuilles de T^B , définis ci-dessous.

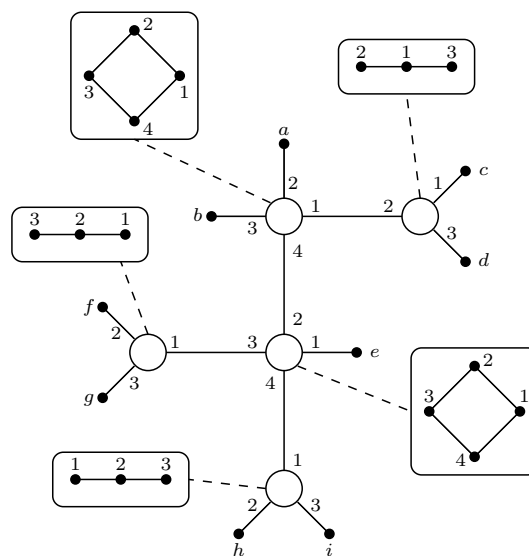
Définition 1.17 *(voir figure 1.13) Soit T un arbre et V l'ensemble de ses feuilles. Un **demi-ensemble par arête** de V est un sous-ensemble $S \subseteq V$ tel qu'il existe une arête e de T telle que S est l'ensemble des feuilles d'un des deux arbres résultant du retrait de e dans T .*



La représentation de [Cun82]



La représentation de [Lan01]



La représentation adoptée dans ce mémoire

FIG. 1.12 – Différentes représentations d'une décomposition d'un graphe.

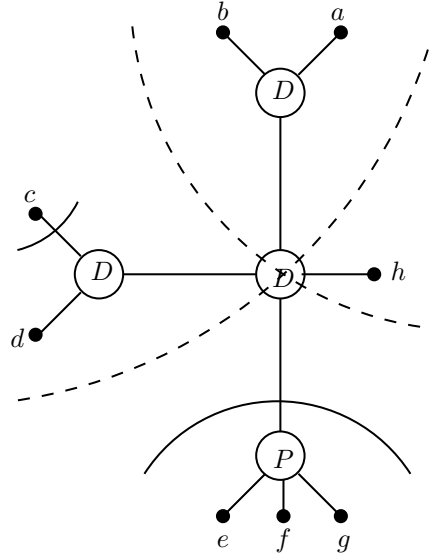


FIG. 1.13 – L'arbre T^B d'une certaine famille bipartitive. Les demi-ensembles par arêtes sont représentés par des traits pleins ($\{c\}$ et $\{e, f, g\}$); et les demi-ensembles par noeud sont représentés par des traits en pointillés ($\{a, b, c, d\}$ et $\{a, b, h\}$). Les noeuds étiquetés P sont les noeuds premiers et ceux étiquetés D les noeuds dégénérés.

Définition 1.18 (voir figure 1.13) Soit T un arbre et V l'ensemble de ses feuilles. Un **demi-ensemble par noeud** de V est un sous-ensemble $S \subseteq V$ tel qu'il existe un noeud p de T et un sous-ensemble A des voisins de p tels que S est l'ensemble des feuilles des arbres contenant les noeuds de A dans la forêt résultant du retrait de p dans T .

[dM03] montre que l'arbre T^B permet de représenter toutes les parties d'une famille bi-partitive, de la manière suivante.

Théorème 1.6 [dM03] Soit une famille bi-partitive $\mathcal{F}$, soit T^B l'arbre représentant ses parties fortes et soit V l'ensemble des feuilles de T^B . Il existe une façon d'étiqueter les noeuds internes de T^B par **premier** ou **dégénéré** de telle sorte que les éléments de $\mathcal{F}$ soient exactement les demi-ensembles par arête de V et les demi-ensembles par noeud de V obtenus à partir des noeuds dégénérés de T^B .

Un exemple d'arbre T^B est donné sur la figure 1.13.

De même que les modules d'un graphe sont une famille partitive, d'après le lemme 1.5 et le théorème 1.5, les demi-coupes d'un graphe sont une famille bi-partitive. Il s'ensuit que les coupes d'un graphe ont une représentation arborescente sympathique comme celle des modules. Elle est aussi compacte car si le nombre de coupes d'un graphe peut atteindre $2^n - 2$, en revanche le nombre de noeuds dans l'arbre les représentant est toujours $O(n)$.

Graphes indécomposables, friables et entièrement décomposables

Les graphes ayant au plus trois sommets n'admettent pas de demi-coupe non triviale, ils sont indécomposables.

Définition 1.19 *Les graphes premiers pour la décomposition en coupes sont ceux possédant au moins quatre sommets et n'admettant pas de demi-coupe non triviale.*

Ainsi, les graphes indécomposables par la décomposition en coupes sont les graphes à au plus trois sommets et les graphes premiers.

A l'opposé, les graphes friables sont ceux dont toute sous-ensemble des sommets (sauf l'ensemble vide et l'ensemble des sommets lui-même) est une demi-coupe.

Théorème 1.7 [Cun82] *Les graphes friables pour la décomposition en coupes sont les graphes complets et les étoiles.*

Nous retrouverons ce résultat comme conséquence du théorème de décomposition que nous établirons.

Les graphes entièrement décomposables par la décomposition en coupe sont ceux dont tout sous-graphe induit par au moins quatre sommets est décomposable. Ce sont aussi ceux qui ne contiennent pas de sous-graphe induit premier. Cette classe de graphes est bien connue sous le nom de **graphes distance héréditaires** [BM86, HM90]. Elle est présentée section 4.7.2.

Décomposition en coupes canonique

Il y a en général plusieurs décompositions en coupes possibles pour un graphe G . Cela vient du fait que lorsque deux demi-coupes se croisent, il faut choisir selon laquelle décomposer le graphe. Cunningham a défini une décomposition en coupes particulière que nous appelons canonique. L'idée est de ne considérer que les décompositions dont les graphes sont soit indécomposables, soit friables. Parmi ces décompositions il en existe une unique dont dérive toutes les autres : c'est celle que l'on appelle la décomposition en coupes canoniques.

Cela signifie que dans cette décomposition canonique aucun choix n'a été fait entre des demi-coupes qui se croisent. Autrement dit, les seules demi-coupes utilisées pour décomposer le graphe sont des demi-coupes fortes. Pour que les graphes de la décomposition qui ne sont pas friables soient premiers, il faut décomposer le graphe selon toutes les demi-coupes fortes. Ainsi, à l'instar de la décomposition modulaire, on peut définir la décomposition en coupes canonique comme étant la décomposition du graphe selon ses demi-coupes fortes. C'est l'angle sous lequel nous attaquons cette définition.

Je tiens à faire remarquer au lecteur que les résultats qui suivent peuvent facilement être retrouvés depuis ceux de [Cun82] et [dM03]. Réciproquement, les résultats contenus dans ces deux documents peuvent se déduire de ceux présentés ici. Ce qui est original dans ce manuscrit, ce ne sont pas les résultats eux mêmes, mais l'enchaînement proposé et la

vision qui s'en dégage. Ce qui suit s'applique à définir la décomposition en coupes canonique comme la décomposition modulaire est définie à partir du théorème de décomposition de Gallai (théorème 1.3). Aussi, l'approche que j'adopte ici consiste à prouver un théorème de décomposition analogue pour la décomposition en coupes (théorème 1.9), qui permet de déduire les autres résultats classiques servant à définir cette décomposition.

Définition 1.20 *Une demi-coupe forte est une demi-coupe qui n'en croise aucune autre. On appelle **demi-coupe forte maximale de G excluant x** , une demi-coupe forte de G ne contenant pas x et qui est maximale pour l'inclusion parmi les demi-coupes fortes non triviales qui ne contiennent pas x . L'ensemble des demi-coupes fortes maximales de G excluant x est noté $\mathcal{CFM}(G, x)$.*

Comme nous l'avons vu précédemment, la famille des demi-coupes d'un graphe G est une famille bi-partitive. Elle admet donc une représentation arborescente [dM03]. Cet arbre T^B est construit à partir des éléments forts de la famille, qui forment une famille arborée au sens de [dM03]. Les feuilles de T^B sont les sommets de G , et les arêtes de T^B définissent les demi-coupes fortes de G . A chaque arête a de T^B correspondent deux demi-coupes fortes de G , complémentaires l'une de l'autre. Ces deux demi-coupes sont les ensembles de feuilles des deux arbres issus du retrait de a dans T^B .

T^B représente toutes les demi-coupes du graphe de manière ensembliste, mais les arêtes du graphe ne sont pas décrites par cette représentation. Le but de ce qui suit est de donner un théorème de décomposition qui permet d'associer à chaque noeud de T^B un graphe. Les graphes ainsi associés aux noeuds de T^B sont exactement ceux de la décomposition canonique de Cunningham.

Quel graphe associer à un noeud u de T^B ? Chaque arête a incidente à u définit une paire de demi-coupes complémentaires : une qui est l'ensemble V_u^a des feuilles de la composante connexe de u après la retrait de a , l'autre qui est l'ensemble $V_{\bar{u}}^a$ des feuilles de la composante connexe ne contenant pas u . Si on veut conserver l'information relative au noeud u , il convient de quotienter, pour chaque arête a qui lui est incidente, par V_u^a . Ainsi, en notant A l'ensemble des arêtes incidentes à u , le graphe associé au noeud u est : $G//\{V_u^a \mid a \in A\}$.

Définition 1.21 *Deux demi-coupes M_1, M_2 d'un graphe G telles que $M_1 \cap M_2 = \emptyset$ sont **inséparables** ssi aucune demi-coupe forte S de G n'est telle que $M_1 \subseteq S \subseteq \overline{M_2}$.*

Lemme 1.8 *Soit G un graphe et $\mathcal{C} = \{M_1, \dots, M_k\}$, avec $k \geq 2$, un ensemble de demi-coupes fortes de G deux à deux disjointes. Soit $S = \{x_1, \dots, x_k\}$ l'ensemble des sommets représentatifs choisis dans chacune des demi-coupes de la partition et soit $R = V \setminus \bigcup_{1 \leq i \leq k} M_i$. Si A et B sont deux demi-coupes de G et se croisent, alors $A \cap (R \cup S)$ et $B \cap (R \cup S)$ sont deux demi-coupes de $G//\mathcal{C}$ et se croisent.*

Preuve : Comme les demi-coupes de $\mathcal{C}$ sont fortes, A et B ne chevauchent aucun des ensembles $M_1, \dots, M_k$. Soient $C, D \in \{A, \bar{A}, B, \bar{B}\}$. Comme $C \cap D \neq \emptyset$, alors $(C \cap D) \cap R \neq \emptyset$ ou $\exists i \in \llbracket 1, k \rrbracket, M_i \subseteq C \cap D$. Par conséquent $(C \cap D) \cap (R \cup S) \neq \emptyset$. On en déduit que $A \cap (R \cup S)$ et $B \cap (R \cup S)$ sont des demi-coupes de $G[R \cup S] = G//\mathcal{C}$ (voir lemme 1.7) et

qu'elles se croisent. ■

Lemme 1.9 *Soit G un graphe et $\mathcal{C}$ une partition en demi-coupes de G telle que toutes les demi-coupes de $\mathcal{C}$ sont fortes et deux à deux inséparables. $G//\mathcal{C}$ n'admet pas de demi-coupe forte non triviale.*

Preuve : On note $\mathcal{C} = \{M_1, \dots, M_k\}$, avec $k \geq 2$, et $\{x_1, \dots, x_k\}$ l'ensemble des sommets représentatifs choisis pour obtenir $G//\mathcal{C}$.

Soit A une demi-coupe non triviale de $G//\mathcal{C}$. On commence par montrer que $B = \bigcup_{\{i \mid x_i \in A\}} M_i$ est une demi-coupe non triviale de G . Soient $y_1, z_1 \in B$ et $y_2, z_2 \in \overline{B}$, tels que y_1 est adjacent à y_2 et z_1 est adjacent à z_2 . Montrons que y_1 est adjacent à z_2 , ce qui, d'après le lemme 1.6, montrera que B est une demi-coupe. Soit $\{M_{y_1}, M_{z_1}, M_{y_2}, M_{z_2}\} \subseteq \mathcal{C}$ tel que $\forall \alpha \in \{y_1, z_1, y_2, z_2\}, \alpha \in M_\alpha$. Comme y_1 est adjacent à y_2 , $y_1 \in \mathcal{F}r(M_{y_1})$ et $y_2 \in \mathcal{F}r(M_{y_2})$. Soit x_1 et x_2 les sommets représentant respectivement M_{y_1} et M_{y_2} . x_1 et x_2 sont adjacents. De même, en notant x_3 et x_4 les sommets représentant respectivement M_{z_1} et M_{z_2} , comme z_1 et z_2 sont adjacents, on en déduit que x_3 et x_4 aussi. Or comme A est une demi-coupe, le lemme 1.6 fournit que x_1 est adjacent à x_4 . Comme $x_1 \in \mathcal{F}r(M_{y_1})$ et $x_4 \in \mathcal{F}r(M_{z_2})$, alors $\mathcal{F}r(M_{y_1})$ et $\mathcal{F}r(M_{z_2})$ sont entièrement adjacents. En particulier, y_1 est adjacent à z_2 .

Maintenant, supposons que A est forte. Comme aucune demi-coupe de $G//\mathcal{C}$ ne croise A , alors, d'après le lemme 1.8, aucune demi-coupe de G ne croise B . B est donc forte. Soit $i, j \in \llbracket 1, k \rrbracket$ tels que $M_i \subseteq B$ et $M_j \subseteq \overline{B}$. Comme A est non triviale, $M_i \subsetneq B$ et $M_j \subsetneq \overline{B}$. On a donc $M_i \subsetneq B \subsetneq \overline{M_j}$. Ce qui contredit le fait que M_i et M_j sont inséparables. Donc A n'est pas forte. ■

Théorème 1.8 *Soit G un graphe connexe. Si G admet une demi-coupe non triviale et si G n'est ni une étoile ni une clique, alors G admet une demi-coupe forte non triviale.*

Le théorème 1.8 est la clef de l'approche que nous présentons ici. Je n'en fournirai pas une preuve directe qui se révèle être assez difficile, comparé par exemple à la preuve du théorème de décomposition pour la décomposition modulaire (théorème 1.3 page 37). Cependant, le théorème 1.8 est de démonstration très simple pour qui connaît déjà la théorie de la décomposition en coupes telle qu'elle est présentée dans [Cun82] et enrichie par [dM03] de la vision des familles bipartitives.

Idée de la preuve. Comme G admet une demi-coupe non triviale, G n'est pas premier. Or G n'est ni une étoile ni une clique. Donc, l'arbre de décomposition en coupe de G comporte strictement plus d'un noeud, et par conséquent comporte au moins une arête a . Les deux demi-ensembles par arête correspondant à a sont des demi-coupes fortes de G . □

Grâce au théorème 1.8, on est en mesure d'établir le théorème de décomposition recherché.

Théorème 1.9 *Soit G un graphe et $\mathcal{C}$ une partition en demi-coupes de G telle que toutes les demi-coupes de $\mathcal{C}$ sont fortes et deux à deux inséparables. Le graphe $G//\mathcal{C}$ est :*

- soit une étoile,
- soit une clique,
- soit un graphe premier.

Preuve : D’après le lemme 1.9, $G//\mathcal{C}$ n’admet pas de coupe forte non triviale. Or d’après le théorème 1.8, si $G//\mathcal{C}$ n’est ni une étoile ni une clique et admet une coupe non triviale, alors $G//\mathcal{C}$ admet une coupe forte non triviale. Par conséquent, si $G//\mathcal{C}$ n’est ni une étoile ni une clique, alors $G//\mathcal{C}$ n’admet aucune coupe non triviale : $G//\mathcal{C}$ est premier. ■

Nous avons décidé d’associer à un noeud $u \in T^B$, le graphe $G//\{V_u^a \mid a \in A\}$, où A est l’ensemble des arêtes incidentes à u dans T^B et pour $a \in A$, V_u^a désigne l’ensemble des feuilles de la composante connexe ne contenant pas u après le retrait de a dans T^B . Par définition de T^B , $\{V_u^a \mid a \in A\}$ ne contient que des demi-coupes fortes. De plus, comme toutes les demi-coupes fortes de G sont des demi-coupes par arêtes, alors les demi-coupes fortes de $\{V_u^a \mid a \in A\}$ sont deux à deux inséparables.

D’après le théorème 1.9, $G//\{V_u^a \mid a \in A\}$ est une étoile, une clique ou un graphe premier pour la décomposition en coupes. T^B est l’arbre de décomposition canonique de Cunningham et les graphes associés à ses noeuds par le théorème 1.9 sont les graphes de cette décomposition canonique. Un exemple d’un graphe avec son arbre de décomposition en coupes canonique est fourni sur la figure 1.14. Les noeuds dont le quotient est une clique sont étiquetés C , ceux dont le quotient est une étoile sont étiquetés E et ceux dont le quotient est un noeud premier sont étiquetés P . Pour les noeuds E , une encoche repère leur voisin qui correspond au centre de l’étoile.

1.2.3 Décomposition en composantes triconnexes

Cette décomposition introduite par Tutte en 1966 [Tut66] vise à décomposer un graphe en graphes ayant des propriétés supérieures de connexité. Dans les lignes qui suivent, je m’efforce pour la présenter de reproduire le schéma d’argumentation adopté pour la décomposition modulaire et la décomposition en coupes. Il est troublant de s’apercevoir que cette nouvelle décomposition souffre très bien cette contorsion. En effet, la ressemblance entre la décomposition en composantes triconnexes et les deux premières décompositions n’est pas frappante a priori. D’ailleurs, certaines propriétés n’y sont plus présentes : les graphes obtenus par décomposition ne sont pas des graphes induits de celui que l’on décompose, et certains d’entre eux ne sont même pas simples, ce sont des multigraphes. Malgré tout, des similarités intéressantes entre cette décomposition et la décomposition modulaire et la décomposition en coupe ressortent. Nous essayons de les mettre en avant par la démarche et le choix du vocabulaire. Mentionnons que cette décomposition a aussi été revisitée par [HT73], et par [BM90, BT96] pour les graphes planaires (voir section 1.4.5).

Définition 1.22 *Soit k un entier non nul. Un graphe $G = (V, E)$ est k -connexe si $\forall S \subseteq V, |S| \leq k - 1 \Rightarrow G[V \setminus S]$ est connexe.*

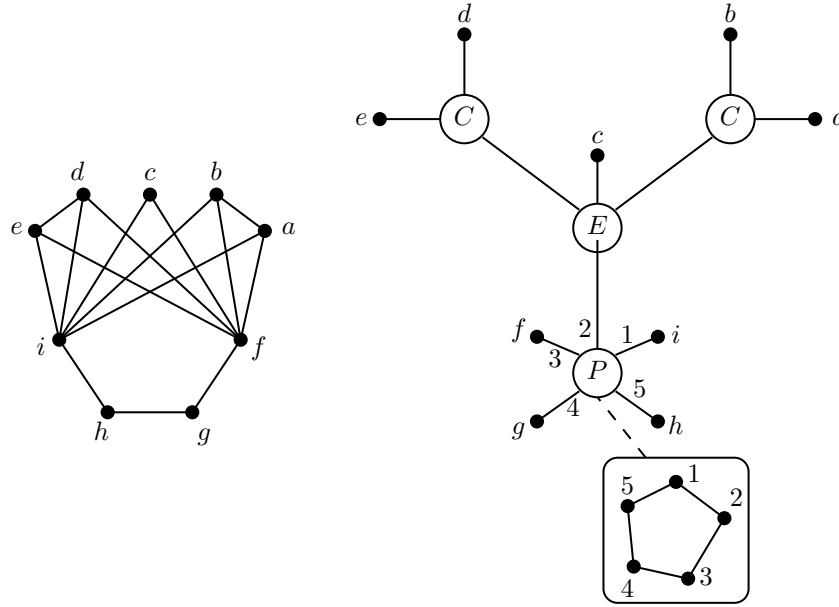


FIG. 1.14 – Un graphe et sa décomposition en coupes canonique.

Remarque 1.8 Un graphe est 1-connexe ssi il est connexe. La 0-connexité peut être considérée comme une propriété triviale satisfaite par tous les graphes.

Remarque 1.9 On dit aussi **biconnexe** à la place de 2-connexe, et **triconnexe** à la place de 3-connexe.

Définition 1.23 Soit $G = (V, E)$ un graphe connexe et soit $S \subseteq V$. On dit que S est un **séparateur** de G si $G[V \setminus S]$ n'est pas connexe.

Il est facile de décomposer un graphe en graphes 1-connexes : il suffit de couper le graphe en morceaux qui sont exactement ses composantes connexes. De même, il n'est pas très difficile de décomposer un graphe 1-connexe en graphes 2-connexes.

Définition 1.24 Soit G un graphe connexe. Une composante biconnexe de G est un sous-ensemble S de sommets maximal tel que $G[S]$ est biconnexe.

Théorème 1.10 [Wes00] Deux composantes biconnexes distinctes ont au plus un sommet en commun.

Un sommet appartenant à au moins deux composantes connexes distinctes est appelé **point d'articulation**.

Théorème 1.11 [Wes00] Soit G un graphe connexe. Soit G_b le graphe biparti d'incidence entre les points d'articulation de G et les composantes biconnexes de G . G_b est un arbre.

Idée de la preuve. Comme G est connexe, G_b est connexe également. Supposons que G_b possède un cycle simple C . Le graphe induit par l'union des composantes biconnexes présentes dans C est biconnexe, ce qui contredit la maximalité de chacune de ces composantes. $\square$

La décomposition en composantes biconnexes d'un graphe connexe G est donc un arbre dont les noeuds sont les points d'articulation de G et ses composantes biconnexes. A chaque noeud correspondant à une composante biconnexe est associé le graphe induit par cette composante.

La prochaine étape est la décomposition d'un graphe biconnexe en composantes triconnexes. Cette décomposition a été introduite par Tutte en 1966 [Tut66], depuis revisitée par [HT73, BM88, BT96] et généralisée au cas des graphes infinis par [Ric04]. [JW95] en donne une présentation succincte fidèle à celle de Tutte.

Dans ce mémoire, j'en donne une nouvelle présentation. Le but recherché ici est de présenter cette décomposition avec une vision la plus proche possible de celle adoptée pour la décomposition en coupes et la décomposition modulaire. La nature de la décomposition en composantes triconnexes est très différente des deux autres citées, mais la démarche de construction adoptée pour les deux premières apporte un éclairage que j'ai jugé intéressant pour la troisième. Il ne sera pas prouvé que cette démarche aboutit à la même décomposition que celle de Tutte.

Lors d'une étape de décomposition d'un graphe en composantes triconnexes, certains des graphes obtenus dans la décomposition ont des arêtes multiples. Pour pouvoir appliquer la décomposition récursivement, il convient donc de décrire une étape de décomposition sur un multigraphe et non pas sur un graphe simple. Dans le reste de cette section, les graphes considérés peuvent avoir des arêtes multiples, si ce n'est pas le cas, on précise qu'ils sont simples.

Nous aurons besoin de la notation qui suit pour décrire la décomposition de Tutte.

Notation 1.1 *Pour un graphe $G = (V, E)$ et deux sommets $x, y \in V$, on note G^{xy} le graphe obtenu à partir de G en ajoutant une arête fictive xy si $xy \notin E$ et en remplaçant l'arête xy de G par une arête fictive xy dans le cas contraire.*

Pour obtenir à partir d'un graphe biconnexe G des graphes triconnexes, l'idée suivie par la décomposition de Tutte est de retirer les **paires séparantes** de G (séparateurs de taille 2). Pour une paire séparante $\{x, y\}$ de G , on procède comme suit. On note $S_1, \dots, S_k$, avec k un entier supérieur à 2, les composantes connexes de $G - \{x, y\}$.

Définition 1.25 [Tut66] *La décomposition simple de G suivant la paire séparante $\{x, y\}$ est un ensemble de graphes. Pour définir cet ensemble, on distingue deux cas suivant le nombre k de composantes connexes de $G - \{x, y\}$.*

- $k = 2$ et xy n'est pas une arête de G . Dans ce cas, la décomposition de G suivant la paire séparante xy est l'ensemble $\{G^{xy}[S_1 \cup \{x, y\}], G^{xy}[S_2 \cup \{x, y\}]\}$. Les deux arêtes fictives xy ajoutées dans les deux graphes de la décomposition sont nouvelles et doivent être considérées comme identiques.

- $k = 2$ et xy est une arête de G . Dans ce cas, les graphes de la décomposition sont les graphes $G^{xy}[S_1 \cup \{x, y\}]$, $G^{xy}[S_2 \cup \{x, y\}]$ et un graphe G_3 à deux sommets x et y reliés par 3 arêtes multiples. Les arêtes fictives xy des graphes $G^{xy}[S_1 \cup \{x, y\}]$ et $G^{xy}[S_2 \cup \{x, y\}]$ sont nouvelles et doivent être considérées comme différentes. G_3 est formé d'une copie de chacune d'entre elles et de l'arête xy présente dans G .
- $k \geq 3$. Dans ce cas, les graphes de la décomposition de G suivant la paire séparante xy sont les graphes $G^{xy}[S_1 \cup \{x, y\}]$, $\dots$, $G^{xy}[S_k \cup \{x, y\}]$ et un graphe G_{k+1} à deux sommets x et y reliés par k arêtes multiples. Les arêtes fictives xy des graphes $G^{xy}[S_i \cup \{x, y\}]$ sont nouvelles et doivent être considérées comme différentes. G_{k+1} est formé d'une copie de chacune d'entre elles.

Grâce aux arêtes fictives, tous les graphes résultant de la décomposition de G selon une de ses paires séparantes $\{x, y\}$ sont biconnexes. De plus, $\{x, y\}$ n'est une paire séparante dans aucun des graphes obtenus. En recommençant l'opération récursivement sur chacun des graphes obtenus par une décomposition simple tant qu'il en existe un qui n'est pas triconnexe, on aboutit à une décomposition dont tous les graphes la composant sont triconnexes.

Définition 1.26 Une **décomposition par paires séparantes** d'un graphe G est un ensemble de graphes défini inductivement de la manière suivante :

- $D = \{G\}$ est une décomposition de G ; et
- si D_1 est une décomposition de G , si $G_1 \in D_1$ et si $\{G_2, \dots, G_k\}$ est une décomposition simple de G_1 , alors $D = (D_1 \setminus \{G_1\}) \cup \{G_2, \dots, G_k\}$ est une décomposition de G .

On dit qu'une décomposition D_1 **dérive** d'une autre D_2 si on peut obtenir D_1 à partir de D_2 par une suite de décompositions simples.

A une décomposition, on peut associer un arbre T de manière univoque : les sommets de T sont les graphes de la décomposition et deux graphes sont reliés par une arête dans T ssi ils partagent une arête fictive commune. Il n'est pas difficile de vérifier que le graphe ainsi obtenu est sans cycle et connexe.

L'opération réciproque de la décomposition selon une paire séparante est la 2-somme.

Définition 1.27 Soit $G = (V_G, E_G)$ et $H = (V_H, E_H)$ deux graphes contenant tous deux l'arête xy . La **2-somme** de G et H est le graphe dont l'ensemble de sommets est $V_G \cup V_H$ et l'ensemble d'arêtes est $(E_G \cup E_H) \setminus \{xy\}$.

En appliquant la 2-somme pour l'arête fictive correspondant à chaque arête de l'arbre de décomposition T , on reconstruit le graphe en entier.

Il n'y a en général pas unicité de la décomposition en paires séparantes, ni par les graphes obtenus ni par l'arbre associé à la décomposition. Par exemple, si $\{x, y\}$ et $\{u, v\}$ sont deux paires séparantes telles que u et v sont dans des composantes connexes différentes de $G - \{x, y\}$, alors commencer par décomposer G selon $\{x, y\}$ ou selon $\{u, v\}$ n'aboutira pas au même résultat final.

La solution que nous proposons ici pour retrouver le caractère unique de la décomposition est de ne décomposer que par des paires séparantes fortes. Les graphes que nous obtenons dans la décomposition canonique décrite ci-dessus sont soit des cycles, soit des multigraphes à deux sommets, soit des graphes triconnexes.

Définition 1.28 *On dit qu'une paire séparante $\{u, v\}$ **démonte** une paire séparante $\{x, y\}$ ssi $\{u, v\} \cap \{x, y\} = \emptyset$ et x, y sont dans des composantes connexes différentes de $G - \{u, v\}$.*

Lemme 1.10 *Soit $\{u, v\}$ et $\{x, y\}$ deux paires séparantes d'un graphe G . Si $\{u, v\}$ démonte $\{x, y\}$, alors $\{x, y\}$ démonte $\{u, v\}$.*

Preuve : On montre que si $\{u, v\}$ est une paire séparante qui sépare x de y et si u et v sont dans la même composante connexe de $G - \{x, y\}$, alors $\{x, y\}$ n'est pas une paire séparante, ce qui par contraposée démontre le lemme. On note S_x et S_y les composantes connexes de $G - \{u, v\}$ contenant respectivement x et y . Soit $\alpha \in x, y$. Comme $G_\alpha = G[S_\alpha \cup \{u, v\}] + uv$ est biconnexe, $\forall a, b \in (S_\alpha \cup \{u, v\}) \setminus \{\alpha\}$, il existe un chemin C_{ab} de a à b dans G_α . Ainsi, si il y a un chemin C_{uv} de u à v dans $G - \{x, y\}$, alors il y a un chemin de a à b dans $G - \{x, y\}$: il suffit de prendre C_{ab} et de remplacer, le cas échéant, l'arête uv de ce chemin par C_{uv} . Donc, si u et v ne sont pas séparés par $\{x, y\}$ alors les sommets de $(S_\alpha \cup \{u, v\}) \setminus \{\alpha\}$ sont dans la même composante connexe de $G - \{x, y\}$. Comme ceci est vrai quelque soit $\alpha \in \{x, y\}$ et comme $((S_x \cup \{u, v\}) \setminus \{x\}) \cup ((S_y \cup \{u, v\}) \setminus \{y\}) = V(G) \setminus \{x, y\}$, alors il n'y a qu'une composante connexe dans $G - \{x, y\}$. Et par conséquent, $\{x, y\}$ n'est pas une paire séparante de G . ■

Définition 1.29 *Une **paire séparante forte** $\{x, y\}$ est une paire séparante qui n'est démontée par aucune autre.*

Lemme 1.11 *Soit $\{x, y\}$ une paire séparante de G et soit G_1 un graphe issu de la décomposition simple de G selon $\{x, y\}$. Soient $\{u, v\} \subseteq V(G_1)$ tel que $\{u, v\} \neq \{x, y\}$. $\{u, v\}$ est une paire séparante de G_1 ssi $\{u, v\}$ est une paire séparante de G . De plus, $\{u, v\}$ est forte dans G_1 ssi $\{u, v\}$ est forte dans G .*

Preuve : Soit $S_1 \subseteq V$ tel que $G_1 = G^{xy}[S_1 \cup \{x, y\}]$. On a donc $\{u, v\} \subseteq S_1 \cup \{x, y\}$. Comme $\{u, v\} \neq \{x, y\}$, on peut supposer sans perte de généralité que $x \notin \{u, v\}$. Comme $G[V \setminus (S_1 \cup \{y\})]$ est connexe, alors $V \setminus (S_1 \cup \{y\})$ est inclus dans une composante connexe de $G[V \setminus \{u, v\}]$ qui contient aussi y si $y \notin \{u, v\}$. Ainsi, si $\{u, v\}$ est séparante dans G , il existe une composante connexe de $G[V \setminus \{u, v\}]$ qui est incluse dans S_1 : $\{u, v\}$ est une paire séparante de G_1 .

Réciproquement, si $\{u, v\}$ est une paire séparante de G_1 , comme x et y sont adjacents dans G_1 , ils sont dans la même composante connexe de $G_1 - \{u, v\}$. Et les composantes connexes de $G - \{u, v\}$ sont les mêmes que celles de $G_1 - \{u, v\}$ à ceci près que la composante connexe contenant x et y est augmentée des sommets de $V \setminus (S_1 \cup \{x, y\})$. $\{u, v\}$ est donc une paire séparante de G . De plus, remarquons que deux sommets $w, z \in S_1 \cup \{x, y\}$

sont dans des composantes connexes différentes de $G_1 - \{u, v\}$ ssi w et z sont dans des composantes connexes différentes de $G - \{u, v\}$.

Soit $\{u, v\}$ une paire séparante de G_1 . Sans perte de généralité, on peut supposer que $y \notin \{u, v\}$.

Soit $\{s, t\}$ une paire séparante de G_1 différente de $\{u, v\}$. D'après ce qui précède, $\{s, t\}$ est une paire séparante de G et si $\{s, t\}$ démonte $\{u, v\}$ dans G_1 alors $\{s, t\}$ démonte $\{u, v\}$ dans G . Donc, si $\{u, v\}$ n'est pas forte dans G_1 , alors $\{u, v\}$ n'est pas forte dans G .

Réciproquement, s'il existe $\{s, t\} \subseteq V$ qui est une paire séparante de G qui démonte $\{u, v\}$, alors par définition, $\{s, t\} \cap \{u, v\} = \emptyset$. Comme $\{u, v\} \subseteq S_1 \cup \{x\}$ et comme $G[S_1 \cup \{x\}]$ et $G[S_1]$ sont connexes, il s'ensuit que nécessairement $\{s, t\} \cap S_1 \neq \emptyset$. Sans perte de généralité, on peut supposer que $s \in \{s, t\} \cap S_1$. Si $t \notin S_1 \cup \{x, y\}$, alors $\{s, y\}$ est aussi une paire séparante de G qui démonte $\{u, v\}$. D'après ce qui précède, $\{s, t\}$ démonte $\{u, v\}$ également dans G_1 . ■

On en tire le corollaire immédiat suivant.

Corollaire 1.3 *Soit D une décomposition par paires séparantes de G . Si $\{x, y\}$ est une paire séparante forte d'un graphe de D , alors $\{x, y\}$ est une paire séparante forte de G .*

Lemme 1.12 *Soit G un graphe et soient $\{x, y\}$ et $\{u, v\}$ deux paires séparantes fortes de G distinctes. La décomposition obtenue en décomposant d'abord selon la paire $\{x, y\}$ puis selon $\{u, v\}$ est la même que celle obtenue en décomposant d'abord selon $\{u, v\}$ puis selon $\{x, y\}$.*

Preuve : Il convient tout d'abord de remarquer que comme $\{x, y\}$ ne démonte pas $\{u, v\}$, il existe un graphe G_1 de la décomposition simple de G selon $\{x, y\}$ qui contient la paire $\{u, v\}$. De plus, comme $\{x, y\} \neq \{u, v\}$, un tel graphe est unique. D'après le lemme 1.11, $\{u, v\}$ est une paire séparante de G_1 . Ainsi, les deux décompositions du lemme sont bien définies et de manière univoque. Il reste à montrer qu'elles sont identiques.

On note $A_1, \dots, A_k$ les composantes connexes de $G - \{x, y\}$ et $B_1, \dots, B_l$ les composantes connexes de $G - \{u, v\}$. On ne traite que le cas de décomposition où $k \geq 3$ et $l \geq 3$ et xy et uv ne sont pas des arêtes de G . Les autres cas sont très similaires mais demanderaient autant de travail pour les traiter dans le détail de leurs particularités. Comme déjà remarqué, le fait que $\{u, v\}$ soit forte et distincte de $\{x, y\}$ implique $\exists! i \in \llbracket 1, k \rrbracket, \{u, v\} \subseteq A_i \cup \{x, y\}$. Sans perte de généralité, on peut supposer que $\{u, v\} \in A_1$ et $\{x, y\} \in B_1$.

La décomposition D_1 de G selon $\{x, y\}$ est $D_1 = \{G^{xy}[A_1 \cup \{x, y\}], \dots, G^{xy}[A_k \cup \{x, y\}]\}$. On pose $G_1 = G^{xy}[A_1 \cup \{x, y\}]$. Comme remarqué dans la preuve du lemme 1.11, les composantes connexes de $G_1 - \{u, v\}$ sont les mêmes que celles de $G - \{u, v\}$ à ceci près que la composante connexe de $G - \{u, v\}$ contenant x et y se voit retirer les sommets de $V \setminus (A_1 \cup \{x, y\})$. Cette composante connexe est donc $B_1 \setminus (V \setminus (A_1 \cup \{x, y\})) = (B_1 \cap A_1) \cup \{x, y\}$. La décomposition de G_1 selon $\{u, v\}$ est $\{G^{xy, uv}[(A_1 \cap B_1) \cup \{x, y, u, v\}], G^{uv}[B_2 \cup \{u, v\}], \dots, G^{uv}[B_l \cup \{u, v\}]\}$. Ainsi, en décomposant selon $\{x, y\}$ puis

selon $\{u, v\}$ on obtient la décomposition $\{G^{xy}[A_2 \cup \{x, y\}], \dots, G^{xy}[A_k \cup \{x, y\}], G^{xy,uv}[(A_1 \cap B_1) \cup \{x, y, u, v\}], G^{uv}[B_2 \cup \{u, v\}], \dots, G^{uv}[B_l \cup \{u, v\}]\}$. Comme $\{x, y\}$ et $\{u, v\}$ jouent des rôles parfaitement symétriques, en décomposant d'abord selon $\{u, v\}$ puis selon $\{x, y\}$, on obtient la décomposition $\{G^{uv}[B_2 \cup \{u, v\}], \dots, G^{uv}[B_l \cup \{u, v\}], G^{xy,uv}[(B_1 \cap A_1) \cup \{u, v, x, y\}], G^{xy}[A_2 \cup \{x, y\}], \dots, G^{xy}[A_k \cup \{x, y\}]\}$. Les deux décompositions sont strictement identiques. ■

Définition 1.30 *Le lemme 1.12 permet de définir la **décomposition de G selon l'ensemble de ses paires séparantes fortes** comme étant une décomposition par paire séparante de G qui utilise toutes, et seulement, les paires séparantes fortes de G .*

La notion de k -anneau sera au centre de la démonstration du lemme qui suit.

Définition 1.31 *On dit qu'un graphe simple $G = (V, E)$ est un **k -anneau**, avec k un entier supérieur ou égal à 3, s'il existe un sous-ensemble $\{x_1, \dots, x_k\}$ de sommets de G et une k -partition au sens large (c.à.d. avec éventuellement des ensembles vides) de $V \setminus \{x_1, \dots, x_k\}$, notée $A_1, \dots, A_k$, tels que, en prenant les indices modulo k , on a les quatre propriétés suivantes :*

1. $\forall i \in \llbracket 1, k \rrbracket, G[A_i \cup \{x_i, x_{i+1}\}]$ est connexe ; et
2. $\forall i, j \in \llbracket 1, k \rrbracket, i \neq j \Rightarrow$ les sommets de A_i sont tous non adjacents à ceux de A_j ; et
3. $\forall i, j \in \llbracket 1, k \rrbracket, i \neq j + 1$ et $i \neq j - 1 \Rightarrow x_i$ n'est pas adjacent à x_j .
4. $\forall i, j \in \llbracket 1, k \rrbracket, i \neq j$ et $i \neq j - 1 \Rightarrow x_i$ n'est adjacent à aucun sommet de A_j .

Lemme 1.13 *Si G est un graphe simple biconnexe qui admet une paire séparante mais pas de paire séparante forte, alors G est un cycle à au moins quatre sommets.*

Preuve : Le but de la première partie de la preuve est de montrer que si G vérifie les hypothèses du lemme, alors G est un 4-anneau.

Soit x, y une paire séparante de $G = (V, E)$ et soit $\{S_1, S_2\}$ une bipartition de $V \setminus \{x, y\}$ telle qu'aucun sommet de S_1 n'est relié à un de S_2 . Comme G est connexe et comme S_1 et S_2 sont des unions de composantes connexes de $G - \{x, y\}$, alors $G[S_1 \cup \{x, y\}]$ et $G[S_2 \cup \{x, y\}]$ sont connexes.

Par hypothèse, la paire séparante $\{x, y\}$ n'est pas forte. Il existe donc une paire séparante $\{u, v\}$ qui démonte $\{x, y\}$, ce qui implique au passage que x n'est pas adjacent à y . Comme $G[S_1 \cup \{x, y\}]$ et $G[S_2 \cup \{x, y\}]$ sont connexes, alors il existe $\alpha_1 \in \{u, v\} \cap S_1$ et $\alpha_2 \in \{u, v\} \cap S_2$. Sans perte de généralité, on peut supposer que $u \in S_1$ et $v \in S_2$. Comme $\{u, v\}$ démonte $\{x, y\}$, nécessairement, u est un sommet séparant de $G[S_1 \cup \{x, y\}]$, et v un sommet séparant de $G[S_2 \cup \{x, y\}]$. Soit $\{S'_{11}, S'_{12}\}$ une partition de $(S_1 \cup \{x, y\}) \setminus \{u\}$ telle qu'aucun sommet de S'_{11} n'est adjacent à un sommet de S'_{12} et $x \in S'_{11}$ et $y \in S'_{12}$; et soit $\{S'_{21}, S'_{22}\}$ une partition de $(S_2 \cup \{x, y\}) \setminus \{v\}$ telle qu'aucun sommet de S'_{21} n'est adjacent à un sommet de S'_{22} et $x \in S'_{21}$ et $y \in S'_{22}$. Posons $S_{11} = S'_{11} \setminus \{x\}$, $S_{12} = S'_{12} \setminus \{y\}$,

$S_{21} = S'_{21} \setminus \{x\}$ et $S_{22} = S'_{22} \setminus \{y\}$. Comme $G[S_1 \cup \{x, y\}]$ est connexe, $G[S_{11} \cup \{u, x\}]$ et $G[S_{12} \cup \{u, y\}]$ le sont également. De même, $G[S_{21} \cup \{v, x\}]$ et $G[S_{22} \cup \{v, y\}]$ sont connexes.

En posant $x_1 = x$, $x_2 = u$, $x_3 = y$ et $x_4 = v$ ainsi que $A_1 = S_{11}$, $A_2 = S_{12}$, $A_3 = S_{22}$ et $A_4 = S_{21}$, on obtient que G est un 4-anneau.

Pour montrer que G est un cycle, il suffit de montrer que G est un n -anneau, où $n = |V|$. Considérons l'entier maximum k tel que G est un k -anneau et supposons que $k < n$. Soit $\{x_1, \dots, x_k\}$ et $A_1, \dots, A_k$ un sous-ensemble de sommet et une partition vérifiant pour G la propriété de k -anneau. Dans ce qui suit, les indices sont à prendre modulo k . Comme $k < n$, il existe $i \in \llbracket 1, k \rrbracket$ tel que $A_i \neq \emptyset$. $\{x_i, x_{i+1}\}$ est donc une paire séparante de G . Par hypothèse, il existe une paire $\{u, v\}$ qui démonte $\{x_i, x_{i+1}\}$. x_i et x_{i+1} ne sont donc pas adjacents. Comme $G[A_i \cup \{x_i, x_{i+1}\}]$ et $G[V \setminus A_i]$ sont connexes, nécessairement $\{u, v\} \cap A_i \neq \emptyset$ et $\{u, v\} \cap (V \setminus (A_i \cup \{x_i, x_{i+1}\})) \neq \emptyset$. Sans perte de généralité, on peut supposer que $u \in A_i$ et $v \in V \setminus (A_i \cup \{x_i, x_{i+1}\})$. u est donc un sommet séparant de $G[A_i \cup \{x_i, x_{i+1}\}]$ et on peut partitionner au sens large $A_i \setminus \{u\}$ en deux ensembles B_1 et B_2 tels que les sommets de B_1 ne sont adjacents à aucun sommet de B_2 et $B_1 \cup \{x_i, u\}$ est connexe et $B_2 \cup \{x_{i+1}, u\}$ est connexe. Le sous-ensemble de sommets $\{x_1, \dots, x_i, u, x_{i+1}, \dots, x_k\}$ et la partition $A_1, \dots, A_{i-1}, B_1, B_2, A_{i+1}, \dots, A_k$ montrent que G est un $k+1$ -anneau : absurde. On en conclut que $k = n$. Et G est un cycle. ■

Théorème 1.12 *Soit G un graphe simple biconnexe. La décomposition de G selon l'ensemble de ses paires séparantes fortes produit uniquement trois sortes de graphes :*

- des graphes à deux sommets reliés par une arête multiple,
- des cycles simples,
- des graphes simples triconnexes.

Preuve : Soit H un graphe de la décomposition de G selon l'ensemble de ses paires séparantes fortes. D'après la définition de la décomposition selon une paire séparante, H est soit un graphe à deux sommets reliés par une arête multiple, soit un graphe simple, qui est le cas à considérer dans la suite de la preuve. D'après le corollaire 1.3, H n'admet aucune paire séparante forte. Si H n'admet aucune paire séparante, H est triconnexe. Si H admet une paire séparante, d'après le lemme 1.13, H est un cycle. ■

Tous les graphes du théorème 1.12 peuvent s'obtenir dans une décomposition selon l'ensemble des paires séparantes fortes d'un graphe. On peut obtenir n'importe quel graphe triconnexe et n'importe quel cycle de longueur supérieure ou égale à 3, car on peut partir d'un tel graphe et dans ce cas la décomposition est triviale. On peut aussi obtenir n'importe quelle multiplicité p d'arête en partant du graphe $K_{2,p}$.

On donne sans le démontrer l'énoncé suivant qui pose l'identité de la décomposition décrite ici avec celle définie par Tutte.

Proposition non démontrée

La décomposition d'un graphe simple biconnexe G selon l'ensemble de ses paires séparantes

fortes est exactement la décomposition de G en graphes triconnexes définie par Tutte dans [Tut66].

Dans [CE80], le théorème d'unicité suivant est énoncé.

Théorème 1.13 *Parmi les décompositions par paires séparantes d'un graphe G dont les graphes sont des multigraphes à deux sommets ou des cycles simples ou des graphes trois connexes, il en existe une unique dont dérive toutes les autres.*

Toujours sans démonstration, on donne la proposition suivante.

Proposition non démontrée

La décomposition d'un graphe simple biconnexe G selon l'ensemble de ses paires séparantes fortes est l'unique décomposition en paires séparantes caractérisée par [CE80].

1.2.4 Décomposition par séparateurs complets

La décomposition par séparateurs complets, qui ne s'applique qu'aux graphes connexes (ou les composantes connexes d'un graphe qui ne l'est pas), repose sur un principe très simple. Il s'agit de trouver une clique dont le retrait sépare le graphe, jusqu'à obtenir des **atomes** indécomposables. Le problème de l'unicité de cette décomposition n'est pas encore résolu. Dans sa version originale due à Tarjan [Tar85], l'arbre de décomposition est binaire et l'ensemble des atomes obtenue n'est pas unique. Depuis, Berry et Bordat [BB97] ont proposé un procédé de décomposition plus contraint qui donne un ensemble d'atomes unique, bien que l'arbre de décomposition ne le soit pas. L'arbre obtenu dans leur version n'est plus binaire. Nous présentons ici les deux variantes de cette décomposition, en commençant par la première citée. Dans tout le reste de la section, les graphes considérés sont connexes.

Une clique séparante d'un graphe connexe $G = (V, E)$ est un sous-ensemble de sommets $C \subseteq V$ qui forme une clique et tel que $G[V \setminus C]$ est non connexe. Lorsque un graphe G admet une clique séparante, on peut partitionner le graphe en trois sous-ensembles A , B et C tels qu'aucun sommet de A ne soit adjacent à un sommet de B . On décompose ainsi le graphe en deux parties : $A \cup C$ et $B \cup C$. On peut représenter cela par un arbre de décomposition dont la racine est la clique séparante utilisée C , qui a deux fils correspondant aux deux parties du graphe $A \cup C$ et $B \cup C$. On continue ainsi récursivement la décomposition sur chacune des deux parties en cherchant une clique séparante dans les graphes $G[A \cup C]$ et $G[B \cup C]$. Si on trouve une clique séparante C_1 , et une tripartition $\{A_1, B_1, C_1\}$, dans $G[A \cup C]$, elle devient fils de la racine à la place de $A \cup C$ et on lui assigne deux fils correspondant aux ensembles $A_1 \cup C_1$ et $B_1 \cup C_1$. On continue ainsi de suite la construction de l'arbre tant qu'on trouve des séparateurs complets. Les feuilles de l'arbre binaire de décomposition ainsi construit correspondent à des sous-ensembles de sommets définissant des graphes induits qui n'ont pas de séparateurs complets. Ce sont des graphes indécomposables, on les appelle les **atomes**. Il est évident que l'arbre de décomposition dépend des cliques choisies pour

décomposer le graphe et de l'ordre dans lesquels on les utilise. Selon ces choix, les atomes obtenus ne sont pas toujours les mêmes.

La décomposition par séparateurs complets ainsi définie ne fournit donc pas de théorème d'unicité. C'est ce à quoi essaye de palier l'approche de [BB97]. Les auteurs restreignent les cliques qu'on peut utiliser pour séparer le graphe. De cette manière, ils ne peuvent garantir l'unicité de l'arbre de décomposition mais obtiennent l'unicité de l'ensemble d'atomes obtenu. La contrainte supplémentaire demandée aux séparateurs complets est qu'elles soient aussi des **séparateurs minimaux** (défini ci-dessous). D'où le nom donné à cette restriction de la décomposition par séparateurs complets : la décomposition par séparateurs minimaux en clique. Précisons la notion de séparateur minimal d'un graphe $G = (V, E)$. Soit $S \subseteq V$. S est un a, b -**séparateur** de G , avec $a, b \in V$ deux sommets non-adjacents, ssi a et b sont dans différentes composantes connexes de $G[V \setminus S]$. S est un a, b -**séparateur minimal** de G ssi il n'existe pas de a, b -séparateur S' strictement inclus dans S . S est un **séparateur minimal** de G ssi il existe un couple de sommet $a, b \in V$ non-adjacents tel que S est un a, b -séparateur minimal de G . Une composante connexe C de $G[V \setminus S]$ est dite S -saturante ssi tout sommet de S a un voisin dans C . S est un séparateur minimal de G ssi $G[V \setminus S]$ a au moins deux composantes connexes S -saturantes [Gol04]. S est un a, b -séparateur minimal de G ssi a et b sont dans des composantes connexes S -saturantes de $G[V \setminus S]$ différentes.

La décomposition définie par [BB97] cherche un séparateur minimal du graphe qui est une clique. Elle a le même pouvoir de décomposition que celle définie par [Tar85]. C'est à dire que les graphes décomposables par l'une ou l'autre des deux décompositions sont les mêmes. En effet, il coule de source que si un graphe a un séparateur minimal en clique, celui-ci est en particulier une clique séparante du graphe. Réciproquement, si il existe une clique séparante dans un graphe G , alors elle contient un séparateur minimal de G , qui est nécessairement une clique. L'arbre de décomposition obtenu par [BB97] n'est pas binaire. Ceci est dû au fait que lorsque un séparateur minimal en clique C est trouvé dans G , un noeud est créé dans l'arbre de décomposition pour chaque composante connexe de $G[V \setminus C]$. Ces noeuds sont fils d'un noeud étiqueté par le séparateur C . Soit A une composante connexe de $G[V \setminus C]$, le noeud correspondant n'est pas nécessairement étiqueté par le sous-ensemble de sommets $A \cup C$. En effet, afin d'obtenir l'unicité de l'ensemble d'atomes issu du procédé de décomposition, Berry et Bordat [BB97] assurent que tous les noeuds de l'arbre sont étiquetés par l'union d'un séparateur minimal C et d'une de ses composantes C -saturantes. Ainsi, si A n'est pas C -saturante, $N(A) \cap C$ est aussi un séparateur minimal et A est $N(A) \cap C$ -saturante. Le noeud de l'arbre correspondant à la composante A est donc étiqueté par $A \cup (N(A) \cap C)$ à la place de $A \cup C$. De cette manière, même s'il est possible de décomposer un graphe donné de différentes manières (et ainsi d'obtenir différents arbres de décomposition), l'ensemble des atomes obtenus est le même dans toutes les décompositions du graphe. Ce qui amène la question suivante :

Question ouverte 1.1 *Est-il possible de définir une restriction de la décomposition par séparateurs complets qui débouche sur un arbre de décomposition unique ?*

Si la réponse devait être positive, l'arbre en question serait sans doute non-enraciné,

comme dans la décomposition en coupe. En effet, il paraît difficile, lors du choix de la première clique séparante, de départager certains séparateurs complets n'ayant pas de rapport entre elles. De la même manière que dans la décomposition en coupe, il est possible de commencer par plusieurs coupes, ce qui n'empêche pas que toute décomposition complète du graphe utilise les mêmes coupes.

Nous avons vu que les graphes décomposables par la décomposition par séparateurs complets et par la décomposition par séparateurs minimaux en cliques sont les mêmes. Il en est donc de même pour les graphes entièrement décomposables définis ainsi.

Définition 1.32 *Un graphe G est dit entièrement décomposable par séparateurs complets ssi tout sous graphe induit de G qui n'est pas une clique admet une clique séparante.*

La classe des graphes entièrement décomposables par séparateurs complets était déjà bien connue avant l'avènement de cette décomposition, il s'agit des graphes **triangulés** (ou chordaux). Les graphes triangulés sont les graphes ne possédant pas de cycles induits de longueur supérieure ou égale à quatre. Autrement dit, ce sont les graphes dans lesquels tout cycle de longueur au moins quatre admet une **corde** (une arête n'appartenant pas au cycle mais reliant deux de ses sommets). Les graphes triangulés peuvent être caractérisés de la manière suivante.

Théorème 1.14 *Un graphe G est triangulé ssi tout séparateur minimal de G est une clique.*

Il s'ensuit qu'un graphe triangulé est entièrement décomposable par séparateurs complets. Réciproquement, si un graphe G est entièrement décomposable par séparateurs complets, les atomes obtenus sont des cliques. Or, une propriété de la décomposition par clique séparante est qu'elle ne sépare pas les cycles induits. C'est à dire que lors d'une étape de décomposition, un cycle induit se retrouve entièrement dans une composante. Ainsi, les cycles induits se retrouvent dans les atomes de la décomposition. Par conséquent, lorsque les atomes sont des cliques, cela signifie que le graphe de départ ne comporte pas de cycle induit de longueur supérieure ou égale à quatre.

1.2.5 Conclusion

Nous venons de présenter quatre décompositions de graphes : la décomposition modulaire, la décomposition en coupes, la décomposition en composantes triconnexes, aussi appelée décomposition de Tutte, et la décomposition par séparateurs complets.

Cette dernière est celle pour laquelle il reste à faire le plus de travail. En effet, malgré les travaux de Berry et Bordat [BB97] qui définissent une méthode de décomposition qui débouche sur l'unicité des atomes produits, il n'existe pas de vraie décomposition canonique pour la décomposition en séparateurs complets.

En revanche, les trois premières admettent des décompositions canoniques qui ont été introduites historiquement de trois manières différentes respectivement dans [Gal67], [Cun82]

et [Tut66]. Je me suis attaché à présenter ces trois décompositions dans un cadre commun, en suivant l'approche proposée par Gallai dans [Gal67]. Selon moi, cette approche prend tout son sens lorsqu'elle est rapprochée de la vision de Cunningham.

Pour Gallai, la notion de décomposition canonique importe peu car il ne définit qu'une décomposition : celle utilisant les modules forts maximaux, qui sont uniques, pour une étape de décomposition. L'unicité de la décomposition est alors acquise par définition. On est même en droit de se poser la question de savoir s'il existe plus d'une décomposition modulaire. La réponse est oui : il n'existe pas que des modules forts, et on peut décomposer un graphe selon certains de ses modules qui ne sont pas forts, en utilisant l'opération de quotient (voir figure 1.6 page 36). L'approche de Gallai ne considère pas ces décompositions. On peut néanmoins espérer pouvoir les retrouver d'une manière ou d'une autre à partir de la décomposition canonique car tous les modules sont lisibles sur l'arbre de décomposition.

A l'opposé, l'approche de Cunningham consiste à d'abord définir ce qu'est une décomposition en coupes quelconque de manière récursive, par l'opération de dérivation d'une décomposition. C'est seulement après qu'est établie l'existence d'une décomposition particulière que l'on dira canonique. La définition de cette décomposition canonique me paraît particulièrement intéressante car elle revêt une idée directrice commune à beaucoup de décompositions. Je dirais même qu'elle répond d'une certaine manière aux questions : pourquoi décomposer un graphe ? jusqu'où décomposer un graphe ?

Pour introduire la définition de la décomposition canonique, Cunningham se concentre sur une famille seulement de décompositions : celles qui ont la propriété de ne contenir que des graphes que l'on peut décomposer de toutes les façons envisageables, les graphes friables ("brittle"), et des graphes que l'on ne peut décomposer d'aucune manière, les graphes premiers. Ces décompositions sont intéressantes car elles sont accomplies : décomposer un graphe friable est vain et décomposer un graphe premier est impossible. Le résultat d'unicité s'énonce ainsi : parmi toutes ces décompositions, il en existe une unique dont dérive toutes les autres.

Essayons de synthétiser les deux approches. Le but d'une décomposition est de décomposer le graphe jusqu'à obtenir des graphes indécomposables. Ce qui menace l'unicité d'une décomposition c'est de faire des choix lors de ce procédé, c'est à dire de décomposer d'une manière qui en interdit d'autres. C'est exactement la définition des éléments forts, que ce soit pour les modules ou les coupes, ce sont les éléments selon lesquels on peut décomposer sans gêner les autres décompositions possibles. Ainsi, lorsqu'on décompose selon tous les éléments forts on obtient une décomposition qui ne contient que des graphes friables et premiers et à partir de laquelle on peut obtenir toutes les décompositions qui ont cette propriété.

Ce que montre l'approche qui a été mise en oeuvre ici pour présenter la décomposition modulaire et la décomposition en coupes est que les deux visions de Gallai et de Cunningham s'applique aussi bien à ces deux décompositions, et apportent des éléments de compréhension complémentaires. Cunningham et Edmonds [CE80] ont mis en oeuvre la deuxième approche pour la décomposition en composantes tri-connexes. Nous avons montré qu'on peut aussi attaquer cette décomposition sous l'angle de Gallai en définissant les paires séparantes fortes, notion absente de [Tut66] et [CE80], bien qu'elle soit implicitement uti-

lisée dans [Tut66] via une définition plus complexe.

Une question qui reste en suspend est celle de savoir quelles autres décompositions peuvent entrer dans ce schéma.

1.3 Modèles d'intersection

Les classes de graphes définies par modèles d'intersection ont concentré beaucoup de travaux. Le livre de McKee et McMorris [MM99] constitue un excellent ouvrage de synthèse sur le sujet.

Ces classes de graphes passionnent les théoriciens par l'esthétisme de leurs définitions et des multiples caractérisations qu'elles admettent souvent. Sans doute sont elles fascinante par le mystère de la correspondance qu'elles permettent d'observer entre les notions géométriques qui les définissent et leurs caractérisations en termes de théorie des graphes. Par exemple, les graphes d'intervalles sont définis comme les graphes d'intersection d'intervalles de la droite réelle. Mais on peut en donner des définitions indépendantes de toute notion géométrique : ce sont les graphes de co-comparabilité sans C_4 [GH64] ; ce sont aussi les graphes triangulés sans triplet astéroïdal [LB62].

D'un point de vue épistémologique, ces classes de graphes, comme toutes les classes restreintes, constituent souvent les premières briques posées vers des résultats sur des classes de graphes plus générales. Il est souvent efficace de s'appuyer sur leurs propriétés géométriques fortes pour faire des preuves que l'on peut ensuite abstraire et généraliser lorsque les propriétés de théorie des graphes qui les sous-tendent ont été dégagées.

Enfin, les classes de graphes définies par modèles d'intersection ont un grand intérêt pratique. Depuis que Cook introduisit la NP-complétude [Coo71], beaucoup d'efforts ont été tournés vers résoudre des problèmes NP-complets sur des classes de graphes restreintes. Cela s'explique par le fait que les données sur lesquelles on veut résoudre ces problèmes n'ont pas toujours la complexité des graphes en général. Ces données proviennent de contextes bien précis qui leur confèrent certaines propriétés structurelles, qui sont parfois celles des classes de graphes auxquelles est dédiée cette section. De surcroît les classes d'intersection géométriques interviennent naturellement dans la modélisation de certains problèmes pratiques : les graphes d'intervalles sont à la base de problèmes de reconstruction de génome et d'emploi du temps, les graphes de permutation jouent un rôle dans l'analyse phylogénétique.

Dans ce mémoire, la question soulevée est celle de la sensibilité algorithmique des modèles géométriques à de légères modifications. Plus précisément, on dispose d'un modèle géométrique d'un graphe, on effectue une modification sur le graphe, qui peut être l'ajout ou le retrait d'une arête ou d'un sommet, et on veut savoir si le nouveau graphe admet toujours un modèle géométrique du même type, et trouver un tel modèle si la première réponse est positive. Nous verrons en section 3.3, que cela nécessite de connaître non pas un mais tous les modèles géométriques du graphe de départ. Dans cette section, nous présentons simplement les définitions des classes de graphes auxquelles nous nous intéresserons dans ce cadre, ainsi que quelques unes de leurs propriétés de base. Dans la section suivante (1.4),

on présente les structures permettant de représenter tous les modèles géométriques de ces classes de graphes.

1.3.1 Graphes d'intervalles

Les graphes d'intervalles (introduits par [Haj57]) sont les graphes d'intersection d'intervalles de la droite des réels (voir [Gol04, BLS99] pour une présentation complète). Un modèle d'intervalles d'un graphe $G = (V, E)$ est un ensemble $\mathcal{I}$ d'intervalles de la droite réelle accompagné d'une bijection entre les sommets de G et les éléments de $\mathcal{I}$ telle que deux sommets de G sont adjacents ssi leurs intervalles correspondants s'intersectent. Un graphe est un graphe d'intervalles ssi il admet un modèle d'intervalles (voir figure 1.15). La classe des graphes d'intervalles reste inchangée si on exige que les intervalles des modèles soient fermés et aient des bornes entières ; c'est encore la même si on demande que les intervalles soient ouverts et aient des bornes entières. Dorénavant, on utilisera que des modèles dont les intervalles sont fermés et les bornes entières, sauf sur les dessins pour des raisons de clarté.

Les graphes d'intervalles admettent plusieurs caractérisations très esthétiques. [LB62] montre qu'ils sont exactement les graphes triangulés sans triplet astéroïdal (voir [Gol04] pour ces définitions). Une autre caractérisation est la suivante.

Théorème 1.15 [GH64] *Soit G un graphe. Les propositions suivantes sont équivalentes.*

1. G est un graphe d'intervalles.
2. G ne contient pas de C_4 (4-cycle sans corde) et son complémentaire $\overline{G}$ est un graphe de comparabilité.
3. Les cliques maximales de G peuvent être ordonnées linéairement de sorte que, pour tout sommet x de G , les cliques maximales contenant x apparaissent consécutivement.

Ordres consécutifs des graphes d'intervalles

Un ordre sur les cliques maximales satisfaisant la propriété 3 du théorème 1.15 est appelé un **ordre consécutif** («consecutive ordering» en anglais). Soit k le nombre de cliques maximales d'un graphe d'intervalles G . En numérotant les cliques maximales avec leur rang dans un ordre consécutif σ et en attribuant à chaque sommet x de G l'intervalle de σ des cliques contenant x on obtient un modèle de G (voir figure 1.15).

Les modèles ainsi obtenus sont ceux dont le nombre d'entiers deux à deux distincts utilisés pour les bornes des intervalles est minimum. Attardons nous un instant sur ce fait.

Définition 1.33 *Soit X un ensemble et $\mathcal{F}$ une famille de sous-ensembles de X . $\mathcal{F}$ satisfait la propriété de Helly si toute sous-famille finie $\mathcal{F}' \subseteq \mathcal{F}$ telle que les éléments de $\mathcal{F}'$ s'intersectent deux à deux est telle que l'intersection des éléments de $\mathcal{F}'$ est non vide.*

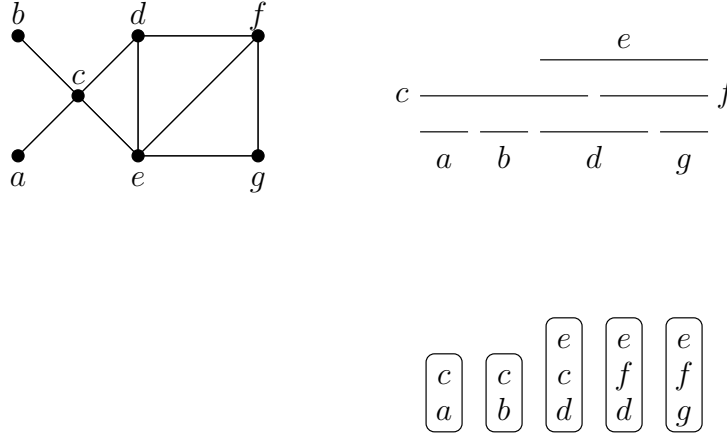


FIG. 1.15 – Un graphe d’intervalles accompagné d’un modèle d’intervalles et d’un arrangement consécutif de ses cliques maximales.

Il n’est pas très dur de vérifier que l’ensemble des intervalles de la droite réelle satisfait la propriété de Helly. Cela implique que dans un modèle d’intervalles fermés avec des bornes entières de G , pour toute clique K de G , il existe un entier p tel que tous les intervalles correspondant aux sommets de K contiennent l’entier p . En particulier, comme les cliques maximales de G sont incomparables pour la relation d’inclusion, les entiers p correspondant à ces cliques sont nécessairement deux à deux distincts. Cela montre que le nombre de cliques maximales est bien le minimum d’entiers deux à deux distincts nécessaires dans un modèle d’intervalles fermés.

En termes d’efficacité de représentation, en associant à chaque sommet les bornes de son intervalle dans un modèle quelconque, on obtient une structure de donnée qui répond à la requête d’adjacence en temps constant et qui ne demande qu’un espace $O(n)$.

Graphes d’intervalles propres et graphes d’intervalles unitaires

En contraignant les modèles que l’on peut utiliser pour les graphes d’intervalles on en définit deux sous classes strictes. Les graphes d’intervalles propres sont ceux qui admettent un modèle dans lequel aucune paire d’intervalle n’est strictement ordonnée par inclusion. Les graphes d’intervalles unitaires sont ceux qui admettent un modèle dans lequel tous les intervalles sont de longueur 1.

Théorème 1.16 [Rob69] *Les graphes d’intervalles propres et unitaires sont les mêmes et ce sont exactement les graphes d’intervalles sans $K_{1,3}$ (griffe).*

Une autre caractérisation remarquable des graphes d’intervalles propres est la suivante.

Théorème 1.17 [DHH96] *Un graphe $G = (V, E)$ est un graphe d’intervalles propres ssi le couple $(V, \{N[x] \mid x \in V\})$ a la propriété d’arrangement consécutif (voir section 1.4.1).*

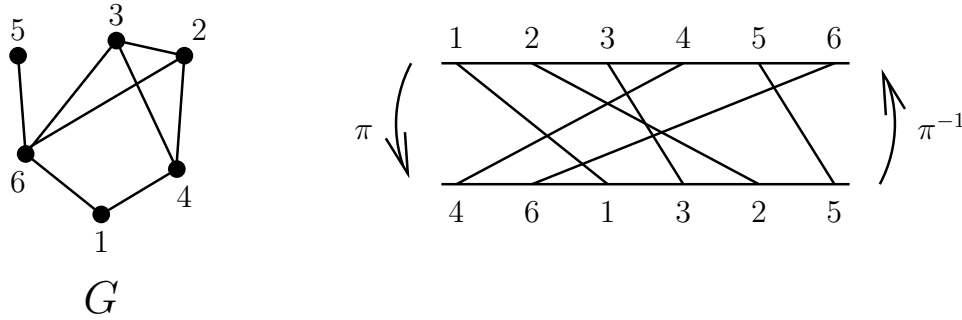


FIG. 1.16 – G est le graphe d'inversion de π^{-1} . Par exemple, $6 > 1$ mais $\pi^{-1}(6) = 2 < 3 = \pi^{-1}(1)$: 6 et 1 sont inversés par π^{-1} , donc les sommets 6 et 1 sont adjacents.

1.3.2 Graphes de permutation

Comme les graphes d'intervalles, les graphes de permutation ont plusieurs caractérisations possibles (voir [Gol04] et [BLS99] pour une introduction complète et des références). On peut les définir comme les graphes d'intersection de segments dont les extrémités sont situées sur deux lignes droites parallèles et distinctes du plan. On peut imposer que les extrémités de tous les segments soient distinctes, la classe définie reste alors la même.

Orientons les deux droites parallèles dans le même sens. On note π_1 (resp. π_2) l'ordre sur les sommets de G défini par l'ordre d'apparition de l'extrémité de leurs segments correspondants sur la première (resp. seconde) droite. Pour un ordre linéaire π sur les sommets d'un graphe G , $\pi(x)$ désigne le rang du sommet x dans π tandis que $\pi^{-1}(i)$ est le sommet occupant le rang i . En attribuant à chaque sommet x le couple $(\pi_1(x), \pi_2(x))$ on peut tester l'adjacence entre deux sommets par comparaison des couples qui leur sont associés. Comme pour les graphes d'intervalles, on obtient ainsi une structure de donnée de taille $O(n)$ qui donne l'adjacence en temps constant.

Une façon classique de représenter un modèle d'intersection pour un graphe de permutation à n sommets est d'attribuer un numéro entre 1 et n à chaque segment et de placer ces numéros sur les deux droites parallèles dans l'ordre où les segments apparaissent (voir figure 1.16). On peut toujours renuméroter les segments pour qu'ils apparaissent sur la première droite dans l'ordre des entiers qui leur sont attribués. Si on adopte cette convention, seule la donnée de leur ordre d'apparition sur la seconde droite est nécessaire. Cet ordre peut être vu comme une permutation des entiers de 1 à n que l'on note π . Les sommets i et j sont adjacents ssi $(j - i)(\pi^{-1}(j) - \pi^{-1}(i)) < 0$. Le graphe G représenté est alors le graphe d'inversion de la permutation π^{-1} , c'est à dire qu'il y a une arête entre i et j ssi i et j sont inversés par π^{-1} . Les graphes de permutation justifient leur nom par le fait que ceux à n sommets sont exactement les graphes d'inversions des permutations de $\llbracket 1, n \rrbracket$.

Un modèle d'intersection pour un graphe de permutation est appelé **diagramme de permutation** ou **réalisateur**. Nous les représenterons par les deux ordres π_1 et π_2 définis ci-dessus (voir figure 1.17). Pour un ordre linéaire σ , on désigne par $\bar{\sigma}$ l'ordre inverse. Remarquons que si $R = (\pi_1, \pi_2)$ est un réalisateur de G , alors (π_2, π_1) , $(\bar{\pi}_1, \bar{\pi}_2)$ et $(\bar{\pi}_2, \bar{\pi}_1)$ sont

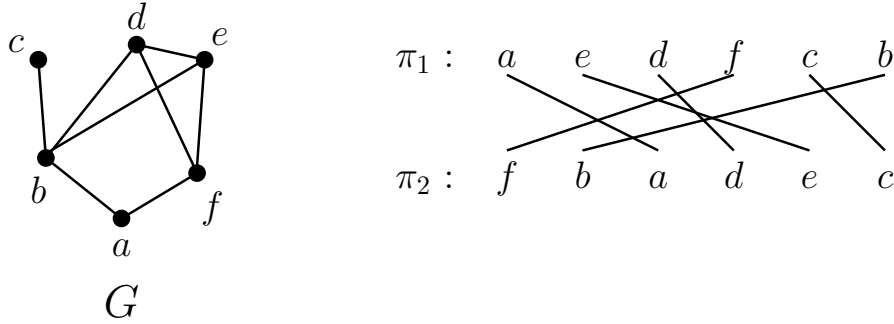


FIG. 1.17 – Diagramme de permutation.

aussi des réalisateurs de G . Dans la suite, ces quatre réalisateurs sont dits **géométriquement équivalents**.

Les graphes de permutation ont aussi une très belle caractérisation en termes d'orientation transitive, ce qui leur donne un rapport très particulier à la décomposition modulaire (voir section 1.4.3).

Un graphe est un graphe de comparabilité ssi ses arêtes peuvent être orientées transitivement [Gal67], on obtient alors un ordre partiel (voir [Tro92] pour de la littérature sur les ordres partiels et la théorie de la dimension).

Théorème 1.18 [EPL72] *Les graphes de permutation sont précisément les graphes de comparabilité dont le complémentaire est aussi de comparabilité.*

La famille des graphes de permutation est donc autocomplémentée. Si (π_1, π_2) est un réalisateur de G , alors $(\pi_1, \bar{\pi}_2)$ est un réalisateur de $\bar{G}$.

La dimension d'un ordre partiel (voir [Tro92]) est un invariant de comparabilité, c'est à dire que deux ordres partiels qui ont le même graphe non orienté sous-jacent (graphe de comparabilité) ont la même dimension.

Théorème 1.19 [EPL72] *Les graphes de permutation sont les graphes de comparabilité dont les orientations transitives sont de dimension deux.*

On passe très simplement d'un réalisateur d'un graphe de permutation à un réalisateur (au sens de la théorie de la dimension) d'une de ses orientations transitives et vice-versa, il suffit d'inverser un des deux ordres. Cette proximité entre les deux objets explique que le mot "réalisateur", issu de la théorie de la dimension, serve aussi à désigner un diagramme de permutation.

Il est amusant de remarquer que les graphes de permutation ont des liens avec les graphes d'intervalles. Ces derniers sont les graphes d'intersection d'intervalles de la droite des réels alors que les graphes de permutation sont les graphes d'inclusion d'intervalles, c'est à dire les graphes G qui admettent un modèle d'intervalles tel que deux sommets de G sont adjacents ssi leurs intervalles sont ordonnés par inclusion. On peut encore demander que les intervalles soient fermés et aient des bornes entières sans changer la classe définie.

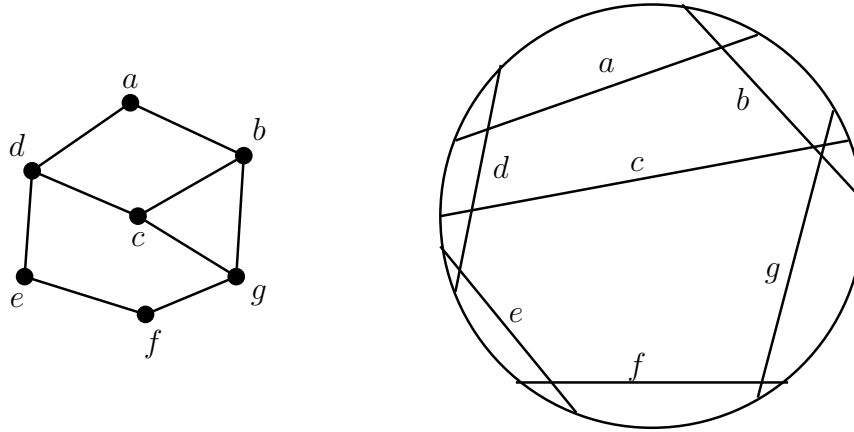


FIG. 1.18 – Un graphe et son modèle d'intersection de cordes d'un cercle.

Comment passer d'un modèle d'intervalles par inclusion à un diagramme de permutation et réciproquement ? Si on dispose d'un modèle d'intervalles, on commence par en construire un autre dans lequel toutes les bornes des intervalles sont distinctes. On obtient alors un réalisateur en prenant pour π_1 l'ordre d'apparition des bornes gauches, et pour π_2 l'ordre d'apparition des bornes droites. Réciproquement, si on dispose d'un réalisateur, on en déduit un modèle d'intervalles par inclusion en prenant $\pi_1(x)$ pour borne gauche de l'intervalle d'un sommet x et $\pi_2(x) + n$ pour sa borne droite.

1.3.3 Graphes de cordes

Les graphes de cordes sont les graphes d'intersection de cordes d'un cercle, figure 1.18. Si dans un diagramme de permutation on relie les extrémités des deux lignes parallèles pour en faire un cercle, on obtient un modèle d'intersection de cordes d'un cercle. Ainsi, les graphes de cordes forment une sur-famille de celle des graphes de permutation.

Comme ces derniers, ils peuvent aussi être représentés par un modèle d'intervalles en considérant que deux sommets sont adjacents si leurs intervalles se chevauchent, c'est à dire leur intersection est non vide et aucun des deux ne contient l'autre.

Les graphes d'intervalles sont les graphes d'intersections d'intervalles, les graphes de permutation sont les graphes d'inclusion d'intervalles et les graphes de cordes sont les graphes de chevauchement d'intervalles. En particulier, cela signifie que l'on peut partitionner les arêtes d'un graphe d'intervalles en deux de manière à ce que le graphe partiel restreint à chacun des deux ensembles d'arêtes soit respectivement un graphe de permutation et un graphe de cordes.

Les graphes de cordes ne sont présents dans ce mémoire que parce qu'ils illustrent de façon remarquable les relations entre décompositions de graphes et représentation de l'ensemble des modèles géométriques d'un graphe. Ce lien est détaillé en section 1.4.4. Cette approche est aussi celle utilisée dans [Cou07] pour construire en logique monadique du second ordre l'ensemble des modèles d'un graphe de cordes. Pour une introduction

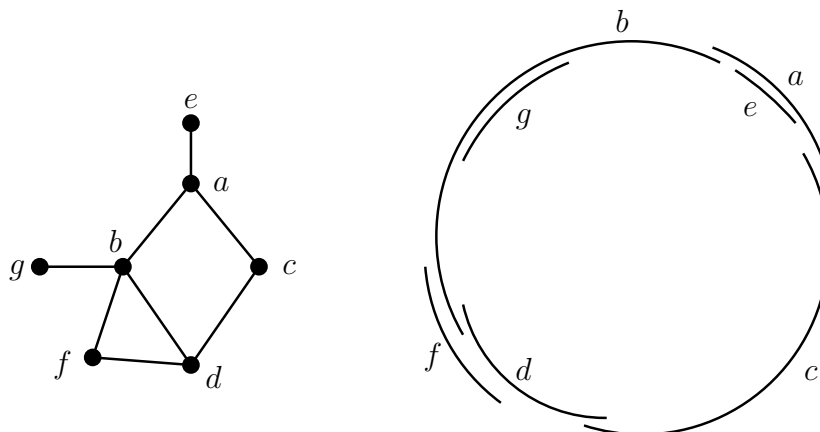


FIG. 1.19 – Un graphe et son modèle d’arcs de cercle.

plus poussée des graphes de cordes et pour d’autres références, le lecteur se reportera à [Gol04, BLS99].

1.3.4 Graphes d’arcs de cercle

Les graphes d’arcs de cercle sont une généralisation des graphes d’intervalles. Ce sont les graphes d’intersections d’arcs d’un cercle, figure 1.19. Il est immédiat de voir que si un graphe admet un modèle d’intervalles il admet aussi un modèle d’arcs de cercle. La réciproque est en général fausse, les graphes d’arcs de cercle ne sont pas tous des graphes d’intervalles. Ils ne sont même pas tenus d’être triangulés : les cycles de longueur arbitraire admettent des modèles d’arcs de cercle.

On pourrait s’attendre, par analogie avec les graphes d’intervalles, à ce que les graphes d’arcs de cercle soient ceux dont les cliques maximales ont la propriété d’arrangement circulaire consécutif (voir section 1.4.2). Malheureusement, le fait d’être sur un cercle plutôt que sur une droite enlève une propriété fondamentale : la propriété de Helly. Les arcs de cercle ne satisfont pas cette propriété et certains graphes d’arcs de cercles n’admettent aucun modèle dont les arcs satisfont la propriété de Helly. Ceux qui admettent au moins un tel modèle sont appelés **graphes d’arcs de cercle de Helly** et sont exactement les graphes dont les cliques maximales ont la propriété d’arrangement circulaire consécutif [Gav74a]. Ils sont un bel exemple de classe de graphes pour lesquels on a une représentation arborescente par degrés de liberté de l’ensemble des modèles géométriques (voir section 1.4.2).

Comme pour les graphes d’intervalles, on peut définir deux classes restreintes de graphes d’arcs de cercle. Les **graphes d’arcs de cercle propres**, qui sont ceux admettant un modèle dans lequel aucune paire d’arcs n’est strictement ordonnée par inclusion, et les **graphes d’arcs de cercle unitaires** qui sont ceux admettant un modèle dont les arcs sont de longueur 1. Contrairement au cas des graphes d’intervalles, la première classe contient strictement la seconde et est contenue dans la classe des graphes de cordes (il suffit pour le voir de remplacer les arcs par les cordes qui joignent leurs extrémités).

De même que les graphes $G = (V, E)$ tels que le couple $(V, \{N[x] \mid x \in V\})$ a la propriété d'arrangement consécutif sont une sous famille des graphes d'intervalles (les graphes d'intervalles propres, voir section 1.3.1), de même les graphes $G = (V, E)$ tels que le couple $(V, \{N[x] \mid x \in V\})$ a la propriété d'arrangement circulaire consécutif sont une sous famille des graphes d'arcs de cercle. C'est ce que montre le théorème suivant dû à Tucker.

Théorème 1.20 [Tuc71] *Si un graphe $G = (V, E)$ est tel que le couple $(V, \{N[x] \mid x \in V\})$ a la propriété d'arrangement circulaire consécutif, alors G est un graphe d'arcs de cercle.*

Il n'existe pas à ma connaissance de caractérisation géométrique de cette classe de graphes.

1.3.5 Graphes triangulés

Les graphes triangulés, ou graphes cordaux ("chordal graphs" en anglais), sont les graphes sans cycles induits de longueur supérieure ou égale à 4. Autrement dit, tout cycle de longueur au moins 4 possède une corde. Nous avons déjà vu des exemples de graphes triangulés avec les graphes d'intervalles qui en sont une sous classe.

Comme la classe des graphes triangulés généralise celle des graphes d'intervalles, on peut se poser la question de savoir si les graphes triangulés admettent des modèles d'intersections, qui seraient plus généraux que ceux des graphes d'intervalles. La réponse est positive. Les graphes d'intervalles sont les graphes d'intersection de sous-chemins d'un chemin et les graphes triangulés sont les graphes d'intersections de sous-arbres d'un arbre. On a le théorème suivant.

Théorème 1.21 [Wal72, Gav74b, Bun74] *Soit $G = (V, E)$ un graphe. Les trois propositions suivantes sont équivalentes :*

1. G est un graphe triangulé.
2. G est le graphe d'intersection d'une famille de sous-arbres d'un arbre.
3. Il existe un arbre T dont l'ensemble des sommets $\mathcal{K}$ est l'ensemble des cliques maximales de G et tel que pour tout $v \in V$, le sous-graphe de T induit par les cliques maximales de G contenant v est connexe, c'est à dire un sous-arbre de T .

Un arbre T vérifiant la condition 3 du théorème 1.21 est appelé un **arbre de cliques** de G .

Lemme 1.14 *Les arbres de cliques sont les modèles d'intersections minimaux des graphes triangulés.*

Preuve : Soit G un graphe triangulé et un modèle d'intersection de G en sous-arbres d'un arbre T . Pour un noeud u de T , on note $V(u)$ l'ensemble des sommets de G dont le sous-arbre contient u . Les sous-arbres d'un arbre ont la propriété de Helly sur les sommets. C'est à dire qu'une famille de sous-arbre qui s'intersectent deux à deux sur les sommets

est telle qu'il existe un sommet commun à tous les arbres de la famille. Cela implique que pour toute clique maximale K de G , il existe un noeud u de T tel que $V(u) = K$. De plus, si pour deux noeuds distincts u et v de T on a $V(u) \subseteq V(v)$, alors, nécessairement, tous les sommets de $V(u)$ sont présents dans tous les $V(p)$ où p est un noeud de T sur le chemin entre u et v . En particulier, en notant q le voisin de u se trouvant sur le chemin de u à v , on a $V(u) \subseteq V(q)$. Par conséquent, tous les sommets u tels que $V(u)$ n'est pas une clique maximale de G ont leur ensemble associé $V(u)$ inclus dans celui d'un de leurs voisins. Or, lorsque u et v sont deux noeuds voisins dans T tels que $V(u) \subseteq V(v)$, en supprimant u de T et en attribuant ses voisins, autres que v , à v , on obtient encore un modèle d'intersection de G . En répétant cette opération pour toutes les cliques non maximales, on obtient un arbre de cliques de G . Les arbres de cliques d'un graphe triangulé sont donc ses modèles d'intersections minimaux en sous-arbres d'un arbre. ■

Comment représenter l'ensemble de ces modèles minimaux ? Une représentation communément utilisée est le graphe des cliques de G . Les sommets de ce graphes sont les cliques maximales de G et les arêtes relient deux cliques dont l'intersection est non vide, elles sont valuées par le cardinal de cette intersection.

Théorème 1.22 [BG81] *Les arbres de cliques d'un graphe triangulé sont exactement les arbres couvrants de poids minimum de son graphe de cliques.*

Les graphes triangulés sont donc définis par un modèle d'intersection et l'ensemble de leur modèle d'intersection admet une représentation compacte. Cependant, cette représentation, l'arbre des cliques, n'est pas une représentation arborescente à degrés de liberté (voir section 1.4) comme pour les autres classes de graphes présentées dans ce chapitre. Remarquons que ce n'est pas le seul point sur lequel cette classe diffère des autres présentées ici : le modèle d'intersection qui la définit n'est pas géométrique, il appartient plutôt au domaine de la théorie des graphes.

1.3.6 Graphes planaires

A l'instar des graphes triangulés, les graphes planaires détonnent un peu dans ce mémoire, de par leur définition qui n'est pas géométrique mais topologique. Pourtant, ils ont des propriétés de représentation très proches des autres classes étudiées ici, comme nous le verrons en section 1.4. Il se trouve même que l'histoire de l'entretien dynamique des graphes planaires illustre parfaitement la problématique et la démarche de ce mémoire (voir section 3.3). Ils ne seront utilisés dans ce mémoire qu'à titre d'exemple, pour de la littérature sur les graphes planaires, le lecteur peut se reporter à [NC88, MT01]. C'est pourquoi on ne donne ci dessous que quelques définitions basiques qui nous seront utiles en section 1.4.5 pour étudier une représentation de l'ensemble des modèles "géométriques"² d'un graphe planaire.

²Nous qualifierons ces modèles de géométriques même si leur nature est plutôt topologique, pour garder une unité de vocabulaire avec les autres classes de graphes

Il est bon de mentionner que le fait que les graphes planaires aient des comportements similaires à ceux des classes de graphes définis de manière géométrique n'est pas si surprenant que cela vu les caractérisations et propriétés géométriques que possèdent ces graphes. La magnifique caractérisation suivante est particulièrement édifiante à ce sujet.

Théorème 1.23 (Koebe, Andreev, Thurston) *Les graphes planaires sont les graphes d'intersections de disques fermés du plan dont les intérieurs sont disjoints.*

Les quelques définitions qui suivent sont très informelles mais suffisent au besoin de ce manuscrit.

Définition 1.34 *Un graphe planaire est un graphe que l'on peut dessiner dans le plan sans croisement d'arêtes.*

De manière équivalente, les graphes planaires sont ceux que l'on peut dessiner sur la sphère sans croisement d'arêtes.

Définition 1.35 *Un dessin d'un graphe planaire sur la sphère découpe celle-ci en régions connexes. L'ordre cyclique alterné des sommets et des arêtes sur le contour d'une telle région est appelé une **face**.*

Définition 1.36 *Deux dessins d'un graphe planaire sur la sphère sont dits équivalents s'ils ont les mêmes faces. Un **plongement** d'un graphe planaire est une classe d'équivalence de dessins.*

1.4 Représentations arborescentes par degrés de liberté

Dans cette section, on présente les structures classiquement utilisées pour représenter tous les modèles géométriques d'un graphe appartenant à une des classes introduites dans la section précédente. Ces structures joueront un rôle majeur dans la reconnaissance dynamique de ces classes de graphes, voir section 3.3. Elles sont utilisées pour élaborer des algorithmes de reconnaissance dynamique pour les graphes de permutation (section 4.5 et les graphes d'intervalles(chapitre 5)).

1.4.1 PQ -arbres et graphes d'intervalles

Les PQ -arbres ont été introduits par Booth et Lueker [BL76] pour reconnaître les graphes d'intervalles. La difficulté d'analyse de leur algorithme a construit injustement une mauvaise réputation aux PQ -arbres. C'est pourtant un objet qui répond à un problème très simple, celui de l'arrangement consécutif, et qui ce faisant met en évidence de remarquables propriétés des ordres linéaires relativement à la notion de contiguïté. J'espère que le chapitre 5 convaincra le lecteur de l'intérêt de cette structure.

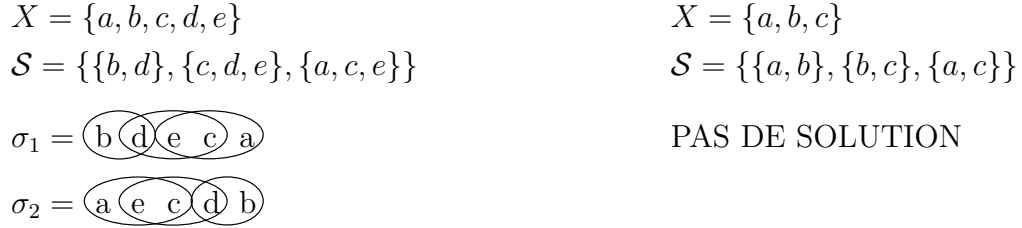


FIG. 1.20 – Un problème d'arrangement consécutif qui admet plusieurs solutions et un autre qui n'en admet aucune.

Etant donné un ensemble fini X et une collection $\mathcal{S}$ de sous-ensembles de X , le problème de l'arrangement consécutif est le suivant : existe-t-il un ordre linéaire σ sur les éléments de X tel que les éléments de chaque sous-ensemble $S \in \mathcal{S}$ apparaissent consécutivement dans σ (voir figure 1.20) ? Autrement dit, on voudrait que les éléments de $\mathcal{S}$ soient des intervalles de σ . Le problème d'arrangement consécutif A sur un ensemble X avec la collection $\mathcal{S}$ de sous-ensembles de X sera noté³ $A = \mathcal{A}_1(X, \mathcal{S})$. On appelle X l'ensemble de référence et $\mathcal{S}$ est l'ensemble des **contraintes de consécuité**.

Le problème de l'arrangement consécutif est non-trivial au sens où sa réponse peut être positive ou négative. Lorsqu'elle est positive, il existe en général plusieurs ordres linéaires σ satisfaisant l'ensemble des contraintes de consécuité imposées par $\mathcal{S}$. L'ensemble de ces ordres, qui sont dits **cohérents**, est noté $\Pi(A)$, ou $\Pi(\mathcal{S})$ lorsqu'il n'y a pas d'ambiguïté sur l'ensemble X auquel on se réfère. Le cardinal de $\Pi(\mathcal{S})$ peut atteindre $k!$, où $k = |X|$. Pourtant, dans tous les cas, $\Pi(\mathcal{S})$ peut être représenté par une structure de taille $O(k)$, appelée PQ -arbre.

Définition 1.37 *Un PQ -arbre T est un arbre enraciné comportant des noeuds internes de deux types distincts : les noeuds P et les noeuds Q . Un noeud P a au moins deux fils et un noeud Q au moins trois. Les feuilles du PQ -arbre sont les éléments de X . Il n'y a pas d'ordre défini sur les fils d'un noeud P , alors que les fils d'un noeud Q sont linéairement ordonnés à renversement près.*

Autrement dit, à chaque noeud Q de T sont associés deux ordres linéaires sur ses fils, inverses l'un de l'autre. Un noeud P n'a pas d'ordre associé car les $k!$ ordres linéaires possibles sur l'ensemble de ses k fils peuvent être utilisés. Sur les figures, un noeud Q sera représenté par un rectangle et les ordres de dessin de ses fils de gauche à droite et de droite à gauche seront précisément les deux ordres associés au noeud Q en question. Les noeuds P seront représentés par des cercles. Un exemple est donné sur la figure 1.21.

Définition 1.38 *On appelle **solidification** d'un PQ -arbre T , l'arbre T dans lequel on a associé à chaque noeud p de T un ordre linéaire sur ses fils : n'importe quel ordre linéaire*

³Dans cette notation le 1 en indice dans $\mathcal{A}_1$ signifie qu'il s'agit de la première formulation du problème d'arrangement consécutif. Nous donnerons plus loin une formulation duale de celle-ci que nous noterons $\mathcal{A}_2$

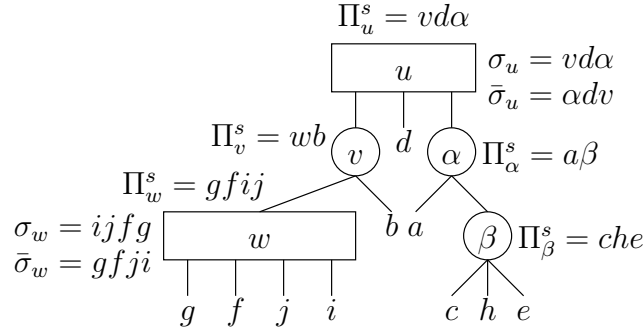


FIG. 1.21 – Une solidification s d'un PQ arbre T et sa frontière $\sigma = gfjibdache$. Par exemple, $\sigma' = dgfbijahec \notin \text{consistant}(T)$.

si p est un noeud P , un des deux ordres linéaires inverses l'un de l'autre si p est un noeud Q .

La **frontière d'une solidification** est l'ordre dans lequel on rencontre les feuilles de T dans un parcours en profondeur de T qui pour tout noeud p de T découvre les fils de p dans l'ordre défini par la solidification. Un ordre linéaire sur l'ensemble des feuilles X d'un PQ -arbre T est dit **consistant** avec T s'il est la frontière d'une solidification de T .

Notation 1.2 On note $\text{Sol}(T)$ l'ensemble des solidifications d'un PQ arbre T et pour $s \in \text{Sol}(T)$, on note $\text{front}(s)$ la frontière de s . On note $\text{consistant}(T) = \{\text{front}(s) \mid s \in \text{Sol}(T)\}$.

Remarque 1.10[BL76] L'application front est une bijection des solidifications d'un PQ arbre T dans $\text{consistant}(T)$.

De manière plus imagée, une solidification d'un PQ arbre T décide d'un ordre de gauche à droite pour dessiner les fils de chaque noeud de T : cet ordre est libre pour les noeuds P , et pour les noeuds Q cet ordre est l'un des deux (inverses l'un de l'autre) associés au noeud en question. La frontière d'une solidification est alors l'ordre de dessin des feuilles de l'arbre de gauche à droite (voir figure 1.21).

Dans la littérature, les noeuds P d'un PQ arbre sont aussi appelés les **noeuds dégénérés**, et les noeuds Q les **noeuds premiers**. Nous utiliserons beaucoup cette terminologie par la suite, d'une part parce qu'elle laisse entrevoir des liens avec la décomposition modulaire et d'autre part parce qu'elle amène moins de lourdeur et d'ambiguïté dans les notations.

Notation 1.3 Pour un noeud premier q d'un PQ arbre, on note σ_q et $\bar{\sigma}_q$ les deux ordres inverses l'un de l'autre associés à q .

Pour un noeud $p \in T$, une solidification s de T et sa frontière σ , on note Π_p^σ ou Π_p^s l'ordre sur les fils de p défini par s . Cela n'entraînera pas d'ambiguïté puisque les solidifications de T et leurs frontières sont en bijection.

Théorème 1.24 [BL76] *Pour toute collection $\mathcal{S}$ de sous-ensembles de X qui vérifie la propriété de l'arrangement consécutif, il existe un unique PQ -arbre T , ayant pour feuilles les éléments de X , tel que $\Pi(\mathcal{S}) = \text{consistant}(T)$. Et réciproquement, pour tout PQ -arbre T dont l'ensemble des feuilles est X , il existe une collection $\mathcal{S}$ de sous-ensembles de X telle que $\Pi(\mathcal{S}) = \text{consistant}(T)$.*

Pouvoir de représentation des PQ -arbres

Comme un noeud interne a au moins deux fils, le nombre total de noeuds dans un PQ -arbre est $O(k)$, k étant son nombre de feuilles. Le PQ -arbre représente donc avec un espace en $O(k)$ un ensemble d'ordres linéaires dont la cardinalité peut atteindre $k!$. Le PQ -arbre n'est pas une représentation en extension des ordres de $\Pi(\mathcal{S})$ mais la donnée de l'arbre permet de retrouver tous les ordres de $\Pi(\mathcal{S})$ et d'en produire la liste explicite.

L'efficacité du PQ -arbre vient du fait qu'il représente un ensemble d'ordres linéaires en décrivant les degrés de liberté sur lesquels on peut jouer pour former les ordres linéaires. Ces degrés de libertés sont de deux types, P et Q , et leur nombre est linéaire en $|X|$. Quand au gouffre entre l'espace de stockage et le nombre de configurations représentées, c'est un faux miracle : pour représenter les $k!$ permutations possibles d'un ensemble à k éléments, il suffit de lister ces éléments, ce qui prends $O(k)$ en espace, et de les faire précéder du symbole P que l'on interprétera comme "tous les arrangements possibles sur...". C'est de ce simple fait que provient l'efficacité en espace des PQ -arbres.

Ce qui est beaucoup plus remarquable, à mon sens, dans la découverte des PQ -arbres, c'est le fait que l'espace des solutions du problème d'arrangement consécutif admette une représentation arborescente par degrés de liberté.

Formulation duale des problèmes d'arrangements consécutifs

Les problèmes d'arrangement consécutif admettent une deuxième formulation. Soit X' un ensemble fini et $\mathcal{S}'$ une collection de sous-ensembles de X' . On se pose la question : existe-t-il un ordre linéaire σ sur les éléments de $\mathcal{S}'$ tel que tout $x' \in X'$ apparaisse dans des éléments de $\mathcal{S}'$ consécutifs ? Cela constitue une autre formulation du problème d'arrangement consécutif que l'on note $\mathcal{A}_2(\mathcal{S}', X')$. Dans cette version, les contraintes de consécutivité sont les éléments de l'ensemble de référence et non pas des sous-ensembles comme c'est le cas dans la première formulation. En fait, les rôles des éléments et des sous-ensembles sont inversés. C'est pour cette raison que les deux formulations sont dites duales.

Ces deux formulations sont équivalentes au sens où tout problème d'arrangement consécutif peut être indifféremment exprimé dans l'une ou l'autre des formulations sans en affecter les solutions, qui se correspondent via le changement de formalisme. La seule difficulté pour passer d'un formalisme à l'autre en conservant l'équivalence est que dans la première formulation, il ne peut pas y avoir deux contraintes identiques, cela est possible dans la seconde formulation. En contre partie, dans la seconde formulation, il ne peut pas y avoir deux éléments de l'ordre σ qui sont soumis aux mêmes contraintes, cela est possible

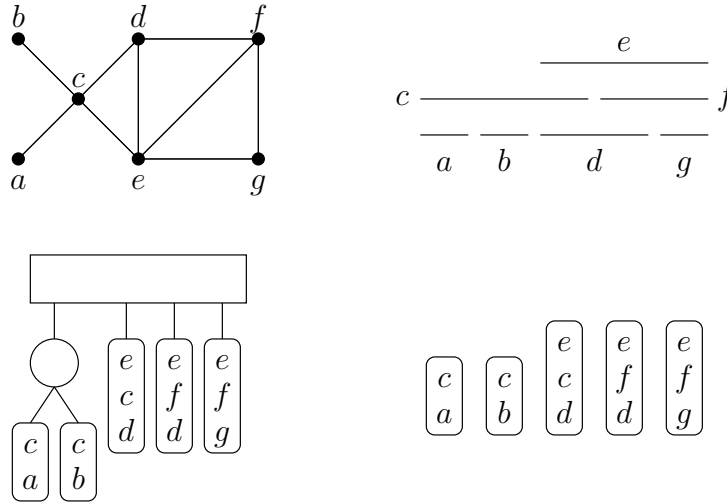


FIG. 1.22 – Un graphe d'intervalle accompagné d'un modèle d'intervalles, d'un arrangement consécutif de ses cliques maximales et de son PQ -arbre.

dans la première.

A titre d'exemple, on montre comment passer de la première à la deuxième formulation, sans gérer le problème des éléments soumis aux mêmes contraintes. Soit $A = \mathcal{A}_1(X, \mathcal{S})$ un problème d'arrangement consécutif. Pour tout élément $x \in X$ on pose $E(x) = \{S \in \mathcal{S} \mid x \in S\}$. Posons $X' = \mathcal{S}$ et $\mathcal{S}' = \{E(x) \mid x \in X\}$. Notons que $E(x)$ est un sous-ensemble de $\mathcal{S}$ et que par conséquent $\mathcal{S}'$ est un ensemble de sous-ensembles de X' . Ainsi, A peut se reformuler dans le second formalisme par $A = \mathcal{A}_2(\mathcal{S}', X')$.

L'emploi de la seconde formulation sera plus naturel pour les graphes d'intervalles, où $\mathcal{S}'$ sera l'ensemble des cliques maximales du graphe et X' l'ensemble de ses sommets.

PQ -arbre d'un graphe d'intervalles

D'après la condition 3 du théorème 1.15, un graphe est un graphe d'intervalles ssi il existe un ordre σ sur ses cliques maximales tel que pour chaque sommet, les cliques qui le contiennent forment un intervalle de σ . Ainsi, si pour un graphe G on note X l'ensemble de ses cliques maximales et si on pose $\mathcal{S} = \{ \{K \in X \mid y \in K\} \mid y \in V(G) \}$, alors G est un graphe d'intervalles si et seulement si le doublet $(X, \mathcal{S})$ a la propriété de l'arrangement consécutif (selon la première formulation du problème, $\mathcal{A}_1(X, \mathcal{S})$).

Nous avons vu en section 1.3.1 que les modèles d'intervalles minimaux d'un graphe d'intervalles G sont ceux basés sur un ordre consécutif des cliques maximales de G . Le PQ -arbre fournit une représentation de l'ensemble des ordres consécutifs des cliques maximales d'un graphe d'intervalles, et donc de l'ensemble des modèles minimaux de G .

Ce PQ -arbre, qu'on appelle le PQ -arbre de G et qu'on note $T^c(G)$, est un arbre enraciné dont les feuilles sont les cliques maximales de G . Ses noeuds internes sont étiquetés P (**noeuds dégénérés**) ou Q (**noeuds premiers**). Tout noeud $q \in T^c(G)$ qui est premier est doté de deux ordres linéaires, inverses l'un de l'autre, sur l'ensemble de ses fils, notés

σ_q et $\bar{\sigma}_q$.

Comme le nombre de cliques d'un graphe d'intervalles G est $O(n)$, le PQ -arbre d'un graphe d'intervalles est de taille $O(n)$. Par contre, si on représente les cliques maximales de G en extension, en stockant pour chaque clique la liste des sommets qu'elle contient, on occupe un espace total pouvant atteindre $O(n + m)$. Cela freine les possibilités algorithmiques de cette représentation. [KM85] ont pallié au problème en stockant les sommets du graphe dans les noeuds internes de l'arbre plutôt que dans ses feuilles, cette structure s'appelle le MPQ -arbre ou la PQ -représentation (voir section 5.2).

1.4.2 PC -arbres et graphes d'arcs de cercle de Helly

Les PC -arbres sont une généralisation des PQ -arbres s'intéressant à la consécutivité dans les ordres circulaires. On définit le problème d'**arrangement consécutif circulaire** de manière analogue à celui de l'arrangement consécutif. La donnée est un ensemble fini X et une collection $\mathcal{S}$ de sous-ensembles de X et la question posée est : existe-t-il un ordre circulaire σ sur les éléments de X tel que les éléments de chaque sous-ensemble $S \in \mathcal{S}$ apparaissent consécutivement dans σ ?

Ce problème est une relaxation de celui de l'arrangement consécutif. En effet, si un ordre linéaire σ répond au problème de l'arrangement consécutif pour le couple $(X, \mathcal{S})$, alors l'ordre circulaire σ' obtenu en bouclant σ répond au problème de l'arrangement consécutif linéaire pour $(X, \mathcal{S})$.

Lorsque la réponse au problème d'arrangement consécutif circulaire est positive, il n'y a en général pas unicité de l'ordre circulaire satisfaisant. Comme dans la version linéaire du problème, l'ensemble des ordres circulaires satisfaisant les contraintes de consécutivité, noté $\Pi(X, \mathcal{S})$, peut être décrit par un arbre à degré de liberté. Cet arbre est appelé **PC -arbre**.

Historiquement, les PQ -arbres ont été introduits par [BL76] pour reconnaître les graphes d'intervalles en temps linéaire. Le même article donnait aussi une application des PQ -arbres à la reconnaissance des graphes planaires. Les PC -arbres ont été introduits [SH99] dans le but de donner un algorithme de reconnaissance des graphes planaires plus simple. Des rapports simples et directs entre les PQ -arbres et les PC -arbres ont été explicités dans [Hsu01, HM03]. Ces deux articles établissent également le rôle que joue les PC -arbres pour le problème d'arrangement consécutif circulaire, qui est présenté ici.

Un PC -arbre T est un arbre non enraciné comportant des noeuds internes de deux types distincts : les noeuds P et les noeuds C . Un noeud P a au moins trois voisins et un noeud C au moins quatre. Les feuilles du PC -arbre sont les éléments de X . Il n'y a pas d'ordre défini sur les voisins d'un noeud P , alors que les voisins d'un noeud C sont circulairement ordonnés à renversement près. Autrement dit, à chaque noeud C de T sont associés deux ordres circulaires sur ses fils, inverses l'un de l'autre.

Pour énoncer ce résultat, on reprend le même vocabulaire que dans la version linéaire.

Définition 1.39 *On appelle **solidification** d'un PC -arbre T , l'arbre T dans lequel on a associé à chaque noeud u de T un ordre circulaire sur ses voisins : n'importe quel ordre circulaire si u est un noeud P , un des deux ordres circulaires inverses l'un de l'autre qui*

sont associés à u si u est un noeud C .

La **frontière d'une solidification** est l'ordre circulaire dans lequel on rencontre les feuilles de T dans un parcours en profondeur de T qui pour tout noeud u de T découvre les fils de u dans l'ordre circulaire défini par la solidification. Un ordre circulaire sur l'ensemble des feuilles X d'un PC -arbre T est dit **consistant** avec T s'il est la frontière d'une solidification de T .

Notation 1.4 On note $Sol(T)$ l'ensemble des solidifications d'un PC arbre T et pour $s \in Sol(T)$, on note $front(s)$ la frontière de s . On note $consistant(T) = \{front(s) \mid s \in Sol(T)\}$.

Remarque 1.11 L'application $front$ est une bijection des solidifications d'un PC -arbre T dans $consistant(T)$.

Une interprétation géométrique de ces notions est donnée par [HM03]. Elle consiste à plonger le PC -arbre dans un disque en répartissant ses feuilles sur le cercle qui en est la limite.

Théorème 1.25 [HM03] Pour toute collection $\mathcal{S}$ de sous-ensembles de X ayant la propriété d'arrangement circulaire, il existe un unique PC -arbre T , ayant pour feuilles les éléments de X , tel que $\Pi(X, \mathcal{S}) = consistent(T)$. Et réciproquement, pour tout PC -arbre T dont l'ensemble des feuilles est X , il existe une collection $\mathcal{S}$ de sous-ensembles de X telle que $\Pi(X, \mathcal{S}) = consistent(T)$.

PC -arbres et graphes d'arcs de cercle de Helly

Comme nous l'avons signalé en section 1.3.4, les graphes d'arcs de cercle ne sont pas, contrairement à ce qu'on pourrait attendre, les graphes dont les cliques maximales ont la propriété de l'arrangement consécutif circulaire. Cela vient du fait que les arcs d'un cercle ne satisfont pas la propriété de Helly (voir définition 1.33). Par contre, il existe des modèles d'arcs de cercle dont l'ensemble d'arcs qu'ils utilisent satisfait la propriété de Helly. Les graphes qui admettent un tel modèle sont appelés graphes d'arcs de cercle de Helly. Ils forment une sous-classe stricte des graphes d'arcs de cercle.

Pour les graphes d'arcs de cercle de Helly, tout se passe comme pour les graphes d'intervalles, à condition de ne s'intéresser qu'à leurs modèles de Helly. Les modèles de Helly minimaux sont ceux basés sur un arrangement consécutif circulaire de leurs cliques maximales. Comme nous l'avons vu dans cette section, les PC -arbres permettent de représenter l'ensemble des arrangements consécutifs circulaires, ils fournissent donc une représentation de l'ensemble des modèles minimaux d'un graphe d'arcs de cercle de Helly.

Ce résultat n'est pas entièrement satisfaisant car les modèles minimaux ne vérifiant pas la propriété de Helly sont exclus de la représentation. Par conséquent, il n'est donc d'aucune utilité pour les graphes d'arcs de cercle qui ne sont pas de Helly. Ce qui soulève la question suivante.

Question ouverte 1.2 Existe-t-il une représentation arborescente à degrés de liberté de l'ensemble des modèles minimaux d'un graphe d'arcs de cercle ?

1.4.3 Décomposition modulaire et graphes de permutation

Il est bien connu que l'arbre de décomposition modulaire d'un graphe de permutation permet d'en représenter tous les réalisateurs. Cette représentation entre dans le cadre des représentations arborescentes à degrés de liberté de la même façon que les PQ -arbres, les PC -arbres et la décomposition en composantes triconnexes d'un graphe planaire biconnexe. La lecture de ce qui suit suppose la connaissance préalable des graphes de permutation, présentés section 1.3.2. Nous reprenons les notations introduites dans la section précitée.

Les graphes de permutation ont un très bon comportement vis à vis des deux opérations fondamentales de la décomposition modulaire : le quotient et la substitution d'un graphe à un sommet.

Lemme 1.15 *La famille des graphes de permutation est héréditaire et fermée par substitution d'un graphe à un sommet.*

Idée de la preuve. Il est aisé de voir que la restriction d'un réalisateur R d'un graphe de permutation G à un sous-ensemble $S \subseteq V$ de sommets, notée $R[S]$, constitue un réalisateur du sous-graphe induit $G[S]$. Comme les graphes de permutation sont les graphes de comparabilité et de co-comparabilité, le fait que la famille soit fermée par substitution d'un graphe à un sommet découle du résultat de [Gal67] sur les graphes de comparabilité. $\square$

De la double propriété de fermeture de la famille exprimée par le lemme 1.15, on déduit directement le résultat suivant.

Théorème 1.26 [Gal67, Möh85] *Un graphe est un graphe de permutation ssi les graphes quotients des noeuds premiers de son arbre de décomposition modulaire sont eux-mêmes des graphes de permutation.*

On dit que la famille des graphes de permutation est **proprement décomposée** par la décomposition modulaire.

Définition 1.40 *Si G et H sont des graphes de permutation qui ont pour réalisateurs respectifs $R_G = (\pi_1, \pi_2)$ et $R_H = (\tau_1, \tau_2)$, alors le graphe de permutation $G_{x \rightarrow H}$ admet pour réalisateur R qui est obtenu à partir de R_G en remplaçant x par τ_1 dans π_1 et par τ_2 dans π_2 . Cette opération est appelée la **composition de réalisateurs**.*

En choisissant un réalisateur pour le graphe quotient de chaque noeud de l'arbre de décomposition modulaire et en appliquant l'opération de composition des réalisateurs (voir définition 1.40) tout au long de l'arbre, on obtient un réalisateur de G (voir figure 1.23).

Les quotients qui sont des cliques ou des stables à k sommets admettent $k!$ réalisateurs distincts ($k!/4$ à équivalence géométrique près. Par contre, il est connu que les graphes premiers n'en admettent qu'un.

Théorème 1.27 [Möh85] *Un graphe de permutation premier a un réalisateur unique à équivalence géométrique près.*

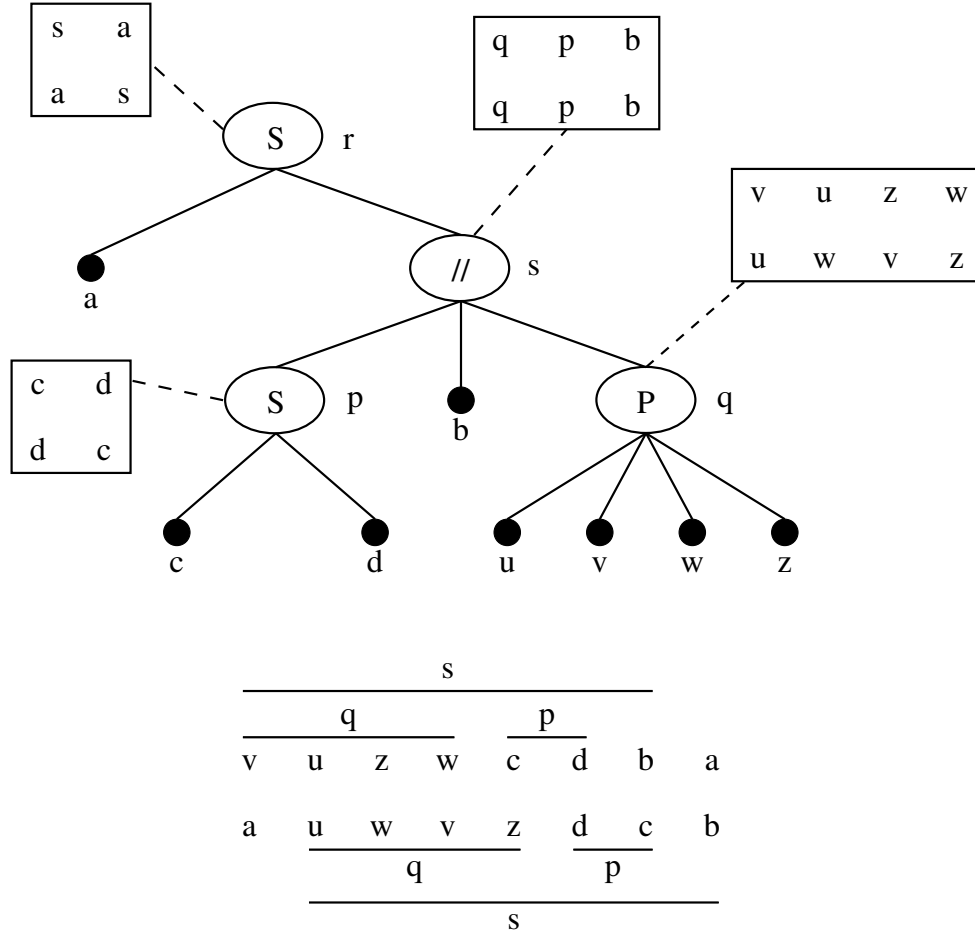
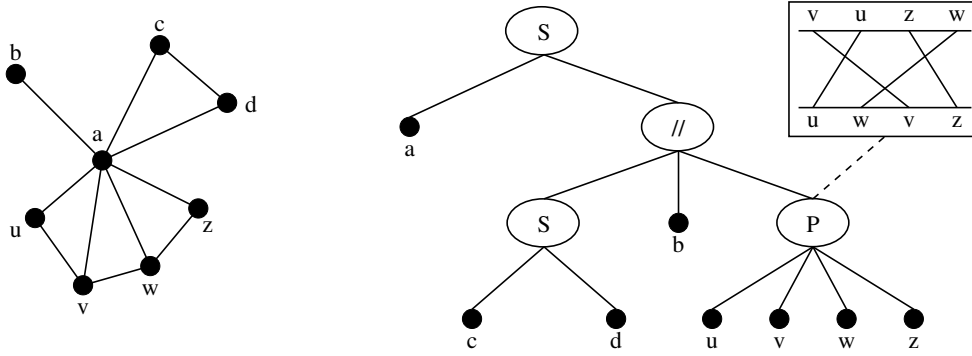


FIG. 1.23 – Une réalisation de l'arbre de décomposition modulaire d'un graphe de permutation et le réalisateur correspondant.

La manière dont l'arbre de décomposition modulaire fournit l'ensemble des réalisateurs d'un graphe de permutation est tout à fait similaire à celle vue pour les arbres à degrés de liberté présentés précédemment.

Définition 1.41 Une **réalisation** de l'arbre de décomposition modulaire T d'un graphe de permutation, est l'arbre T dans lequel on a associé à chaque noeud u un réalisateur de son quotient : (π, π) (resp. $(\pi, \bar{\pi})$) si u est un noeud parallèle (resp. série), où π désigne un ordre quelconque sur les fils de u , et un des quatre réalisateurs possibles si u est premier. On note $\text{consistant}(T)$ l'ensemble des réalisateurs de G obtenus par composition de réalisateurs le long d'une réalisation de T .

Théorème 1.28 L'ensemble des réalisateurs de G est précisément $\text{consistant}(T)$.

FIG. 1.24 – Un graphe de permutation et sa MD -représentation.

Ce qui suit vise à donner une idée de la preuve de ce théorème, qui appartient au folklore.

Définition 1.42 *Etant donnée une paire (π_1, π_2) d'ordres linéaires, un **intervalle commun** [UY00] est un ensemble I qui est à la fois un intervalle de π_1 et de π_2 . Un **intervalle commun fort** est un intervalle commun n'en chevauchant aucun autre.*

A l'évidence, les intervalles communs d'un réalisateur $R = (\pi_1, \pi_2)$ d'un graphe de permutation G sont des modules de G . La réciproque est fausse mais :

Théorème 1.29 [dM03] *Les modules forts d'un graphe de permutation $G = (V, E)$ sont exactement les intervalles communs forts d'un quelconque de ses réalisateurs.*

Idée de la preuve du théorème 1.28 : Soit R un réalisateur de G et M un module fort de G . D'après le théorème 1.29, M est un intervalle commun de R . Soit $\{M_1, \dots, M_k\}$ l'ensemble des modules forts maximaux de $G[M]$. $\forall i \in \llbracket 1, k \rrbracket, M_i$ est un intervalle commun de $R[M]$. Soit $\{x_1, \dots, x_k\}$ un ensemble de représentants de $\{M_1, \dots, M_k\}$. $R' = R[\{x_1, \dots, x_k\}]$ est un réalisateur de $G[M]/\mathcal{MFM}(G[M])$. En substituant les réalisateurs $R[M_i]$, pour $i \in \llbracket 1, k \rrbracket$, dans R' on obtient $R[M]$. Et comme M est un intervalle commun de R , cela montre que R s'obtient bien par substitution des réalisateurs des quotients tout au long de l'arbre de décomposition modulaire de G . ■

MD -représentation d'un graphe de permutation

Dans la MD -représentation d'un graphe de permutation G , on peut représenter les quotients des noeuds premiers par un de leurs quatre réalisateurs géométriquement équivalents. Ce faisant, on obtient une représentation de G dont la taille est $O(n)$ (voir figure 1.24).

Nous avons vu en section 1.4.3, que l'arbre de décomposition modulaire permet de représenter tous les réalisateurs d'un graphe de permutation. D'un point de vue algorithmique, étant donné une réalisation de la MD -représentation d'un graphe de permutation,

calculer le réalisateur correspondant en appliquant la composition de réalisateurs selon toutes les arêtes de l'arbre prend un temps $O(n)$. Obtenir un réalisateur à partir de la seule MD -représentation prend évidemment le même temps car il suffit de choisir d'abord un réalisateur pour les noeuds dégénérés afin d'obtenir une réalisation.

Dans [BXHP05], l'algorithme de [UY00] a été revisité de sorte à ce que l'arbre d'inclusion des intervalles communs forts soit calculé en temps $O(n)$ à partir des deux ordres linéaires. Le même résultat est obtenu dans [BCdMR05] par une autre méthode. D'après le théorème 1.29, cet arbre d'inclusion est l'arbre de décomposition modulaire du graphe de permutation correspondant. En outre, du réalisateur donné pour le graphe en entier, il est possible d'extraire les réalisateurs des quotients des noeuds premiers dans le même temps de $O(n)$.

Ainsi, étant donné la MD -représentation d'un graphe de permutation G on peut obtenir un réalisateur de G en temps $O(n)$ et réciproquement, étant donné un réalisateur de G , on peut obtenir la MD -représentation de G en temps $O(n)$. Or, un réalisateur comme la MD -représentation ont une taille de $O(n)$: ces deux structures sont donc linéairement équivalentes au sens de la définition qui suit.

Définition 1.43 *Deux structures de donnée D_1 et D_2 sont dites **linéairement équivalentes** si on peut passer de D_1 à D_2 en temps linéaire par rapport à la taille de D_1 et réciproquement.*

De la définition, il s'ensuit que des structures équivalentes ont une taille équivalente, c'est à dire $|D_1| = O(|D_2|)$ et $|D_2| = O(|D_1|)$.

1.4.4 Décomposition en coupes et graphes de cordes

L'ensemble des modèles d'un graphe de cordes connexe admet une représentation arborescente. Pour les graphes de permutation, ce rôle est joué par l'arbre de décomposition modulaire. Pour les graphes de cordes connexes, c'est l'arbre de décomposition en coupes qui convient. La contrainte de connexité vient du fait que la décomposition en coupes ne s'applique qu'aux graphes connexes. Même si cela n'est pas précisé, tous les graphes de cordes considérés jusqu'à la fin de la section sont connexes.

Ce qui suit reprend les résultats de [Bou87, GSH89] qui montrent comment la décomposition en coupes décompose un modèle de cordes. Ici, ces approches ont été complétées pour montrer que grâce à l'arbre de décomposition en coupes on peut obtenir une et une seule fois tous les modèles de cordes d'un graphe. A la manière de [Bou87], on manipulera les modèles de cordes comme des mots circulaires dont chacune des lettres a exactement deux occurrences. Chaque occurrence d'une lettre correspond à une extrémité d'une corde et le mot circulaire est l'ordre dans lequel on rencontre ces extrémités en tournant dans le sens trigonométrique.

Les graphes de cordes sont compatibles avec la décomposition en coupes car la composition $*$ se définit naturellement pour des modèles de cordes. Comme le fait remarquer [Bou87], la composition $*$ de deux modèles de cordes fournit plusieurs résultats.

Définition 1.44 Soient G_1 et G_2 deux graphes de cordes ayant un sommet s en commun. Soient $sAsB$ et $sXsY$ des modèles de cordes respectifs de G_1 et G_2 . On définit deux compositions $*$ des modèles de G_1 et G_2 qui sont les modèles $AXBY$ et $AYBX$.

Lemme 1.16 Les deux compositions $*$ des modèles de G_1 et G_2 sont les mêmes ssi G_1 ou G_2 est une clique.

Preuve : Si $AXBY$ et $AYBX$ définissent les deux mêmes ordres circulaires, alors $AXBY$ et $AYBX$ définissent les mêmes ordres linéaires ou $AXBY$ et $BXAY$ définissent les mêmes ordres linéaires. Ce qui implique que $X = Y$ ou $A = B$, c'est à dire G_1 ou G_2 est une clique. La réciproque est vrai car si G est une clique contenant s alors n'importe lequel de ses modèles s'écrit $sAsA$ où A est un ordre linéaire sur ses sommets autres que s . ■

Les deux lemmes qui suivent montrent le bon comportement des graphes de cordes par rapport à la composition $*$.

Lemme 1.17 [Bou87] Les deux compositions $*$ des modèles de G_1 et G_2 sont des modèles de $G_1 * G_2$.

Lemme 1.18 [Bou87] Soient G_1 et G_2 deux graphes connexes partageant un sommet commun. $G_1 * G_2$ est un graphe de corde ssi G_1 et G_2 sont des graphes de cordes.

On en déduit le théorème suivant qui montre que les graphes de cordes sont proprement décomposés par la décomposition en coupes.

Théorème 1.30 [Bou87, GSH89] Un graphe est un graphe de cordes ssi les quotients des noeuds premiers de son arbre de décomposition en coupes sont eux-mêmes des graphes de cordes.

Idée de la preuve. Comme la famille des graphes de cordes est héréditaire et comme les graphes quotients des noeuds premiers de l'arbre de décomposition en coupes sont des graphes induits, nécessairement, ces graphes sont des graphes de cordes. La réciproque provient directement du lemme 1.18. □

Lorsqu'on possède un modèle de cordes, on en obtient un en général différent par renversement, c'est à dire en inversant l'ordre dans lequel on rencontre les extrémités des cordes en faisant le tour du cercle dans le sens trigonométrique. Même à renversement près, il n'y a en général pas unicité du modèle de cordes d'un graphe, sauf pour les graphes premiers.

Théorème 1.31 [Bou87, GSH89] Un graphe de cordes G possède un unique modèle, à renversement près, ssi G est premier pour la décomposition en coupes.

Cela signifie que tous les degrés de liberté dans le choix d'un modèle de cordes pour un graphe donné vont être concentrés dans les noeuds dégénérés de son arbre de décomposition en coupes. Pour une étoile, on obtient tous ses modèles une et une seule fois en prenant les modèles de la forme $aBa\overline{B}$, où a est le centre de l'étoile et B un ordre linéaire sur les feuilles de l'étoile, pris à renversement près. Pour une clique, on obtient tous ses modèles une et une seule fois en prenant les modèles de la forme AA , où A est un ordre circulaire sur les sommets de la clique.

Définition 1.45 Une **assignation** de l'arbre T de décomposition en coupes d'un graphe de cordes est l'arbre T dans lequel on a associé :

1. à chaque noeud un modèle de cordes du graphe qui lui correspond ;
2. à chaque arête un de ses choix de composition * possible.

Les arêtes dont un des noeuds extrémités ont pour graphe associé une clique n'offrent qu'un choix possible de composition *, les autres en offrent deux (lemme 1.16). L'ensemble des modèles de cordes obtenus par application de la composition * choisie le long des arêtes d'une assignation de T est noté $\text{consistant}(T)$.

Le lemme qui suit vise à montrer que deux assignations différentes de l'arbre de décomposition en coupes d'un graphes de cordes donnent deux modèles de cordes différents.

Lemme 1.19 Soient G et H deux graphes de cordes contenant le sommet s . Soit $\mathcal{M}_{G*H}$ un modèle de cordes de $G*H$. Il existe un unique couple $(\mathcal{M}_G, \mathcal{M}_H)$ de modèles de cordes de G et H tel que $\mathcal{M}_{G*H}$ s'obtient comme composition * de $\mathcal{M}_G$ et $\mathcal{M}_H$.

Preuve : Il suffit de remarquer que $\mathcal{M}_{G*H}$ restreint aux sommets de G et de H impose les modèles de cordes $\mathcal{M}_G$ et $\mathcal{M}_H$. ■

Grâce aux lemmes 1.17 et 1.19, on déduit le résultat suivant.

Théorème 1.32 Soit G un graphe de cordes et soit T son arbre de décomposition en coupes. L'ensemble des modèles de cordes de G est exactement $\text{consistant}(T)$ et deux assignations de T différentes donnent lieu à deux modèles de cordes de G différents.

1.4.5 Décomposition en composantes triconnexes et graphes planaires

Dans cette section, on montre, à la manière de [Tut66], comment la décomposition en composantes triconnexes permet de représenter tous les plongements d'un graphe planaire. Il est à noter que les relations entre cette décomposition et les graphes planaires ont été utilisées et revisitées dans [BM90, BT96]

En particulier, [BT96] utilise une structure appelée *SPQR*-arbre pour résoudre le problème de la reconnaissance dynamique des graphes planaires. Les *SPQR*-arbres de [BT96] servent à représenter tous les plongements d'un graphe planaire biconnexe qui a été

orienté de manière acyclique de sorte à avoir une unique source et un unique puits. C'est ce que [BT96] appelle les *st*-graphes planaires ("planar *st*-graphs"). Cette orientation du graphe n'est pas nécessaire pour pouvoir en représenter tous les plongements. On a la même propriété avec la décomposition classique de Tutte. Pour compléter le parallèle entre les *SPQR*-arbres et la décomposition en composantes triconnexes, il convient de remarquer que les noeuds typés *R* ne sont pas absolument nécessaires aux *SPQR*-arbres. En effet, ce sont des noeuds triviaux de décomposition qui correspondent simplement aux arêtes du graphe décomposé. Les noeuds *S*, *P* et *Q* correspondent aux trois types de graphes de la décomposition de Tutte :

- les noeuds *S* correspondent aux cycles,
- les noeuds *P* correspondent aux multigraphes à deux sommets et
- les noeuds *Q* correspondent aux graphes triconnexes.

Ces correspondances sont de plus biaisée par l'orientation et le *SPQR*-arbre d'un *st*-graphe planaire ne correspond pas forcément à la décomposition par paires séparantes canonique⁴ du graphe non orienté sous-jacent car les choix des paires séparantes utilisées dans la décomposition en *SPQR*-arbres sont dictés par l'orientation des arêtes.

Pour ces raisons, je préfère présenter la vision basée sur la décomposition de Tutte. Dans ce qui suit, on montre succinctement les relations entre cette décomposition et la planarité, qui ont été explicitées dans [Tut66]. Dans toutes la suite, lorsqu'on parle de dessin d'un graphe planaire, on sous-entend dessin sans croisement d'arêtes.

La décomposition d'un graphe biconnexe en composantes triconnexes (voir section 1.2.3) est compatible avec la planarité.

Théorème 1.33 *Un graphe biconnexe est planaire ssi les graphes triconnexes issus de sa décomposition en composantes triconnexes sont planaires.*

Idée de la preuve. Les graphes de la décomposition de *G* selon une paire séparante $\{x, y\}$ sont obtenus en remplaçant un sous-graphe connexe contenant *x* et *y* par l'arête *xy*. Clairement, cette opération préserve la planarité. Réciproquement, la 2-somme, qui recolle deux graphes par une arête commune, conserve la planarité. Comme les cycles et les multigraphes à deux sommets sont planaires, le résultat suit. $\square$

La façon dont la décomposition de Tutte représente l'ensemble des plongements des graphes planaires bi-connexes répond au même schéma que celui mis en oeuvre sur les autres classes de graphes traitées dans cette section. Il faut choisir un plongement pour chacun des noeuds de l'arbre de décomposition et les combiner pour obtenir un plongement du graphe lui-même. Dans ce cas, l'opération de combinaison utilisée est une spécialisation de la 2-somme aux plongements, nous en donnons la définition ci-après. L'ensemble des plongements du graphe décomposé est l'ensemble des combinaisons obtenues en prenant tous les choix possibles pour chaque noeud de l'arbre. Comme pour les *PQ*-arbres et les *PC*-arbres, certains noeuds offrent une grande liberté dans le choix de leur plongement,

⁴La décomposition par paires séparantes canoniques d'un graphe est sa décomposition en composantes triconnexes, voir section 1.2.3

alors que d'autres offrent un choix très restreint : un multigraphe à deux sommets admet autant de plongements que d'ordres circulaires sur ses arêtes, alors qu'un cycle admet un unique plongement et qu'un graphe planaire triconnexe admet un unique plongement à renversement près [Whi32, Whi33].

Afin de facilement décrire la construction de dessins de graphes planaires sur la sphère, on introduit la notion de sphère particularisée.

Définition 1.46 *La sphère particularisée est une sphère dans laquelle on a choisi un équateur, deux points notés A et B diamétralement opposés sur cet équateur, ainsi qu'une demi-sphère ouverte nord et une demi-sphère ouverte sud.*

Définition 1.47 *Soient G et H deux graphes planaires contenant l'arête xy . Soient $\mathcal{E}_G$ et $\mathcal{E}_H$ respectivement des plongements de G et H . La 2-somme de plongements de $\mathcal{E}_G$ et $\mathcal{E}_H$ est le plongement de la 2-somme de G et H correspondant au dessin sur la sphère particularisée obtenu ainsi :*

- faire un dessin de $\mathcal{E}_G$ qui place x et y respectivement en A et B , l'arête xy sur l'équateur et tous les autres sommets et arêtes de G dans la demi-sphère ouverte nord,
- faire un dessin de $\mathcal{E}_H$ qui place x et y respectivement en A et B , l'arête xy sur l'équateur et tous les autres sommets et arêtes de H dans la demi-sphère ouverte sud, et
- retirer du dessin les arêtes xy de G et H .

Il convient de remarquer que cette définition est bien formée car quels que soient les dessins remplissant les conditions demandées que l'on choisit comme représentants de $\mathcal{E}_G$ et $\mathcal{E}_H$, on aboutit à des dessins de la 2-somme de G et H correspondant au même plongement.

La propriété suivante est la raison pour laquelle la décomposition de Tutte représente exactement tous les plongements d'un graphe planaire.

Lemme 1.20 *Soient G et H deux graphes planaires qui contiennent l'arête xy , et tels que G n'admet pas $\{x, y\}$ comme paire séparante et possède xy comme arête simple. Soit $\mathcal{E}_{GH}$ un plongement de la 2-somme de G et H , notée S_{GH} . Il existe un unique couple $(\mathcal{E}_G, \mathcal{E}_H)$ de plongements de G et H tel que $\mathcal{E}_{GH}$ s'obtient comme 2-somme de plongements de $\mathcal{E}_G$ et $\mathcal{E}_H$.*

Idée de la preuve. Pour montrer l'existence, considérons un dessin de $\mathcal{E}_{GH}$ sur la sphère particularisée qui place x et y respectivement en A et B . Comme xy n'est pas une paire séparante de G , nécessairement, les sommets de $V(G) \setminus \{x, y\}$ sont tous dans la même face de la restriction $\mathcal{E}'_{GH}$ de $\mathcal{E}_{GH}$ aux sommets de H . De plus, comme xy est une arête simple de G il ne reste aucune arête xy provenant de G dans $\mathcal{E}'_{GH}$. Il s'ensuit que tous les sommets et arêtes provenant de G sont dans la même face de $\mathcal{E}'_{GH}$. Ainsi, il est possible de déplacer les sommets et arêtes de G de sorte qu'ils soient dans la demi-sphère ouverte nord et les sommets et arêtes de H de sorte qu'ils soient dans la demi-sphère ouverte sud.

Pour ce qui est de l'unicité, on donne l'argument suivant. Dans un dessin de $\mathcal{E}_{GH}$ sur la sphère particularisée, en remplaçant tour à tour le dessin de $G - xy$ par l'arête xy et le dessin de $H - xy$ par xy , on remarque que le plongement $\mathcal{E}_{GH}$ impose les faces des dessins de G et H . Ce qui conclut à l'unicité des plongements respectifs dont ils sont issus. $\square$

La notion suivante correspond à celle de solidification pour les PQ -arbres et les PC -arbres.

Définition 1.48 *Une assignation de l'arbre T de décomposition en composantes triconnexes d'un graphe planaire biconnexe est l'arbre T dans lequel on a associé à chaque noeud un plongement du graphe qui lui correspond. L'ensemble des plongements obtenus par application de la 2-somme de plongements le long des arêtes d'une assignation de T est noté $\text{consistant}(T)$.*

Du lemme 1.20, on déduit le résultat suivant.

Théorème 1.34 [Tut66] *Soit G un graphe planaire biconnexe et soit T son arbre de décomposition en composantes triconnexes. L'ensemble des plongements de G est exactement $\text{consistant}(T)$ et deux assignations de T différentes donnent lieu à deux plongements de G différents.*

1.5 Conclusion

Dans ce chapitre ont été présentés trois types de représentations de graphes : décompositions, modèles géométriques et arbres à degrés de liberté. Je me suis attaché à mettre en parallèle, sur les exemples de plusieurs classes de graphes, les relations connues entre ces différents objets. Les trois exemples les plus proches sont ceux des graphes de permutation, des graphes de cordes et des graphes planaires. Le schéma général qui s'en dégage peut être résumé ainsi.

Pour chacune de ces trois classes de graphes, on peut représenter l'ensemble des modèles géométriques d'un graphe à l'aide d'une décomposition générale de graphes : la décomposition modulaire pour les graphes de permutation, la décomposition en coupes pour les graphes de cordes et la décomposition en composantes triconnexes pour les graphes planaires.

Chacune des trois classes est proprement décomposée par la décomposition qu'on lui applique. C'est à dire qu'on a un théorème du type :

un graphe appartient à la classe ssi les graphes indécomposables issus de sa décomposition appartiennent eux mêmes à la classe.

Dans les trois cas, ces théorèmes font usage du fait que les autres graphes issus de la décomposition sont dans la classe : les cliques et les stables pour les graphes de permutation, les cliques et les étoiles pour les graphes de cordes, et les cycles et les multigraphes à deux sommets pour les graphes planaires. Ainsi, on peut associer un modèle à chacun des graphes issus de la décomposition, c'est à dire à chacun des noeuds de l'arbre de décomposition.

Ces théorèmes s'appuient aussi grandement sur les bonnes propriétés des modèles géométriques de la classe par rapport à la décomposition associée. En particulier, dans les trois cas, on peut définir une opération de composition sur les modèles géométriques qui est le pendant naturel de l'opération de composition définie pour les graphes en général. Cette opération est celle utilisée pour reconstruire un modèle du graphe de départ à partir des modèles associés à chacun des noeuds de l'arbre de décomposition.

Un des faits marquants en commun dans les trois cas est que les graphes indécomposables n'admettent qu'un (à une constante près) modèle alors que les autres graphes issus de la décomposition en admettent beaucoup⁵ (en nombre factoriel par rapport à celui de leur sommets!). C'est la raison de la terminologie "par degrés de liberté" : les noeuds dégénérés de la décomposition concentrent la liberté dont on dispose dans le choix des modèles, alors que les noeuds premiers n'offrent aucun choix.

La façon dont la décomposition représente tous les modèles est commune aux trois classes de graphes : l'ensemble des modèles du graphe est l'ensemble des modèles qu'on peut obtenir en choisissant un modèle pour chacun des noeuds de l'arbre et en les assemblant par application de la composition de modèles selon chacune des arêtes de l'arbre de décomposition.

En ce qui concerne les graphes d'intervalles et les graphes d'arcs de cercle de Helly, ils admettent aussi une représentation arborescente par degrés de liberté, qui sont respectivement le PQ -arbre et le PC -arbre de leurs cliques maximales. On peut remarquer deux différences essentielles avec les représentations précédentes :

1. les PQ -arbres et PC -arbres ne sont pas définis à partir d'une décomposition ;
2. les feuilles de ces arbres ne sont pas les sommets du graphe mais ses cliques maximales.

Pourtant, c'est une idée communément admise dans la communauté des graphes que le PQ -arbre d'un graphe d'intervalle a un rapport avec son arbre de décomposition modulaire, dans le sens où on peut se servir de l'un comme de l'autre pour résoudre efficacement certains problèmes. En section 5.3, nous expliciterons les relations entre ces deux objets. Nous verrons que malgré leurs différences notables, ces deux arbres sont mathématiquement et algorithmiquement très proches.

La question reste ouverte pour les PC -arbres, sont ils proches de l'arbre d'une certaine décomposition appliquée aux graphes d'arcs de cercle de Helly ? D'une certaine manière, vu les rapports entre les définitions des graphes de permutation, d'intervalles, de cordes et d'arcs de cercle, et vu les rapports qu'ont les trois premières classes avec les décompositions modulaire et en coupes, il serait plaisant que le PC -arbre des graphes d'arcs de cercle de Helly soit lié à leur décomposition en coupes. Le lecteur ne me tiendra pas rigueur du fait que la formulation de la question ouverte suivante ne repose que sur la légèreté de l'observation ci-dessus.

Question ouverte 1.3 *Y a-t-il un rapport entre le PC -arbre des cliques maximales d'un graphe d'arcs de cercle de Helly et son arbre de décomposition en coupes similaire à celui*

⁵Une seule exception à signaler à cette règle : les cycles, qui n'admettent qu'un plongement planaire.

exprimé en section 5.3 entre le PQ-arbre des cliques maximales d'un graphe d'intervalles et son arbre de décomposition modulaire ?

Chapitre 2

Algorithmes dynamiques

La spécificité de ce mémoire est le maintien dynamique de représentations de graphes lors de changements élémentaires du graphe en question. Nous avons déjà présenté les représentations que nous utiliserons. Ce chapitre introduit la problématique de l'algorithmique dynamique et les notions qui s'y rapportent. En dernière section, ces notions sont appliquées sur une étude du comportement dynamique des deux représentations de graphes les plus classiques : la matrice d'adjacence et les listes d'adjacence. Pour ces dernières, on discute de quelques modifications d'implémentation possibles en vue d'améliorer leur dynamisme.

2.1 Problématique de l'algorithmique dynamique

Des algorithmes dynamiques

Il convient pour commencer de lever une possible ambiguïté sur le terme d'**algorithme dynamique**. En algorithmique de graphe, la mention dynamique pour un algorithme se rapporte en fait au graphe sur lequel porte l'algorithme et non pas à l'algorithme lui-même. En effet, les algorithmes qui seront présentés dans ce mémoire n'ont aucune dynamique, en ce sens qu'ils ne se modifient pas de quelque manière que ce soit.

Ce qui se modifie, en algorithmique dynamique de graphe, c'est le graphe considéré. Cette branche de la théorie se donne pour but l'entretien du résultat d'un calcul sur un graphe lorsque ce dernier subit une modification. Par opposition, un **algorithme statique** (c'est le cas classique) doit fournir le résultat du calcul à partir de la donnée du problème, mais ne possède pas pour ce faire le résultat du calcul sur un autre graphe. Tout problème de graphe peut être considéré du point de vue dynamique. On peut par exemple souhaiter actualiser, lorsque le graphe considéré est modifié, la liste des composantes connexes d'un graphe, un circuit hamiltonien, l'ordre de rencontre des sommets dans un parcours en profondeur, le résultat d'un problème de décision. Connaissant le résultat d'un calcul sur un graphe et étant donné une modification de ce graphe, à quel coût peut-on déduire le résultat du calcul sur le nouveau graphe ? C'est la question fondamentale de l'algorithmique dynamique.

Envisager n'importe quelle modification possible du graphe conduirait à entretenir le résultat du calcul en passant d'un graphe à un autre graphe sans rapport avec le premier. Dans de tels cas, la démarche perdrait de son sens et le problème ne serait sans doute pas différent de celui du calcul statique sur le second graphe. Pour ces raisons, les modifications du graphe considérées sont "légères". Ainsi, lorsque le graphe modifié est proche du graphe initial, peut être est-il possible de tirer parti du résultat afin de déduire le résultat du calcul pour le graphe modifié, à un coût moindre de celui du calcul statique sur le graphe modifié. Toute modification du graphe conservant une certaine proximité du graphe modifié avec le graphe initial peut être considérée. Néanmoins, dans la plupart des algorithmes dynamiques (dont ceux présentés dans ce manuscrit), les modifications élémentaires du graphe autorisées sont parmi les quatre suivantes :

- ajout d'une arête entre deux sommets existants,
- suppression d'une arête entre deux sommets existants,
- ajout d'un nouveau sommet avec les arêtes définissant son voisinage,
- suppression d'un sommet avec les arêtes qui lui sont incidentes.

Les algorithmes supportant les opérations d'ajout et de suppression sont dits **entièrement dynamiques**. Par exemple, on parle d'algorithme entièrement dynamique sur les sommets ou sur les arêtes.

Intérêt pratique

D'un point de vue pratique, le besoin d'algorithmes dynamiques se fait ressentir dès lors que les graphes modélisent des systèmes se modifiant dans le temps. Cette situation peut se rencontrer dans tous les domaines d'application des graphes mais prend une ampleur particulière dans le domaine des réseaux de communication. Lorsque l'algorithmique dynamique est appliquée aux problèmes de réseaux, elle est en général qualifiée d'«algorithmique online». Le lecteur désireux d'en savoir plus sur cette branche de la théorie se référera à [BEY98, FW98].

Le graphe physique d'internet et le graphe des communications directes dans un réseau ad'hoc sont des exemples de graphes ayant une forte dynamique. Dans ces cas où le graphe considéré se modifie très vite, la complexité du calcul dynamique est en compétition directe avec la fréquence moyenne des modifications du graphe. En effet, si on ne veut pas être submergé par les modifications, il faut être capable de traiter chacune d'entre elles suffisamment rapidement pour maintenir leur file d'arrivée à un niveau stable. Ainsi, les progrès de l'algorithmique dynamique pourraient fournir les outils nécessaires à l'observation de systèmes modélisables par des graphes et se modifiant à une fréquence élevée.

Un autre domaine privilégié d'application de l'algorithmique dynamique est celui de la gestion des structures de données employées dans les programmes informatiques. En effet, au cours de l'exécution d'un programme, il est rare que les structures de données soient figées, et il peut être nécessaire au cours de leur modification de savoir si elles vérifient une propriété donnée ou d'en entretenir une représentation particulière. Dès lors, les algorithmes utilisés sont dynamiques.

Intérêt Théorique

D'un point de vue théorique, l'algorithmique dynamique de graphe cherche à déterminer la sensibilité d'un calcul à une modification de sa donnée. Ceci correspond en quelque sorte à une notion de continuité, ou de stabilité, d'un calcul : si une légère modification est opérée sur la donnée, le nouveau résultat recherché est-il séparé du premier par un temps de calcul faible ou important ?

Des problèmes de complexité très différentes ont été étudiés d'un point de vue dynamique : des problèmes *NP*-complets jusqu'aux problèmes pour lesquels on connaît un algorithme de complexité linéaire. Pourtant, si le nombre d'algorithmes dynamiques est important, une poignée de problèmes ont concentré une attention particulière. C'est le cas des problèmes de connexité, de biconnexité, de triconnexité, de 2-arête connexité, ou encore les problèmes de bipartition, de fermeture transitive, de plus courts chemins et d'arbre couvrant de poids minimal. Dans la section suivante, nous présentons une autre catégorie de problèmes dynamiques qui est celle des problèmes de représentation et de reconnaissance de classes de graphes.

L'intérêt du domaine est accru par le fait que les problèmes algorithmiques de graphes ont des comportements dynamiques très variés. Certains sont naturellement dynamiques alors que pour d'autres il paraît difficile de faire mieux que refaire entièrement le calcul sur le graphe modifié. Parmi les problèmes difficiles à traiter dynamiquement de manière efficace, on trouve des problèmes de graphes extrêmement simples. C'est le cas du calcul des composantes connexes d'un graphe non orienté. Il existe un algorithme qui résout le problème en temps linéaire : il suffit de faire un parcours en largeur des sommets du graphe. Par contre, d'un point de vue dynamique, comment déterminer si une composante connexe est scindée en deux par le retrait d'une arête ? Faire un parcours en largeur aboutirait à une complexité dans le pire des cas qui est celle du problème statique. Pourtant, Holm, de Lichtenberg et Thorup [HdLT98] donnent un algorithme dynamique traitant l'ajout ou le retrait d'une arête en temps amorti $O(\log^2 n)$ par modification. Leur algorithme fait appel à des structures de données très sophistiquées et complexes, et son analyse est difficile. La conception d'algorithmes dynamiques est souvent difficile et pousse à explorer plus loin les possibilités algorithmiques.

Cadre formel des problèmes dynamiques de graphe

La définition formelle d'un problème dynamique n'est pas si aisée. Un problème statique, est défini par une donnée, qui sert de base au calcul, et une question, qui exprime ce qu'on attend du calcul. Un problème dynamique comporte aussi une question qui est du même type que celle qu'on trouve dans la définition d'un problème statique. C'est en ce qui concerne la donnée d'un problème dynamique que survient une difficulté supplémentaire. A mon sens, il convient de distinguer deux parties dans la donnée : la modification du graphe d'une part et la **donnée résiduelle** d'autre part.

La modification du graphe. C'est une donnée objective du problème, le formalisme dans lequel elle est décrite fait partie intégrante de la définition du problème et n'est pas laissé au choix de celui qui le résoud.

Comme dans le cas statique, la façon dont est codée la donnée, ici la modification du graphe, a une influence sur la complexité que peut atteindre un algorithme résolvant le problème. Par exemple, lors de l'ajout d'un sommet, si son voisinage est un ensemble de sommets consécutifs dans un ordre fixé, on peut ne donner que son premier et dernier voisin dans cet ordre. Ou encore, lors du retrait d'un sommet, on peut préciser ou non son voisinage.

Dans toute la suite de ce mémoire, nous n'exploitons pas la piste laissée par la question de la représentation de la modification du graphe : les arêtes sont données par leurs deux sommets incidents, et les voisinages, lors de l'ajout d'un sommet, sont donnés par la liste explicite des sommets qu'ils contiennent. Lors du retrait d'un sommet, son voisinage n'est pas précisé.

Reste une question en suspens concernant la façon dont est donnée la modification : un sommet est-il désigné par son identifiant ou par un pointeur sur une zone mémoire qui lui est propre ? Dans la littérature, la réponse à cette question est rarement précisée par les auteurs. Pourtant, elle n'est pas sans enjeux. Si le sommet est donné par son identifiant, il est souvent nécessaire de le rechercher dans la structure de donnée, alors que s'il est désigné par un pointeur sur la zone de la structure de donnée lui étant dédiée, cette recherche peut être évitée. Bien sûr, pour les problèmes dynamiques dont la résolution est plus coûteuse que les recherches, cette question tombe à l'eau. Ce n'est pas le cas de tous les algorithmes présentés dans ce mémoire, qui évoluent dans des domaines de complexités très basses et pour lesquels le coût de la recherche d'un sommet donné par son identifiant n'est pas négligeable. Le choix fait dans ce travail est de désigner les sommets par leur identifiant. Il arrivera cependant, notamment en section 4.3, que l'on abandonne ce choix pour désigner les sommets par un pointeur sur la zone qu'ils occupent dans la structure de donnée. Cela évite que la complexité du problème envisagé ne soit masquée par le coût de la recherche initiale dans la structure de donnée. Ces exceptions seront explicitement mentionnées.

Le véritable enjeu de la désignation des sommets dépasse le cadre strict de l'algorithme, il s'agit de savoir quelle est l'entité qui commande les modifications du graphe au programme implémentant l'algorithme dynamique. S'il s'agit d'un autre programme, il n'est pas difficile d'envisager que les deux correspondent en désignant les sommets par leurs adresses mémoires. Par contre, s'il s'agit d'une entité d'une autre nature, telle qu'un humain, celle-ci n'a a priori pas accès aux adresses mémoires et devra donc utiliser les identifiants pour parler des sommets. Dans ce dernier cas, il ne faut pas que le programme, au cours de son exécution, modifie ces identifiants sans en prévenir la personne humaine avec qui il interagit. Même en étant prévenue, cette dernière peut être décontenancée par un changement des noms des sommets. Les réaffectations d'identifiant s'avèrent pourtant bien utiles dans certains cas, notamment pour préserver la consécuitivité des identifiants au cours de l'algorithme. Remarquons qu'il est alors toujours possible d'entretenir un arbre binaire de recherche équilibré qui fait la correspondance entre les identifiants du point de vue utilisateur et ceux du point de vue programme. Cela ayant pour effet, au pire, d'aug-

menter toutes les opérations d'un facteur $\log n$. Le programmeur préférera sûrement faire la correspondance à l'aide de structures de données du type table de hachage garantissant de meilleurs résultats en espérance. Puisque cette question n'intéresse ni l'algorithmicien ni le programmeur, laissons-la de côté. Je ne l'ai soulevée que par souci de ne pas cacher au lecteur les problèmes attachés à la thématique.

La donnée résiduelle. C'est ce qui est présent en mémoire avant la modification demandée. Elle dépend entièrement de l'algorithme conçu pour répondre au problème et ne fait pas partie de la définition du problème. L'algorithme a le choix des structures qu'il entretient. La représentation du graphe adoptée peut être complète ou partielle, elle peut comporter tout objet jugé utile par le concepteur de l'algorithme. Le problème n'exprime aucune contrainte sur la structure employée, sa seule exigence est que l'algorithme réponde à la question posée pour le graphe résultant de la modification. On peut même choisir -bien que cette question soit sans enjeu à mon avis- de ne pas laisser en mémoire la réponse à la question posée par le problème sur le graphe avant la nouvelle modification. Le choix de la donnée résiduelle est entièrement libre.

C'est là que réside la différence majeure avec l'algorithmique statique : l'essentiel de la donnée n'est pas imposée par le problème mais laissée à l'appréciation de l'algorithmicien. Cela accroît encore la difficulté à savoir (ou prouver) si la complexité d'un algorithme pour résoudre un problème est optimale, car il faut envisager tous les choix possibles de la donnée résiduelle. Cette liberté donne une importance particulière aux structures de données, et les algorithmes dynamiques sont gourmands et producteurs de structures efficaces. Les complexités atteintes pour résoudre les problèmes dynamiques sont parfois surprenantes (même lorsqu'elles sont justes).

2.2 Le problème de représentation et reconnaissance dynamique d'une classe de graphes

Les **problèmes de représentation et reconnaissance dynamique d'une classe de graphes** (PRRD en abrégé) sont une sous-famille des problèmes dynamiques de graphes. Dans ce type de problème sur une classe de graphe $\mathcal{F}$, la question posée est : le graphe modifié appartient-il à $\mathcal{F}$? et si oui, en donner une représentation (fixée ou libre).

Dans un PRRD, la représentation demandée requiert souvent la condition que le graphe appartienne à $\mathcal{F}$. Par exemple, le problème de reconnaissance et de représentation dynamique pour la classe des graphes de permutation peut s'exprimer de la manière suivante : le graphe modifié est-il un graphe de permutation ? si oui, en donner un réalisateur (voir section 1.3.2). Dans ce cas, l'existence d'un réalisateur est, par définition, subordonnée à l'appartenance à la classe.

Les PRRD présentent une restriction importante par rapport aux problèmes dynamiques en général. N'importe quelle séquence de modifications ne peut pas être envisagée, car l'algorithme s'arrête dès lors que le graphe n'appartient plus à $\mathcal{F}$. Cette restriction qui

peut paraître fortement contraignante est en fait difficile à éviter. En effet, l'hypothèse que le graphe avant la modification appartient à la classe est pour beaucoup dans la possibilité de déterminer efficacement si le graphe modifié est dans la classe. Ceci vient du fait que l'appartenance à la classe offre une représentation (au sens large, voir le chapitre 1) particulière du graphe et qu'ainsi le problème de savoir si le nouveau graphe est dans la classe se ramène à vérifier si la modification respecte la représentation propre à la classe.

Si on s'autorise, lors de la série de modifications du graphe, à sortir de la classe, alors on peut être amené à considérer n'importe quel graphe et le problème prend une tout autre forme. Il s'agit maintenant de trouver une structure qui décrit non plus que le graphe a telle propriété mais plutôt qui mesure la "distance" à laquelle il est de cette propriété. L'exercice est beaucoup plus difficile, d'autant que l'objectif est toujours de faire mieux que la reconnaissance statique. A ma connaissance, il n'existe pas d'algorithme dynamique de reconnaissance d'une classe de graphes qui autorise à sortir de la classe. En revanche, le PRRD, dans sa définition présentée ici, a été étudié pour de nombreuses classes de graphes [SS04, CP05, NPP06, CP06, HSS02, Iba01, Hsu96, Iba99, BT96, EGIS98]. Une grande partie de ces algorithmes sont abordés dans les chapitres suivants.

2.3 Remarques sur la complexité des algorithmes dynamiques

Afin de donner des repères et les moyens de comparer les complexités des différents algorithmes dynamiques dont il sera question dans le mémoire, on apporte quelques éléments de réponses aux questions suivantes : y a-t-il des opérations plus délicates que d'autres à traiter parmi les quatre modifications élémentaires ? est-il plus difficile de concevoir un algorithme entièrement dynamique qu'un algorithme monotone, (ne manipulant que les insertions ou que les suppressions) ? L'étude de la classe des graphes d'intervalles unitaires illustre à merveille certaines propriétés et comportements des algorithmes dynamiques en rapport avec ces questions.

Il existe un algorithme de reconnaissance et de représentation entièrement dynamique pour la classe des graphes d'intervalles unitaires [HSS02]. Cet algorithme traite l'insertion d'un sommet de degrés d en $O(d + \log n)$, la suppression d'un sommet en $O(\log n)$, l'insertion d'arête en $O(\log n)$ et enfin la suppression d'arête en $O(1)$. Il est analysé plus longuement en section 4.7.1.

Remarquons d'abord la dissymétrie entre les cas d'insertion et de suppression. Ici, les insertions coûtent plus cher, mais dans d'autres problèmes dynamiques, le contraire peut arriver. Il n'y a en fait aucune raison pour que les complexités d'ajouts et de suppression soient les mêmes, ni pour qu'elles soient en faveur de l'une ou l'autre des opérations : cela dépend entièrement du problème traité.

Plus intéressant encore, [HSS02] propose un algorithme purement incrémental sur les arêtes où chaque insertion est traitée en temps $O(1)$ dans le pire des cas. Cela illustre le fait qu'un algorithme monotone, qui ne considère que des insertions ou que des sup-

pressions, peut prétendre à une meilleure complexité dans le pire des cas qu'un algorithme qui manipule les deux opérations alternativement. Ceci dit, cela pourrait venir du fait qu'il est plus difficile de concevoir un algorithme entièrement dynamique qui atteint la même complexité que l'algorithme monotone, sans pour autant signifier qu'un tel algorithme dynamique n'existe pas. Dans le cas présent, [HSS02] montre une borne inférieure de $\Omega(\log n / (\log \log n + \log b))$ par opération pour la reconnaissance entièrement dynamique sur les arêtes des graphes d'intervalles unitaires, dans le modèle de calcul "cell probe" avec une taille de mots b . Pour plus de détails sur le modèle de calcul et la technique de preuve utilisée, [HSS02] renvoie à [Yao81, FS89, HF98].

Ce résultat fournit l'exemple d'un problème dynamique dont la complexité d'un algorithme monotone, résolvant le problème soit pour l'insertion soit pour la suppression, ne peut pas être atteinte par un algorithme traitant à la fois l'insertion et la suppression.

2.4 Des listes d'adjacence dynamiques

Examinons le comportement des listes et de la matrice d'adjacence soumises à des modifications élémentaires du graphe. Les comportements dynamiques des deux structures sont sans rapport. Cela vient du fait que la matrice d'adjacence est un tableau alors que les listes d'adjacences sont composées principalement de listes, même si un tableau peut être s'avérer utile pour leur implémentation.

Comportements dynamiques des listes et des tableaux

Les listes. Les listes n'offrent aucune résistance à l'insertion et à la suppression d'éléments, ce qui s'avère souvent avantageux pour les algorithmes dynamiques. Leur atout majeur est de permettre l'ajout et le retrait d'un élément à une position donnée en temps constant. Dit ainsi, les listes semblent la structure de donnée naturellement appropriée pour les algorithmes dynamiques. La situation est en réalité loin d'être si tranchée. La gestion dynamique des listes pose une difficulté qui provient du fait que l'on connaît rarement la position à laquelle on veut effectuer la modification. C'est pourquoi, dans beaucoup de situations, il est nécessaire de parcourir la liste afin de trouver cette position de modification, ce qui demande un temps proportionnel à la longueur de la liste.

Les tableaux. À l'opposé, le tableau est une structure purement statique très réticente aux modifications mais qui permet une recherche rapide. L'avantage et l'essence même du tableau est de permettre un accès en temps constant au contenu d'une de ses cases dont on connaît l'index. Cette propriété remarquable est rendue possible par le fait que la partie de la mémoire occupée par le tableau est formée de cases mémoires consécutives. Il n'est donc pas possible d'insérer ou de supprimer une case au milieu d'un tableau sans déplacer toutes les cases précédant ou suivant la position d'insertion ou de suppression. Pire, il n'est pas non plus possible d'ajouter une case en début ou en fin du tableau de manière efficace car rien ne garantit que les cases mémoires qui jouxtent la zone occupée par le tableau

soient libres. La seule possibilité consiste alors à déplacer entièrement le tableau vers une zone mémoire offrant le nombre suffisant de cases mémoires libres et consécutives. Ainsi, les opérations d'ajout et de suppression d'une case dans un tableau prennent toutes deux un temps $O(l)$ dans le pire des cas, où l est le nombre de cases du tableau, car le déplacement entier du tableau peut être nécessaire. Pour les tableaux à plusieurs dimensions, la situation n'est pas différente : le retrait ou l'ajout d'une "ligne" demande de réécrire entièrement le tableau dans le pire des cas.

Entretien dynamique de la matrice d'adjacence

En ce qui concerne la matrice d'adjacence qui est un tableau à 2 dimensions et n^2 cases, le retrait ou l'ajout d'un sommet implique d'enlever ou d'ajouter une ligne et une colonne à la matrice, et donc de déplacer toute la structure. Le temps demandé pour mettre à jour la matrice d'adjacence lors d'une modification de sommet est $\Omega(n^2)$. Inversement, du fait de l'accès en temps constant à une case, les modifications d'arêtes ne demandent qu'un temps $O(1)$. Pour garantir cet accès en temps constant à partir de la donnée des identifiants des sommets, il est nécessaire de ré-attribuer ces identifiants, après chaque retrait de sommet, de sorte que tous les numéros utilisés restent consécutifs.

Entretien dynamique des listes d'adjacence

Les listes d'adjacence peuvent être implémentées de plusieurs manières. La liste des sommets du graphe peut être stockée dans un tableau, dans une liste, dans un arbre de recherche ou toute autre structure ; de même pour la liste des voisins d'un sommet.

Quelle que soit l'implémentation choisie, son comportement dynamique peut être amélioré en introduisant quelques pointeurs supplémentaires que nous appellerons des **pointeurs de position**. Lorsque un sommet x est présent dans la liste des voisins d'un sommet y , on ajoute un pointeur de la cellule de x dans la liste de y vers la cellule de y dans la liste de x . Ainsi, lors de la suppression de x , on peut accéder en temps constant à sa cellule dans la liste de y et l'en supprimer.

La seule des implémentations des listes d'adjacence présentées ici qui fournisse la fonctionnalité de donner la liste des voisins d'un sommet x en un temps proportionnel à son degrés est celle qui utilise un tableau pour stocker la liste des sommets du graphe. Dans les autres implémentations, le coût d'une telle requête est augmenté de celui de la recherche de x dans la liste des sommets du graphe. Cependant, la difficulté d'entretien dynamique du tableau incite à envisager d'autres implémentations plus faciles à actualiser.

L'implémentation classique. Commençons par l'implémentation la plus classique des listes d'adjacence : celle utilisant un tableau. Sous les modifications d'arête, cette structure est relativement aisée à maintenir. Lorsque la suppression de l'arête xy est demandée, il faut supprimer x dans la liste des voisins de y et vice versa. Cela peut être fait en temps $O(\min(d(x), d(y)))$ en parcourant les deux listes simultanément jusqu'à trouver y dans la liste des voisins de x , ou x dans la liste des voisins de y . Les pointeurs de position entre

x et y permettent alors de trouver l'autre cellule recherchée. L'ajout d'une arête peut se faire en temps constant (si on suppose que l'arête n'existe pas déjà) car on peut ajouter le nouveau voisin de x et celui de y au début de leurs listes respectives, avec les pointeurs de position nécessaires.

En ce qui concerne les modifications de sommet, comme déjà évoqué, la difficulté d'entretien vient de la nécessité d'ajouter ou de retirer une case du tableau, ce qui nécessite un temps $O(n)$. Le reste de l'actualisation peut être fait plus rapidement. Ajouter le sommet x et la liste de ses voisins prends un temps $O(d(x))$; et ajouter x dans les listes respectives de ses voisins, avec les pointeurs de position correspondants, peut également se faire en $O(d(x))$ en ajoutant x au début de chaque liste. Le temps total demandé par un ajout de sommet est donc $O(n)$. Lors du retrait, effacer x des listes de ses voisins peut être fait en temps $O(d(x))$ grâce aux pointeurs de position qui donne accès en temps constant à la position de x dans la liste d'un quelconque de ses voisins. Le temps total de l'opération de retrait est donc $O(n)$, car il faut toujours modifier le tableau stockant le voisinage de chaque sommet.

Des implémentations plus dynamiques ? On le voit, l'implémentation ci-dessus souffre d'un coût de maintenance élevé lors des modifications de sommet du à la présence du tableau. Cela est le prix à payer pour accéder à la liste des voisins d'un sommet en temps constant. Si on relâche cette exigence, on peut imaginer une structure qui offre un meilleur compromis entre rapidité de recherche et entretien dynamique.

C'est le but avoué des **arbres binaires de recherche équilibrés** (ABRE). Dans ces structures, les sommets sont stockés dans les noeuds d'une arborescence dont toutes les branches sont de longueur $O(\log n)$. Des propriétés supplémentaires de l'arborescence garantissent la possibilité de trouver un élément en partant de la racine et en suivant une seule branche, ce qui donne une recherche en temps $O(\log n)$. De plus, ces structures peuvent être maintenues lors de l'ajout et du retrait d'élément en temps $O(\log n)$, en conservant toutes les propriétés nécessaires, dont la longueur des branches. Parmi ces structures, on trouve par exemple les arbres rouge et noir [CLR94].

Ainsi en stockant les sommets du graphes dans un arbre binaire de recherche équilibré, on diminue le sur-coût dû aux recherches de $O(n)$ à $O(\log n)$. Il peut être intéressant de stocker également les listes de voisins dans un tel arbre, c'est la solution que nous étudions ici. Dans cette version des listes d'adjacence, l'énumération des voisins d'un sommet x donné requiert un temps $O(d(x) + \log n)$.

Pour ajouter ou supprimer une arête xy , il faut rechercher les deux extrémités dans la liste des sommets du graphe, ce qui prends $O(\log n)$. Pour l'ajout, on insère chacune des extrémités de l'arête dans l'ABRE stockant les voisins de l'autre extrémité, avec le pointeur de position adéquat, cela prends $O(\log n)$. Pour le retrait, on recherche et on retire x de l'ABRE des voisins de y , et vice versa, en $O(\log n)$. Les modifications d'arête prennent donc un temps $O(\log n)$.

En ce qui concerne le retrait de sommet, on recherche d'abord le sommet concerné ($O(\log n)$) puis on le retire de l'ABRE de chacun de ses voisins (auxquels on accède en $O(1)$)

grâce aux pointeurs de position). Cela prends donc un temps total de $O(\log n + d \log D)$, où D est le degrés maximum de G . Lorsqu'on ajoute un sommet x de degrés d , son insertion dans l'ABRE des sommets du graphe coûte $O(\log n)$; l'ABRE de ses voisins peut être construit en $O(d \log d)$; et pour chacun des voisins y de x , il faut trouver y dans l'ABRE des sommets du graphe ($O(\log n)$), puis insérer x dans l'ABRE de y , ce qui prends au plus $O(\log D)$. La complexité de l'ajout de sommet est donc $O(d \log n)$.

Chapitre 3

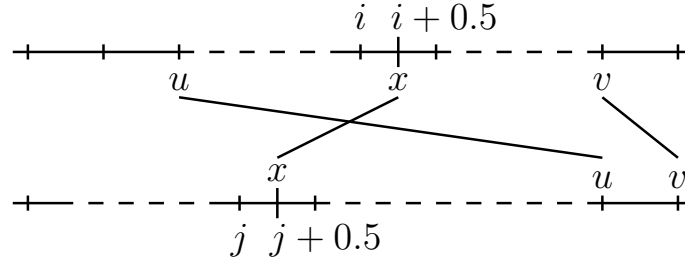
Dynamacité de modèles géométriques de graphes

Ce chapitre est une première étape vers le maintien dynamique de représentations géométriques de graphes. Nous avons vu dans le chapitre 1, différentes classes de graphes qui admettent des modèles géométriques, telles que les graphes d'intervalles et de permutation. Lors d'un algorithme de reconnaissance dynamique d'une classe de graphes qui admettent des modèles géométriques, l'enjeu est de vérifier après chaque opération si le graphe admet toujours un modèle du type souhaité. La difficulté à laquelle on se heurte est que le nouveau graphe peut admettre un modèle qui n'a aucun rapport avec l'ancien modèle. Par exemple, pour les graphes d'intervalles, si on possède un modèle d'intervalles du graphe initial, lors de l'insertion d'un sommet x dans G , on ne peut pas se contenter de tester si on peut attribuer un intervalle convenable à x . Pour obtenir un modèle du nouveau graphe, il peut être nécessaire de changer les intervalles qui étaient attribués aux sommets de G . Ainsi, le problème de maintenir un modèle géométrique donné, sans le bouleverser, est bien plus facile que celui de la reconnaissance dynamique de la classe.

Dans ce chapitre on considère le problème restreint d'entretien dynamique d'un modèle fixé. La section 3.1 traite le cas des graphes de permutation et la section 3.2 celui des graphes d'intervalles. La section 3.3 relate l'histoire des problèmes dynamiques sur les graphes planaires, de laquelle se dégage une idée qui nous permettra de passer, pour d'autres classes de graphes, de l'entretien d'un modèle fixé au problème plus général de la reconnaissance dynamique de la classe. Pour les graphes de permutation et d'intervalles, le problème général est résolu respectivement dans la section 4.5 et le chapitre 5.

3.1 Les graphes de permutation

Cette section est dédiée à l'entretien dynamique d'un réalisateur de graphe de permutation lors d'une modification de sommet. La version du problème envisagée ici est une version restreinte, le problème dans sa version la plus générale est étudié en section 4.5. La donnée est un réalisateur d'un graphe de permutation, et la question posée n'est pas de savoir si

FIG. 3.1 – Une position d'insertion pour x .

le graphe correspondant reste de permutation lors de la modification, mais de savoir si le réalisateur donné peut être restreint ou étendu en un réalisateur du nouveau graphe. La réponse à cette dernière question étant moins souvent affirmative qu'à la première.

Définition 3.1 Soit $G = (V, E)$ un graphe de permutation. On dit qu'un réalisateur R' est la **restriction** du réalisateur R au sous-ensemble de sommets $S \subseteq V$ si R' s'obtient à partir de R en retirant les sommets de S des deux ordres linéaires composant R . Réciproquement, on dit que R' est une **extension** de R si R est une restriction de R' .

Avec cette définition on peut donner une formulation plus rigoureuse du problème que l'on considère ici.

Problème.

Donnée : Un graphe de permutation avec un de ses réalisateurs R et une insertion ou suppression de sommet dans le graphe.

Question : Calculer, si possible, un réalisateur du graphe modifié qui soit une extension ou restriction de R , renvoyer faux si cela n'est pas possible.

Le cas de la suppression d'un sommet se traite très simplement : on peut toujours restreindre le réalisateur donné et cette restriction se calcule en temps $O(n)$. Plus délicat est le cas de l'insertion d'un sommet x . La routine *InsPrime* que nous présentons ici est destinée à le résoudre : elle détermine en $O(n)$ s'il existe une extension du réalisateur (on dit que R est **extensible** à x) et en calcule une, le cas échéant, en temps $O(n)$. Il est à noter qu'en cas d'échec, cette routine ne fait pas la différence entre les cas où $G + x$ n'est pas de permutation et les cas où $G + x$ est de permutation mais le réalisateur donné ne peut pas être étendu.

Routine *InsPrime*.

Etendre un réalisateur consiste à insérer le sommet x dans les deux ordres linéaires composant le réalisateur. Les positions d'insertion dans un ordre linéaire sur n sommets seront désignées par $i + 0.5$, avec $i \in \llbracket 0, n \rrbracket$.

Lemme 3.1 (voir figure 3.1) Soit $R = (\pi_1, \pi_2)$ un réalisateur d'un graphe de permutation $G = (V, E)$, avec $|V| = n$, et $x \notin V$ un sommet à insérer. R est extensible ssi $\exists(i, j) \in \llbracket 0, n \rrbracket^2$ tel que

$$\forall u \in N(x), \pi_1(u) \leq i \text{ iff } \pi_2(u) > j \quad \text{et} \quad \forall v \in \overline{N}(x), \pi_1(v) \leq i \text{ ssi } \pi_2(v) \leq j$$

Un réalisateur étendu s'obtient en insérant x à la position $i + 0.5$ dans π_1 et à la position $j + 0.5$ dans π_2 .

Preuve : Soit $G' = G + x$ un graphe de permutation et soit $R' = (\pi'_1, \pi'_2)$ un réalisateur de G' . La restriction $R'[V]$ est un réalisateur de G qui, par définition d'un réalisateur, satisfait la condition du lemme. Réciproquement, si pour un réalisateur R de G , il existe i et j qui satisfont la condition du lemme, alors R peut clairement être étendu à un réalisateur de G' comme décrit dans le lemme. ■

Définition 3.2 Un intervalle commun initial d'un réalisateur $R = (\pi_1, \pi_2)$ est un intervalle commun de R (définition 1.42 page 84) qui contient $\pi_1^{-1}(1)$ et $\pi_2^{-1}(1)$. Par convention, l'ensemble vide est aussi un intervalle commun initial.

Remarquons que le nombre d'intervalles communs initiaux d'un réalisateur est $O(n)$. Ils jouent un rôle important dans la détection des positions d'insertion pour x .

Remarque 3.1 Soit $(i, j) \in \llbracket 0, n \rrbracket^2$. Les ensembles $\overline{N}_1(x) = \{v \in \overline{N}(x) \mid \pi_1(v) \leq i\}$ et $N_1(x) = \{u \in N(x) \mid \pi_1(u) \leq i\}$ sont respectivement des intervalles communs initiaux de $R[\overline{N}(x)]$ et $R[N(x)]$ ssi i et j satisfont la condition du lemme 3.1.

Notation 3.1 (voir figure 3.2) Soit $R = (\pi_1, \pi_2)$ un réalisateur d'un graphe de permutation. Soit I un intervalle commun initial de $R[\overline{N}(x)] = (\pi_1[\overline{N}(x)], \pi_2[\overline{N}(x)])$. On note $\bar{a}_1$ la borne droite de I dans $\pi_1[\overline{N}(x)]$ et $\bar{b}_1$ le successeur de $\bar{a}_1$ dans $\pi_1[\overline{N}(x)]$. $\bar{a}_2$ désigne la borne droite de I dans $\pi_2[\overline{N}(x)]$ et $\bar{b}_2$ le successeur de $\bar{a}_2$ dans $\pi_2[\overline{N}(x)]$. Soit J un intervalle commun initial de $R[N(x)] = (\pi_1[N(x)], \pi_2[N(x)])$. On note a_1 la borne droite de J dans $\pi_1[N(x)]$ et b_1 le successeur de a_1 dans $\pi_1[N(x)]$. b_2 désigne la borne droite de J dans $\pi_2[N(x)]$ et a_2 le successeur de b_2 dans $\pi_2[N(x)]$.

Si $I = \emptyset$, $\bar{a}_1$ est indéfini, et si $I = \overline{N}(x)$, $\bar{b}_1$ est indéfini. De façon similaire, $\bar{a}_2$ et $\bar{b}_2$ peuvent être indéfinis. Ces cas particuliers peuvent être évités en ajoutant des sommets fictifs y_0 et y_{n+1} respectivement aux positions 0 et $n+1$ dans R , qui sont considérés comme appartenant à la fois à $N(x)$ et à $\overline{N}(x)$. Un intervalle commun initial I de $R[\overline{N}(x)]$ devient $I \cup \{y_0\}$. Lorsque I est vide, $\bar{a}_1 = y_0$, et lorsque $I = \overline{N}(x)$, $\bar{b}_1 = y_{n+1}$; et de même pour les intervalles communs initiaux J de $R[N(x)]$.

Cela nous permet de ne pas avoir à considérer dans la suite les cas dans lesquels $I = \emptyset$ ou $I = \overline{N}(x)$, de même que les cas dans lesquels $J = \emptyset$ ou $J = N(x)$.

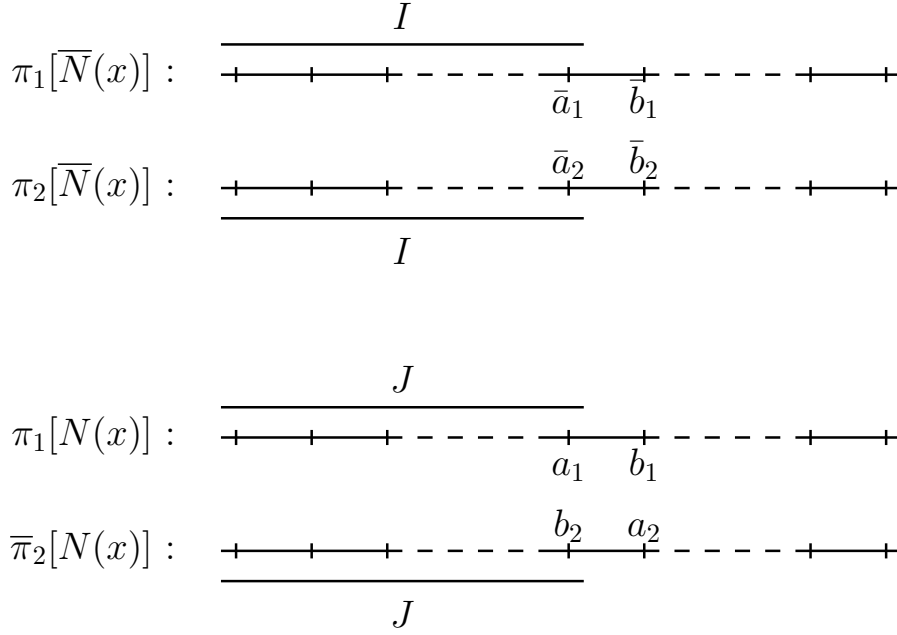


FIG. 3.2 – Illustration de la notation 3.1.

Remarque 3.2 Par définition, des intervalles $[\bar{a}_1^i, \bar{b}_1^i]$ différents correspondant à des intervalles communs initiaux I_i de $R[\bar{N}(x)]$ différents ne peuvent s'intersecter que sur leurs bornes. Cela est vrai également pour des intervalles $[\bar{a}_2^i, \bar{b}_2^i]$ différents, de même que pour des intervalles $[a_1^j, b_1^j]$ différents et des $[a_2^j, b_2^j]$ différents qui correspondent à des intervalles communs initiaux J_j de $\bar{R}[N(x)]$ différents.

Lemme 3.2 Soit $R = (\pi_1, \pi_2)$ un réalisateur d'un graphe de permutation. Soient I et J des intervalles communs initiaux de respectivement $R[\bar{N}(x)]$ et $\bar{R}[N(x)]$. Si, dans π_1 , $[\bar{a}_1, \bar{b}_1]$ et $[a_1, b_1]$ s'intersectent, alors $|\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]| = 2$. Si, dans π_2 , $[\bar{a}_2, \bar{b}_2]$ et $[a_2, b_2]$ s'intersectent, alors $|\bar{a}_2, \bar{b}_2] \cap [a_2, b_2]| = 2$.

Preuve : On donne la preuve pour π_1 , celle pour π_2 est similaire. Comme $\{\bar{a}_1, \bar{b}_1\} \subseteq \bar{N}(x)$ et $\{a_1, b_1\} \subseteq N(x)$, alors $[\bar{a}_1, \bar{b}_1]$ et $[a_1, b_1]$ ne peuvent pas partager de borne. Par conséquent, $|\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]| \geq 2$. Sans perte de généralité, on peut supposer que $a_1 <_{\pi_1} \bar{a}_1$. Comme $[\bar{a}_1, \bar{b}_1]$ et $[a_1, b_1]$ s'intersectent, alors $\bar{a}_1 <_{\pi_1} b_1$. Si le successeur de $\bar{a}_1$ dans π_1 est un non voisin de x , alors, par définition, c'est $\bar{b}_1$. Donc, $|\bar{a}_1, \bar{b}_1]| = 2$ et la preuve est terminée. Sinon, si le successeur s de $\bar{a}_1$ dans π_1 est un voisin de x , alors, comme $\bar{a}_1 <_{\pi_1} b_1$ il s'ensuit que $s \leq_{\pi_1} b_1$. Comme b_1 est le successeur de a_1 dans π_1 restreint aux voisins de x , alors $s = b_1$. Donc, $|\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]| = 2$. ■

Le corollaire 3.1 reformule le lemme 3.1 en tenant compte de la discussion ci-dessus.

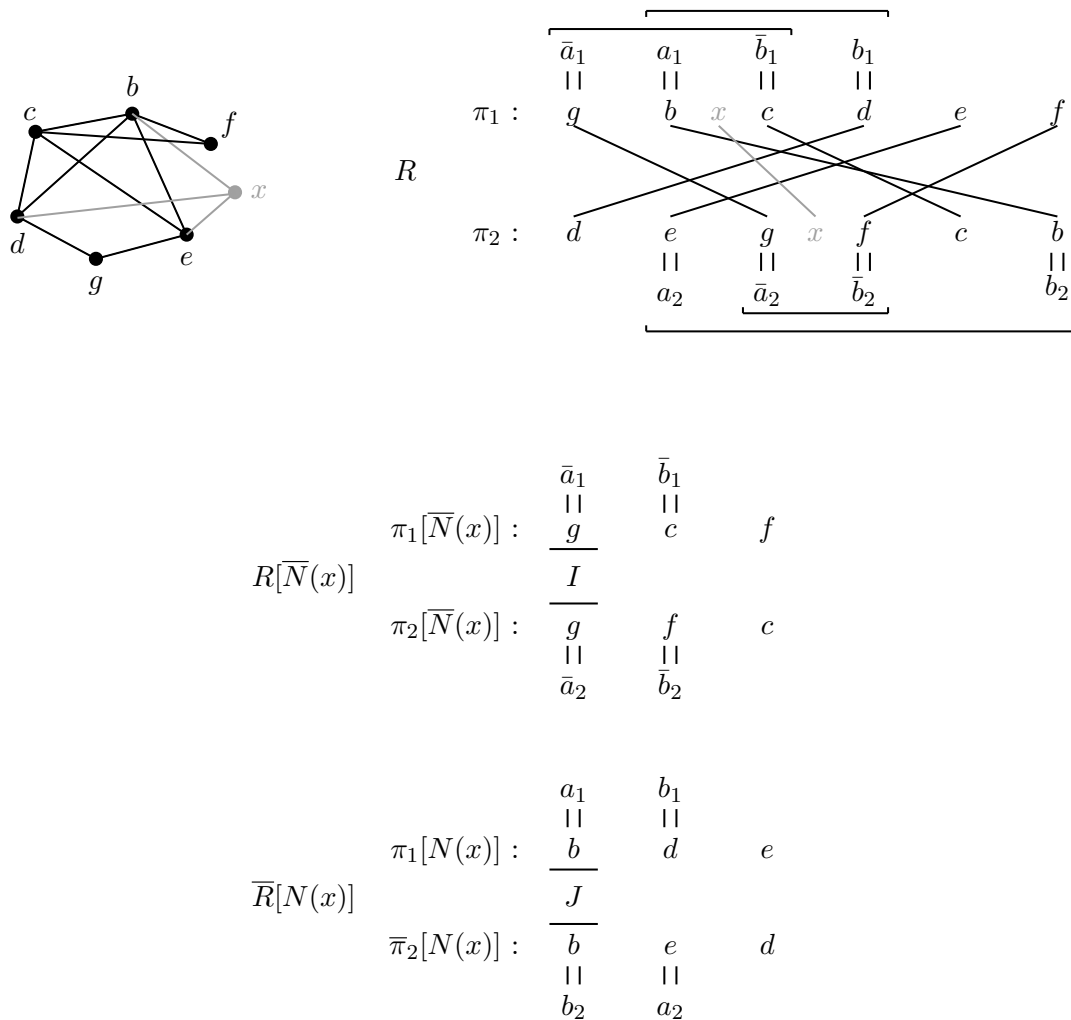


FIG. 3.3 – Illustration du corollaire 3.1.

Corollaire 3.1 (voir figure 3.3) Soit $R = (\pi_1, \pi_2)$ un réalisateur d'un graphe de permutation et x un sommet à insérer. R est extensible à x ssi il existe I et J des intervalles communs initiaux de respectivement $R[\overline{N}(x)]$ et $\overline{R}[N(x)]$, tels que $[\bar{a}_1, \bar{b}_1]$ et $[a_1, b_1]$ s'intersectent dans π_1 et $[\bar{a}_2, \bar{b}_2]$ et $[a_2, b_2]$ s'intersectent dans π_2 . Et si de tels I, J existent, on obtient un réalisateur de $G + x$ en insérant x entre les deux éléments de $[\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]$ dans π_1 , et entre les deux éléments de $[\bar{a}_2, \bar{b}_2] \cap [a_2, b_2]$ dans π_2 .

Preuve : Si R est extensible à x , il existe i, j qui satisfont la condition du lemme 3.1. Alors, $I = \overline{N}_1(x) = \{v \in \overline{N}(x) \mid \pi_1(v) \leq i\}$ et $J = N_1(x) = \{u \in N(x) \mid \pi_1(u) \leq i\}$ sont respectivement des intervalles communs initiaux de $R[\overline{N}(x)]$ et $\overline{R}[N(x)]$. D'après la notation 3.1 et la définition de $\overline{N}_1(x)$ et $N_1(x)$, il découle que $[\bar{a}_1, \bar{b}_1]$ et $[a_1, b_1]$ contiennent tous deux $\{\pi_1^{-1}(i), \pi_1^{-1}(i+1)\}$, et donc s'intersectent. De même, $[\bar{a}_2, \bar{b}_2]$ et $[a_2, b_2]$ s'intersectent. Réciproquement, si il existe I, J qui satisfont la condition du corollaire, alors d'après le lemme 3.2, $||[\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]| = 2$ et $||[\bar{a}_2, \bar{b}_2] \cap [a_2, b_2]| = 2$. Il n'est pas difficile de voir qu'insérer x entre les deux sommets de $[\bar{a}_1, \bar{b}_1] \cap [a_1, b_1]$ dans π_1 , et entre les deux sommets de $[\bar{a}_2, \bar{b}_2] \cap [a_2, b_2]$ dans π_2 , produit les adjacences correctes à la fois entre x et son voisinage et entre x et son non voisinage. ■

Remarque 3.3 Il y a une bijection entre l'ensemble des couples (I, J) d'intervalles communs initiaux de respectivement $R[\overline{N}(x)]$ et $\overline{R}[N(x)]$ qui vérifient les conditions du corollaire 3.1 et l'ensemble des positions d'insertion pour x dans R .

Il est facile de concevoir un algorithme qui fournit tous les intervalles communs initiaux en temps $O(l)$, où l est le nombre d'éléments du réalisateur.

La routine *InsPrime* comporte trois étapes.

1. Extraire de $R = (\pi_1, \pi_2)$, les réalisateurs $R[\overline{N}(x)]$ et $\overline{R}[N(x)]$ et calculer leurs intervalles communs initiaux.
2. Pour chaque intervalle commun initial I de $R[\overline{N}(x)]$, repérer dans π_1 l'intervalle d'insertion correspondant parmi les non voisins de x : i.e. associer à $\bar{a}_1$ (resp. $\bar{b}_1$), un identifiant pour I , des pointeurs vers $\bar{a}_2$ et $\bar{b}_2$ et un pointeur vers $\bar{b}_1$ (resp. $\bar{a}_1$). Procéder de la même façon pour les intervalles communs initiaux J de $\overline{R}[N(x)]$.
3. Parcourir π_1 en cherchant les intersections d'un intervalle de type $[\bar{a}_1^i, \bar{b}_1^i]$ avec un intervalle $[a_1^j, b_1^j]$. Lorsqu'une telle intersection est trouvée, vérifier si les $[\bar{a}_2^i, \bar{b}_2^i]$ et $[a_2^j, b_2^j]$ correspondant s'intersectent dans π_2 . Si c'est le cas, renvoyer la position d'insertion pour x dans R . Continuer à parcourir π_1 .

Repérer l'intervalle d'insertion à l'étape 2 peut se faire en temps constant pour chaque intervalle commun initial. Le test d'intersection à l'étape 3 coûte $O(1)$. Comme les intervalles d'insertion parmi les non voisins de x ne s'intersectent pas, hormis sur leurs bornes, de même que les intervalles d'insertion parmi les voisins de x , alors le parcours de π_1 décrit ci-dessus requiert un temps $O(n)$. Ainsi, la routine *InsPrime* appliquée sur un réalisateur

de graphe de permutation détermine si celui-ci est extensible et renvoie toutes les positions d'insertions possibles pour x , en temps $O(n)$.

Application de *InsPrime* sur le réalisateur d'un graphe de permutation premier.

Considérons le cas particulier d'application de la routine *InsPrime* sur le réalisateur d'un graphe de permutation premier.

Théorème 3.1 *Soit $G = (V, E)$ un graphe de permutation premier et $R = (\pi_1, \pi_2)$ son réalisateur. Soit x un sommet à insérer dans G tel que V n'est pas uniforme rel. à x . Il y a au plus deux positions d'insertion pour x dans R . De plus, il y a deux positions d'insertion différentes ssi x a un jumeau dans $G + x$.*

Preuve : Considérons les deux positions d'insertion différentes pour x dans R . Effectuons simultanément les deux insertions correspondantes de x_1 et x_2 dans R (x_1 et x_2 représentent x dans les deux positions d'insertion possibles). Soit $R_{ins} = (\pi_1^{ins}, \pi_2^{ins})$ le réalisateur obtenu. Soit $a \in V$ un sommet se trouvant entre x_1 et x_2 dans un des deux ordres de R_{ins} . Alors, a est entre x_1 et x_2 dans l'autre ordre de R_{ins} . Sinon, a ne serait pas lié de la même façon à x_1 et x_2 ce qui est une contradiction avec le fait que x_1 et x_2 représentent tous deux le sommet x . Par conséquent, l'ensemble B des sommets se trouvant entre x_1 et x_2 dans π_1^{ins} est le même que l'ensemble de ceux se trouvant entre x_1 et x_2 dans π_2^{ins} . B est un intervalle commun de R , donc B est un module de G qui est premier. Le fait que V soit mixte implique que $B \neq V$. Il s'ensuit que $|B| = 1$. Cela montre que deux positions d'insertion ne peuvent pas être séparées par plus d'un sommet dans chacun des deux ordres du réalisateur. Par conséquent, il y a au plus deux positions d'insertion pour x dans R .

On note a l'unique élément de B . Pour chacune des deux positions d'insertion possibles pour x , $\{a, x\}$ est un intervalle du réalisateur R' de $G' = G + x$ correspondant. Donc, $\{a, x\}$ est un module de $G + x$ et a est un jumeau de x dans $G + x$. Réciproquement, si x a un jumeau a dans G' , alors $\{a, x\}$ est un module fort de G' . Dans tout réalisateur de G' , $\{a, x\}$ est un intervalle commun. En fait, comme G est premier, $\{a, x\}$ est l'unique module de G' . D'où G' admet exactement deux réalisateurs, chacun d'entre eux s'obtenant à partir de l'autre en interchangeant les positions de a et x dans les deux ordres. La restriction d'un quelconque de ces deux réalisateurs de G' donne l'unique réalisateur de G . On en déduit que x a deux positions d'insertion différentes dans R . ■

D'après le théorème 3.1, appliquée au réalisateur R d'un graphe de permutation premier, la routine *InsPrime* trouve au plus deux positions d'insertion possibles pour x dans R . D'après le corollaire 3.1, et comme le réalisateur d'un graphe premier est unique, si aucune position d'insertion n'est trouvée, alors $G + x$ n'est pas un graphe de permutation. Si on trouve exactement une position d'insertion, alors $G + x$ est un graphe de permutation et il existe une unique façon d'insérer x dans R , $G + x$ est donc également premier. Si on trouve deux positions d'insertion, alors $G + x$ est un graphe de permutation, et d'après le théorème 3.1 x a un jumeau a dans $G + x$, qui peut être trouvé en temps constant.

Pour résumer, lorsqu'elle est appliquée au réalisateur d'un graphe premier, la routine *InsPrime* détermine si $G + x$ est un graphe de permutation, en temps $O(n)$, et si c'est le cas elle retourne, dans la même complexité, une paire de listes doublement chaînées formant un réalisateur de $G + x$, ainsi que le jumeau de x s'il existe.

3.2 Les graphes d'intervalles

Dans cette section, le but recherché est celui de l'entretien dynamique d'un modèle d'intervalles minimal lors d'une modification de sommet. Nous n'étudions pas ici ce problème dans sa généralité, ce qui sera fait au chapitre 5, mais nous considérons une version restreinte du problème qui s'exprime comme suit. La donnée est un modèle d'intervalles minimal avec une modification du graphe qui peut être l'ajout ou le retrait d'un sommet et la question posée est : le modèle d'intervalle minimal donné peut-il être restreint ou étendu pour obtenir un modèle minimal du nouveau graphe ? Précisons ce que nous entendons par restriction et extension d'un modèle d'intervalles minimal.

Définition 3.3 Soit $G = (V, E)$ un graphe d'intervalles et soit $S \subseteq V$. Soient σ un ordre consécutif de $\mathcal{K}(G)$ et σ' un ordre consécutif de $\mathcal{K}(G[S])$. On dit que σ' est la **restriction** de σ à S si σ' s'obtient de σ en retirant les sommets de S des cliques maximales de G puis en retirant de σ les cliques qui ne sont plus maximales. Réciproquement, on dit que σ' est une **extension** de σ si σ est une restriction de σ' . Pour les modèles d'intervalles minimaux, le vocabulaire restriction et extension se rapporte à ces notions pour les ordres consécutifs sous-jacents.

Avec cette définition on peut donner une formulation plus rigoureuse du problème que l'on considère ici.

Problème.

Donnée : Un graphe d'intervalles avec un de ses modèles d'intervalles minimaux $\mathcal{I}$ et une insertion ou suppression de sommet dans le graphe.

Question : Calculer, si possible, un modèle d'intervalles minimal du graphe modifié qui soit une extension ou restriction de $\mathcal{I}$, renvoyer faux si cela n'est pas possible.

Il y a une dissymétrie totale entre le cas de l'ajout et celui de la suppression d'un sommet. En effet, toute restriction d'un modèle d'intervalles de G à un sous-ensemble de sommets S résulte en un modèle minimal de $G[S]$. En revanche, il n'est pas toujours possible lors de l'ajout d'un sommet x d'étendre un modèle d'intervalles minimal donné pour obtenir un modèle d'intervalles minimal de $G + x$, même lorsque $G + x$ est un graphe d'intervalles. C'est ce qui nous poussera, au chapitre 5, à considérer tous les modèles d'intervalles minimaux possibles pour G , car il en existe toujours un que l'on peut étendre, lorsque $G + x$ est un graphe d'intervalles. Cette dissymétrie peut s'expliquer par le fait que

lorsqu'on retire un sommet, on retire une contrainte : le modèle précédent qui satisfaisait plus de contraintes satisfera à fortiori les nouvelles. Dans l'autre sens, lorsqu'on ajoute un sommet, on ajoute une contrainte que ne respectait pas forcément le modèle précédent. Nous cherchons donc maintenant à déterminer les cas dans lesquels le modèle minimal donné peut être étendu lors de l'ajout du sommet x .

3.2.1 Insertion d'un sommet dans un modèle d'intervalles

Certaines notions et notations introduites ici sont définies ailleurs dans le mémoire, au chapitre 5. Le but recherché est l'indépendance de la section courante. Que le lecteur se rassure, cela n'entraînera pas de conflit de vocabulaire ou de notation. Des notions ou notations identiques désignent bien les mêmes concepts ou mêmes objets tout au long du mémoire.

Définition 3.4 Soit G un graphe d'intervalles, soit $K \in \mathcal{K}(G)$ et soit x un sommet à insérer dans G . K est dite **pleine** si tous les sommets de K sont adjacents à x .

Notation 3.2 Soit $G = (V, E)$ un graphe d'intervalles, soit σ un ordre consécutif de $\mathcal{K}(G)$ et x un sommet à insérer dans G . On note $\mathcal{P}_G$ l'ensemble des cliques maximales de G qui sont pleines. Lorsque $\mathcal{P}_G \neq \emptyset$, on note F_σ (resp. L_σ), si elle existe, la clique maximale de G précédant la première (resp. suivant la dernière) clique de $\mathcal{P}_G$ dans σ . Pour un sommet $y \in V$, on note K_y^1 (resp. K_y^2) la première (resp. dernière) clique maximale de G contenant y dans σ .

Définition 3.5 Soit $G = (V, E)$ un graphe d'intervalles, soit σ un ordre consécutif de $\mathcal{K}(G)$ et x un sommet à insérer dans G . Soit $K \in \mathcal{K}(G)$, K vérifie la **propriété gauche** (resp. **droite**) si pour tout sommet $y \in N(x)$, on a $K_y^2 \geq_\sigma K$ (resp. $K_y^1 \leq_\sigma K$). K vérifie la **propriété gauche (resp. droite) stricte** si K vérifie la propriété gauche (resp. droite) et il existe $y \in N(x)$ tel que $K_y^2 = K$ (resp. $K_y^1 = K$).

Théorème 3.2 Soit G un graphe d'intervalles, soit σ un ordre consécutif de $\mathcal{K}(G)$ et x un sommet à insérer dans G . Il existe une extension de σ à $G + x$ ssi une des deux conditions suivantes est vérifiée :

1. $\mathcal{P}_G \neq \emptyset$ est un intervalle de σ et F_σ et L_σ satisfont respectivement les propriétés gauche et droite ; ou
2. $\mathcal{P}_G = \emptyset$ et il existe $K_f, K_l \in \mathcal{K}(G)$ telles que K_f est le prédécesseur de K_l dans σ , et K_f et K_l satisfont respectivement la propriété gauche et droite et $K_f \cap K_l \subseteq N(x)$.

Preuve : $\Rightarrow$.

Comme σ est extensible, alors σ est la restriction d'un ordre consécutif σ' de $\mathcal{K}(G + x)$. On note C_x^1 et C_x^2 respectivement la première et dernière clique de σ' contenant x . σ est formé des cliques restant maximales après retrait de x . Nous analyserons plus loin dans le document (section 5.2.5) les conséquences du retrait d'un sommet x dans un ordre consécutif.

En particulier, le lemme 5.14 page 206 montre que les deux seules cliques pouvant disparaître de σ' lors de la suppression de x sont C_x^1 et C_x^2 . Par conséquent, $\mathcal{P}_G$ contient tous les éléments de l'ensemble $\{K \setminus \{x\} \mid C_x^1 <_{\sigma'} K <_{\sigma'} C_x^2\}$, plus éventuellement $C_x^1 \setminus \{x\}$ et $C_x^2 \setminus \{x\}$, et ne contient aucune autre clique.

Ceci montre que si $\mathcal{P}_G$ est non vide, $\mathcal{P}_G$ est un intervalle de σ . On note K_1 le prédécesseur de C_x^1 dans σ' et K_2 le successeur de C_x^2 . On a alors $F_\sigma = K_1$ et $L_\sigma = K_2$. Comme pour tout sommet y adjacent à x , la dernière clique contenant y dans σ' est supérieure ou égale à C_x^1 , alors la dernière clique E_y^1 contenant y dans σ est telle que $E_y^1 \geq_\sigma F_\sigma$. Par conséquent, F_σ vérifie la propriété gauche. Par symétrie, L_σ vérifie la propriété droite. Ainsi, la condition 1 est satisfaite.

Si $\mathcal{P}_G$ est vide, on pose $K_f = K_1$ et $K_l = K_2$. Comme $\mathcal{P}_G$ est vide, C_x^1 et C_x^2 sont consécutifs dans σ' et K_l est le successeur de K_f dans σ . D'après ce qui précède, K_f et K_l satisfont respectivement la propriété gauche et droite. Si $\exists y \in K_1 \cap K_2$, alors comme σ' est un ordre consécutif, $y \in C_x^1$ et donc y est adjacent à x . Ainsi, la condition 2 est satisfaite.

$\Leftarrow$.

Si σ vérifie une des deux conditions du théorème, on peut construire un ordre consécutif σ' de $\mathcal{K}(G + x)$ qui étend σ . On étudie les deux cas séparément.

1. Si $F_\sigma \cap N(x) \neq \emptyset$, pour obtenir σ' , on ajoute la clique maximale $(F_\sigma \cap N(x)) \cup \{x\}$ entre F_σ et sa suivante dans σ . De même, si $L_\sigma \cap N(x) \neq \emptyset$, pour obtenir σ' , on ajoute la clique maximale $(L_\sigma \cap N(x)) \cup \{x\}$ entre L_σ et sa précédente dans σ . Enfin, on ajoute x dans toutes les cliques de $\mathcal{P}_G$. Il est clair qu'ainsi on obtient exactement les cliques maximales de $G + x$. Il faut vérifier que les contraintes de consécuité imposées par les sommets de $F_\sigma \setminus N(x)$ ne sont pas violées par l'insertion de la clique $(F_\sigma \cap N(x)) \cup \{x\}$. Ces contraintes sont bien respectées car la suivante de F_σ dans σ appartient à $\mathcal{P}_G$: elle ne contient que des sommets adjacents à x et par conséquent ne contient aucun sommet de $F_\sigma \setminus N(x)$. Pour les mêmes raisons, les contraintes de consécuité imposées par les sommets de $L_\sigma \setminus N(x)$ sont respectées. σ' est donc bien un ordre consécutif de $\mathcal{K}(G + x)$.
2. Dans ce cas, on procède comme dans le cas précédent en faisant jouer à K_f et K_l le rôle de F_σ et L_σ . Il convient de nouveau de s'assurer que l'insertion de la clique $(K_f \cap N(x)) \cup \{x\}$ ne menace pas les contraintes de consécuité imposées par les sommets de $K_f \setminus N(x)$. Ces contraintes sont respectées car la suivante de K_f dans σ est K_l . D'après la condition 2 du théorème, $K_f \cap K_l$ ne contient aucun sommet qui soit non adjacent à x , et donc K_l ne contient aucun sommet de $K_f \setminus N(x)$. σ' est donc bien un ordre consécutif de $\mathcal{K}(G + x)$.

Dans les deux cas, il est immédiat de voir que la restriction de σ' aux sommets de V est précisément σ . ■

3.2.2 Algorithme et complexité

La difficulté est de tester si σ vérifie une des deux conditions du théorème 3.2 et de déterminer le couple (F_σ, L_σ) ou le couple (K_f, K_l) en temps $O(n)$. Ceci fait, rajouter dans σ les cliques nécessaires pour obtenir σ' dans la même complexité ne pose aucun problème.

L'implémentation d'un modèle minimal d'intervalles utilisée dans cet algorithme est celle décrite en section 5.3.1. Cette représentation s'appuie sur une liste doublement chaînée dont chaque cellule est une clique maximale, l'ordre des cellules étant celui de l'ordre consécutif σ choisi. Chaque cellule contient son rang dans la liste. Chaque sommet y du graphe possède deux pointeurs vers les deux cellules (éventuellement identiques), notées K_y^1 et K_y^2 , qui sont respectivement la première et la dernière clique maximale de σ contenant y . Lorsque $y \in K$, on dit aussi que y **couvre** K .

La procédure *PartNonCouv*($\pi, \mathcal{I}$)

Définition 3.6 On appelle **partition intervallaire** d'un sous-ensemble S d'un ensemble E muni d'un ordre total, l'unique partition de S dont le nombre d'éléments est minimal et dont tous sont des intervalles de E .

Notation 3.3 Pour un ordre π sur un ensemble sous-jacent E , et pour un ensemble $\mathcal{I}$ d'intervalle de π , on note $\mathcal{NC}(\pi, \mathcal{I}) = \{x \in E \mid \forall I \in \mathcal{I}, x \notin I\}$.

La procédure *PartNonCouv* prend deux paramètres : π qui est un ordre linéaire sur un ensemble fini d'éléments E , et $\mathcal{I}$ qui est un ensemble d'intervalles de π . Et elle retourne la partition intervallaire de $\mathcal{NC}(\pi, \mathcal{I})$.

On donne ci-dessous la description de cette procédure. On commence par trier les intervalles I de $\mathcal{I}$ par ordre de $f(I)$ croissant. Comme les $f(I)$ sont compris entre 1 et $|E|$, cela requiert un temps $O(|E| + |\mathcal{I}|)$. On note L l'ordre ainsi obtenu.

En parcourant les $I \in \mathcal{I}$ à $f(I)$ croissant, on entretient une partition intervallaire de l'ensemble des éléments de E qui ne sont présents dans aucun des $I \in \mathcal{I}$ examinés jusque là. On note $\mathcal{P}_i$ cette partition après examen des i premiers sommets de L .

Initialement, la partition $\mathcal{P}_0$ ne contient qu'un intervalle qui est π lui même. A tout moment, $\mathcal{P}_i$ est stockée comme une liste dont les éléments sont des paires de pointeurs sur $f(J)$ et $l(J)$ pour chaque intervalle J de la partition. Cette liste est rangée par ordre croissant des $f(J)$ (et donc des $l(J)$ car les intervalles ne s'intersectent pas). L'algorithme de parcours s'arrête lorsque tous les intervalles de $\mathcal{I}$ ont été examinés ou lorsque le dernier élément de π n'est plus présent dans $\mathcal{P}_i$. Ainsi, lorsque l'algorithme s'arrête, $\mathcal{P}_i$ est la partition intervallaire de $\mathcal{NC}(\pi, \mathcal{I})$. On montre ci-dessous que, lors de l'examen du $i+1^{\text{ème}}$ intervalle $I \in \mathcal{I}$, mettre à jour $\mathcal{P}_i$ pour obtenir $\mathcal{P}_{i+1}$ ne demande qu'un temps constant. Ainsi, le parcours de $\mathcal{I}$ demande un temps total de $O(|\mathcal{I}|)$.

On fait usage des deux invariants immédiats suivants. Lors du parcours de L , juste avant d'examiner un intervalle I :

1. le dernier élément de π appartient au dernier intervalle de $\mathcal{P}_i$; et

2. $f(I)$ est strictement plus grand dans π que tous les éléments des intervalles de $\mathcal{P}_i$ qui ne sont pas dans le dernier intervalle de $\mathcal{P}_i$.

Cette dernière propriété vient du fait qu'on examine les intervalles $I \in \mathcal{I}$ à $f(I)$ croissant. Ainsi, en examinant I , on est sûr que les seuls éléments de $\mathcal{P}_i$ qu'il contient, s'il en contient, se trouvent dans le dernier intervalle de $\mathcal{P}_i$. On peut restreindre cet intervalle aux éléments de E strictement supérieures à $l(I)$ dans π en temps constant.

Le temps total d'exécution de la procédure *PartNonCouv* est donc $O(|E| + |\mathcal{I}|)$.

Test de la condition 1 du théorème 3.2

Dans ce test, le plus difficile est de vérifier que $\mathcal{P}_G$ forme un intervalle de σ . Par définition, $\mathcal{P}_G = \{K \in \mathcal{K}(G) \mid \forall y \in \overline{N}(x), y \notin K\}$. Donc, en posant $\mathcal{I} = \{\llbracket K_y^1, K_y^2 \rrbracket \mid y \in \overline{N}(x)\}$, on a $\mathcal{P}_G = \mathcal{NC}(\sigma, \mathcal{I})$. Ainsi, en utilisant la procédure *PartNonCouv*, on obtient la partition intervallaire de $\mathcal{P}_G$. Si cette partition contient strictement plus d'un intervalle, u ne vérifie pas la condition 1. Si elle en contient exactement un, alors on détermine F_σ et L_σ qui sont simplement les deux sommets précédant et suivant cet intervalle. Il reste à vérifier que F_σ et L_σ satisfont respectivement la propriété gauche et droite, ce qui est très facile en examinant les pointeurs de tous les sommets de G , cela prend un temps $O(n)$. L'exécution de *PartNonCouv* prend un temps $O(|\mathcal{K}(G)| + |\mathcal{I}|)$. Comme $|\mathcal{K}(G)| = O(n)$ et $|\mathcal{I}| = O(n)$, l'appel à la procédure *PartNonCouv* coûte un temps $O(n)$.

On peut donc tester si σ satisfait la condition 1 du théorème 3.2 et déterminer, le cas échéant, F_σ et L_σ en temps $O(n)$.

Test de la condition 2 du théorème 3.2

Remarquons d'abord que si une clique maximale K_1 vérifie la propriété gauche, alors $\forall K \in \mathcal{K}(G), K \leq_\sigma K_1 \Rightarrow K$ vérifie la propriété gauche. De plus, si K_1 vérifie la propriété gauche stricte (voir définition 5.9), alors $\forall K \in \mathcal{K}(G), K_1 <_\sigma K \Rightarrow K$ ne vérifie pas la propriété gauche. Par symétrie, on a les assertions correspondantes pour la propriété droite. On note $C_f = \min\{K_y^2 \mid y \in N(x)\}$, et $C_l = \max\{K_y^1 \mid y \in N(x)\}$. D'après ce qui précède, l'ensemble des cliques maximales de G vérifiant la propriété gauche est $\{K \in \mathcal{K}(G) \mid K \leq_\sigma C_f\}$ et l'ensemble des cliques maximales de G vérifiant la propriété droite est $\{K \in \mathcal{K}(G) \mid C_l \leq_\sigma K\}$. Trouver C_f et C_l se fait en parcourant les sommets de G en temps $O(n)$.

- Examinons le cas où $C_f < C_l$. Si C_f n'est pas le prédécesseur de C_l , alors σ ne satisfait pas la condition 2. Si au contraire C_f est le prédécesseur de C_l , alors on teste si σ satisfait la condition 3(c)ii en parcourant les sommets de G et en vérifiant qu'aucun sommet de $\overline{N}(x)$ ne couvre C_f et C_l simultanément.
- Dans le cas où $C_l \leq C_f$, il peut y avoir plusieurs couples de sommets successeurs prétendant à être le couple (K_f, K_l) . On note k la longueur de l'intervalle $\llbracket C_l, C_f \rrbracket$ et $\{K_i\}_{1 \leq i \leq k} = \llbracket C_l, C_f \rrbracket$, avec $C_l = K_1 <_\sigma K_2 <_\sigma \dots <_\sigma K_k = C_f$. Lorsqu'ils existent, on note K_0 le prédécesseur de C_l et K_{k+1} le successeur de C_f . Les couples

possibles pour (K_f, K_l) sont donc les couples (K_i, K_{i+1}) pour $0 \leq i \leq k$. σ satisfait la condition 2 ssi au moins un de ces couples n'est entièrement couvert par aucun sommet de $\overline{N}(x)$. Ainsi, en posant $\mathcal{I} = \{\llbracket K_y^1, \text{pred}(K_y^2) \rrbracket \mid y \in \overline{N}(x)\}$, la condition 2 est satisfaite ssi $\mathcal{NC}(\llbracket K_0, K_k \rrbracket, \mathcal{I}) \neq \emptyset$. Un appel à *PartNonCouv*($\llbracket K_0, K_k \rrbracket, \mathcal{I}$) détermine la partition intervallaire $\mathcal{NC}(\llbracket K_0, K_k \rrbracket, \mathcal{I})$ en temps $O(k + |\mathcal{I}|) = O(n)$. Si $\mathcal{NC}(\llbracket K_0, K_k \rrbracket, \mathcal{I}) = \emptyset$, la condition 2 n'est pas satisfaite. Sinon, la condition 2 est satisfaite et n'importe quel couple formé d'un élément de $\mathcal{NC}(\llbracket K_0, K_k \rrbracket, \mathcal{I})$ et de son suivant dans σ convient pour (K_f, K_l) .

En conclusion, on peut tester si σ satisfait la condition 2 et trouver un couple (K_f, K_l) le cas échéant en temps $O(n)$.

3.3 L'exemple des graphes planaires

L'histoire des algorithmes dynamiques pour les graphes planaires illustre assez bien la problématique de ce mémoire et introduit les idées générales que nous mettrons en oeuvre sur les graphes de permutation et d'intervalles.

Beaucoup d'algorithmes dynamiques pour les graphes planaires fonctionnent sur ce que l'on appelle les graphes plan. Les graphes plan sont des graphes planaires dont les sommets et les arêtes sont plongés dans le plan (sans croisement d'arêtes). Autrement dit, ce sont des graphes planaires pour lesquels on a fixé un dessin planaire. Différents problèmes dynamiques ont été étudiés sur les graphes plans [EIT⁺92, GI91, GIS92, GI96, HRS94, Tam88] parmi lesquels [GIS92, Tam88] traitent de la reconnaissance dynamique. Ces algorithmes ne traitent que les modifications élémentaires telles que le graphe G' obtenu admette un dessin planaire qu'on peut obtenir à partir du dessin du graphe initial G sans modifier le dessin de la partie du graphe qui n'est pas modifiée. C'est à dire qu'on doit pouvoir obtenir un dessin planaire de G' sans modifier les dessins des sommets et arêtes de G qui ne sont pas touchés par la modification. Si cette condition n'est pas remplie, l'algorithme ne peut pas traiter la modification.

Lors du retrait d'une arête ou d'un sommet, cette condition est toujours remplie, par contre, lors d'un ajout, cette condition peut être violée alors que le graphe G' est planaire.

Ainsi donc, les algorithmes dynamiques sur les graphes plans présentent l'inconvénient majeur de ne pas pouvoir traiter toutes les modifications du graphe qui le laissent planaire. Le fait de travailler sur un dessin planaire du graphe paraît indispensable pour pouvoir exploiter les propriétés particulières à la classe. Mais pour pouvoir traiter toutes les modifications, il faudrait considérer tous les dessins possibles du graphe et cela entraînerait une perte d'efficacité rédhibitoire.

Pour la question de la reconnaissance dynamique, plusieurs algorithmes ont dépassé ce problème et sont capables de traiter toutes les modifications élémentaires résultant en un graphe planaire.

Il y a deux approches différentes. La première est celle de [EGIS98, GIS99], elle consiste à découper le graphe en sous-graphes peu denses (ayant peu d'arêtes par rapport au nombre de sommets). Les morceaux étant peu denses, ils peuvent être traités efficacement. Pour le

recollement entre les différents morceaux, il faut considérer toutes les façons possibles. Les découpages sont ainsi faits que ces recollements peuvent être traités de manière efficace. Les algorithmes de [EGIS98] et [GIS99] sont entièrement dynamiques : ils permettent de traiter les opérations de suppression et d'insertion d'une arête en temps $O(n^{2/3})$ pour [GIS99] et $O(n^{1/2})$ pour [EGIS98].

La deuxième approche est celle de [BT96], dont l'algorithme n'est qu'incrémental. C'est pourtant cette approche qui sera de plus grand intérêt pour ce mémoire. Dans [EGIS98, GIS99], les découpages utilisés et les recollements qui en découlent ne sont pas fortement liés à la structure des plongements planaires du graphe. Les découpages visent uniquement à "creuser" le graphe. En revanche, l'algorithme de [BT96] est basé sur une décomposition du graphe (*SPQR*-arbre) dont le but est de représenter tous les plongements planaires du graphe. Cette décomposition est directement inspirée de la décomposition en composantes triconnexes des graphes planaires biconnexes. Nous avons présenté cette décomposition pour les graphes en général en section 1.2.3, et son application aux graphes planaires en section 1.4.5. L'algorithme de [BT96] est nettement plus rapide que ceux de [EGIS98, GIS99], il fonctionne en $O(\log n)$ par insertion d'arête. Malheureusement il ne permet pas de traiter la suppression d'arête, ce qui laisse la question ouverte suivante.

Question ouverte 3.1 *Existe-t-il un algorithme de reconnaissance des graphes planaires entièrement dynamique sur les arêtes qui soit basé sur le *SPQR*-arbre et qui fonctionne en temps $O(\log n)$ par modification ?*

Cette approche qui consiste à résoudre un problème dynamique sur une classe de graphe définie géométriquement en utilisant une structure qui représente tous les modèles géométriques du graphe sera reproduite dans ce manuscrit pour les graphes de permutation et d'intervalles. Pour les graphes de permutation, comme nous l'avons vu en section 1.4.3, cette structure est l'arbre de décomposition modulaire. Pour les graphes d'intervalles, ce rôle est rempli par le *PQ*-arbre des cliques maximales, voir section 1.3.1.

Chapitre 4

Aspects dynamiques de la décomposition modulaire

La motivation principale de cette section est la recherche d'un algorithme de maintien entièrement dynamique de la décomposition modulaire d'un graphe lors de modifications élémentaires sur ses sommets et arêtes : ajout et suppression d'une arête ou d'un sommet (avec les arêtes définissant son voisinage).

Quel que soit le problème dynamique considéré, il admet une solution triviale qui consiste à lancer un algorithme statique sur le graphe obtenu après la modification. Ainsi, la complexité d'un algorithme statique est une borne supérieure de la complexité dynamique. Pour cette raison, on compare toujours la complexité d'un algorithme dynamique avec la meilleure complexité statique connue.

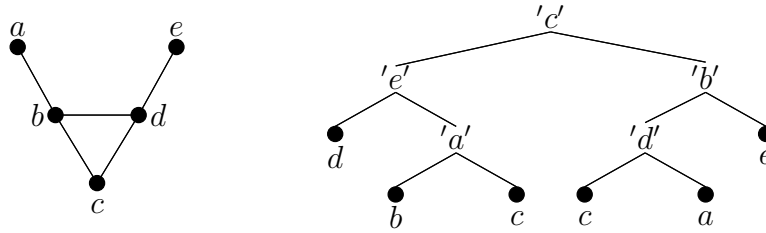
Dans le cas de la décomposition modulaire, une complexité intéressante pour un algorithme dynamique doit donc être inférieure à la borne linéaire $O(n + m)$ de [DGM01]. A l'heure actuelle on ne connaît pas d'algorithme dynamique traitant les quatre opérations avec une complexité dans le pire des cas qui soit meilleure que $O(n + m)$.

Ce mémoire ne fournit pas un tel algorithme. Cependant, nous verrons que pour des classes de graphes ayant de bonnes propriétés de décomposition, on peut atteindre des temps de mise à jour bien inférieurs : $O(1)$ ou $O(n)$ pour les modifications d'arête ; $O(d)$ ou $O(n)$ pour les modifications de sommet (où d est le degré du sommet considéré). Grâce à cette démarche, nous serons en mesure de dégager certaines opérations critiques qu'il est nécessaire et suffisant de traiter rapidement pour obtenir un algorithme dynamique efficace, voir section 4.6.

En outre, ce travail sur la décomposition modulaire ouvre la perspective de traiter de manière similaire d'autres décompositions et les classes de graphes qui leur sont reliées.

4.1 Le cas des graphes quelconques

L'objectif envisagé ici est de réaliser l'entretien dynamique de l'arbre de décomposition modulaire pour un graphe quelconque en un temps inférieur à celui du calcul statique qui

FIG. 4.1 – Un exemple de N -représentation.

se fait en $O(n + m)$. Quelle complexité dans le pire des cas peut-on atteindre pour un algorithme dynamique ?

Dans [MS89], Muller et Spinrad montrent que si on se restreint à ne traiter que les ajouts de sommet, on peut obtenir une complexité dans le pire des cas de $O(n)$ par insertion. Malheureusement, la structure de donnée qu'ils utilisent ne paraît pas permettre de traiter les suppressions de sommet dans la même complexité. L'étude de leur algorithme n'en reste pas moins de grand intérêt. La clef de la complexité réside dans la représentation adoptée pour les noeuds premiers. La lecture de ce chapitre et des algorithmes produits sur des classes de graphes particulières confortera cette conviction. Pour les graphes quelconques, il est difficile de trouver une représentation des noeuds premiers qui les rendent facilement manipulables. C'est ce que réussit à faire la N -représentation de [MS89]. Une de ses propriétés algorithmiques essentielles est qu'elle permet, lors de l'ajout d'un sommet x , de déterminer si x a un jumeau dans un graphe premier en temps $O(n)$.

La N -représentation est un arbre binaire dont les feuilles sont les sommets du graphe premier G et chaque noeud interne est étiqueté par un sommet de G . Cet arbre est construit ainsi. On choisit arbitrairement un sommet v du graphe qui étiquette la racine, et le fils gauche de la racine représente l'ensemble de sommets $N(v) \cup \{v\}$, alors que le fils droit représente l'ensemble $\overline{N}(v) \cup \{v\}$. On continue récursivement de la manière suivante : si un ensemble S de sommets représenté par une feuille f de l'arbre n'est pas un singleton, alors comme ce n'est pas non plus un module (le graphe est premier) il existe un sommet $u \notin S$ tel que $S \cap N(u) \neq \emptyset$ et $S \cap \overline{N}(u) \neq \emptyset$. u est l'étiquette du noeud interne p qui remplace la feuille f . Le fils gauche de p représente l'ensemble $S \cap N(u)$ et son fils droit $S \cap \overline{N}(u)$. On continue récursivement jusqu'à ce que toutes les feuilles soient des singletons. Remarquons que le sommet étiquetant la racine est présent deux fois dans les feuilles de l'arbre, et que c'est le seul dans ce cas. Cela s'expliquera par l'usage de cette structure.

La N -représentation permet, lorsqu'on sait déterminer l'adjacence entre deux sommets en temps $O(1)$, de déterminer si un sommet x à insérer avec son voisinage dans un graphe premier G a un jumeau (vrai ou faux jumeau) dans G . Pour cela on suit un chemin dans l'arbre depuis la racine vers une feuille. Lorsqu'on arrive sur un noeud étiqueté par le sommet v , si x est adjacent à v , on continue le chemin par le fils gauche de v , sinon par le fils droit. On arrive ainsi à une feuille de l'arbre qui représente un sommet w de G qui est l'unique prétendant des sommets de G à être un jumeau de x . Il reste à tester s'ils le sont vraiment en comparant leurs voisinages en $O(n)$ tests d'adjacence. Le chemin suivi

dans l'arbre est de longueur $O(n)$ car l'arbre est binaire et son nombre de feuilles est $n + 1$. Donc, si l'adjacence peut être testée en temps constant, cela prend $O(n)$ de déterminer si x a un jumeau et de le trouver. La nécessité de mettre le sommet v étiquetant la racine dans son sous-arbre droit et dans son sous-arbre gauche vient du fait qu'un sommet x qui est jumeau avec v peut aussi bien l'être en étant adjacent à v qu'en lui étant non adjacent.

La section 4.6 fera apparaître la nécessité de déterminer si x a un jumeau dans un graphe premier et de le trouver le cas échéant en temps $O(n)$ pour pouvoir traiter l'insertion de sommet dans cette même complexité.

L'algorithme de [MS89] présente une contrainte dont on aimerait s'affranchir. L'utilisation de la N -représentation suppose que l'on puisse répondre à la requête d'adjacence en temps constant. Pour ce faire, [MS89] utilise la matrice d'adjacence. Ce choix est tout à fait acceptable pour eux car leur objectif est la complexité statique de $O(n^2)$ pour le calcul sur le graphe complet. Par contre, pour un algorithme purement dynamique, le coût d'entretien de la matrice d'adjacence lors de l'insertion d'un sommet est rédhibitoire, il est de $\Omega(n^2)$ (voir section 2.4). Une façon de pallier à ce problème est de réserver de l'espace en mémoire pour contenir un tableau à deux dimension de taille fixée à l'avance et de l'utiliser pour contenir la matrice d'adjacence du graphe à l'instant courant. Si on fait cela, l'ajout d'un sommet dans la matrice d'adjacence ne demande plus qu'un temps $O(n)$, et on sauvegarde la complexité de [MS89] pour l'ajout de sommet.

Cela présente néanmoins un double inconvénient : le nombre de sommets maximum que peut contenir le graphe est fixé par avance au début de l'algorithme ; et l'espace réservé en mémoire n'est jamais libéré avant la fin de l'algorithme, même si le graphe contient peu de sommets. Le problème de trouver un algorithme dynamique ne présentant pas cet inconvénient et traitant l'insertion de sommet en temps $O(n)$ est encore ouvert.

Laissons pour l'instant de côté le cas des graphes en général. Si on se restreint à des classes de graphes bien décomposables par la décomposition modulaire, il est possible de concevoir des algorithmes entièrement dynamiques sur les arêtes et les sommets avec des complexités faibles. On commence par examiner le cas des cographes et des cographes orientés. Pour ces deux classes, l'absence de noeuds premiers dans l'arbre de décomposition rend possible des complexités optimales : $O(1)$ pour les modifications d'arête et $O(d)$ pour les modifications de sommets. Cette complexité est encore possible à atteindre sur la classe des graphes P_4 -sparse qui autorise les noeuds premiers, mais sous de fortes contraintes. Le cas se rapprochant le plus des graphes quelconques est celui des graphes de permutation, qui s'entretiennent en $O(n)$ par modification d'arête ou de sommet. Les contraintes dans la décomposition modulaire des graphes de permutation ne sont pas trop fortes. Contrairement aux classes cités précédemment, elles ne s'appliquent pas à l'arbre de décomposition lui même mais uniquement aux quotients des noeuds premiers. L'étude de cette classe nous donnera des éléments importants pour envisager le traitement des graphes quelconques, que nous rediscuterons en conclusion.

4.2 Les cographes

Les cographes sont les graphes entièrement décomposables par la décomposition modulaire.

Définition 4.1 *Un graphe G est un cographe ssi tout sous-graphe induit de G ayant au moins trois sommets admet un module non-trivial.*

De cette définition, on déduit le théorème de caractérisation des cographes le plus connu.

Théorème 4.1 [CLSB81] *Un graphe G est un cographe ssi G ne contient pas de P_4 induit.*

Preuve : De la définition 4.1, il découle que les cographes sont les graphes n'ayant aucun sous-graphe induit premier. Or, d'après le Théorème 1.4 page 40, tout graphe premier contient un P_4 . Comme le P_4 est un graphe premier, il vient qu'un graphe G ne possède aucun sous-graphe induit premier ssi G ne possède aucun P_4 induit. ■

La classe est aussi caractérisée par le fait que les cographes sont les graphes sans noeud premier dans leur arbre de décomposition modulaire. Ce résultat appartient au folklore.

Théorème 4.2 *Un graphe G est un cographe ssi son arbre de décomposition modulaire ne comporte pas de noeud premier.*

4.2.1 Performance du coarbre

Le fait que les cographes se comportent si bien vis à vis de la décomposition modulaire leur confère des propriétés remarquables. Au niveau représentation, l'arbre de décomposition modulaire d'un cographe G , souvent appelé **coarbre** [CLSB81], est une représentation complète du graphe qui prends un espace $O(n)$. Cette représentation se comporte à peu près comme les listes d'adjacences au regard des requêtes de voisinage et d'adjacence.

L'adjacence entre deux sommets x et y peut être déterminée en trouvant $ppca(x, y)$ dans l'arbre, ce qui peut être fait en suivant les chemins depuis les feuilles x et y vers la racine de l'arbre en temps $O(\text{Max}\{d(x), d(y)\})$. En effet, le chemin d'une feuille correspondant au sommet z à la racine de l'arbre a une longueur en $O(d(z))$. Cela vient du fait que sur ce chemin, les noeuds séries et parallèles alternent et que le nombre de noeuds séries est $O(d(z))$.

Les cographes peuvent également être représentés plus efficacement vis à vis des requêtes d'adjacence. Ce sont en effet des graphes de permutation. Par conséquent, un réalisateur donne l'adjacence en temps constant et requiert un espace $O(n)$ (voir section 1.3.2).

Pour lister tous les voisins d'un sommet donné x , on peut remonter le chemin C depuis la feuille de x jusqu'à la racine de l'arbre. A chaque noeud série rencontré sur ce chemin, pour chacun de ses fils f , hormis celui qui est sur le chemin C , on liste toutes les feuilles du

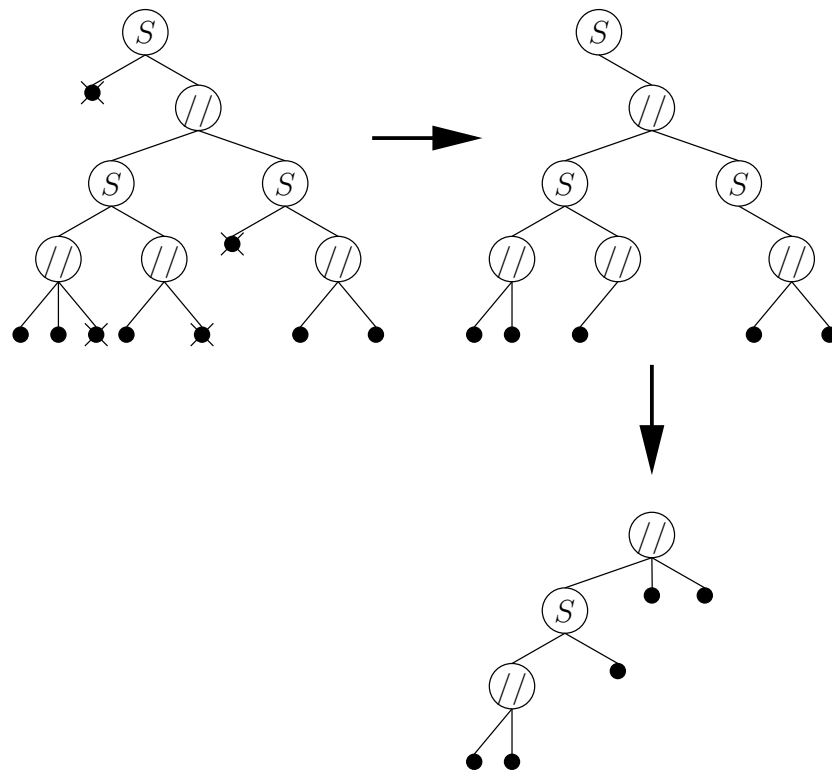


FIG. 4.2 – Obtenir le coarbre d'un sous-graphe induit.

sous-arbre enraciné en f . Comme tout sous-arbre a un nombre de noeuds qui est dominé par son nombre de feuilles, lister toutes les feuilles de tous les sous-arbres rencontrés le long du chemin C prends un temps $O(d(x))$. Quant au chemin lui-même, il est de longueur $O(d(x))$. En effet, comme l'arbre de décomposition ne contient pas de noeud premier, les noeuds séries et parallèles alternent sur tout chemin depuis une feuille à la racine. Et comme à chaque noeud série sur le chemin C peut être associé un voisin de x différent, la longueur du chemin de la feuille de x à la racine de l'arbre est $O(d(x))$. Au bout du compte, les voisins d'un sommet x quelconque peuvent être énumérés en temps $O(d(x))$.

Les cographes admettent donc des représentations efficaces pour les critères d'espace et de temps de traitement des requêtes d'adjacence et de voisinage. Ces représentations ont, de plus, de bonnes propriétés dynamiques. C'est particulièrement vrai pour le coarbre. Pour preuve, le premier algorithme de reconnaissance en complexité linéaire ($O(n + m)$) des cographes est incrémental. Il est dû à Corneil, Perl et Stewart en 1985, [CPS85]. Leur algorithme considère les sommets du graphe G donné un après l'autre, dans n'importe quel ordre $x_1, \dots, x_i, \dots, x_n$. Il construit à chaque étape le coarbre de $G[\{x_1, \dots, x_i\}]$ si celui-ci est un cographe ou retourne *faux* sinon, dans un temps $O(d(x_i))$ à l'étape numéro i . Ainsi, cet algorithme constitue la clef de voûte d'un algorithme de reconnaissance entièrement dynamique de la classe des cographes. Ce dernier travail a été réalisé par Shamir et Sharan [SS04], qui traitent la suppression de sommet, en temps $O(d(x))$, ainsi que la suppression et l'ajout d'arête en temps $O(1)$.

Comme nous le verrons en section 4.5, le meilleur algorithme connu pour l'entretien dynamique du réalisateur d'un graphe de permutation fonctionne en temps $O(n)$ par modification. Ce qui laisse la question suivante pour les cographes.

Question ouverte 4.1 *Existe-t-il, pour les cographes, une structure de donnée permettant de répondre au test d'adjacence en temps constant et dont l'entretien dynamique ne coûte que $O(d)$ par modification élémentaire (ajout ou suppression d'arête ou de sommet), où d est le nombre d'arêtes impliquées dans l'opération ?*

4.2.2 Entretien dynamique

Nous présentons de manière synthétique les idées majeures des algorithmes de [CPS85] et [SS04]. Une étude détaillée, qui généralise ces résultats au cas des cographes orientés, est donnée dans la section 4.3 et a été publiée dans [CP06].

Remarque importante : Contrairement au reste du mémoire, nous faisons, dans cette section et dans la suivante, l'hypothèse que les sommets du graphe nous sont donnés non pas par leur identifiant mais par un pointeur sur leur feuille correspondante dans le coarbre.

Se basant sur le caractère héréditaire de la famille des cographes, [CPS85] se ramène à considérer le problème suivant : étant donné un cographe $G = (V, E)$ et son coarbre T^m , ainsi qu'un sommet $x \notin V$ et son voisinage $N(x) \in V$, le graphe $G + x$ est-il un cographe ? si oui, en donner son coarbre.

Ainsi, en partant du graphe à un sommet qui est un cographe et dont on connaît le coarbre, on reconnaît incrémentalement la famille des cographes. On ajoute les sommets du graphe donné un après l'autre et on détermine si le nouveau sous-graphe obtenu est un cographe. S'il ne l'est pas, par la propriété d'hérédité, le graphe entier lui-même ne l'est pas. Si la réponse est positive, l'algorithme continue en ajoutant un nouveau sommet. L'algorithme s'arrête soit lorsqu'il obtient une réponse négative sur un sous-graphe, soit lorsqu'il obtient une réponse positive pour le graphe entier.

La clef de l'algorithme de [CPS85] est la procédure *Mark*. Nous nous servirons de variantes de cette procédure pour les cographes orientés, les graphes de permutation et les graphes d'intervalles. Cette procédure marque et démarque, au plus une fois, certains noeuds de T^m .

Procédure *Mark*

- La procédure commence par marquer les feuilles de T^m correspondant aux voisins de x , puis les démarque.
- Lorsqu'un noeud de T^m (feuille ou noeud interne) est démarqué, il propage l'information à son père qui devient alors marqué, ou le reste s'il l'était déjà.
- Lorsqu'un noeud interne a tous ses fils qui ont été démarqués, il est lui même démarqué.
- Le processus prend fin lorsqu'aucun sommet ne reste à être démarqué.

Définition 4.2 *A la fin de la procédure *Mark*, on qualifie de **non marqués** à la fois les noeuds qui ont été marqués puis démarqués et les noeuds qui n'ont jamais été marqués.*

L'examen des noeuds de T^m qui sont marqués et de ceux qui sont non marqués suffit à déterminer si $G + x$ est un cographe. Remarquons que le nombre de noeuds démarqués au cours du processus est $O(d(x))$ car le sous-arbre enraciné en un noeud démarqué ne contient que des feuilles voisines de x . Les noeuds restant marqués après *Mark* sont aussi en nombre $O(d(x))$ car ils ont au moins un fils démarqué. Cette propriété numéraire est essentielle à la complexité de l'algorithme entier.

Définition 4.3 *Un noeud série de T^m qui reste marqué après *Mark* est dit **correctement marqué** si tous ses fils sauf un ont été démarqués.*

Définition 4.4 *Un chemin d'un coarbre marqué est appelé **chemin alterné légitime** s'il est constitué de noeuds série correctement marqués et de noeuds parallèle non marqués qui alternent le long du chemin (voir figure 4.3 tirée de [CPS85]).*

Soit α un noeud marqué de niveau minimal (le plus bas) dans l'arbre. La propriété fondamentale, dont nous retrouveront le pendant dans le cas orienté, est que si $G + x$ est un cographe, les noeuds restants marqués ne peuvent pas être disséminés partout dans l'arbre.

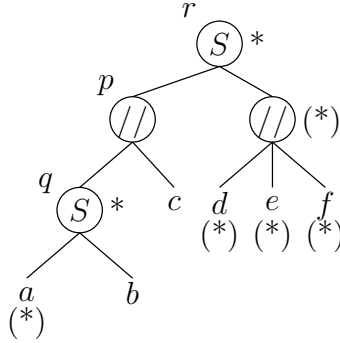


FIG. 4.3 – Exemple de coarbre marqué. $*$ symbolise les noeuds marqués et $(*)$ ceux qui ont été démarqués. La racine r est un noeud correctement marqué. Le chemin rpq est un chemin alterné légitime.

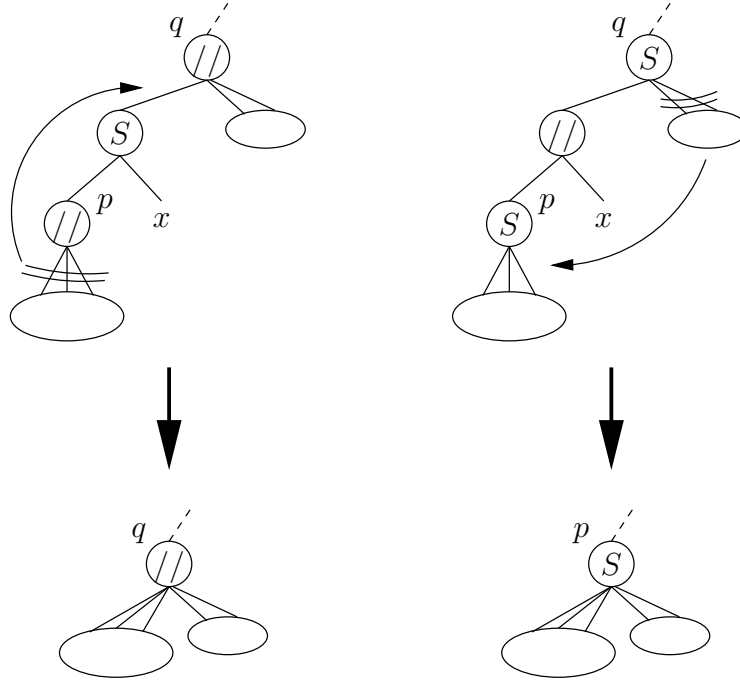
Théorème 4.3 [CPS85] *Si $G + x$ est un cografe, alors les noeuds marqués privés de α sont exactement les noeuds séries d'un chemin alterné légitime ayant la racine pour extrémité.*

[CPS85] fournit même une condition nécessaire et suffisante, en termes de noeuds marqués et démarqués, pour que $G + x$ soit un cografe. [CPS85] montre comment vérifier cette condition et comment construire le nouveau coarbre si la réponse est positive, en temps $O(d(x))$.

Dans l'optique d'un algorithme entièrement dynamique de reconnaissance des cografes, l'algorithme de [CPS85] couvre exactement le cas de l'ajout de sommet. [SS04] complète l'approche en envisageant la suppression de sommet ainsi que l'ajout et la suppression d'arête. La suppression de sommet ne pose que peu de problèmes. Comme la famille des cografes est héréditaire, si G est un cografe, le graphe $G - x$ en est toujours un. Son coarbre est obtenu comme expliqué plus haut et illustré sur la figure 4.2. La seule difficulté se présente lorsque le sommet à supprimer x a un unique frère p dans T^m et que ce dernier a la même étiquette que le grand-père de x dans T^m (voir figure 4.4). Pour obtenir la complexité $O(d(x))$ souhaitée il faut alors prendre garde en construisant le nouvel arbre à ne manipuler que des noeuds dont le sous-arbre contient des voisins de x .

En ce qui concerne les modifications d'arête, [SS04] fait remarquer que si on sait traiter soit l'ajout, soit la suppression d'arête en utilisant seulement le coarbre, alors on sait faire l'autre opération dans la même complexité. En effet, enlever une arête de G revient à en ajouter une dans $\overline{G}$. Or le coarbre du complémentaire d'un cografe s'obtient à partir de celui du graphe lui-même en inversant les noeuds séries et parallèles. Pour manipuler le coarbre de $\overline{G}$ sans avoir à le calculer, il suffit d'interpréter les noeuds séries de T^m comme des noeuds parallèles, et vice versa.

Considérons par exemple le cas de l'insertion d'une arête uv dans le cografe G . Le graphe résultant de la modification est rarement un cografe, au sens où la condition pour qu'il le soit est très contraignante. En particulier, [SS04] montre qu'il faut (mais ce n'est pas suffisant) que le $ppca$ de u et v dans T^m soit à distance au plus deux des

FIG. 4.4 – Cas délicats de la suppression de x .

deux feuilles u et v . Cela permet de le trouver en temps constant. Le reste de la condition nécessaire et suffisante de [SS04] se vérifie aisément en temps constant. Si le graphe modifié est un cographe, la modification du coarbre à accomplir est locale et ne bouleverse pas sa structure : elle peut aussi être accomplie en temps constant. Ce qui donne un algorithme optimal pour le retrait et l'ajout d'arête.

Théorème 4.4 [SS04] *Il existe un algorithme entièrement dynamique sur les arêtes de complexité optimale pour reconnaître les cographes et maintenir leur arbre de décomposition modulaire, qui traite les modifications d'arête en temps constant.*

L'algorithme de [CPS85] pour l'ajout de sommet est aussi optimal. Seule l'optimalité de la complexité de la suppression de sommet peut être remise en cause : la donnée est de taille constante (identifiant du sommet à supprimer) et l'algorithme la traite en temps $O(d(x))$.

4.3 Les cographes orientés

Nous venons de le voir, il est possible d'obtenir un algorithme entièrement dynamique de maintien du coarbre d'un cographe qui soit de complexité optimale : $O(d)$ par modification d'un sommet de degrés d et $O(1)$ par modification d'arête. On s'en doute, une telle efficacité est due à l'absence de noeud premier dans le coarbre, car ce sont eux qui contiennent la plus grande part de la "complexité" d'un graphe.

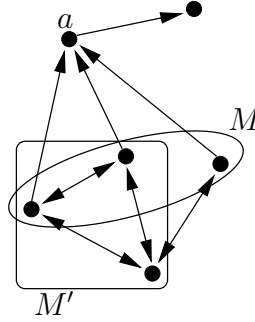


FIG. 4.5 – M est un module, mais M' n'en est pas un (à cause de a par exemple).

C'est en tous cas ce que suggère le résultat de [CPS85, SS04] sur les cographes qui peut être interprété ainsi : si on écarte la complexité introduite par les noeuds premiers, il est possible de maintenir l'arbre de décomposition de manière optimale.

La question à laquelle nous voulons répondre dans cette section est : quelle est la complexité introduite par l'orientation des arêtes ? Autrement dit, si on s'intéresse au problème équivalent dans le cadre des graphes orientés, peut-on toujours atteindre cette complexité optimale, ou bien la difficulté introduite par le caractère non symétrique de la relation binaire est elle significativement plus importante ?

Cette question nous amène à considérer la classe des cographes orientés pour laquelle nous montrons qu'au prix d'une analyse un peu plus complexe, il est possible d'obtenir un algorithme ayant la complexité optimale de celui pour les cographes non orientés. Le travail présenté dans cette section a été publié dans [CP04, CP06].

4.3.1 Définition et premières propriétés

La décomposition modulaire s'applique parfaitement aux graphes orientés. Elle est même définie pour des structures plus générales appelées **2-structures** [ER90b] (voir aussi [EHR99] pour de la littérature sur les 2-structures).

Définition 4.5 *Un module d'un graphe orienté $G = (V, A)$ est un sous-ensemble de sommets $S \subseteq V$ tel que $\forall x \in V \setminus S, \forall y, z \in S, (xy \in A \Leftrightarrow xz \in A)$ et $(yx \in A \Leftrightarrow zx \in A)$.*

Dit plus prosaïquement, un module d'un graphe orienté, comme d'un graphe non orienté, est un ensemble S de sommets tel que tout sommet extérieur à S voit tous les sommets de S de la même façon (voir figure 4.5). La famille des modules d'un graphe orienté n'est pas une famille partitionnée, mais faiblement partitionnée (c'est à dire qu'elle vérifie les conditions du théorème 4.5). Il n'est plus vrai que la différence symétrique de deux modules qui se chevauchent est un module, mais cela reste vrai pour les différences simples.

Théorème 4.5 *Si A et B sont deux modules d'un graphe orienté G et se chevauchent alors :*

1. $A \cap B$ est un module de G
2. $A \cup B$ est un module de G
3. $A \setminus B$ et $B \setminus A$ sont des modules de G

Les définitions de module fort et de module fort maximal restent identiques au cas non orienté (voir section 1.2.1). Le théorème de décomposition pour les graphes orientés s'énonce ainsi.

Théorème 4.6 *Soit $G = (V, A)$ un graphe orienté et $x \in V$. Le graphe $G/\mathcal{MFM}(G)$ est :*

- soit un stable,
- soit une clique (un graphe orienté complet),
- soit un ordre total,
- soit un graphe orienté premier.

On définit l'arbre de décomposition modulaire comme dans le cas non orienté. C'est la réduction transitive de l'ordre d'inclusion des modules forts et on étiquette chaque noeud p selon la nature du quotient $G[P]/\mathcal{MFM}(G[P])$: *parallèle, série, ordre* ou *premier*.

Les modules d'un graphe orienté se lisent sur son arbre de décomposition modulaire de la manière suivante.

Théorème 4.7 *Soit G un graphe orienté. En confondant les noeuds de l'arbre de décomposition modulaire et le sous-ensemble de sommets qu'ils représentent, on a : les modules de G sont exactement les noeuds de l'arbre (les modules forts), les unions d'un sous-ensemble des fils d'un noeud série ou parallèle, et les unions de fils consécutifs d'un noeud ordre.*

La classe des graphes orientés entièrement décomposables par la décomposition modulaire s'appelle les **cographes orientés**. Ce sont les graphes qui n'ont pas de noeuds premiers dans leur arbre de décomposition modulaire, qu'on appelle un **coarbre orienté**. Dans un coarbre orienté, il n'y a pas d'ordre sur les fils des noeuds séries et parallèles, mais les fils des noeuds ordres p sont totalement ordonnés, en cohérence avec l'ordre total du quotient $G[P]/\mathcal{MFM}(G[P])$. Un exemple de c graphe orienté avec son coarbre orienté est donné par la figure 4.6. Les cographes orientés sont aussi caractérisés par sous-graphes exclus. Les graphes à exclure sont les graphes orientés premiers minimaux, ils sont au nombre de huit et sont illustrés sur la figure 4.7.

Théorème 4.8 *Un graphe orienté est un c graphe orienté ssi il ne contient aucun des graphes de la figure 4.7 comme sous-graphe induit.*

La famille des cographes orientés, comme celle des cographes, est donc héréditaire.

Faisons une relecture du théorème de décomposition (théorème 4.6) spécifique aux cographes orientés. Un graphe orienté $G = (V, A)$ est un **k -ordre**, avec $k \in \mathbb{N}^*$, s'il existe une partition $V_1 \sqcup \dots \sqcup V_k$ de V telle que pour tout $x \in V_i$ et pour tout $y \in V_j$, si $i < j$

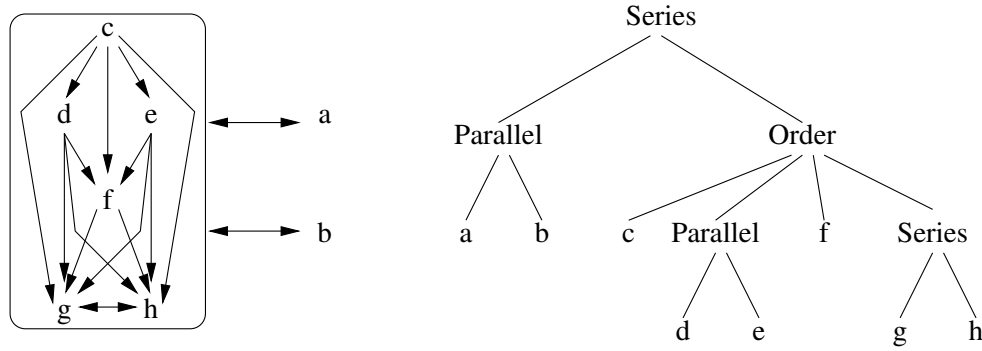


FIG. 4.6 – Un cographe orienté et son coarbre orienté.

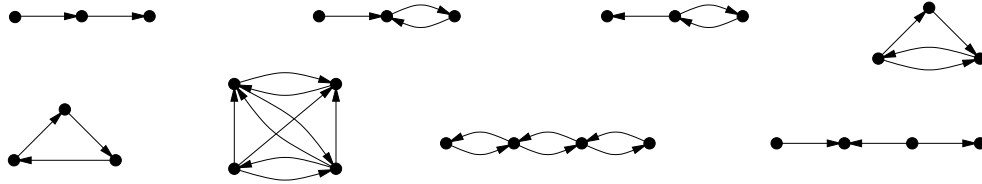


FIG. 4.7 – Ensemble des graphes orientés premiers minimaux. Ce sont les sous-graphes exclus pour la famille des cographes orientés.

alors $xy \in A$ et $yx \notin A$. Soit k l'entier maximum tel que G est un k -ordre, alors il existe une unique partition telle que G est un k -ordre. Les ensembles de cette partition unique sont appelés les **composantes ordres** de G . Les composantes ordres sont naturellement ordonnées par l'orientation du graphe de la première V_1 à la dernière V_k . Dans la suite, V_1 et V_k seront appelées **composantes ordres extrémales**. Le complémentaire d'un graphe orienté $G = (V, A)$ est le graphe orienté $\bar{G} = (V, V^2 \setminus A)$. Un graphe orienté est dit co-connexe ssi son complémentaire est connexe.

Théorème 4.9 *Un cographe orienté G est :*

- soit non connexe, alors ses modules forts maximaux sont ses composantes connexes,
- soit non co-connexe, alors ses modules forts maximaux sont ses composantes co-connexes,
- soit connexe et co-connexe, alors G est un k -ordre, pour un certain $k \in \mathbb{N} \setminus \{0, 1\}$ et ses modules forts maximaux sont ses composantes ordres.

Il en découle que les cographes orientés sont les graphes définis récursivement à partir du graphe à un sommet sous la fermeture des trois opérations suivantes : l'**union disjointe**, la **composition série** et la **composition ordre**. L'union disjointe de deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ est le graphe $G_{12} = (V_1 \cup V_2, E_1 \cup E_2)$. La composition série de G_1 et G_2 est l'union disjointe dans laquelle on ajouté toutes les arcs entre V_1 et V_2 , dans

les deux sens. Enfin, la composition ordre de V_1 et V_2 est l'union disjointe dans laquelle on a ajouté tous les arcs de V_1 vers V_2 . Nous utiliserons ce vocabulaire dans les preuves.

Soit un graphe orienté $G = (V, A)$, un sommet $x \notin V$ et deux sous-ensembles de sommets $S_1 \in V$ et $S_2 \in V$. On note $G + x$ le graphe $G' = (V \cup \{x\}, A \cup \{xz \mid z \in S_1\} \cup \{zx \mid z \in S_2\})$, de telle sorte que S_1 et S_2 sont respectivement les voisinages sortant et entrant de x dans $G + x$. Si $xy \in E$, $G - xy$ est le graphe $G' = (V, E \setminus \{xy\})$. Les graphes $G - x$ et $G + xy$ sont définis de manière analogue. Le lemme suivant se passe de démonstration.

Lemme 4.1 *Soit $G = (V, A)$ un graphe orienté et $x \notin V$ un sommet à insérer dans G . Soit $M \subseteq V$ tel que $M \cup \{x\}$ est un module de $G + x$, alors M est un module de G .*

Les notions que nous introduisons maintenant joueront un rôle essentiel dans la conception et l'analyse de l'algorithme. Un ensemble $S \subseteq V$ de sommets est **uniforme** relativement à un sommet $x \notin S$ si $S \subseteq N^+(x)$ ou $S \cap N^+(x) = \emptyset$, et $S \subseteq N^-(x)$ ou $S \cap N^-(x) = \emptyset$. Autrement dit, S est uniforme ssi c'est un module de $G[S \cup \{x\}]$. Lorsque S n'est pas uniforme on dit qu'il est **mixte**. Pour un noeud p d'un coarbre T^m , la notion d'uniformité se rapporte à celle du module fort qu'il représente. Un ensemble S de sommets est **lié** à un sommet $x \notin S$ si $S \cap N(x) \neq \emptyset$. Si S est uniforme et lié, on dit que S est **uniformément lié**. Lorsqu'aucune confusion n'est possible, on omet le sommet auquel se rapporte les notions ci-dessus.

Les notions précédentes seront employées aussi pour les noeuds p de T^m en se référant au sous-ensemble de sommets $P \subseteq V$ qu'ils représentent. Dans le coarbre orienté T^m , le chemin entre un noeud p et la racine r de T^m est noté P_p^r . M_{xy} désigne le plus petit module de G (pour l'inclusion) contenant les sommets x et y . Comme M_{xy} n'est pas nécessairement fort, c'est un sous ensemble de P_{xy} , où p_{xy} est le *ppca* de x et y dans T^m . Pour tout sommet y de G , la feuille de T^m correspondant à y sera notée l_y .

4.3.2 Structure de données

La représentation que nous maintenons d'un cografe orienté est basée sur son coarbre. Nous maintenons aussi une permutation factorisante du graphe (voir définition 1.7 page 39) qui n'est pas utile pour la reconnaissance à proprement parler mais permet de fournir un certificat lorsque le graphe n'est plus un cografe orienté.

Comme illustré sur la figure 4.8, chaque noeud q du coarbre stocke six pointeurs :

- un pointeur vers son père p dans le coarbre, et un pointeur vers sa position dans la liste des fils de p ;
- un pointeur vers le premier et le dernier élément de la liste de ses fils dans le coarbre ;
- un pointeur vers le premier et le dernier sommet de Q dans la permutation factorisante.

Les listes de fils et la permutation factorisante sont des listes doublement chaînées (par soucis de clarté sur la figure 4.8, elles sont représentées comme des listes simplement chaînées). La liste des fils de tout noeud ordre est ordonnée en cohérence avec l'ordre défini

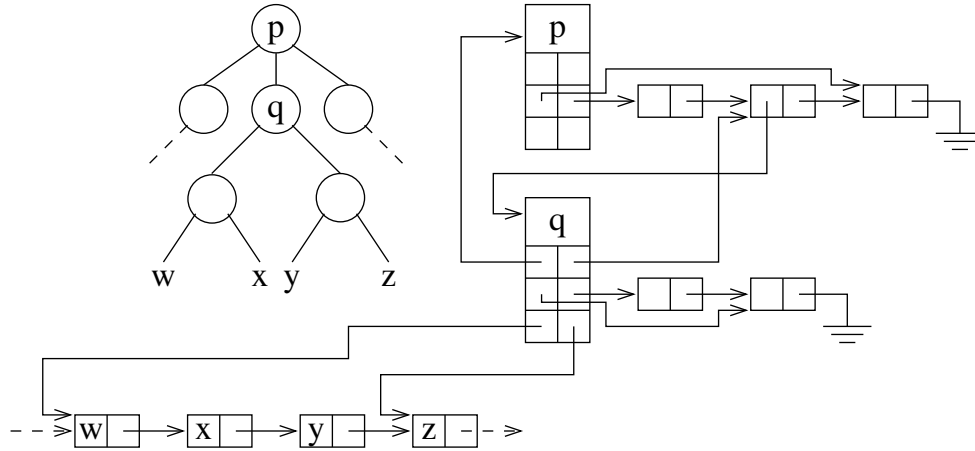


FIG. 4.8 – La structure de donnée maintenue par l’algorithme. Bien que ce ne soit pas représenté ainsi sur la figure, les listes de fils d’un noeud ainsi que la permutation factorisante sont des listes doublement chaînées. En plus de ses pointeurs, chaque noeud du coarbre stocke le nombre de ses fils.

par le noeud sur ses fils, de la première composante ordre à la dernière. En plus de la liste de ses fils, chaque noeud stocke leur nombre.

Remarquons que la méthode de lecture de l’adjacence dans le coarbre orienté se fait en temps $O(d(x))$, comme dans le coarbre non orienté. Pour connaître la relation d’adjacence entre x et y , on trouve d’abord le $ppca(x, y)$. Comme pour les cographes, la longueur du chemin entre la feuille d’un sommet x et la racine du coarbre orienté est $O(d(x))$. Si $ppca(x, y)$ est série ou parallèle, on conclut, sinon, il faut déterminer laquelle des composantes ordres de x et de y est la première dans l’ordre défini par le $ppca$. On peut parcourir toute la liste car le nombre de fils d’un noeud ordre q est $O(d(x))$ pour tout sommet $x \in Q$. Les pointeurs d’un noeud q vers la permutation factorisante permettent d’accéder depuis q à la liste des sommets de Q en temps constant, ce qui n’est pas possible seulement avec le coarbre.

4.3.3 Opérations dynamiques sur les sommets

Cette section traite de l’insertion et la suppression d’un sommet dans un cographe orienté. Lors d’une suppression de sommet, le graphe résultant $G - x$ est toujours un cographe orienté. L’algorithme consiste alors simplement à mettre à jour le coarbre et la permutation factorisante. La connaissance de la manière dont se modifie le coarbre soumis à une suppression de sommet est utile pour caractériser les cas dans lesquels l’insertion d’un sommet x est possible. Le théorème 4.10 constitue la base de l’algorithme d’insertion qui met à jour le coarbre et la permutation factorisante, ou trouve un certificat garantissant que $G + x$ n’est pas un cographe orienté. Par soucis de simplicité, le certificat est un ensemble de quatre sommets qui induisent un sous-graphe contenant un des sous-graphes

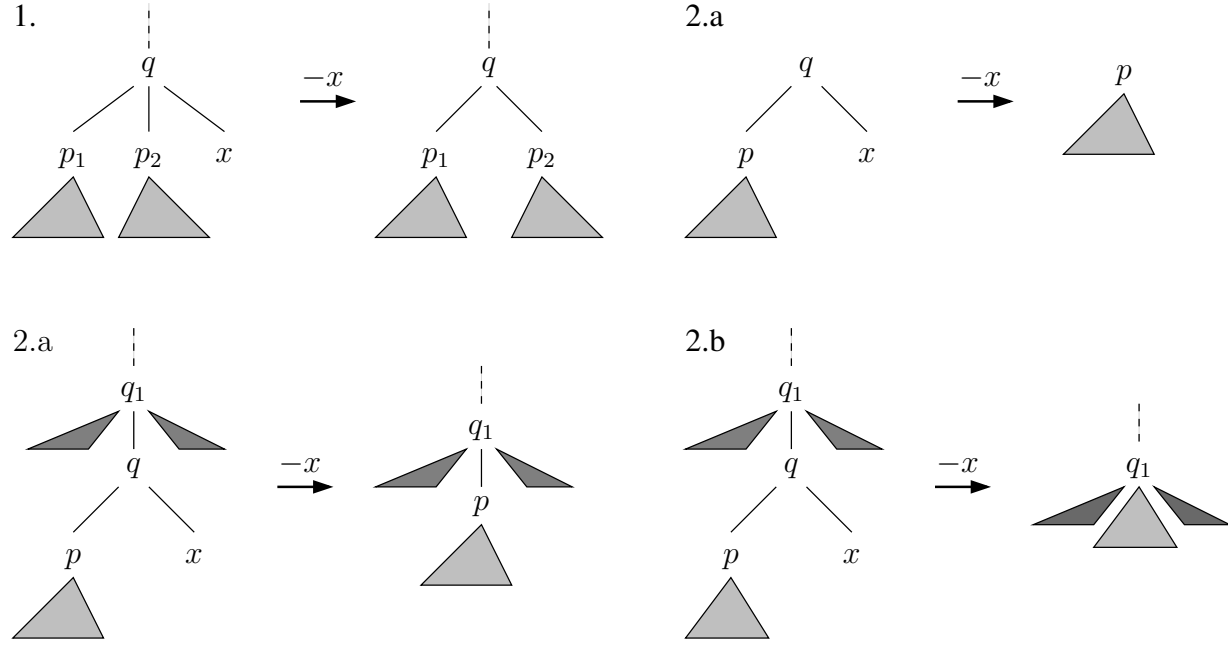


FIG. 4.9 – Modifications du coarbre soumis à une suppression de sommet.

induits interdits de la figure 4.7. Cependant, en complétant l'algorithme, un certificat exact peut être retourné. La complexité de l'algorithme de suppression, comme celle de celui d'insertion, est $O(d(x))$.

Suppression d'un sommet

Comme remarqué précédemment, l'opération de suppression ne demande que la mise à jour du coarbre T^m de G afin d'obtenir le coarbre T' de G' (voir figure 4.9). Cela peut être fait en temps $O(d(x))$ de la manière suivante (voir [SS04] pour un algorithme similaire).

Le cas où x est le seul sommet du graphe est trivial. Dans le cas contraire, soit q le père de x dans T^m .

1. Si x a au moins 2 frères, alors x est supprimé de T^m .
2. Sinon, soit p l'unique frère de x .
 - (a) Si q est la racine de T^m ou si l'étiquette de $\text{parent}(q) = \tilde{q}$ est différente de celle de p , les noeuds l_x et q sont supprimés de T^m . Si q est la racine de T^m , alors c'est p qui devient racine de T' . Sinon, p est inséré dans les fils de $\tilde{q}$ précisément à la place de q (c'est crucial lorsque $\tilde{q}$ est un noeud ordre).
 - (b) Si $\text{label}(\tilde{q}) = \text{label}(p)$, les noeuds l_x , q et p sont supprimés de T^m . Les fils de p sont insérés dans les fils de $\tilde{q}$, à la place de q . Si $\tilde{q}$ est un noeud ordre, alors l'ordre relatif des fils de p est respecté et ces derniers deviennent un intervalle des fils de $\tilde{q}$ inséré précisément à la place de q .

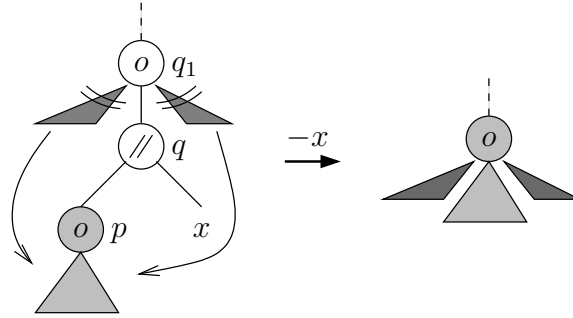


FIG. 4.10 – Mise à jour du coarbre en temps $O(d)$ lors d’une suppression de sommet. Les noeuds sont étiquetés par O pour ordre et par // pour parallèle.

Le lecteur se convaincra que le nouvel arbre T' ainsi construit est bien l’unique coarbre de G' en vérifiant que les propriétés suivantes sont satisfaites : i) aucun noeud de T' n’a la même étiquette que son père, tous les noeuds internes ont au moins deux fils, et tous les noeuds sont étiquetés série, ordre ou parallèle (i.e. T' est un coarbre valide) ; ii) la relation d’adjacence définie par T' (méthode de l’ancêtre) est exactement celle de $G[V \setminus \{x\}]$.

Pour des raisons de complexité, le cas où p et $\tilde{q}$ ont la même étiquette (cas 2b ci-dessus) doit être traité avec attention : on ne doit toucher que des noeuds de T^m dont l’arbre contient des voisins de x . Si q est un noeud parallèle, les fils de p sont liés à x . Ils peuvent être déconnectés de p et substitués à q dans les fils de $\tilde{q}$. Si q est un noeud parallèle, ses frères sont liés à x . Ils peuvent être déconnectés de $\tilde{q}$ et reconnectés en tant que fils de p (à la place convenable si $\tilde{q}$ est un noeud ordre, voir figure 4.10). Enfin, p remplace $\tilde{q}$.

Mettre à jour la permutation factorisante se réduit à en supprimer x . Remarquons que les seuls noeuds restant dans T^m après la suppression de x et qui doivent actualiser leurs pointeurs sont les ancêtres p de x pour lesquels x est une borne du segment correspondant à p dans la permutation factorisante. Pour ces noeuds, leur pointeur vers x doit être changé pour un pointeur sur le sommet précédant (resp. suivant) x dans la permutation factorisante, lorsque x est le dernier (resp. le premier) sommet du segment correspondant au noeud p en question. Comme le nombre d’ancêtres de x est $O(d(x))$, la mise à jour entière des pointeurs vers la permutation factorisante peut être effectuée en temps $O(d(x))$. Seuls le père et le grand-père de x dans T^m pourraient avoir à actualiser leur nombre de fils. Dans le cas 1, $|\mathcal{C}(q)|$ est décrémenté d’un ; dans le cas 2.b, $|\mathcal{C}(\tilde{q})|$ est augmenté de $|\mathcal{C}(q)|$. Cela ne prends qu’un temps constant.

Insertion d’un sommet

La difficulté principale de l’algorithme entièrement dynamique présenté ici est le maintien du coarbre orienté lors de l’insertion d’un sommet. Le théorème 4.10 caractérise les cas dans lesquels, étant donnés un cographe orienté G , un sommet x et ses voisinages, le graphe $G + x$ reste un cographe orienté. Comme dans [CPS85], l’algorithme effectue d’abord une étape de marquage du coarbre orienté T^m de G . Ensuite, il teste si les marques satisfont

le théorème 4.10. Si oui, le coarbre est mis à jour ; sinon, un certificat prouvant que $G + x$ n'est pas un cografe orienté est retourné.

Théorème 4.10 *Soit $G = (V, A)$ un cografe orienté et T^m son coarbre. Soit $x \notin V$ un sommet donné avec ses voisinages $N^-(x)$, $N^+(x)$. $G' = G + x$ est un cografe orienté ssi pour tout noeud p de T^m , une des conditions suivantes est vraie :*

1. P est uniforme rel. à x ;
2. P est mixte rel. à x et a un unique fils mixte f tel que $F \cup \{x\}$ est un module de $G'[P \cup \{x\}]$;
3. P est mixte rel. à x , n'a aucun fils mixte et une des conditions suivantes est vraie :
 - (a) il existe un unique ensemble non vide $\mathcal{S} \subsetneq \mathcal{C}(p)$ de fils de p tel que $S = \bigcup_{k \in \mathcal{S}} K$ est uniforme rel. à x et $S \cup \{x\}$ est un module de $G'[P \cup \{x\}]$, ou
 - (b) il existe un ensemble non vide $\mathcal{S} \subsetneq \mathcal{C}(p)$ de fils de p tel que $S \cup \{x\}$ et $(P \setminus \mathcal{S}) \cup \{x\}$ sont des modules de $G'[P \cup \{x\}]$, avec $S = \bigcup_{k \in \mathcal{S}} K$.

Un noeud de T^m qui satisfait la condition 2 du théorème 4.10 est appelé noeud **simplement mixte**, et noeud **mixte terminal** s'il satisfait la condition 3.

Il est à noter que les cas 3.(a) et 3.(b) du Théorème 4.10 sont mutuellement exclusifs. Dans le cas 3.(b), p est nécessairement un noeud ordre. Sinon, p serait uniforme rel. à x . Or, un noeud ordre p qui satisfait la condition 3.(b) ne satisfait pas la condition 3.(a). En effet, l'unicité de l'ensemble $\mathcal{S}$ de la condition 3.(a) ne serait pas assurée : les ensembles $\mathcal{S}$ et $\mathcal{C}(p) \setminus \mathcal{S}$ de la condition 3.(b) conviendraient tous les deux.

Le corollaire 4.1 ci-dessous montre que si $G + x$ est un cografe orienté, les noeuds mixtes ne peuvent pas être disséminés n'importe où dans T^m , et qu'il y a un unique noeud mixte terminal.

Le théorème 4.10 est une équivalence. Le corollaire 4.1 découle de l'implication directe et est utile pour montrer l'implication réciproque du théorème 4.10. Aussi, nous prouvons d'abord que les conditions du théorème 4.10 sont nécessaires, puis nous prouvons que ces conditions impliquent le corollaire 4.1, et pour finir, nous prouvons l'implication réciproque du théorème 4.10.

Preuve de l'implication directe du théorème 4.10 : $\implies$. Soit $G' = G + x$ qui est un cografe orienté. Soit T' son coarbre et q' le père de x dans T' . En supprimant x de G' , on obtient, comme décrit page 131 et sur la figure 4.9, le coarbre T^m de $G' - x = G$. La transformation de T' en T^m fait apparaître des relations entre les modules forts de G' et les modules forts de G . Nous utilisons ces relations pour montrer que les conditions du théorème 4.10 sont satisfaites. La discussion qui suit utilise les notations de la figure 4.9. Avant d'étudier l'un après l'autre les trois cas de la suppression de sommet (page 131), nous écartons d'abord le cas où q' est la racine de T' , qui est un cas particulier des cas 1 et 2.a de la suppression de sommet. Dans ce cas, les modules forts de G sont exactement les modules forts de G' qui ne contiennent pas x . Plus précisément, ce sont les modules forts

de G' correspondant aux noeud de T' différents de la racine et de l_x . Par conséquent, ils sont tous uniformes et satisfont la condition 1 du théorème 4.10.

A partir de maintenant, nous supposons donc que q' n'est pas la racine de T' . Soit $\tilde{q}'$ son père. Après la suppression de x , $\tilde{q}'$ reste un noeud du coarbre (voir figure 4.9). Dans T^m , nous renommons ce noeud en $\tilde{q}$. En d'autres termes, $\tilde{q}$ est le noeud de T^m correspondant au module fort $\tilde{Q}' \setminus \{x\}$ de G .

Proposition 4.1 $\tilde{q}$ est mixte rel. à x .

Preuve : En effet, les étiquettes de $\tilde{q}'$ et q' sont différentes, ce qui implique que les sommets de $\tilde{Q}' \setminus Q'$ et ceux de $Q' \setminus \{x\}$ n'ont pas la même relation d'adjacence avec x . Il s'ensuit que $\tilde{Q} = \tilde{Q}' \setminus \{x\}$ est mixte. $\square$

Proposition 4.2 Soit $k \in \text{Anc}_T(\tilde{q}) \setminus \{\tilde{q}\}$, k est simplement mixte.

Preuve : Soit k_m l'unique fils de k qui soit un ancêtre de $\tilde{q}$. Comme k_m est un ancêtre de q , k_m est mixte. L'étude de la suppression d'un sommet (voir page 131 et figure 4.9) nous apprend que $K_m \cup \{x\}$ est un module fort de G' , c'est donc un module de $G'[K \cup \{x\}]$. Par conséquent, k est simplement mixte. $\square$

Remarque 4.1 Par voie de conséquence, tout fils f de k différent de l'unique fils mixte k_m de k est uniforme, et ses descendants aussi.

- Si x a au moins deux frères dans T' (voir cas 1. de la suppression de sommet et figure 4.9), alors $Q' \setminus \{x\}$ est un module fort de G . Nous notons q son noeud correspondant dans T^m . Examinons $\tilde{q}$ et ses descendants. Soit u un fils de $\tilde{q}$ différent de q , alors U est un module de G' qui ne contient pas x . Il s'ensuit que u et ses descendants sont uniformes. Pour les mêmes raisons, les fils de q sont uniformes. En ce qui concerne $\tilde{q}$ et q , il nous faut distinguer deux cas. Premièrement, si q' est un noeud ordre et si x n'est pas une composante ordre extrême de q' , soit $\mathcal{S} = \{f \in \mathcal{C}(q') \mid f <_{q'} l_x\}$, où $<_{q'}$ est l'ordre défini par q' sur ses fils, et soit $S = \bigcup_{f \in \mathcal{S}} F$. $S \cup \{x\}$ et $(Q \setminus S) \cup \{x\}$ sont des modules de $G'[Q \cup \{x\}] = G'[Q']$. Donc, par définition, q est mixte terminal, il satisfait la condition 3.(b) du théorème 4.10. Comme $Q \cup \{x\} = Q'$ est un module de $G'[\tilde{Q} \cup \{x\}] = G'[\tilde{Q}']$, $\tilde{Q}$ est simplement mixte. Dans le deuxième cas, c'est à dire si q' est un noeud série ou parallèle ou si q' est un noeud ordre mais que x est une composante ordre extrême de q' , alors $Q = Q' \setminus \{x\}$ est un module de G' qui ne contient pas x . Donc, q est uniforme, il satisfait la condition 1 du théorème 4.10. Et comme $Q \cup \{x\} = Q'$ est un module de $G'[\tilde{Q} \cup \{x\}] = G'[\tilde{Q}']$, $\tilde{q}$ est mixte terminal, il satisfait la condition 3.(a) du théorème 4.10 avec $\mathcal{S} = \{q\}$. Un tel sous-ensemble $\mathcal{S}$ de fils de $\tilde{q}$ est unique. En effet, comme S doit être uniforme, d'après le cas 1 de la suppression de sommet (voir figure 4.9), $S \subseteq Q$. Comme $\mathcal{S}$ est un sous-ensemble de fils de $\tilde{q}$, nécessairement $\mathcal{S} = \{q\}$.

- Si x a un unique frère p' dans T' et si p' et $\tilde{q}'$ ont des étiquettes différentes (voir cas 2.(a) de la suppression de sommet et la figure 4.9), alors P' est un module fort de G . On note p son noeud correspondant dans T^m . De manière similaire avec le cas précédent, les fils de $\tilde{q}$ différent de p sont uniformes. De plus, P est un module fort de G' et P ne contient pas x , donc p est uniforme. Enfin, tout descendant de $\tilde{q}$ est uniforme. Comme $P \cup \{x\} = Q'$ est un module de $G'[\tilde{Q} \cup \{x\}] = G'[\tilde{Q}']$, et comme $\tilde{q}$ est mixte, alors il est mixte terminal, il satisfait la condition 3.(a) du théorème 4.10 avec $\mathcal{S} = \{p\}$. De nouveau, un tel ensemble $\mathcal{S}$ est unique. En effet, S doit être uniforme, d'où $S \subseteq P'$ ou $S \subseteq (\tilde{Q}' \setminus Q')$. $S \cup \{x\}$ doit aussi être un module de $G'[\tilde{Q} \cup \{x\}] = G'[\tilde{Q}']$, donc $S = P' = P$.
- Si x a un unique frère p' dans T' et si p' et $\tilde{q}$ ont la même étiquette (voir cas 2.(b) de la suppression de sommet et la figure 4.9), alors, comme ci-dessus, tous les descendants de $\tilde{q}$ sont uniformes. Soit $\mathcal{S} = \mathcal{C}(p')$ et $S = P'$. $S \cup \{x\} = Q'$ est un module de $G'[\tilde{Q} \cup \{x\}] = G'[\tilde{Q}']$. Comme $\tilde{q}$ est mixte, il satisfait la condition 3.(a) du théorème 4.10 avec $\mathcal{S} = \{p\}$. La preuve d'unicité d'un tel ensemble $\mathcal{S}$ dans le cas présent est identique à celle du cas précédent.

■

Corollaire 4.1 *Si $G + x$ est un cographe orienté, il existe un unique noeud q tel que l'ensemble des noeuds mixtes de T^m est exactement $\text{Anc}(q)$. q est l'unique noeud mixte terminal, et l'unique noeud mixte sans fils mixte.*

Preuve : Si $G + x$ est un cographe orienté, alors les conditions du théorème 4.10 sont satisfaites. Supposons qu'il existe deux noeuds mixtes distincts q_1 et q_2 de T^m tels que $q_1 \notin \text{Anc}(q_2)$ et $q_2 \notin \text{Anc}(q_1)$. Alors $p = \text{ppca}(q_1, q_2)$ est différent de q_1 et de q_2 . Soit p_1 (resp. p_2) l'unique noeud appartenant à $\mathcal{C}(p) \cap \text{Anc}(q_1)$ (resp. $\mathcal{C}(p) \cap \text{Anc}(q_2)$). Par définition d'un noeud mixte, tout ancêtre d'un noeud mixte l'est aussi. Comme $p_1 \in \text{Anc}(q_1)$, p_1 est mixte, et de la même manière p_2 est mixte. Le noeud p est mixte et a deux fils mixtes, ce qui réfute les conditions du théorème 4.10. En conclusion, les noeuds mixtes sont totalement ordonnés par la relation "est ancêtre de".

Soit q le plus bas noeud mixte dans T^m . Tous les noeuds mixtes appartiennent à $\text{Anc}(q)$, et comme tout ancêtre d'un noeud mixte est mixte, l'ensemble des noeuds mixtes est exactement $\text{Anc}(q)$.

■

Preuve de l'implication réciproque du théorème 4.10 : $\Leftarrow$. Si les conditions du théorème 4.10 sont vérifiées, nous construisons un coarbre T' en insérant x dans T^m . Nous montrons que les relations d'adjacence définies par T' entre les sommets de V sont celles de G , et que les relations d'adjacence définies par T' entre x et les sommets de V sont les relations définies par la donnée des voisinages $N^+(x)$ et $N^-(x)$ de x . Comme T' est construit uniquement avec des noeuds parallèles, séries et ordres, et comme T' est un arbre de décomposition modulaire, T' est un coarbre. De plus, par unicité de l'arbre de décomposition modulaire, T' est l'arbre de décomposition modulaire de G' qui est donc un

cographe orienté.

Comme établi dans la preuve du corollaire 4.1, si les conditions du théorème 4.10 sont vérifiées, alors il existe un unique noeud mixte q de T^m tel que les noeuds mixtes de T^m sont précisément les noeuds du chemin P_q^r de q à la racine r de T^m .

Lemme 4.2 $Q \cup \{x\}$ est un module fort de G' .

Preuve : $\forall p \in \text{Anc}(q)$, p est mixte. Si p n'est pas la racine, son père $\tilde{p}$ est mixte et a un fils mixte. Puisque les conditions du théorème 4.10 sont vérifiées, $\tilde{p}$ est simplement mixte. Par conséquent, p est l'unique fils mixte de $\tilde{p}$ et $P \cup \{x\}$ est un module de $G'[\tilde{P}]$. Par récurrence, on montre que $Q \cup \{x\}$ est un module de G' . Montrons que $Q \cup \{x\}$ est fort. Supposons qu'il existe un module M' de G' qui chevauche $Q \cup \{x\}$. Nécessairement, $M = M' \setminus \{x\} \neq \emptyset$. Il s'ensuit que M est un module de G , et comme Q est fort, M ne le chevauche pas. Comme $M' \setminus (Q \cup \{x\}) \neq \emptyset$, alors $M \not\subseteq Q$. D'où, $M \cap Q = \emptyset$ et $x \in M'$, ou $Q \subseteq M$. Dans le premier cas, $(Q \cup \{x\}) \setminus M' = Q$ est un module de G' . Dans le second cas ($Q \subseteq M$), comme M' chevauche $Q \cup \{x\}$, alors $x \notin M'$. Donc, $(Q \cup \{x\}) \cap M' = Q$ est un module de G' . Comme Q ne contient pas x , dans les deux cas Q est uniforme : absurde. ■

Par voie de conséquence, construire T' se réduit à insérer x dans T_q . De cette façon, on obtient les adjacences désirées entre x et les sommets de $V \setminus Q$. Nous discutons maintenant de la façon d'insérer x dans T_q (voir figure 4.11).

- Si q satisfait la condition **3.(b) du théorème 4.10**, alors $S \cup \{x\}$ et $(Q \setminus S) \cup \{x\}$ sont des modules de $G'[Q \cup \{x\}]$. D'après le lemme 4.1, on a S et $Q \setminus S$ qui sont des modules de $G[Q]$. D'où, $\mathcal{S}$ et $\mathcal{C}(q) \setminus \mathcal{S}$ sont des intervalles de l'ordre défini par q sur ses fils. Sans perte de généralité, supposons que $\mathcal{S}$ est le premier intervalle dans cet ordre. Comme $S \cup \{x\}$ est un module de G' , $Q \setminus S \subseteq N^+(x)$ et comme $(Q \setminus S) \cup \{x\}$ est un module de G' , $S \subseteq N^-(x)$. Il en découle qu'en insérant la feuille l_x en tant que fils de q entre $\mathcal{S}$ et $\mathcal{C}(q) \setminus \mathcal{S}$, on obtient les relations d'adjacence souhaitées entre x et les sommets de Q .
- Si q satisfait la condition **3.(a) du théorème 4.10**, alors $S \cup \{x\}$ est un module de $G'[Q \cup \{x\}]$. D'après le lemme 4.1, S est un module de $G[Q]$. Donc, les sommets de $Q \setminus S$ ont la même relation d'adjacence avec x qu'avec les sommets de S . Pour cette raison, et comme S est uniforme, on peut insérer x comme indiqué ci-après.
- Si $|\mathcal{S}| \geq 2$, les noeuds de $\mathcal{S}$ deviennent les fils d'un nouveau noeud p_1 , en prenant soin de ne pas changer leur ordre relatif si $\text{label}(q) = \text{ordre}$. On donne à p_1 la même étiquette que q . Un nouveau noeud p_2 est créé, auquel est assigné l'étiquette correspondant à la relation d'adjacence entre x et les sommets de S . Plus formellement, la relation de correspondance entre les étiquettes et les types (défini dans la procédure de marquage) des noeuds sera notée $\equiv$ et est définie par : série $\equiv$ InOut, parallèle $\equiv$ None, ordre $\equiv$ In et ordre $\equiv$ Out. Remarquons que l'étiquette de p_2 est nécessairement différente de celle de p_1 , sinon $\mathcal{S}$ ne serait pas unique comme exigé par la condition **3.(a)**. l_x et p_1 deviennent fils de p_2 , dans l'ordre qui convient si

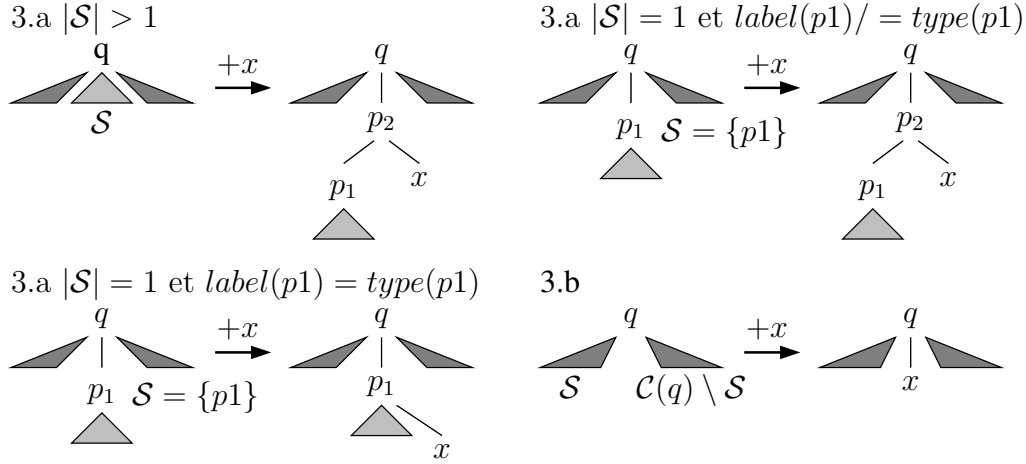


FIG. 4.11 – Modification du coarbre orienté par une insertion de sommet. $\equiv$ dénote la relation d'équivalence entre les étiquettes et les types des noeuds.

$label(p_2) = ordre$. Finalement, p_2 devient un fils de q . Si q est un noeud ordre, les noeuds de $\mathcal{S}$ forment un intervalle dans l'ordre défini par q , et p_2 est inséré dans les fils de q à la place de cet intervalle.

- Si $|\mathcal{S}| = 1$, soit p_1 son unique élément. Si la relation d'adjacence entre x et les sommets de P_1 correspond à l'étiquette de p_1 , alors l_x devient fils de p_1 , à la place qui convient si p_1 est un noeud ordre. Sinon, un nouveau noeud p_2 est créé et on lui donne l'étiquette correspondant à la relation d'adjacence entre x et les sommets de P_1 . l_x et p_1 deviennent fils de ce nouveau noeud, dans l'ordre qui convient si $label(p_2) = ordre$.

Le lecteur peut vérifier qu'en insérant x dans T^m nous n'avons pas changé les relations d'adjacence entre les sommets de V et nous avons obtenu les relations souhaitées entre x et les sommets de V . Comme nous n'avons fait usage que de noeuds parallèles, séries et ordres, nous avons obtenu un coarbre orienté qui est l'arbre de décomposition modulaire de G' . Donc G' est un cografe orienté. ■

La procédure de marquage.

La première étape de l'algorithme colore les noeuds de l'arbre de décomposition modulaire T^m suivant le voisinage du sommet x à insérer. Cette étape préliminaire est une extension directe de la procédure de marquage de [CPS85].

Initialement, toutes les feuilles l_y telles que $y \in N(x)$ sont colorées en rouge. Selon la relation d'adjacence entre y et x , ces feuilles reçoivent un **type** : $type(l_y) = In$ si $yx \in E$ et $xy \notin E$; $type(l_y) = Out$ si $xy \in E$ et $yx \notin E$; ou $type(l_y) = InOut$ si $xy \in E$ et $yx \in E$. La procédure est un parcours de bas en haut de l'arbre : chaque noeud rouge p transmet son type à son père q et selon les différents types reçus par q , une couleur lui est attribuée. La première fois qu'un noeud interne reçoit un type, il est coloré en noir. Un noeud q devient rouge si tous ses fils ont le même type (i.e. l'ensemble de sommets Q est uniformément lié). Un noeud rouge se voit attribuer le type de ses enfants. Une fois qu'un noeud rouge

```

Type( $G, T^m, \mathcal{R}$  un ensemble de noeuds rouges typés)
1. Tant que il existe un noeud rouge  $p$  Faire
2.    $couleur(p) \leftarrow gris$ 
3.   Si  $p$  n'est pas la racine de  $T^m$  Alors
4.     Soit  $q$  le père de  $p$ 
5.     Ajouter  $p$  à la liste  $filsgris(q)$ 
6.     Ajouter un à  $\#type(q, type(p))$ 
7.     Si  $\#type(q, type(p)) = \#fils(q)$  Alors
8.        $couleur(q) \leftarrow rouge$  et  $type(q) \leftarrow type(p)$ 
9.     Sinon  $couleur(q) \leftarrow noir$ 
10. Fin Tant que

```

FIG. 4.12 – Marking process.

a transmis son type a son père, il devient gris. La racine ne peut pas suivre cette règle : elle devient grise sitôt qu'elle est devenue rouge (si cela arrive). Afin de préparer l'insertion éventuelle de x , chaque noeud atteint par le procédé maintient une liste de ses fils gris. La procédure s'arrête lorsqu'il ne reste plus de noeuds rouges.

Pour des raisons de convenance, nous dirons que la couleur par défaut d'un noeud est blanc. L'absence de type peut être considéré comme un type particulier et on note $type(p) = None$. Pour des raisons de complexité, il est important de remarquer que les noeuds visités par la procédure de marquage se retrouvent affublés d'une couleur autre que blanc et d'un type autre que $None$ à la fin de la procédure. Cela vient du fait que seules les feuilles liées à x sont visitées et colorées en rouge au début de la procédure.

Dans notre procédure de marquage, chaque noeud stocke la liste et le nombre de ses fils gris, ainsi que trois compteurs $\#type(q, t)$ pour tout type $t \in \{In, Out, InOut\}$ (eg. $\#type(q, In)$ indique le nombre de fils de q dont le type est In).

Nous affirmons, sans en fournir de preuve, les propriétés de bases du coarbre coloré T^c résultant de la procédure de marquage. Elle sont nécessaires pour la compréhension de l'algorithme d'insertion et de la production d'un certificat.

Lemme 4.3 *Les noeuds de T^c qui sont uniformément liés à x sont exactement les noeuds gris ; les noeuds de T^c qui sont non liés à x sont blancs ; et les noeuds noirs sont mixtes.*

Remarque 4.2 Un noeud blanc peut être mixte. Il l'est ssi il est lié.

Il s'ensuit que les noeuds mixtes sont noirs ou blancs. L'ensemble des noeuds noirs est noté $\mathcal{B}$.

Lemme 4.4 *Après la procédure de marquage, un noeud blanc qui est lié à x a un descendant noir.*

Preuve : Soit w un noeud blanc lié à x . Comme w est blanc, il n'est pas uniformément lié à x : il est mixte. Soit v un noeud mixte minimal de T_w , c'est à dire un noeud mixte n'ayant pas de descendant mixte. Les fils de v sont tous uniformes et v est mixte, par conséquent au moins un de ses fils v_c est uniformément lié à x . V_c est gris et v est noir. ■

Le temps d'exécution de la procédure de marquage (procédure **Type** de la figure 4.12) est $O(d(x))$, et les nombres de noeuds gris et de noeuds noirs sont bornés par $O(d(x))$. La partie du coarbre T^m parcourue par la procédure **Type** est constituée des noeuds gris et noirs et des arêtes entre ces noeuds. Considérons d'abord l'arbre restreint aux noeuds gris. C'est une forêt dans laquelle les feuilles sont liées à x et les noeuds internes ont au moins deux fils. comme le nombre de feuilles est $O(d(x))$, c'est aussi le cas du nombre de noeuds gris. Comme les noeuds noirs ont au moins un fils gris et comme un noeud a au plus un père, alors le nombre de noeuds noirs est aussi $O(d(x))$. Chaque arête de la restriction de T^m aux noeuds noirs et gris est traversée une seule fois au cours de la procédure **Type** et chaque noeud est traité en temps constant. La procédure **Type** s'exécute donc en temps $O(d(x))$.

Tester l'insertion.

Afin de tester si l'insertion de x est possible ou non, le théorème 4.11 exprime les conditions du théorème 4.10 en termes de noeuds colorés. L'algorithme vérifie les conditions du théorème 4.11 dans T^c , en ne considérant que les couleurs des noeuds.

Théorème 4.11 *Soit G un cographe orienté et T^m son coarbre orienté. $G + x$ est un cographe orienté ssi il existe un noeud noir q de T^m tel que :*

1. *tous les noeuds noirs appartiennent à P_q^r ,*
2. *tout noeud noir de $P_{parent(q)}^r$ est un noeud série ou ordre simplement mixte,*
3. *tout noeud blanc de $P_{parent(q)}^r$ est un noeud parallèle et*
4. *q est mixte terminal.*

A l'évidence, les conditions du théorème 4.11 sont très proches des conditions du théorème 4.10. Le lemme 4.5 établit les relations, lorsque $G + x$ est un cographe orienté, entre la couleur et l'étiquette des noeuds simplement mixtes de T^c . La différence principale entre les deux théorèmes est que les conditions du théorème 4.11 ne demandent pas que les noeuds parallèles blancs de $P_{parent(q)}^r$ soient simplement mixtes. Le lemme 4.6 montre que cela n'est pas utile car sous les conditions du théorème 4.11, les noeuds parallèles blancs de $P_{parent(q)}^r$ sont nécessairement simplement mixtes. Le fait que l'algorithme n'est pas à tester cette condition est crucial pour la complexité finale de l'insertion ($O(d(x))$).

Lemme 4.5 *Soit p un noeud simplement mixte de T^c . p est soit un noeud parallèle blanc, soit un noeud série noir, soit un noeud ordre noir.*

Preuve : Soit p un noeud série ou ordre simplement mixte, et f son unique fils mixte. Alors, $F \cup \{x\}$ est un module de $G'[P]$. Il en découle que tout fils h de p différent de f est

uniformément lié à x . Donc, h est gris et p est noir.

Soit maintenant p un noeud parallèle simplement mixte. Alors, $F \cup \{x\}$ est un module de $G'[P]$. Ce qui implique que tout fils h de p différent de f est non lié à x . Donc h est blanc. Comme f est mixte, f n'est pas gris. Par conséquent p est blanc. ■

Lemme 4.6 *Si il existe un noeud noir q tel que tout noeud noir appartient à P_q^r et que tout noeud blanc de $P_{parent(q)}^r$ est parallèle, alors les noeuds blancs de P_q^r sont simplement mixtes.*

Preuve : Soit p un noeud blanc de P_q^r . Comme $p \in Anc(q) \setminus \{q\}$, p est mixte et possède un fils mixte $p_1 \in Anc(q)$. Soit $h \in \mathcal{C}(p) \setminus \{p_1\}$. Supposons que h est lié à x , alors, d'après le lemme 4.4, h a un descendant f coloré noir, qui n'est pas sur le chemin P_q^r : absurde. Donc h n'est pas lié à x , de même que ses descendants, qui sont tous blanc (voir lemme 4.3). De plus, comme p est un noeud blanc de P_q^r , p est parallèle. p a unique fils mixte p_1 et ses autres fils ne sont pas liés à x . Par conséquent, $P_1 \cup \{x\}$ est un module de $G'[P \cup \{x\}]$: p est simplement mixte. ■

Grâce aux deux lemmes précédents, on montre l'équivalence entre les conditions du théorème 4.10 et celles du théorème 4.11, ce qui prouve ce dernier.

Preuve du théorème 4.11 : $\implies$. Si $G+x$ est un cographe orienté, alors les conditions du théorème 4.10 sont satisfaites. D'après le corollaire 4.1, les noeuds mixtes de T^m induisent un chemin de la racine à un certain noeud mixte q , qui est coloré en noir. En effet, q est mixte et ses fils sont uniformes. Donc, au moins un de ses fils est uniformément lié à x . Ce fils est gris et q est noir. Comme les noeuds noirs sont mixtes, et comme, d'après le corollaire 4.1, les noeuds mixtes sont sur le chemin P_q^r , alors les noeuds noirs appartiennent tous à P_q^r . La condition 1 est vérifiée. Les conditions du théorème 4.10 impliquent que les noeuds de $P_{parent(q)}^r$ sont simplement mixtes. Le lemme 4.5 implique que les noeuds simplement mixtes noirs sont séries ou ordres, et les noeuds simplement mixtes blanc sont parallèles. Les conditions 2 et 3 sont vérifiées. La condition 4 est strictement identique à la condition 3 du théorème 4.10.

$\impliedby$. On montre que si les conditions du théorème 4.11 sont vérifiées, alors les conditions du théorème 4.10 le sont aussi. Comme q est mixte terminal, par définition, il satisfait la condition 1 du théorème 4.10. D'après la condition 2 du théorème 4.11, les noeuds noirs de $P_{parent(q)}^r$ sont simplement mixtes. D'après les conditions 1 et 3 du théorème 4.11, et comme le noeud q est noir, le lemme 4.6 s'applique. On en déduit que les noeuds blancs de $P_{parent(q)}^r$ sont simplement mixtes. Comme $P_{parent(q)}^r$ ne contient que des noeuds noirs ou blancs (tous les noeuds de $P_{parent(q)}^r$ sont mixtes), les noeuds de $P_{parent(q)}^r$ vérifient la condition 2 du théorème 4.10. Il s'ensuit que les noeuds de $T \setminus (Anc(q) \cup Desc(q))$ sont uniformes, ils vérifient la condition 1 du théorème 4.10. ■

La procédure **Check** (voir figure 4.13) teste si les conditions du théorème 4.11 sont vérifiées ou non. Si ces conditions sont réunies, l'insertion du sommet x est réalisée par

```

Check( $G, T^c, \mathcal{B}, x$ )
1.  $bottom \leftarrow r$ , où  $r$  est la racine de  $T^c$ 
2. Tant que il existe un noeud  $q$  dans  $\mathcal{B}$  Faire
3.    $p \leftarrow q$  et retirer  $q$  de  $\mathcal{B}$ 
4.   Tant que  $p \neq bottom$  Faire
5.      $p \leftarrow parent(p)$ 
6.     Si  $p$  a déjà été visité Alors Find-Certificate( $p$ )
7.     Si  $p$  est un noeud blanc non-parallèle Alors Find-Certificate( $p$ )
8.     Si  $p \in \mathcal{B}$  n'est pas un noeud simplement mixte Alors Find-Certificate( $p$ )
9.     Si  $p \in \mathcal{B}$  Alors retirer  $p$  de  $\mathcal{B}$ 
10.    Marquer  $p$  comme étant visité
11.  Fin Tant que
12.   $bottom \leftarrow q$ 
13. Fin Tant que
14. Si  $q$  est un noeud mixte terminal Alors Insert( $x, q, T^c$ )
15. Sinon Find-Certificate( $q$ )

```

FIG. 4.13 – Test d'insertion du sommet x . T^c est le coarbre orienté coloré de G et $\mathcal{B}$ est l'ensemble des noeuds noirs de T^c .

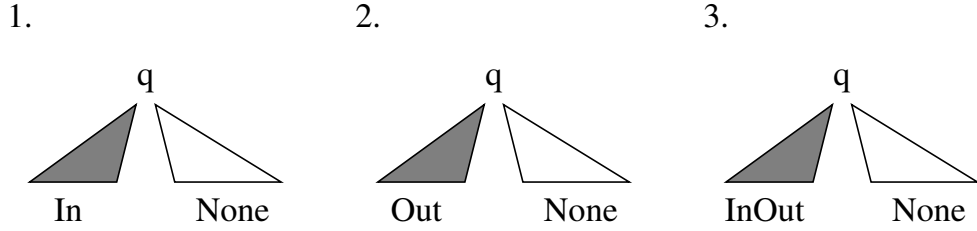
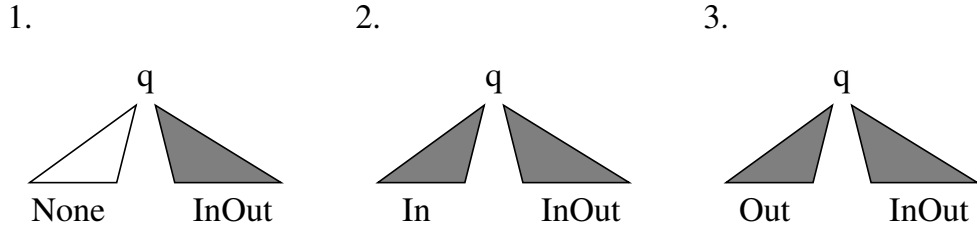
la procédure **Insert**. Si l'une d'elle n'est pas vérifiée, alors un appel à la procédure **Find-Certificate** permet de trouver un ensemble Z de trois sommets tels que $G'[Z \cup \{x\}]$ contient un des sous-graphes exclus de la figure 4.7, avec $G' = G + x$. Les procédures **Insert** et **Find-Certificate** sont décrites plus loin.

Soit p le noeud courant de la procédure **Check**. Si p a déjà été visité (test ligne 6), la condition 1 du théorème 4.11 n'est pas satisfaite et G' n'est pas un cographe orienté. Les tests des lignes 7 et 8 testent si p vérifie les conditions 2 et 3 du théorème 4.11. La condition 4 est testée à la ligne 14.

Détaillons le test vérifiant si un noeud série ou ordre noir est simplement mixte (ligne 8), et celui vérifiant si q est mixte terminal (ligne 14).

Soit p un noeud série noir ou un noeud ordre noir. Pour que p soit simplement mixte, il faut que tous ses fils sauf un soient gris. Si p est un noeud série, ses fils distincts de son unique fils non gris q doivent être de type *InOut*. Si p est un noeud ordre, ses fils qui se trouvent avant (resp. après) q dans l'ordre défini par p doivent être de type *In* (resp. *Out*). Soit q le noeud testé par la procédure **Check** à la ligne 14. Il n'y a aucune contrainte sur l'étiquette de q . Pour tout $t \in \{In, Out, InOut\}$, on note $\#type(t)$ le nombre de fils de q de type T^m , et on note $\#grey$ le nombre de fils de q colorés en gris. On peut tester si q est mixte terminal de la manière suivante :

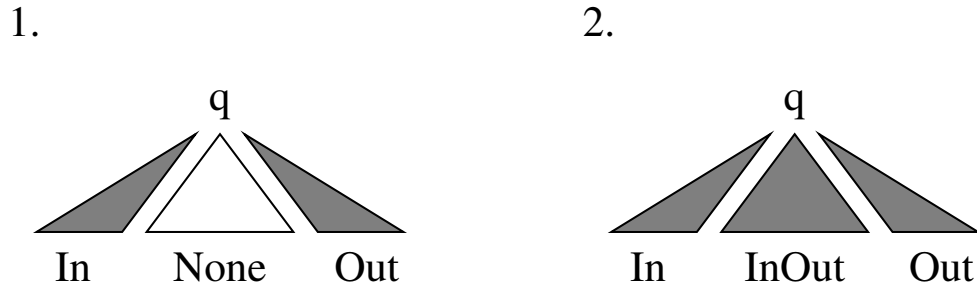
- si q est un noeud parallèle (voir figure 4.14) : on vérifie que $\#type(In) = \#grey$ ou $\#type(Out) = \#grey$ ou $\#type(InOut) = \#grey$ (puisque dans ce cas, si q est mixte

FIG. 4.14 – Les trois cas dans lesquels q est un noeud parallèle mixte terminal.FIG. 4.15 – Les trois cas dans lesquels q est un noeud série mixte terminal.

terminal, tout noeud de $\mathcal{S}$ est gris, où $\mathcal{S}$ est l'ensemble défini dans la condition 3.a du théorème 4.10 ;

- Si q est un noeud série (voir figure 4.15) : on vérifie que $\#type(In) + \#type(InOut) = |\mathcal{C}(q)|$ ou $\#type(Out) + \#type(InOut) = |\mathcal{C}(q)|$ ou $\#type(In) = \#type(Out) = 0$;
- si q est un noeud ordre (voir figure 4.16) : On vérifie d'abord que $\#type(InOut) + \#type(In) + \#type(Out) = |\mathcal{C}(q)|$ ou bien $\#type(InOut) = 0$. Ensuite on vérifie que les $\#type(In)$ premiers fils de q (dans l'ordre défini par le noeud q) sont typés In et que les $\#type(Out)$ derniers sont typés Out .

Tester si un noeud série noir p est simplement mixte peut être fait en temps constant. Si p est un noeud ordre, il faut parcourir ses fils lorsque $|\mathcal{C}(p)| = \#grey + 1$ (sinon, p n'est pas simplement mixte), cela prend un temps $O(d(x))$. Le test pour déterminer si un noeud série ou parallèle q est mixte terminal s'effectue en temps constant. En revanche, si q est un noeud ordre, le test s'effectue en deux recherches dans la liste des fils de q . La première vérifie que les $\#type(In)$ premiers fils de q sont de type In et la seconde vérifie

FIG. 4.16 – Les deux cas dans lesquels q est un noeud ordre mixte terminal.

que les $\#type(Out)$ derniers fils de q sont de type *out*. Si l'une des deux recherches trouve un noeud blanc, alors elle prend fin. Par voie de conséquence, seuls les fils gris de q sont visités plus éventuellement un noeud blanc. Comme le nombre de noeuds gris est $O(d(x))$, la procédure **Check** s'exécute en temps $O(d(x))$.

Insertion d'un sommet.

Lorsque la procédure **Check** détermine que $G + x$ est un cographe orienté, la procédure **Insert** doit effectuer l'insertion de x dans le coarbre T^m de G afin d'obtenir le coarbre T' de $G' = G + x$. Dans ce cas, les conditions du théorème 4.10 sont vérifiées. Soit q l'unique noeud mixte terminal (le noeud *bottom* de la procédure **Check**). d'après le lemme 4.2, $Q \cup \{x\}$ est un module de G' . On en déduit que les modifications de l'arbre se produisent dans le sous-arbre T_q de T^m enraciné en q . La mise à jour de T_q est réalisée exactement comme décrit dans la preuve du théorème 4.10, et illustré sur la figure 4.11. Il est à remarquer que lorsque l'insertion est possible, l'ensemble $\mathcal{S}$ du théorème 4.10 est donné par la procédure **Check**. Comme dans le cas de la suppression de sommet, la mise à jour du coarbre dans le cas 3.a quand $|\mathcal{S}| \geq 2$ (voir figure 4.11) demande une attention particulière afin de ne pas déplacer le non voisinage de x . Si les noeuds de $\mathcal{S}$ n'ont pas de type, on déconnecte les noeuds de $\mathcal{C}(p) \setminus \mathcal{S}$ de p et on les reconnecte sur un nouveau noeud ; dans le cas contraire, la manipulation de déconnexion-reconnexion est appliquée aux noeuds de $\mathcal{S}$.

A ce stade, nous devons faire la distinction entre le coarbre orienté, qui est un concept mathématique, et son implémentation en mémoire. La différence principale entre les deux est que la liste des fils d'un noeud p utilisée dans l'implémentation ordonne les fils de p d'une manière qui n'est pas pertinente lorsque p est série ou parallèle. Dans le paragraphe précédent, nous avons montré comment maintenir le coarbre (i.e. la relation de parenté) mais nous n'avons pas précisé la manière dont on peut obtenir une implémentation du coarbre, car il est très aisé d'imaginer une façon de le faire. Nous nous occupons maintenant d'actualiser la permutation factorisante et les pointeurs des noeuds du coarbre vers cette permutation. Ce faisant, nous actualisons aussi l'implémentation du coarbre (celle obtenue par la mise à jour du coarbre décrite au paragraphe précédent) de telle sorte que nous conservons la propriété selon laquelle la permutation factorisante est l'ordre dans lequel on rencontre les feuilles de T' dans un parcours en profondeur qui suit l'ordre des listes de fils de l'implémentation.

La mise à jour de la permutation factorisante, des pointeurs des noeuds du coarbre vers cette permutation, et de l'implémentation du coarbre peut être faite en temps $O(d(x))$. Nous montrons, en exemple, comment effectuer cette mise à jour dans le cas où le noeud mixte terminal q vérifie la condition 3.a du théorème 4.10 et $|\mathcal{S}| \geq 2$. Les autres cas sont plus simples car, pour ceux-là, il n'est pas nécessaire de trier les fils de q avant d'insérer x dans la permutation factorisante. Dans ce qui suit, p_1 et p_2 sont les noeuds introduits dans la preuve de l'implication réciproque du théorème 4.10 (page 136), et illustrés sur la figure 4.11.

- q est un noeud parallèle (voir l'exemple sur la figure 4.17), on déplace le noeud p_2 au début de la liste des fils de q . Ensuite, pour chaque fils u de p_1 , on découpe son

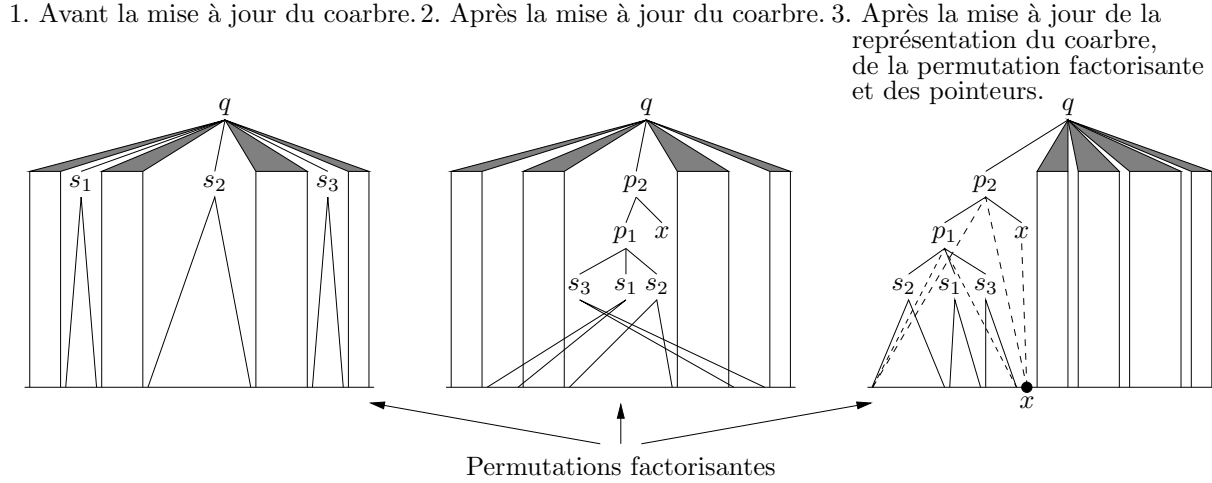


FIG. 4.17 – Modification de la structure de donnée après insertion d'un sommet. q est le noeud mixte terminal et $\mathcal{S} = \{s_1, s_2, s_3\}$. Les pointeurs sont symbolisés par des lignes (en pointillés pour la feuille x et les noeuds p_1 et p_2) qui vont des noeuds à la permutation factorisante.

intervalle dans la permutation et on le place au début de l'intervalle de q ; on place u au début de la liste des fils de p_1 . Cela assure la complexité recherchée car, dans ce cas (q parallèle), les fils de p_1 sont gris et par conséquent leur nombre est $O(d(x))$. p_1 se voit attribuer le premier pointeur de son fils déplacé en dernier et le second pointeur de son fils déplacé en premier.

- Si p_2 est un noeud ordre et x est son premier fils, alors x est inséré dans la permutation factorisante juste avant le premier sommet de P_1 , et p_2 reçoit un pointeur vers x et un vers le dernier sommet de P_1 . La feuille de x est déplacée au début de la liste des fils de p_2 .
- Sinon, x est inséré après le dernier sommet de P_1 , et p_2 reçoit un pointeur vers le premier noeud de P_1 et un autre vers x . La feuille de x est déplacée à la fin de la liste des fils de p_2 .
- Si q est un noeud série, on procède de façon similaire. On déplace p_2 au début de la liste des fils de q . Pour préserver la complexité, on ne peut pas toucher aux fils de p_1 , qui pourraient être colorés en blanc. Au lieu de cela, pour chaque fils de q différent de p_2 , on découpe son intervalle dans la permutation factorisante, et on le place à la fin de l'intervalle de q . La place attribuée à x et les pointeurs de p_2 sont les mêmes que dans le cas précédent.
- Si q est un noeud ordre, l'ordre de ses fils n'a pas besoin d'être modifié. p_1 reçoit le premier pointeur du premier noeud de $\mathcal{S}$, et le second pointeur du dernier noeud de $\mathcal{S}$. x est inséré après le dernier sommet de P_1 , et p_2 reçoit le premier pointeur de p_1 et un pointeur vers x . La feuille de x est déplacée à la fin de la liste des fils de p_2 .

La mise à jour du nombre de fils des noeuds du coarbre se fait en temps constant. Dans le cas 3.a lorsque $|\mathcal{S}| \geq 2$, si les noeuds de $\mathcal{S}$ sont liés à x , on connaît leur nombre grâce aux

compteurs utilisés dans la procédure de marquage. Si les noeuds de $\mathcal{S}$ ne sont pas liés à x , les noeuds de $\mathcal{C}(q) \setminus \mathcal{S}$ sont liés et leur nombre est connu. Dans les deux cas, par différence, on déduit à la fois $|\mathcal{S}|$ et $|\mathcal{C}(q) \setminus \mathcal{S}|$. Et ainsi, on peut assigner leurs valeurs correctes aux nombres de fils de q, p_1 et p_2 . Les autres cas sont laissés au lecteur, ils sont encore plus simples.

Production d'un certificat.

Afin d'éviter une analyse de cas lourde, on donne une version de la procédure **Find-Certificate**(p) qui ne fournit pas un certificat exact, mais un ensemble de quatre sommets dont le graphe induit contient un des graphes exclus de la figure 4.7.

Lemme 4.7 *Si $G' = G + x$ n'est pas un cografe orienté, on peut trouver en temps $O(d(x))$ un ensemble $Z = \{a, b, c\}$ de trois sommets tel que $G'[Z \cup \{x\}]$ contient un des graphes exclus de la figure 4.7.*

Pour chaque appel à **Find-Certificate**, on montre sur deux exemples comment extraire un certificat minimal de l'ensemble $Z \cup x$ retourné. Il est laissé le soin au lecteur de compléter l'analyse de cas et de s'assurer qu'il est toujours possible de trouver un graphe exclu minimal dans la même complexité.

La procédure **Find-Certificate**(p) fait aussi usage de P_{bottom}^r et P_q^p où *bottom* et q sont les noeuds définis respectivement aux lignes 12 et 2 de la procédure **Check**. Grâce aux listes des fils gris de chaque noeud de T_c et à la permutation factorisante, la recherche s'effectue en temps $O(d(x))$. L'appel à **Find-Certificate** se produit : à la ligne 6, si le noeud courant p a déjà été visité auparavant ; aux lignes 7 et 8, si le noeud p n'est pas simplement mixte ; à la ligne 15, si le dernier noeud visité q n'est pas mixte terminal. Pour chaque appel, on montre comment trouver les trois sommets a, b, c de Z .

Les noeuds noirs ne se répartissent pas sur un chemin se terminant à la racine (Ligne 6). dans ce cas p est nécessairement un noeud parallèle, sinon, **Check** aurait découvert que p n'est pas simplement mixte, puisque p a au moins deux fils mixtes. Ces deux fils mixtes ont déjà été visités par la procédure **Check**. Ce sont le fils h de p sur le chemin P_{bottom}^r , et le fils h' de p sur le chemin P_q^p . Les noeuds h et h' sont noirs, sinon, **Check** se serait arrêtée du fait qu'il ne sont pas parallèles. Ils ont donc tous les deux reçu le type d'un de leur fils gris, notés respectivement k et k' . Soit a un sommet de K et b un sommet de K' . Comme h' est mixte et k' est uniforme, il existe un sommet $c \in H' \setminus K'$ tel que $type(c) \neq type(b)$. Voir les exemples de la figure 4.18.

Le noeud courant n'est pas simplement mixte (lignes 7 ou 8). Soit p le noeud actuellement visité par la procédure **Check**. S'il s'agit de la première occurrence de la boucle interne (lignes 4 - 11), alors p est le père du noeud q choisi parmi les noeuds noirs restants dans $\mathcal{B}$, à la ligne 2. Donc p a un fils mixte. Si nous ne sommes pas dans la première occurrence de la boucle interne alors, au cours de l'occurrence précédente de la boucle, un noeud h a été visité. Si h n'est pas noir, alors c'est un noeud parallèle blanc, sinon l'algorithme se serait arrêté. On en déduit que h a un fils noir. En effet, si h a un fils déjà visité h' , h' est série ou ordre, puisque h est parallèle, et h' est noir, sinon l'algorithme se serait arrêté en le visitant. Si h n'a pas de fils déjà visité, alors h a été visité lors de la

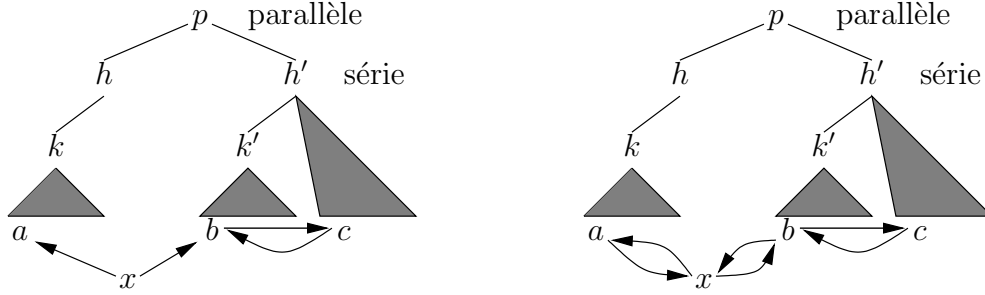


FIG. 4.18 – Comme p est un noeud parallèle, h' est soit série soit ordre. Supposons qu'il soit série, dans ce cas, les arcs bc et cb sont tous les deux présents. Dans le premier exemple, le certificat est induit par $\{b, c, x\}$, dans le second il est induit par $\{a, b, c, x\}$.

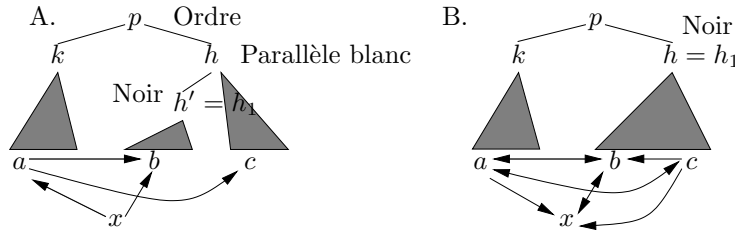


FIG. 4.19 – Dans le cas A., comme $type(c) \neq Out$, $G'[\{a, c, x\}]$ est un certificat. Dans le cas B., $G'[\{a, b, c, x\}]$ est un certificat.

première occurrence de la boucle interne et possède un fils noir qui est le noeud q choisi parmi ceux de $\mathcal{B}$ à la ligne 2 pour initialiser la boucle interne. Nous venons de prouver que p a soit un fils noir, soit un petit-fils noir, que l'on note $\tilde{h}$ dans les deux cas. Soit b un sommet appartenant à un fils gris de $\tilde{h}$. Soit c un sommet de H tel que $type(c) \neq type(b)$. Si p n'est pas simplement mixte, alors il existe $k \in \mathcal{C}(p) \setminus \{q\}$ tel que $type(k)$ ne correspond pas à l'étiquette de p . Plus précisément, si p est un noeud parallèle, $type(k) \neq None$; si p est un noeud série, $type(k) \neq InOut$; et si p est un noeud ordre, $type(k) \neq In$ et k est avant q dans l'ordre défini sur les fils de p , ou $type(k) \neq Out$ et k arrive après q dans les fils de p . Soit a un sommet de K dont le type ne correspond pas à l'étiquette de p . Remarquons que si p est un noeud parallèle, k est uniformément lié à x et n'importe quel sommet $a \in K$ convient. Voir les exemples de la figure 4.19.

Le noeud *bottom* n'est pas mixte terminal (Ligne 15). A ce stade de l'algorithme, nous savons que les noeuds noirs se répartissent sur un chemin P_q^r depuis un noeud q jusqu'à la racine r de T^m . Comme q est le plus bas des noeuds noirs, ses fils sont gris ou blancs. Le lemme 4.4 implique que les fils blanc de q sont uniformes. L'appel à **Find-Certificate** se produit lorsque q n'est pas mixte terminal.

- Si q est un noeud parallèle ou série
Alors q a deux fils k et k' de types différents tels que leurs types soient différents de celui correspondant à l'étiquette de q .

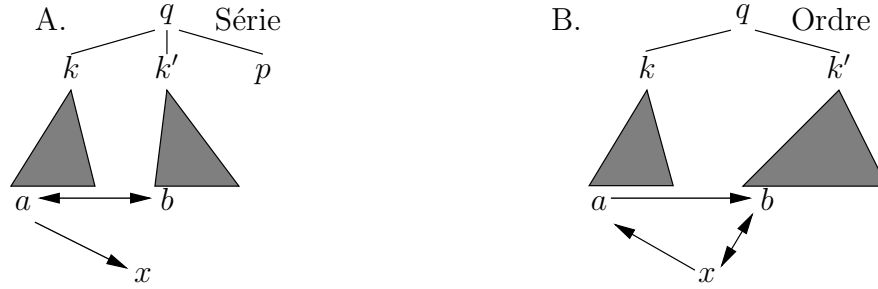


FIG. 4.20 – Dans le cas A., comme $\text{type}(k) = \text{In}$ et $\text{type}(k') = \text{None}$, $G'[\{a, b, x\}]$ est un certificat. Dans le cas B., comme $\text{type}(k) = \text{Out}$, $\text{type}(k') = \text{InOut}$ et $k < k'$ dans l'ordre défini par q , alors $G'[\{a, b, x\}]$ est un certificat.

- Si q est un noeud ordre
 - Si q a un fils gris de type InOut et un fils blanc, soient respectivement k et k' ces fils. Sinon, q a deux fils k et k' avec $k < k'$ et une des conditions suivantes est vérifiée :
 - $\text{type}(k) = \text{Out}$ et $\text{type}(k') = \text{In}$
 - $\text{type}(k) = \text{Out}$ et $\text{type}(k') = \text{none}$ ou InOut
 - $\text{type}(k) = \text{none}$ ou InOut et $\text{type}(k') = \text{In}$

On prend $a \in K$ et $b \in K'$. Dans cet appel à **Find-Certificate**, c n'est pas nécessaire car $G'[\{a, b, x\}]$ est un graphe exclu de la figure 4.7. Voir les exemples de la figure 4.20.

Pour chacun des appels à **Find-Certificate**, l'ensemble Z décrit ci-dessus peut être trouvé en temps $O(d(x))$. A cette fin, on parcourt l'arbre depuis le noeud p sur lequel l'appel s'effectue, en considérant un nombre constant de noeuds qui sont fils ou petit-fils de p . Pour chaque noeud considéré, on fait au plus une recherche dans ses fils gris, et au plus une recherche dans son segment de la permutation factorisante. Cette dernière recherche pourrait menacer la complexité requise de $O(d(x))$. Heureusement, lors d'une recherche dans la permutation factorisante, on cherche un sommet dont le type est différent d'un type spécifié choisi parmi In , Out et InOut , mais qui ne peut pas être None . Il s'ensuit que tous les noeuds visités lors de cette recherche, sauf éventuellement le dernier, sont liés à x . Remarquons la nécessité des pointeurs des noeuds du coarbre vers la permutation pour garantir l'accès en temps constant à l'ensemble de sommets représenté par un noeud. Comme annoncé, la complexité de la procédure **Find-Certificate** est $O(d(x))$.

4.3.4 Opérations dynamiques sur les arcs

Dans cette section, nous montrons comment traiter les modifications d'arc en temps $O(1)$. Nous ne présentons que la suppression d'arc. En complétant et en adaptant l'argument de [SS04], on obtient, à partir de l'algorithme de suppression, un algorithme d'insertion ayant la même complexité.

Comme la famille des cographes orientés est fermée par complémentarité, le graphe $G + xy$ est un cographe orienté ssi le graphe $\overline{G} - xy$ en est un. De même, un certificat

prouvant que $\overline{G} - xy$ n'est pas un cographe orienté, est un certificat pour $G + xy$. Le coarbre orienté du complémentaire d'un cographe orienté g s'obtient du coarbre de G en intervertissant les étiquettes séries et parallèles, et en renversant l'ordre des fils des noeuds ordres. La structure de donnée que nous utilisons permet de faire ces changements en temps constant pour chaque noeud du coarbre orienté. Comme notre algorithme de suppression ne manipule que le coarbre, il peut être adapté pour traiter l'insertion d'arc avec la même complexité.

Suppression d'un arc.

On peut distinguer deux types de modification d'arc. Le premier est la suppression simultanée de deux arcs symétriques xy et yx . Cette modification peut être comparée à la suppression d'une arête dans un cographe non orienté, voir [SS04]. La preuve du théorème 4 de [SS04] est parfaitement adaptable au cas plus général des cographes orientés (et au cas de la suppression d'après la discussion ci-dessus). Soit q_x (resp. q_y) le fils de p_{xy} ¹ contenant x (resp. y).

Théorème 4.12 *Le graphe $G' = G - \{xy, yx\}$ est un cographe orienté ssi $|Q_x| = 1$ et $Q_y \setminus \{y\} \subseteq \overline{N}(y)$, ou $|Q_y| = 1$ et $Q_x \setminus \{x\} \subseteq \overline{N}(x)$.*

Le théorème 4.13 étend le théorème 4.12 de manière à caractériser toute modification d'arc valide d'un cographe orienté. Rappelons que M_{xy} est le plus petit module de G contenant à la fois x et y ($M_{xy} \subseteq P_{xy}$).

Théorème 4.13 *Le graphe $G' = G - xy$ est un cographe orienté ssi une des conditions 1 et 2 est vérifiée.*

1. p_{xy} est un noeud ordre et $M_{xy} = Q_x \cup Q_y$ et une des deux conditions suivantes est vérifiée :
 - (a) $|Q_x| = 1$ et $Q_y \setminus \{y\} \subseteq \overline{N}(y)$, ou
 - (b) $|Q_y| = 1$ et $Q_x \setminus \{x\} \subseteq \overline{N}(x)$.
2. p_{xy} est un noeud série et une des deux conditions suivantes est vérifiée :
 - (a) $|Q_x| = 1$ et $Q_y \setminus \{y\} \subseteq N^+(y) \setminus N^-(y)$, ou
 - (b) $|Q_y| = 1$ et $Q_x \setminus \{x\} \subseteq N^-(x) \setminus N^+(x)$.

Preuve : On peut se restreindre à l'étude du sous-graphe $G[M_{xy}]$ de G induit par M_{xy} . En effet, comme M_{xy} contient à la fois x et y , alors M_{xy} est aussi un module de $G' = (V, E')$, avec $E' = E \setminus \{xy\}$. Donc $G' = (V, E')$ est un cographe orienté ssi $G'[M_{xy}]$ en est un.

$\implies$. Comme les modules de G' contenant à la fois x et y sont exactement les modules de G contenant x et y , alors M_{xy} est aussi un module de G' contenant x et y . Il s'ensuit que le module fort maximal Q'_x de $G'[M_{xy}]$ qui contient x et le module fort Q'_y qui contient y sont distincts. On note p'_{xy} la racine de l'arbre de décomposition modulaire de $G'[M_{xy}]$.

¹Pour mémoire, p_{xy} a été défini comme le $ppca(x, y)$

Si p_{xy} est un noeud ordre, $M_{xy} = \bigcup_{q_x \leq h \leq q_y} H$ en notant $\leq$ l'ordre sur les fils de p_{xy} . Après la suppression de l'arc xy , x et y ne sont plus adjacents. Comme Q'_x et Q'_y sont des modules forts distincts de $G'[M_{xy}]$, d'après le théorème 4.6, Q'_x et Q'_y sont des composantes connexes de $G'[M_{xy}]$. Donc, p'_{xy} est un noeud parallèle et ses fils sont exactement q'_x et q'_y , car M_{xy} est le module minimal de $G'[M_{xy}]$ qui contient x et y . Montrons que q_x et q_y sont en fait les seuls fils de p_{xy} . Supposons que $\exists h \in \mathcal{C}(p_{xy})$ tel que $q_x < h < q_y$. Soit $u \in H$, xu et uy appartiennent tous deux à E , et par conséquent à E' , ce qui contredit le fait que Q'_x et Q'_y sont des composantes connexes distinctes de $G'[M_{xy}]$. Donc, $M_{xy} = Q_x \cup Q_y$. Soit $u \in Q_x \setminus \{x\}$. uy appartient à E , et par conséquent à E' , donc u est dans la composante connexe Q'_y de y dans $G'[M_{xy}]$. Il s'ensuit que $u \in \overline{N}(x)$. De même, tout $v \in Q_y \setminus \{y\}$ appartient à Q'_x et $v \in \overline{N}(y)$. Finalement, supposons qu'il existe à la fois un sommet $u \in Q_x \setminus \{x\}$ et un sommet $v \in Q_y \setminus \{y\}$. Comme nous avons montré ci-dessus, dans $G'[M_{xy}]$, u est dans la composante connexe Q'_y de y et v dans la composante connexe Q'_x de x , qui sont distinctes. Mais uv appartient à E , donc à E' . Cela contredit le fait que Q'_x et Q'_y sont des composantes connexes distinctes de $G'[M_{xy}]$. En conclusion, si G' est un cographe orienté, $Q_x \setminus \{x\} = \emptyset$ ou $Q_y \setminus \{y\} = \emptyset$.

Si p_{xy} est un noeud série, alors $M_{xy} = Q_x \cup Q_y$. Après la suppression de l'arc xy , x et y sont liés par l'arc yx . Comme Q'_x et Q'_y sont des modules forts distincts de $G'[M_{xy}]$, d'après le théorème 4.6, il découle que Q'_x et Q'_y sont des composantes ordres de $G'[M_{xy}]$. Q'_y est la première, et Q'_x la dernière, dans l'ordre défini par p'_{xy} . Soit $u \in Q_x \setminus \{x\}$, comme uy et yu sont tous deux dans E , et par conséquent dans E' , alors u est dans la composante ordre Q'_y de y . D'où $u \in N^-(x)$. De manière analogue, tout $v \in Q_y \setminus \{y\}$ appartient à Q'_x et donc $v \in N^+(y)$. Pour finir, supposons qu'il existe à la fois un sommet $u \in Q_x \setminus \{x\}$ et un sommet $v \in Q_y \setminus \{y\}$. Comme nous l'avons montré précédemment, dans $G'[M_{xy}]$, u est dans la composante ordre Q'_y de y et v dans la composante ordre Q'_x de x , qui sont distinctes. Mais uv et vu appartiennent à E , et donc à E' . Ce qui contredit le fait que Q'_x et Q'_y sont des composantes ordres distinctes de $G'[M_{xy}]$. En conclusion, si G' est un cographe orienté, $Q_x \setminus \{x\} = \emptyset$ ou $Q_y \setminus \{y\} = \emptyset$.

$\Leftarrow$. Montrons que si les conditions du théorème 4.13 sont vérifiées, $G'[M_{xy}]$ est un cographe orienté (voir figure 4.21).

Si p_{xy} est un noeud ordre et si $|Q_x| = 1$, $\{x\}$ est une composante ordre de $G[M_{xy}]$. Comme $Q_y \setminus \{y\} \subseteq \overline{N}(y)$, après la suppression de xy , y n'est plus adjacent à x et $\{y\}$ devient une composante connexe de $G'[M_{xy}]$. L'autre composante connexe est $\{x\} \cup (Q_y \setminus \{y\}) = Q'_x$ et $\{x\}$ est en composition ordre avec $Q_y \setminus \{y\}$. Comme $G'[Q'_x \setminus \{x\}] = G[Q_y \setminus \{y\}]$ est un cographe orienté, alors $G'[M_{xy}]$ est un cographe orienté. Le cas où $|Q_y| = 1$ est similaire.

Si p_{xy} est un noeud série et si $|Q_x| = 1$, $\{x\}$ est une composante connexe de $G[M_{xy}]$. Comme $Q_y \setminus \{y\} \subseteq N^+(y) \setminus N^-(y)$, après la suppression de xy , x appartient à $N^+(y) \setminus N^-(y)$ et $\{y\}$ devient une composante ordre de $G'[M_{xy}]$. L'autre composante ordre est $\{x\} \cup (Q_y \setminus \{y\}) = Q'_x$ et $\{x\}$ est en composition série avec $Q_y \setminus \{y\}$. Comme $G'[Q'_x \setminus \{x\}] = G[Q_y \setminus \{y\}]$ est un cographe orienté, alors $G'[M_{xy}]$ est un cographe orienté. Le cas où $|Q_y| = 1$ est similaire.

■

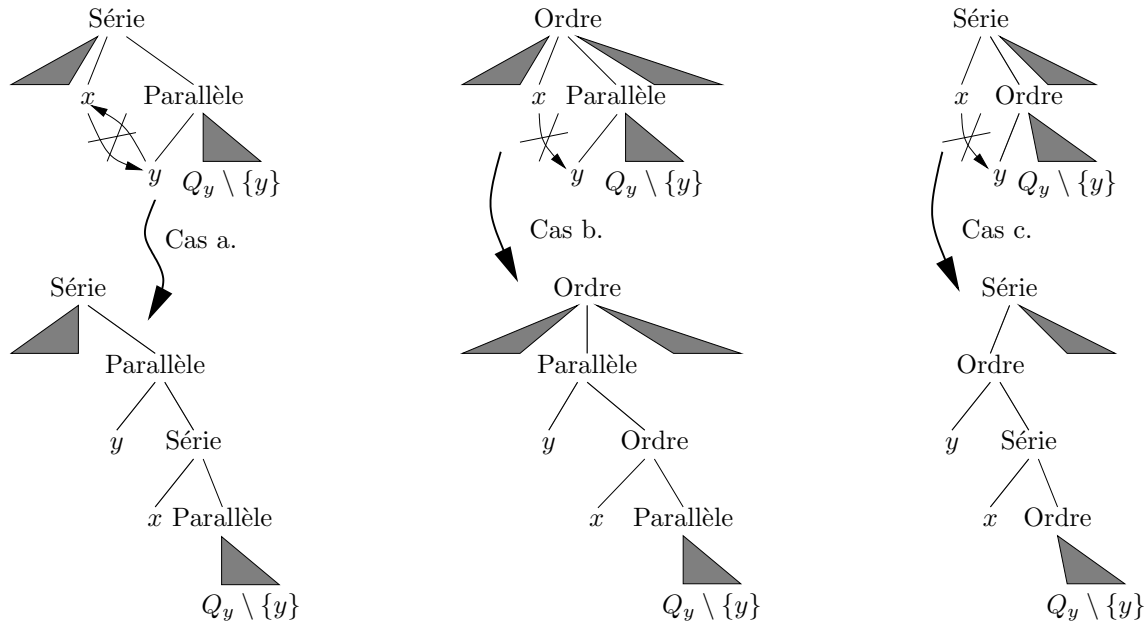


FIG. 4.21 – Le cas a. illustre la modification entraînée par la suppression simultanée de deux arcs symétriques (voir théorème 4.12); les cas b. et c. illustrent la suppression de l’arc xy décrite au théorème 4.13. Selon le nombre de frères de la feuille de y , le cographe orienté obtenu peut contenir moins de noeuds que représenté ci-dessus.

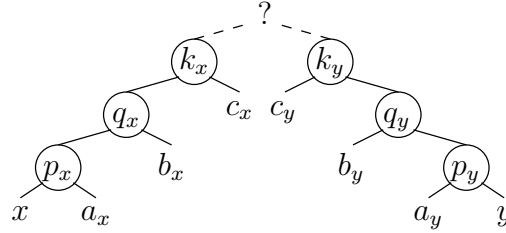
Des théorèmes 4.12 et 4.13 découle que l’on puisse tester si la suppression d’arc débouche sur un cographe orienté en temps $O(1)$. En effet, si c’est le cas, x et y doivent être fils et petit-fils de p_{xy} , ou tous deux fils de p_{xy} . Ensuite, il suffit de tester l’étiquette de p_{xy} et éventuellement de son fils q_y qui est le père de y . Enfin, si p_{xy} est un noeud ordre, ses fils q_x et q_y doivent être consécutifs dans l’ordre qu’il définit. Si la suppression est possible, les modifications du coarbre orienté sont effectuées en temps constant, comme illustré sur la figure 4.21.

Production d’un certificat.

Supposons que le test de suppression de xy (ou d’arcs symétriques xy et yx) échoue. Comme fait pour les modifications de sommet, l’algorithme retourne un sous-graphe de petite taille contenant un des graphes exclus de la figure 4.7. Grâce à la permutation factorisante, les sommets de ce sous-graphe peuvent être trouvés en temps constant. Si un certificat exact est souhaité, il est possible d’en trouver un en temps $O(\text{Max}(d(x), d(y)))$.

Lemme 4.8 *Si $G' = G - xy$ n’est pas un cographe orienté, un ensemble Z d’au plus six sommets tel que $G'[Z \cup \{x, y\}]$ contient un des graphes de la figure 4.7 peut être trouvé en temps $O(1)$.*

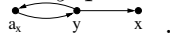
Le même ensemble Z de sommets, décrit ci-dessus, fournit aussi un certificat pour le cas où la suppression d’arcs symétriques xy et yx échoue. Dans la suite, nous ne détaillons

FIG. 4.22 – Une configuration possible de Z .

pas ce cas qui est très similaire avec la suppression d'un arc simple xy .

Donnons la définition de Z . Soit p_x (resp. p_y) le père de x (resp. y) dans T^m . Si $p_x \neq r$ (resp. $p_y \neq r$), soit q_x (resp. q_y) le père de p_x (resp. p_y) dans T^m . Si $q_x \neq r$ (resp. $q_y \neq r$), soit k_x (resp. k_y) le père de q_x (resp. q_y) dans T^m . On définit, si possible, six sommets a_x, b_x, c_x et a_y, b_y, c_y . Le sommet a_x appartient à $P_x \setminus \{x\}$, de plus, si p_x est un noeud ordre dont l_x n'est pas la dernière composante ordre, a_x doit être choisi dans la composante ordre qui succède à l_x dans l'ordre défini par p_x . Les sommets b_x et c_x appartiennent respectivement à $Q_x \setminus P_x$ et $K_x \setminus Q_x$, si ces ensembles existent. Les trois autres sommets a_y, b_y, c_y sont définis de la même façon mais par rapport à y . Si c'est possible, a_y doit être choisi dans la composante ordre précédant y dans l'ordre défini par p_y . Il est à remarquer que, lorsqu'ils existent, ces sommets peuvent ne pas être tous distincts (e.g. il se pourrait que $a_x = c_y$).

Le tableau de la figure 4.23 synthétise l'analyse de cas pour trouver un graphe exclu de la figure 4.7, dans le cas où $ppca(x, y)$ est un noeud série. Le cas où $ppca(x, y)$ est un noeud ordre est similaire. Détaillons à titre d'exemple le cas où $ppca(x, y)$ est un noeud série et où ni x ni y ne sont fils de $ppca(x, y)$ et où p_x est un noeud parallèle (première ligne du tableau). Dans ce cas, comme p_x est parallèle, il n'y a aucun arc entre x et a_x . Par contre, il y a des arcs dans les deux directions entre a_x et y , car $ppca(a_x, y) = ppca(x, y)$ est série. Après la suppression de l'arc xy , x et y par un arc de y à x . Donc, les sommets a_x, x, y induisent le graphe exclu suivant :



Manifestement, les sommets $a_x, b_x, c_x, a_y, b_y, c_y$ peuvent être trouvés en temps constant. Si un certificat exact est souhaité, nous avons besoin de trouver le $ppca$ de x et y . Comme il se peut que ce noeud ne soit pas parmi $p_x, q_x, k_x, p_y, q_y, k_y$ (cf. figure 4.22), cela peut prendre un temps $O(\text{Max}(d(x), d(y)))$ de le trouver. Une fois $ppca(x, y)$ localisé, en examinant les étiquettes de $p_x, q_x, k_x, p_y, q_y, k_y$ et $ppca(x, y)$, il est possible de déterminer en temps constant un sous-ensemble $\tilde{Z}$ de Z tel que $G'[\tilde{Z} \cup \{x, y\}]$ est un sous-graphe exclu minimal.

4.4 Les graphes P_4 -sparse

Nous venons de le voir, les cographes et cographes orientés, qui sont entièrement décomposables par la décomposition modulaire, ne possèdent pas de noeud premier dans leur arbre de décomposition modulaire. Cela permet de traiter sur ces classes nombre de problèmes de manière très efficace. Le maintien dynamique de leur arbre de décomposition

$x, y, ppca(x, y)$	p_x, q_x, p_y, q_y	k_x, k_y	sous-graphe exclu
ni x ni y n'est un fils de $ppca(x, y)$	parmi p_x, q_x, p_y, q_y au moins un est parallèle		 ou 
	p_x ou q_x est ordre et p_y ou q_y est ordre		 ou 
x est un fils de $ppca(x, y)$ mais y n'est ni un fils ni un petit fils de $ppca(x, y)$	p_y ou q_y est parallèle		
	p_y est série et q_y est ordre		 ou 
	p_y est ordre et q_y est série	k_y est parallèle	
		k_y est ordre	 ou 
y est un fils de $ppca(x, y)$ mais x n'est ni un fils ni un petit fils de $ppca(x, y)$	idem que pour le cas précédent		
x est un fils de $ppca(x, y)$ et y est un petit fils de $ppca(x, y)$	p_y est parallèle		
	p_y est ordre mais y n'est pas son premier fils		
y est un fils de $ppca(x, y)$ et x est un petit fils de $ppca(x, y)$	p_x est parallèle		
	p_x est ordre mais x n'est pas son dernier fils		

FIG. 4.23 – Analyse de cas pour trouver un sous-graphe exclu minimal, lorsque $ppca(x, y)$ est un noeud série.

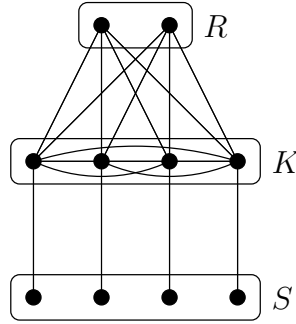


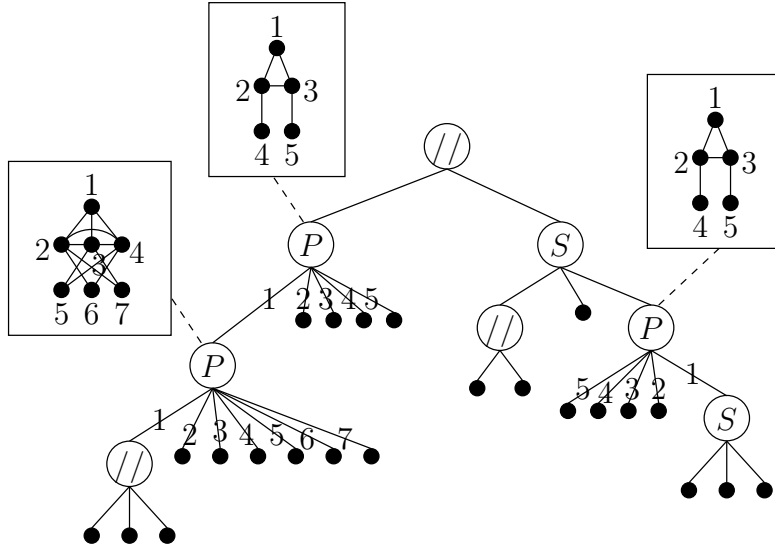
FIG. 4.24 – Une araignée fine.

modulaire peut être réalisé en temps $O(d(x))$ pour les opérations sur les sommets, où x est le sommet à ajouter ou retirer du graphe, et en temps $O(1)$ pour les opérations sur les arêtes. Si on relâche la contrainte sur la famille de graphe étudiée et qu'on l'autorise à posséder des noeuds premiers dans son arbre de décomposition, quelle complexité peut-on espérer atteindre pour le problème du maintien dynamique de l'arbre de décomposition ? A l'évidence, cela dépend de la contrainte définissant la classe de graphes considérée. Rappelons que, pour le cas des graphes en général, c'est à dire lorsque la contrainte est nulle, on ne connaît pas à ce jour d'algorithme entièrement dynamique de mise à jour de l'arbre de décomposition modulaire dont la complexité dans le pire des cas pour certaines opérations élémentaires soit meilleure que celle de l'algorithme statique qui calcule l'arbre, c'est à dire $O(n + m)$ (voir section 4.1). Pour les graphes P_4 -sparse [Ho83], pour lesquels la contrainte demandée est que tout sous-ensemble à cinq sommets induise au plus un P_4 (chemin à quatre sommets), [NPP06] montre qu'on peut encore entretenir l'arbre de décomposition modulaire en temps proportionnel au degré pour les modifications de sommet, et en temps constant pour les modifications d'arête. Comment la contrainte de définition des graphes P_4 -sparse se répercute-t-elle sur leur arbre de décomposition ? On sait que puisqu'ils peuvent contenir des P_4 induits, il peut y avoir des noeuds premiers dans leur arbre de décomposition. Quels peuvent être leurs quotients ? [JO92] répond à la question (théorème 4.14).

Définition 4.6 Une **araignée** est un graphe qui peut être partitionné en trois ensembles de sommets S , K et R , avec R éventuellement vide, tels que :

1. $2 \leq |S| = |K|$ et S est un stable et K est une clique ;
2. tous les sommets de R sont adjacents à tous les sommets de K et à aucun sommet de S ;
3. il existe une bijection $f : S \rightarrow K$ telle que :
 - (a) soit $\forall v \in S, N(v) \cap K = \{f(v)\}$,
 - (b) soit $\forall v \in S, N(v) \cap K = K \setminus \{f(v)\}$.

Une araignée qui vérifie la condition 3a est dite **fine**, et **épaisse** si elle vérifie la condition 3b. Un exemple d'araignée fine est donné sur la figure 4.24.

FIG. 4.25 – La décomposition modulaire d'un graphe P_4 -sparse.

Lemme 4.9 *Une araignée est un graphe premier ssi $|R| \leq 1$.*

Théorème 4.14 [JO92] *Soit G un graphe et T^m son arbre de décomposition modulaire. G est P_4 -sparse ssi pour tout noeud premier de T^m , le graphe quotient associé est une araignée première (S, K, R) et aucun sommet de $S \cup K$ ne correspond à un noeud interne de T^m .*

Corollaire 4.2 *Soit G un graphe P_4 -sparse et T^m son arbre de décomposition modulaire. Tout noeud premier de T^m a au plus un fils non feuille.*

Un exemple d'arbre de décomposition modulaire d'un graphe P_4 -sparse est donné sur la figure 4.25. On le voit, les quotients des noeuds premiers des graphes P_4 -sparse sont très contraints. Remarquons qu'une araignée première à n sommets peut être codée à isomorphisme près par $\lceil \log n \rceil + 1$ bits, ce qui est la taille de l'entier n . Pour cela on utilise un bit pour signifier que R est vide ou contient un unique sommet, un bit pour dire si l'araignée est fine ou épaisse, et $\lceil n \rceil - 1$ bits pour donner le nombre de sommets dans S . En fait, à isomorphisme près, il n'y a que deux araignées premières parmi les graphes à n sommets ! Devant la force de la contrainte à laquelle se plient les graphes P_4 -sparse, on comprend mieux qu'on puisse atteindre pour l'entretien dynamique de leur arbre de décomposition modulaire, la même complexité que pour les cographes et les cographes orientés.

Les propriétés constatées sur les cographes et les cographes orientés qui sont déterminantes pour la complexité atteinte sur ces classes de graphes se retrouvent chez les graphes P_4 -sparse. Deux propriétés que les P_4 -sparse ont en commun avec les cographes et les cographes orientés, et qui leur confère un comportement sympathique vis à vis de la décomposition modulaire, sont les suivantes : la famille des graphes P_4 -sparse est héréditaire et stable par complémentation. L'hérédité facilite le retrait de sommet (dont on sait qu'il

débouche toujours sur un graphe de la classe) et la stabilité par complémentation permet de ne traiter qu'un des deux cas de modification d'arête. Pour être complet, signalons que la stabilité par complémentation à elle seule n'assure pas que l'on puisse éviter de traiter une des deux modifications d'arête. Il faut aussi qu'on puisse obtenir le quotient d'un noeud premier du complémentaire $\overline{G}$ en temps constant à partir du quotient du noeud premier correspondant dans l'arbre de G . Cela assure de pouvoir manipuler localement l'arbre du complémentaire sans avoir à le calculer, et ne change pas la complexité. Lors d'une modification d'arête portant sur xy , ce qui permet d'avoir une complexité de traitement en temps constant est le fait que x et y sont à distance bornée par une constante dans l'arbre de décomposition T^m . Ainsi, l'algorithme est capable de déterminer si le graphe modifié reste P_4 -sparse en n'examinant qu'une partie bornée (par une constante) autour de leurs feuilles correspondantes. Lorsque la réponse est positive, la modification résultante de l'arbre est aussi locale. La suppression de sommet est plutôt aisée, et la modification de l'arbre en cas de succès est légère. Le cas difficile est encore celui de l'insertion de sommet. Remarquons que pour le traiter, la clef est encore une fois le procédé de marquage de [CPS85] destiné à identifier certains noeuds mixtes et certains noeuds uniformes relativement au sommet inséré x (voir section 4.3). La complexité de cette procédure est toujours $O(d(x))$ même sur un graphe quelconque. Par contre, comme elle ne détermine que les noeuds uniformément liés et les noeuds mixtes qui ont au moins un fils uniformément lié, elle ne fournit pas toujours une information suffisante pour conclure, selon la classe de graphes sur laquelle on l'applique. Pour le cas des graphes P_4 -sparse, [NPP06] montre que tout se passe bien. Cela repose sur une propriété remarquable déjà présente chez les cographes et les cographes orientés. Il s'agit du fait que pour que $G + x$ soit dans la classe, il faut que les noeuds mixtes se trouvent tous sur un même chemin se terminant à la racine de T^m . C'est une contrainte suffisamment forte pour que l'on puisse déterminer si tous les noeuds de l'arbre sont correctement reliés à x en examinant seulement les noeuds touchés par le procédé de marquage. Pour plus de détail, le lecteur est invité à se reporter à [NPP06].

4.5 Les graphes de permutation

Dans ce qui suit, on se focalise sur l'entretien de la MD -représentation du graphe de permutation considéré. Comme notre algorithme fonctionne en $O(n)$ par modification d'arête ou de sommet, et comme on peut obtenir un réalisateur de G à partir de $MD(G)$ en temps $O(n)$ (voir section 1.3.2), alors, dans la même complexité, notre algorithme entretient aussi un réalisateur du graphe. Une version courte de ce travail a été publiée dans [CP05].

Rappelons quelques notations que nous avons déjà introduites et utilisées. Elles peuvent être retrouvées, ainsi que d'autres, dans les préliminaires généraux du manuscrit et en section 1.2.1. On note $MD(G)$ la MD -représentation d'un graphe G . Si G est un graphe de permutation, les quotients des noeuds premiers de $MD(G)$ sont représentés par un réalisateur. On note $\mathcal{R}_G$ l'ensemble des réalisateurs des noeuds premiers de $MD(G)$. $T^m(G)$ dénote l'arbre de décomposition modulaire de G . S'il n'y a aucune ambiguïté sur le graphe auquel on se réfère, on notera simplement T^m . Pour un noeud p de T^m , on note P le module

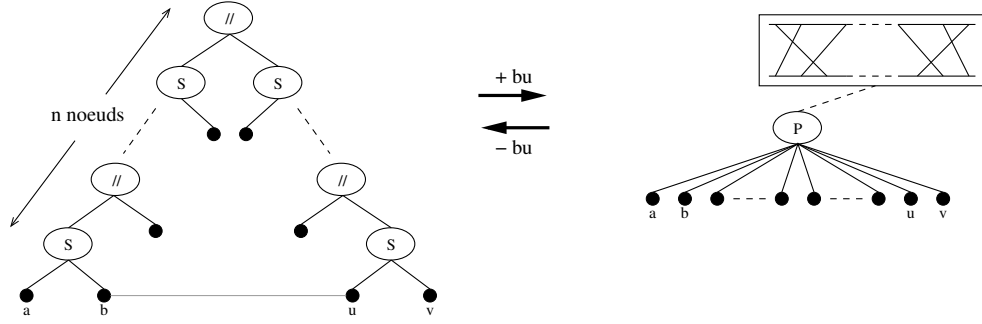


FIG. 4.26 – Une insertion/suppression d’arête provoquant $O(n)$ changements dans la MD -représentation. Lorsque l’arête bu est insérée dans le cographe dont l’arbre de décomposition est donné à gauche, celui-ci devient un graphe premier tout en restant un graphe de permutation. Le nombre de noeuds dans son arbre de décomposition chute de $2n - 1$ à 1. Réciproquement, lorsque l’arête bu est supprimée du graphe de permutation premier représenté à droite, celui-ci devient un cographe, et le nombre de noeuds dans son arbre de décomposition passe de 1 à $2n - 1$.

fort correspondant. La feuille de T^m correspondant à x sera noté l_x , lorsqu’on ne fait pas purement et simplement l’assimilation entre le sommet x de G et la feuille le représentant dans $T^m(G)$. Le graphe quotient d’un noeud premier p est noté G_p . Le sous arbre de T^m enraciné en p est noté T_p^m .

Notation 4.1 Pour un graphe G , on notera R_G un réalisateur de G . Et pour tout noeud premier p de $T^m(G)$, R_p est le réalisateur de $G[P]/\mathcal{MFM}(G[P])$ qui est associé au noeud p dans $MD(G)$.

Opérations dynamiques sur les arêtes

Malheureusement, une modification d’arête peut provoquer $O(n)$ changements dans l’arbre de décomposition modulaire (figure 4.26). Puisque nous proposons un algorithme en temps $O(n)$ les opérations d’insertion et de suppression de sommets, les opérations sur une arête e incidente à un sommet x seront traitées en supprimant d’abord x puis en le réinsérant avec son voisinage modifié.

4.5.1 Suppression de sommet

Soit $G' = G - x$ le graphe résultant de la suppression d’un sommet x dans le graphe G . Comme la famille des graphes de permutation est héréditaire, G' est toujours un graphe de permutation et supprimer x se réduit à mettre à jour la MD -représentation de G' à partir de celle de G . Il nous faut distinguer le cas où le père p de x dans $T^m(G)$ (s’il existe) est un noeud premier du cas où p est un noeud dégénéré.

Le cas dégénéré. Dans [SS04], un algorithme est proposé pour traiter la suppression de sommet dans un cografe dynamique. Avec une légère modification (le cas 4(b) ci-dessous), Il s'applique aussi au cas des graphes de permutation quand un sommet x fils d'un noeud dégénéré est supprimé. Généralisons donc l'algorithme de [SS04].

1. Si $T^m(G)$ n'a qu'une feuille l_x , alors $T^m(G')$ est vide.
2. Si p a au moins trois fils, alors $T^m(G')$ s'obtient en retirant la feuille l_x de $T^m(G)$.
3. Si p n'a que deux fils l_x et l et si l est une feuille de $T^m(G)$, alors $T^m(G')$ s'obtient à partir de $T^m(G)$ en retirant l_x et en remplaçant p par l .
4. si p n'a que deux fils l_x et q_1 et si q_1 est un noeud interne de $T^m(G)$:
 - (a) Si p est la racine de $T^m(G)$, alors $T^m(G')$ est le sous-arbre de $T^m(G)$ enraciné en q_1 .
 - (b) Si p n'est pas la racine de $T^m(G)$. Soit q_2 le père de p .
 - Si q_1 et q_2 sont deux noeuds série ou deux noeuds parallèles, alors $T^m(G')$ s'obtient en retirant de $T^m(G)$ la feuille l_x et les noeuds p et q_2 , et en connectant les fils de q_2 à q_1 .
 - Sinon, $T^m(G')$ s'obtient en retirant la feuille l_x de $T^m(G)$ et en remplaçant p par q_1 .

Remplacer un noeud p par un noeud q dans l'arbre de décomposition modulaire consiste à faire deux opérations : le père de p devient le père de q et p est supprimé. Le noeud q garde ses fils. Ceci se fait en temps $O(1)$. Remarquons que tous les cas, sauf le 4(b), peuvent être traités en temps constant. Le nombre d'opérations dans le cas 4(b) est clairement borné par $O(n)$.

Finalement, pour obtenir $MD(G')$, il reste à mettre à jour (si besoin) l'ensemble $\mathcal{R}_G$ des réalisateurs de $MD(G)$. Le seul cas à traiter se produit lorsque le père de p est un noeud premier. Alors q remplace p dans le réalisateur $R_{parent(p)}$ du quotient de $parent(p)$. Pour prouver la correction de l'algorithme, il est facile de vérifier que les relations d'adjacence entre toute paire de sommets de $V \setminus \{x\}$ sont préservées par les opérations décrites ci-dessus.

Le cas premier. Si p est un noeud premier, alors la suppression de x peut créer des modules dans le graphe quotient de p . Il s'ensuit que l'algorithme n'a que deux cas à traiter :

1. Si $G_p - x$ est un graphe premier, alors $T^m(G')$ s'obtient simplement en retirant la feuille l_x de $T^m(G)$. De façon similaire, retirer x du réalisateur R_p donne le réalisateur du noeud premier modifié.
2. Sinon, $MD(G')$ s'obtient en remplaçant d'abord dans $T^m(G)$ le noeud p par la racine de $T^m(G_p - x)$ et en remplaçant ensuite toutes les feuilles de $T^m(G_p - x)$ par le fils correspondant de p . $\mathcal{R}_{G'}$ est l'union de $\mathcal{R}_G \setminus \{R_p\}$ et de l'ensemble des réalisateurs de $MD(G_p - x)$.

Comme dans le cas dégénéré, les adjacences des sommets autres que x sont préservées par ces opérations. Par conséquent, le problème se réduit au calcul de $MD(G')$ lorsque G est

un graphe premier, avec $G' = G - x$. Comme nous l'avons déjà mentionné en section 1.3.2, en utilisant les algorithmes de [BXHP05, BCdMR05], l'arbre de décomposition modulaire d'un graphe de permutation peut être calculé en $O(n)$ si le réalisateur est donné. Or, supprimer x du réalisateur de G fournit un réalisateur de G' . Ainsi nous pouvons utiliser les algorithmes de [BXHP05, BCdMR05] pour nos besoins. Il est à noter que les réalisateurs de tous les noeuds premiers de $T^m(G_p - x)$ peuvent être extraits du réalisateur de G en temps $O(n)$.

Théorème 4.15 *La mise à jour de la MD-représentation d'un graphe de permutation après la suppression d'un sommet coûte un temps $O(n)$.*

La propriété suivante nous sera utile par la suite.

Définition 4.7 *Soit σ un ordre linéaire sur un ensemble X , soit $S \subseteq X$ et soit $a \in X \setminus S$. On dit que S **entoure** a s'il existe un élément de S inférieur à a et un élément de S supérieur à a dans σ .*

Lemme 4.10 *Soit $G = (V, E)$ un graphe de permutation premier et x un sommet. Les modules forts non triviaux de $G' = G - x$ peuvent être partitionnés en deux familles (éventuellement vides) totalement ordonnées par inclusion.*

Preuve : Notons $R = (\pi_1, \pi_2)$ un réalisateur de G et R' un réalisateur de G' obtenu en supprimant x dans R . D'après le théorème 1.29 page 84, il est équivalent de montrer que les intervalles communs forts de R' peuvent être divisés en deux familles (éventuellement vides) totalement ordonnées par inclusion. Toujours d'après le théorème 1.29, tout intervalle commun de R est non trivial. Soit I un intervalle commun non trivial de R' . Comme I n'est pas un intervalle commun de R , alors I entoure x dans au moins un des deux ordres π_1, π_2 de R . En d'autres termes, $I \cup \{x\}$ est un intervalle de π_1 (ou π_2) et x n'est pas une borne de cet intervalle. Comme les intervalles communs forts d'un réalisateur ne se chevauchent pas, ceux de R' qui entourent x dans l'ordre π_1 de R (resp. dans π_2) sont totalement ordonnés par inclusion. ■

Comme il y a au plus deux modules forts maximaux, la racine de $T^m(G')$ a au plus deux fils qui ne sont pas des feuilles, et chaque noeud interne de $T^m(G')$ a au plus un fils qui n'est pas une feuille. De plus, on peut montrer que tous les noeuds dégénérés de $T^m(G')$ ont au plus deux fils feuilles. Par conséquent, le nombre de modules (pas nécessairement forts) de G' est $O(n)$, et il y a aussi $O(n)$ intervalles communs dans le réalisateur de G' .

Cette structure très particulière au cas de la suppression d'un sommet dans le réalisateur d'un noeud premier permettrait de se passer des algorithmes de [BXHP05, BCdMR05] et de les remplacer, dans ce cas, par un significativement plus simple. Nous ne poursuivons pas cette piste.

4.5.2 Insertion de sommet

Afin d'obtenir la MD -représentation d'un graphe de permutation après une insertion de sommet, nous allons : 1) actualiser l'arbre de décomposition modulaire $T^m(G)$ et ainsi obtenir $T^m(G')$; 2) tester si $G' = G + x$ est un graphe de permutation; et 3) si tel est le cas, calculer l'ensemble des réalisateurs $\mathcal{R}_{G'}$ à partir de $\mathcal{R}_G$ et $T^m(G)$. L'algorithme que nous proposons a une complexité de $O(n)$ par insertion de sommet. Dans sa présentation, cette section suit les trois étapes précitées.

Arbre de décomposition modulaire de $G + x$

[MS89] a proposé un algorithme incrémental² qui met à jour l'arbre de décomposition modulaire lors de l'ajout d'un sommet x . Alors que l'approche de [MS89] est algorithmique, dans cette section, nous donnons une description mathématique de l'arbre de décomposition modulaire du graphe augmenté, indépendamment de tout aspect calculatoire.

Les résultats que nous obtenons dans cette section s'appliquent à tous les graphes et ne sont pas restreints aux graphes de permutation. L'intérêt principal de notre approche est de séparer le problème du maintien de l'arbre de décomposition modulaire du problème du maintien de la représentation choisie pour les graphes quotients des noeuds premiers. Pour effectuer le maintien de l'arbre en $O(n)$ lors de l'ajout de sommet, il faut que cette représentation permette de déterminer si x a un jumeau dans le graphe quotient d'un noeud premier p en temps $O(|\mathcal{C}(p)|)$. C'est le cas de la N -représentation de [MS89]. Cependant, dans [MS89], comme la structure associée à un noeud premier (N -représentation) est une représentation partielle, elle ne permet pas de maintenir l'arbre lors de la suppression de sommet. En effet, seulement certaines arêtes du graphe peuvent être retrouvées à partir de sa N -représentation.

Dans notre cas, le fait que nous considérons uniquement la famille des graphes de permutation nous fournit une représentation complète des graphes quotients qui nous permet de maintenir l'arbre à la fois lors du retrait et de l'ajout d'un sommet. Quelle que soit la représentation choisie pour les quotients des noeuds premiers, les résultats que nous donnons sur la structure de l'arbre de décomposition modulaire du graphe augmenté sont valides et déterminent entièrement cet arbre.

Notation 4.2 $\mathcal{C}_l(p)$ (resp. $\mathcal{C}_{nl}(p)$) dénote l'ensemble des fils pleins (resp. creux) de p et $\mathcal{C}_m(p)$ l'ensemble des fils mixtes de p . Pour $t \in \{m, l, nl\}$, on note $F_t(p) = \bigcup_{f \in \mathcal{C}_t(p)} F$.

Noeud d'insertion. Nous montrons qu'insérer x dans G se réduit à mettre à jour seulement un certain sous-arbre de l'arbre de décomposition modulaire. Commençons par identifier la racine de ce sous-arbre.

²L'algorithme de [MS89] n'est pas purement incrémental, il est plus précisément présenté et discuté en section 4.1

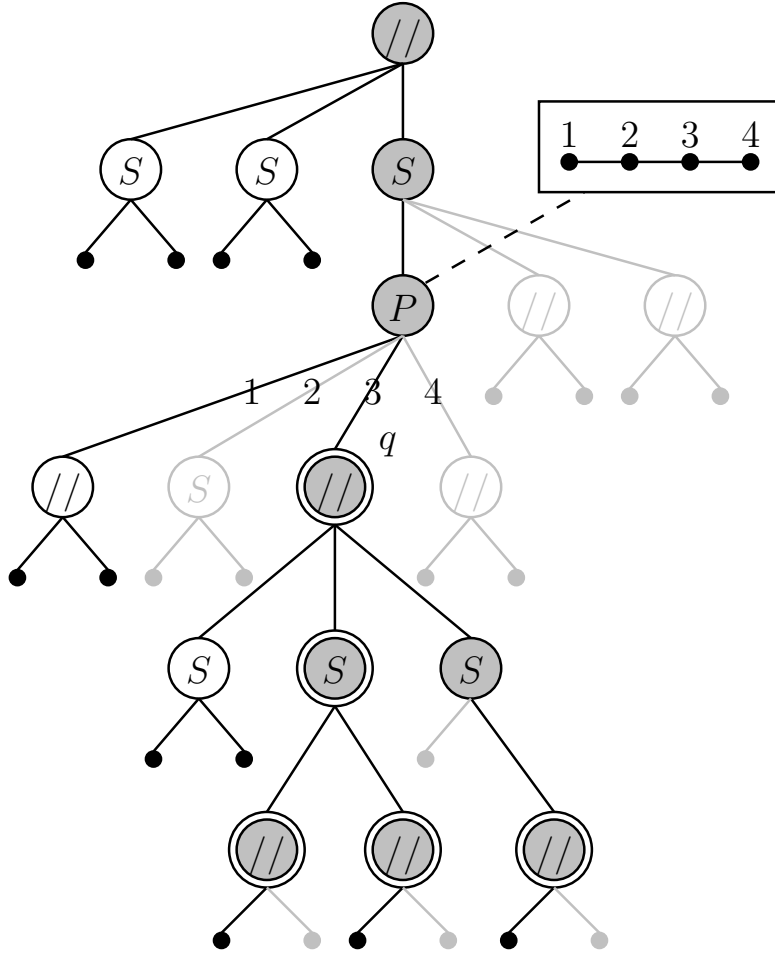


FIG. 4.27 – Les feuilles grises sont adjacentes à x , les noires ne le sont pas. Les noeuds au contour gris sont pleins, ceux au contour noir sont creux et ceux au contour noir qui sont remplis de gris sont mixtes. Les noeuds non propres sont ceux qui ont un double contour. Le noeud q est le noeud d'insertion.

Définition 4.8 (voir figure 4.27) Un noeud p de $T^m(G)$ est **propre** ssi p est uniforme relativement à x ou p est un noeud mixte ayant un unique fils mixte f , qui est tel que $F \cup \{x\}$ est un module de $G'[P \cup \{x\}]$. Dans le cas contraire, p est dit **non propre**.

Lemme 4.11 Soit $G = (V, E)$ un graphe et x un sommet devant être inséré dans G . Si tous les noeuds de $T^m(G)$ sont propres, alors V est uniforme relativement à x .

Preuve : On prouve la contraposée. Si V est mixte, alors la racine de $T^m(G)$ est mixte. Tout noeud mixte de $T^m(G)$ possède un descendant mixte p qui n'a que des fils uniformes. Par la définition 4.8, un tel noeud p est non propre. ■

Il découle du Lemme 4.11 que, dans le cas où il n'y a aucun noeud mixte dans T^m , x est un sommet universel ou isolé. Par conséquent, l'arbre de décomposition modulaire est facile à mettre à jour en temps constant. Ce cas ne sera plus considéré dans ce qui suit. A partir de maintenant, nous supposons l'existence d'au moins un noeud non propre dans T^m .

Observation 4.1 *Le plus petit ancêtre commun q des noeuds non propres de T^m est un noeud non propre.*

Preuve : Soient p_1 et p_2 deux noeuds non propres. Si $ppca(p_1, p_2) \in \{p_1, p_2\}$, alors la propriété est vérifiée. Sinon, comme tout ancêtre d'un noeud mixte est mixte, $ppca(p_1, p_2)$ a au moins deux fils mixtes. Donc $ppca(p_1, p_2)$ est non propre. ■

Lemme 4.12 *Le $ppca$ q des noeuds non propres de T^m est tel que $Q' = Q \cup \{x\}$ est un module fort de $G' = G + x$.*

Preuve :

Nous montrons par récurrence que pour tout ancêtre p_1 de q , Q' est un module de $G[P'_1]$, avec $P'_1 = P_1 \cup \{x\}$. En effet, Q' est un module de $G'[Q']$. Soit p_1 un ancêtre de q tel que Q' est un module de $G'[P'_1]$. Si $p_1 \neq r$ alors soit p_2 le père de p_1 . Comme q est mixte et p_2 est un ancêtre de q , alors p_2 est un noeud mixte propre (par définition de q). Il en découle que p_1 est l'unique fils mixte de p_2 et P'_1 est un module de $G'[P'_2]$. Comme Q' est un module de $G'[P'_1]$ et P'_1 est un module de $G'[P'_2]$, alors, par définition d'un module, Q' est un module de $G'[P'_2]$. Cela termine la récurrence et montre que Q' est un module de G' .

Montrons que Q' ne chevauche aucun autre module M' de G' . Soit M' un module de G' . Supposons que M' chevauche Q' . Si $x \notin M'$, alors M' est un module de G . Comme Q est un module fort de G , M' et Q ne se chevauchent pas. Ainsi, comme M' chevauche Q' , alors $Q \subsetneq M'$. Q n'est pas uniforme relativement à x , donc M' ne l'est pas non plus : contradiction avec le fait que M' est un module de G' qui ne contient pas x . Si $x \in M'$, alors $M = M' \setminus \{x\}$ est un module de G (voir le lemme 1.1 page 39). Comme Q est un module fort de G , il ne chevauche pas M . Supposons que $M \cap Q = \emptyset$. Comme M' chevauche Q' , $Q' \setminus M' = Q$ est un module de G' . Ce qui constitue une contradiction puisque $x \notin Q$ et Q n'est pas uniforme relativement à x . Donc $M \cap Q \neq \emptyset$, et comme Q ne chevauche pas M , alors $Q \subseteq M$ ou $M \subseteq Q$. Dans les deux cas, Q' ne chevauche pas M' : contradiction finale. ■

Lemme 4.13 *Soit $G' = G + x$ et soit q le $ppca$ des noeuds non propres de $T^m(G)$. $T^m(G')$ s'obtient à partir de $T^m(G)$ en remplaçant le sous-arbre $T_q^m(G)$ par $T^m(G'[Q']) = T^m(G[Q] + x)$.*

Preuve : Comme Q est un module fort de G et comme, d'après le lemme 4.12, $Q' = Q \cup \{x\}$ est un module fort de $G' = G + x$, alors $G/\{Q\}$ et $G'/\{Q'\}$ sont bien définis. Ces deux

quotients sont égaux car en choisissant le sommet représentatif de Q' parmi les sommets de Q , on obtient le même ensemble de sommets représentatifs pour les deux quotients. Notez que $T^m(G/\{Q\})$ s'obtient à partir de $T^m(G)$ en remplaçant le noeud q par une feuille correspondant au sommet représentatif de Q . En d'autres termes $T^m(G/\{Q\})$ est, à une feuille près, la partie complémentaire de $T_q^m(G)$ dans $T^m(G)$. De manière similaire, $T^m(G'/\{Q'\})$ est la partie complémentaire de $T^{m'}_{q'}(G')$ dans $T^m(G')$. Comme $G/\{Q\} = G'/\{Q'\}$, il s'ensuit que $T^m(G')$ s'obtient à partir de $T^m(G)$ en remplaçant le sous-arbre $T_q^m(G)$ par $T^{m'}_{q'}(G'[Q'])$. ■

Définition 4.9 *Le noeud d'insertion q est le ppca des noeuds non propres de T^m .*

Dans la suite, q désignera toujours le noeud d'insertion. D'après le lemme 4.13, on conclut que la mise à jour de $T^m(G)$ lors de l'insertion de x se réduit à insérer x dans $T^m(G)[Q]$, d'où le nom de noeud d'insertion pour q .

Arbre de décomposition modulaire de $G'[Q']$. Distinguons différentes situations pour le noeud d'insertion q .

Définition 4.10 (voir figure 4.28) *Le noeud d'insertion q de $T^m(G)$ est **coupé** ssi q est :*

1. *un noeud dégénéré sans fils mixte mais des fils de deux types (i.e. pleins et creux), ou*

2. *q est un noeud premier sans fils mixte et dont un des fils est jumeau de x dans G_q .*

*Si non, q est **non coupé**. Précisément, q est un noeud dégénéré ayant au moins un fils mixte, ou q est un noeud premier ayant au moins un fils mixte ou dont aucun fils n'est un jumeau de x dans G_q .*

Le cas où le noeud d'insertion est un noeud dégénéré coupé est similaire au cas, considéré par [CPS85], du maintien de l'arbre de décomposition modulaire d'un cograph lors de l'insertion d'un sommet. Si q est un noeud série (resp. parallèle), la racine q' de $T^m(G'[Q'])$ est un noeud série (resp. parallèle). Les fils de q' sont les fils pleins (resp. creux) de q et un nouveau noeud parallèle (resp. série) q'_1 . Les fils de q'_1 sont $\{x\}$ et les fils restants de q , i.e. ceux qui sont creux (resp. pleins).

Le cas dans lequel le noeud d'insertion est un noeud premier coupé est facile à traiter. Dans les fils de q , le jumeau f de x est remplacé par un nouveau noeud dégénéré q_1 (i.e. q_1 prend la place de f dans le réalisateur de q). L'étiquette de q_1 est série si f est plein, et parallèle si f est creux. $\{x\}$ et f deviennent fils de q_1 .

Considérons maintenant le cas où le noeud d'insertion q est non coupé.

Notation 4.3 *On définit l'ensemble de sommets Q_s comme étant l'ensemble Q si q est un noeud premier et l'ensemble $F_m(q) \cup F_{nl}(q)$ (resp. $F_m(q) \cup F_l(q)$) si q est un noeud série (resp. parallèle). Comme d'habitude, Q'_s désignera l'ensemble $Q_s \cup \{x\}$. On note $T_{q_s} = T^m(G[Q_s])$ et q_s sa racine ; ainsi que $T'_{q'_s} = T^m(G'[Q'_s])$ et q'_s sa racine.*

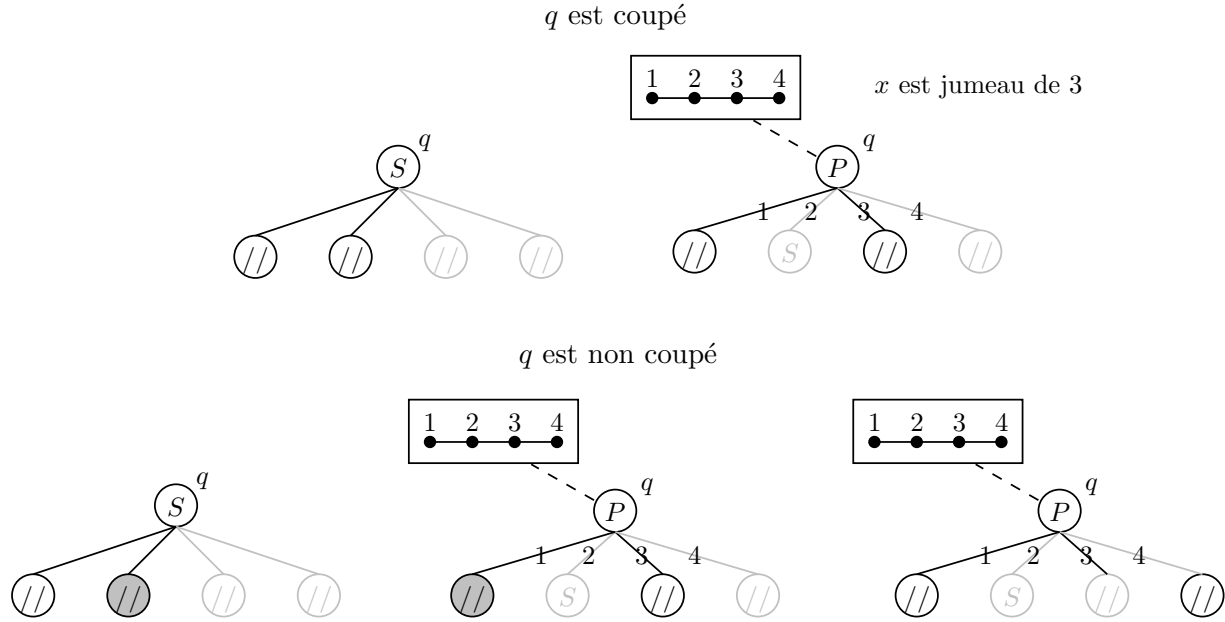


FIG. 4.28 – Les différentes façons pour le noeud d'insertion d'être coupé ou non. Le code des couleurs des noeuds est le même que sur la figure 4.27.

Déterminer la décomposition modulaire de $G'[Q'_s]$ (voir le théorème 4.16 ci-dessous) est crucial afin d'obtenir l'arbre de décomposition modulaire de $G'[Q']$. A partir de maintenant, on note $\mathcal{MUM}(G)$ l'ensemble des modules uniformes (rel. à x) maximaux d'un graphe G . Il convient de remarquer que $\mathcal{MUM}(G)$ est, par définition, une partition de congruence de G . Les deux lemmes suivants sont utiles dans la preuve du théorème 4.16.

Lemme 4.14 *Soit M un module de G et M' un module de $G' = G + x$, tel que $x \in M'$ et $M \cap M' \neq \emptyset$. Alors $M \cup M'$ est un module de G' .*

Preuve : Soit $G = (V, E)$ et $G' = G + x = (V', E')$. Soit $y \in V' \setminus (M \cup M')$ et $z \in M \cap M'$. Comme M est un module de G , $\{y, z\} \in E'$ ssi $\forall u \in M, \{y, u\} \in E'$ et comme M' est un module de G' , $\{y, z\} \in E'$ ssi $\forall v \in M, \{y, v\} \in E'$. D'où, $\forall t \in M \cup M', yt \in E'$ ssi $yz \in E'$. ■

Lemme 4.15 *Si $G' = G + x$ est connexe et co-connexe et si $\{x\}$ est un module fort maximal de G' , alors l'ensemble des modules forts maximaux de G' différents de $\{x\}$ est exactement l'ensemble des modules uniformes (rel. à x) maximaux de G . $\mathcal{MFM}(G') \setminus \{x\} = \mathcal{MUM}(G)$.*

Preuve : Comme G' est connexe et co-connexe, ses modules forts maximaux sont ses modules maximaux. Comme x est l'un d'eux, les autres ne contiennent pas x . Donc, les modules maximaux de G' différents de $\{x\}$ sont les éléments maximaux (pour l'inclusion) de l'ensemble des modules de G' qui ne contiennent pas x . Montrons que l'ensemble de

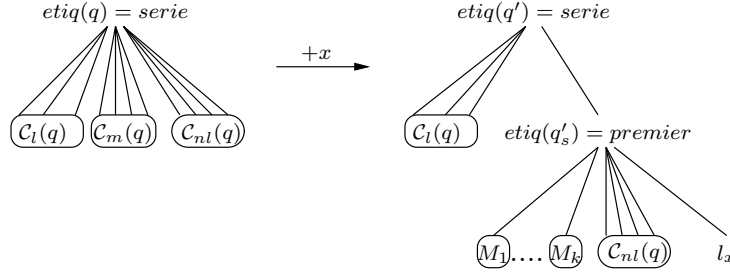


FIG. 4.29 – Mise à jour de l'arbre de décomposition modulaire lorsque le noeud d'insertion est un noeud série. Les modules $M_1 \dots M_k$ sont les modules uniformes maximaux de $G[Q_s]$

ces éléments maximaux est exactement $\mathcal{MUM}(G)$. Soit M un module uniforme de G , M est un module de G' . Réciproquement, soit M' un module de G' qui ne contient pas x , M' est un module uniforme de G . L'ensemble des modules uniformes de G est exactement l'ensemble des modules de G' qui ne contiennent pas x , et donc les éléments maximaux de ces deux ensembles sont les mêmes. ■

Théorème 4.16 *Soit x un sommet à insérer dans un graphe G . Si le noeud d'insertion q de l'arbre de décomposition modulaire T^m de G est non coupé, alors $G'[Q'_s]$ est connexe et co-connexe et $\mathcal{MF}\mathcal{M}(G'[Q'_s]) = \mathcal{MUM}(G[Q_s]) \cup \{\{x\}\}$.*

Preuve : Grâce au lemme 4.15, il nous suffit de montrer que $G'[Q'_s]$ est connexe et co-connexe, et que $\{x\}$ est un module fort maximal de $G'[Q'_s]$.

Si q est un noeud premier, par définition, $Q'_s = Q'$. $G[Q]$ est connexe et co-connexe, et comme Q est mixte, alors $G'[Q']$ est connexe et co-connexe.

Si q est un noeud série, par définition de $Q_s = \bigcup_{f \in \mathcal{C}_{nl}(q) \cup \mathcal{C}_m(q)} F$, $G[Q_s]$ est connexe. De plus, comme q est non coupé, alors $\mathcal{C}_m(q) \neq \emptyset$. Il en découle qu'il existe $y \in Q_s$ tel que xy est une arête. Donc $G'[Q'_s]$ est connexe. Comme q est un noeud série, alors tout fils f de q est parallèle ou premier. Par conséquent, $G[F]$ est co-connexe. En outre, pour chaque $f \in \mathcal{C}_{nl}(q) \cup \mathcal{C}_m(q)$, par définition, il existe $y \in F$ tel que xy n'est pas une arête. Il s'ensuit que $G'[Q'_s]$ est co-connexe.

Si q est un noeud parallèle, en considérant le complémentaire de G , on aboutit à la même conclusion que dans le cas série. Donc $G'[Q'_s]$ est connexe et co-connexe, nous montrons maintenant que $\{x\}$ est un module fort maximal de $G'[Q'_s]$.

Proposition 4.3 *Si le noeud d'insertion q est un noeud premier non coupé qui n'a pas de fils mixte, alors $\{x\}$ est un module fort maximal de $G'[Q'_s]$.*

Preuve : Soit M' le module fort maximal de $G'[Q']$ contenant x . Comme $G'[Q']$ est connexe et co-connexe, M' est aussi le module maximal de $G'[Q']$ contenant x . On note $M = M' \setminus \{x\}$. Supposons que $M \neq \emptyset$, alors, d'après le lemme 1.1 page 39, M est un module de $G[Q]$. Soit Q_M le module maximal de $G[Q]$ contenant M . Q_M et M' satisfont les hypothèses du lemme 4.14 dans $G[Q]$, donc $Q_M \cup M'$ est un module de $G'[Q']$. Comme M'

est maximal, $Q_M \cup M' = M'$. Comme, par définition de M et Q_M , $Q_M \cup M' = Q_M \cup \{x\}$, alors $Q_M \cup \{x\} = M'$ et $M = Q_M$. Comme $Q_M \cup \{x\}$ est un module de $G'[Q']$, q_M est un jumeau de x dans le quotient de q . Ceci est impossible car q est non coupé. Donc, $M = \emptyset$ et $\{x\}$ est un module fort maximal de $G'[Q']$. $\square$

Proposition 4.4 Si le noeud d'insertion q est un noeud premier non coupé qui a au moins un fils mixte, alors $\{x\}$ est un module fort maximal de $G'[Q'_s]$.

Preuve : Soit M' le module fort maximal de $G'[Q']$ contenant x . On note $M = M' \setminus \{x\}$. Supposons $M \neq \emptyset$. Comme dans le cas où q n'a pas de fils mixte, on note Q_M le module maximal de $G[Q]$ contenant M et on obtient que $M = Q_M$, car la preuve du cas précédent reste valide. Si q a un unique fils mixte q_t , comme q est un noeud non propre et $Q_M \cup \{x\}$ est un module de $G'[Q']$, alors q_M n'est pas cet unique fils mixte de q , par définition d'un noeud non propre, $q_M \neq q_t$. Si q a au moins deux fils mixtes, l'un deux q_t est différent de q_M . Dans les deux cas, il existe un fils mixte q_t de q tel que $q_t \neq q_M$. Soit $u \in Q_t$ tel que $\{x, u\} \in E'$ et $v \in Q_t$ tel que $\{x, v\} \notin E'$. Soit $z \in Q_M$, comme $Q_M \cup \{x\}$ est un module de $G'[Q']$ et $\{x, u\} \in E'$, alors $\{z, u\} \in E'$. Comme $Q_M \cup \{x\}$ est un module de $G'[Q']$ et $\{x, v\} \notin E'$, alors $\{z, v\} \notin E'$. Cela contredit le fait que Q_t est un module de $G[Q]$. Donc, $M = \emptyset$ et $\{x\}$ est un module fort maximal de $G'[Q']$. $\square$

Proposition 4.5 Si le noeud d'insertion q est un noeud série non coupé, alors $\{x\}$ est un module fort maximal de $G'[Q'_s]$.

Preuve : Soit M' le module fort maximal de $G'[Q'_s]$ contenant x . Comme $G'[Q'_s]$ est connexe et co-connexe, M' est aussi le module maximal de $G'[Q'_s]$ contenant x . On note $M = M' \setminus \{x\}$. Remarquons que, par définition, $M' \neq Q'_s$ et $M \neq Q_s$. Supposons que $M \neq \emptyset$, alors, d'après le lemme 1.1 page 39, M est un module de $G[Q]$. Comme $M \neq Q_s$, il est possible de trouver $f_j \in \mathcal{C}_{nl}(q) \cup \mathcal{C}_m(q)$ tel que $F_j \cap M = \emptyset$. Comme F_j est mixte ou creux, il existe $y \in F_j$ tel que xy n'est pas une arête. Soit $z \in M$, comme f_j est un fils de q qui est un noeud série et comme M est un module de $G[Q]$ tel que $F_j \cap M = \emptyset$, yz est une arête. Comme M' est un module de $G'[Q']$, $y \notin M'$, $\{x, z\} \subseteq M'$ et xy n'est pas une arête, alors yz n'est pas une arête : absurde. Donc $M = \emptyset$ et $\{x\}$ est un module fort maximal de $G'[Q'_s]$. $\square$

Proposition 4.6 Si le noeud d'insertion q est un noeud parallèle non coupé, alors $\{x\}$ est un module fort maximal de $G'[Q'_s]$.

La preuve est tout à fait similaire à celle du cas où q est un noeud série non coupé (considérer le complémentaire de $G'[Q']$). $\blacksquare$

L'arbre de décomposition modulaire $T^m(G'[Q'])$ est formé comme suit. Si q est un noeud premier, alors, d'après le théorème 4.16, $G'[Q']$ est connexe et co-connexe. Par conséquent, q' est un noeud premier et ses fils correspondent aux modules uniformes maximaux de $G[Q]$

(théorème 4.16). Si q est dégénéré, alors q' est dégénéré et a la même étiquette que q . Si q est un noeud série (resp. parallèle), l'ensemble des fils de q' est $\{q'_s\} \cup \mathcal{C}_l(q)$ (resp. $\{q'_s\} \cup \mathcal{C}_{nl}(q)$) où q'_s est un nouveau noeud représentant les sommets de Q'_s . D'après le théorème 4.16, q'_s est un noeud premier et ses fils correspondent aux modules uniformes maximaux de $G[Q]$ (théorème 4.16). Remarquons que lorsque q est série (resp. parallèle), il se peut que $\mathcal{C}_l(q)$ (resp. $\mathcal{C}_{nl}(q)$) soit vide. Dans ce cas, $Q' = Q'_s$ et on ne crée pas le nouveau noeud q'_s . Etant donné un module uniforme maximal M de $G[Q_s]$, l'arbre de décomposition modulaire de $G'[M]$ est la partie de $T^m(G)$ restreinte à M .

Nous avons entièrement déterminé l'arbre de décomposition modulaire $T^m(G')$. Mais nous ne savons toujours pas si G' est un graphe de permutation, et, le cas échéant, comment construire les réalisateurs de $\mathcal{R}_{G'}$. C'est le but de la section suivante.

Caractérisation dynamique des graphes de permutation

Dans le cas où x est un sommet isolé ou universel, G' est un graphe de permutation et la MD -représentation est facile à maintenir en temps constant. Nous ne considérons pas ce cas et nous nous concentrons sur celui dans lequel $T^m(G)$ contient au moins un noeud non propre, voir lemme 4.11. Comme précédemment, on note q le noeud d'insertion. Dans la section précédente, nous avons montré que l'insertion de x dans $T^m(G)$ se réduit à l'insertion de x dans $T^m(G[Q])$. Le lemme 4.16 montre que, de façon similaire, insérer x dans $MD(G)$ se réduit à insérer x dans $MD(G[Q])$.

Lemme 4.16 *Soit $G' = G + x$ et soit q le noeud d'insertion. G' est un graphe de permutation ssi $G'[Q']$ en est un. De plus, si G' est un graphe de permutation, $MD(G')$ s'obtient de $MD(G)$ en remplaçant le sous-arbre $T_q^m(G)$ par $T^m(G'[Q']) = T^m(G[Q] + x)$, et en remplaçant, dans $\mathcal{R}_G$, les réalisateurs des noeuds premiers de $T_q^m(G)$ par ceux des noeuds premiers de $T^m(G'[Q'])$.*

Preuve : Comme la famille des graphes de permutation est héréditaire, si G' est un graphe de permutation alors $G'[Q']$ en est un. Réciproquement, comme la famille des graphes de permutation est fermée par substitution d'un graphe à un sommet et comme $G/\{Q\} = G'/\{Q'\}$ (voir preuve du lemme 4.13), si $G'[Q']$ est un graphe de permutation alors G' en est un également. Dans la preuve du lemme 4.13, on a déjà montré que $T^m(G')$ s'obtient à partir de $T^m(G)$ en remplaçant le sous-arbre $T_q^m(G)$ par $T^m(G'[Q']) = T^m(G[Q] + x)$. Pour chaque noeud de $T^m(G')$ qui n'est pas dans $T_q^m(G')$, on peut trouver un ensemble de sommets représentatifs pour le quotient qui ne contient pas x . Ainsi, les graphes quotients de ces noeuds sont les mêmes que ceux des noeuds de $T^m(G)$ correspondants. Donc, on obtient $MD(G')$ en remplaçant, dans $\mathcal{R}_G$, les réalisateurs des quotients des noeuds premiers de $T_q^m(G)$ par ceux des noeuds premiers de $T^m(G'[Q'])$. ■

Nous établissons maintenant une caractérisation, basée sur $MD(G'[Q'])$, des cas où G' est un graphe de permutation (théorème 4.17). Si on veut que G' soit un graphe de permutation, les noeuds mixtes de $T_q^m(G)$ ne peuvent pas être disséminés n'importe où

dans l'arbre. Le lemme 4.17 affirme qu'il y a au plus deux branches de noeuds mixtes dans $T_q^m(G)$, qui sont enracinées en q . Ces deux branches correspondent aux deux familles du lemme 4.10.

Lemme 4.17 *Si G' est un graphe de permutation alors le noeud d'insertion q a au plus deux fils mixtes et tout noeud $p \neq q$ de $T_q^m(G)$ a au plus un fils mixte.*

Preuve : Si q est coupé, alors q n'a pas de descendants mixtes et l'énoncé est vrai.

Prenons donc le cas où q est non coupé. Par définition de Q_s , T_{q_s} s'obtient de $T_q^m(G)$ en supprimant certains fils uniformes de q . De manière équivalente, on peut donc montrer le lemme sur T_{q_s} plutôt que sur $T_q^m(G)$. D'après le théorème 4.16, $H' = G'[Q'_s]/\mathcal{MFM}(G'[Q'_s])$ est un graphe de permutation premier. Toujours d'après le théorème 4.16, on a $H = G[Q_s]/\mathcal{MUM}(G[Q_s]) = H' - x$. D'après le théorème 4.16, les modules forts non triviaux de H peuvent être partitionnés en au plus deux familles entièrement ordonnées par inclusion. Par conséquent la racine de $T^m(H)$ a au plus deux fils qui ne sont pas des feuilles et tout noeud de $T^m(H)$ distinct de sa racine a au plus un fils qui n'est pas une feuille.

Par définition, tout module fort mixte de $G[Q_s]$ n'est pas un singleton et est l'union de certains modules de $\mathcal{MUM}(G[Q_s])$. Dès lors, le lemme 1.4 s'applique : $\mathcal{M} \subseteq \mathcal{MUM}(G[Q_s])$ est un module fort non trivial de H ssi $P = \cup_{M \in \mathcal{M}} M$ est un module fort non trivial de $G[Q_s]$. Il s'ensuit que l'ensemble des modules forts mixtes de $G[Q_s]$ et l'ensemble des modules forts de H qui ne sont pas des singletons sont en bijection. Pour terminer la preuve, il nous faut prouver que cette bijection respecte la relation de parenté entre les noeuds de $G[Q_s]$. Formellement, on montre que si p_1 et p_2 sont deux noeuds mixtes de T_{q_s} tel que p_1 est le père de p_2 , alors les noeuds non feuilles $\tilde{p}_1$ et $\tilde{p}_2$ qui leur correspondent dans $T^m(H)$ sont tels que $\tilde{p}_1$ est le père de $\tilde{p}_2$. Soit $\mathcal{M}_1 \subseteq \mathcal{MUM}(G[Q_s])$ tel que $P_1 = \cup_{M \in \mathcal{M}_1} M$. $\mathcal{M}_2$ est défini de manière similaire. Remarquons que $\mathcal{M}_1 = \tilde{P}_1$ et $\mathcal{M}_2 = \tilde{P}_2$. Comme $P_2 \subseteq P_1$, nécessairement, $\mathcal{M}_2 \subseteq \mathcal{M}_1$. Supposons qu'il existe $\mathcal{M}_3 \subseteq \mathcal{MUM}(G[Q_s])$ tel que $\mathcal{M}_3$ est un module fort de H et $\mathcal{M}_2 \subseteq \mathcal{M}_3 \subseteq \mathcal{M}_1$. Alors, d'après le lemme 1.4, $P_3 = \cup_{M \in \mathcal{M}_3} M$ est un module fort de $G[Q_s]$. De plus, $P_2 \subseteq P_3 \subseteq P_1$: contradiction avec le fait que p_1 est le père de p_2 . Donc, $\tilde{p}_1$ est le père de $\tilde{p}_2$. ■

Malheureusement, les conditions du lemme 4.17 ne sont pas suffisantes pour que G' soit un graphe de permutation. Le théorème 4.17 donne des conditions nécessaires et suffisantes.

Notation 4.4 *Etant donné un graphe $G = (V, E)$, $S \subsetneq V$ et $y \in V \setminus S$, on note $G - yS = (V, E \setminus \{\{y, z\} \mid z \in S\})$.*

Notation 4.5 *Pour tout noeud p de $T_q^m(G)$, on note $P' = P \cup \{x\}$. Et on note $H'_p = (G'[P'] - xF_m(p))/(\mathcal{MFM}(G[P]) \cup \{\{x\}\})$.*

Remarque 4.3 Comme les modules forts maximaux de $G[P]$ sont uniformes relativement à x dans $G'[P'] - xF_m(p)$, ce sont des modules de $G'[P'] - xF_m(p)$ et H'_p est bien défini.

Les conditions du théorème 4.17 s'appliquent aux graphes H'_p avec p un noeud de T_q^m .

Théorème 4.17 *Soit x un sommet à insérer dans un graphe de permutation G . $G' = G + x$ est un graphe de permutation ssi le noeud d'insertion q de l'arbre de décomposition modulaire T^m de G est coupé, ou bien q n'est pas coupé et les deux conditions suivantes sont remplies :*

1. q satisfait une des trois conditions suivantes :

- (a) q a deux fils mixtes f_1 et f_2 , et H'_q est un graphe de permutation qui admet un réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 , et x et f_2 sont consécutifs dans π_2 .
- (b) q a un unique fils mixte f_1 , et H'_q est un graphe de permutation qui admet un réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 .
- (c) q n'a pas de fils mixte et $H'_q = G'[Q]/(\mathcal{MFM}(G[Q]) \cup \{\{x\}\})$ est un graphe de permutation.

2. et tout noeud $p \neq q$ de T_q^m satisfait une des deux conditions suivantes :

- (a) p a un unique fils mixte f_1 , et H'_p est un graphe de permutation qui admet un réalisateur $R_{H'_p} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 , et x est le premier élément de π_2 .
- (b) p n'a pas de fils mixte, et H'_p est un graphe de permutation qui admet un réalisateur $R_{H'_p} = (\pi_1, \pi_2)$ tel que x est le premier élément de π_2 .

Preuve :

$\Rightarrow$. Si q n'est pas coupé, on montre que les noeuds de T_q^m satisfont les conditions du théorème 4.17. Les trois propositions ci-dessous traitent les cas où le noeud de T_q^m considéré a au moins un fils mixte, les cas où il n'en a pas se déduisent comme cas particulier des cas traités dans les propositions qui suivent.

Soit f un fils mixte de q et F le module fort de G correspondant. Soit $R' = (\pi'_1, \pi'_2)$ un réalisateur du graphe quotient $G'_{q'_s}$. Comme $\mathcal{MFM}(G'[Q'_s]) = \mathcal{MUM}(G[Q_s]) \cup \{\{x\}\}$ (voir théorème 4.16), retirer x de R' donne un réalisateur R de $G[Q_s]/\mathcal{MUM}(G[Q_s])$. Comme F est un module fort mixte de $G[Q_s]$, F n'est pas un singleton et le lemme 1.4 s'applique. Donc, l'ensemble $A = \{f' \in \mathcal{C}(q'_s) \mid F' \in \mathcal{MUM}(G[F])\}$ est un module fort de $G[Q_s]/\mathcal{MUM}(G[Q_s])$ et d'après le théorème 1.29 page 84, A est un intervalle commun du réalisateur R . De plus, comme F est mixte, A entoure x dans π'_1 ou π'_2 (voir définition 4.7).

Proposition 4.7 *Si G' est un graphe de permutation et si le noeud d'insertion q de T^m est non coupé et a deux fils mixtes f_1 et f_2 , alors H'_q est un graphe de permutation qui admet un réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 , et x et f_2 sont consécutifs dans π_2 . Preuve :* D'après la discussion ci-dessus, l'ensemble $A_1 = \{f \in \mathcal{C}(q'_s) \mid F \in$

$\mathcal{MUM}(G[F_1])\}$ et l'ensemble $A_2 = \{f \in \mathcal{C}(q'_s) \mid F \in \mathcal{MUM}(G[F_2])\}$ sont des intervalles communs du réalisateur R . Comme F_1 et F_2 sont tous les deux mixtes, nécessairement, A_1 et A_2 entourent x dans des ordres linéaires de R' différents. Sans perte de généralité, on peut supposer que A_1 entoure x dans π'_1 et A_2 dans π'_2 et que les éléments de A_1 sont plus petits que ceux de A_2 dans π'_1 (sinon, il suffit d'échanger et/ou de renverser les deux ordres du

réalisateur). Supposons que les sommets de F_1 ne sont pas adjacents à ceux de F_2 (l'autre cas est similaire). Alors les éléments de A_1 sont antérieurs à ceux de A_2 dans π'_2 . Il découle aussi que q est soit un noeud parallèle soit un noeud premier tel que f_1 n'est pas adjacent à f_2 dans G_q .

- Si q est un noeud premier, alors $Q = Q_s$ et $\mathcal{MUM}(G[Q_s]) = \mathcal{MUM}(G[F_1]) \cup \mathcal{MUM}(G[F_2]) \cup (\mathcal{MFM}(G[Q]) \setminus \{F_1, F_2\})$. Le réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ de H'_q s'obtient comme suit. On retire x de π'_1 et π'_2 . On remplace les éléments de A_1 par un unique élément f_1 et ceux de A_2 par f_2 . Finalement, on réinsère x juste après f_1 dans π_1 et juste avant f_2 dans π_2 .
- Si q est un noeud parallèle, alors par définition $Q \neq Q_s$ et on a $\mathcal{MUM}(G[Q_s]) = \mathcal{MUM}(G[F_1]) \cup \mathcal{MUM}(G[F_2]) \cup \{F_l\}$ (voir notation 4.2). Pour obtenir le réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ de H'_q , on procède d'abord comme dans le cas où q est premier. Ensuite, on remplace l'élément f_l représentant F_l par l'ensemble $\mathcal{C}_l(q)$, avec le même ordre relatif dans π_1 et dans π_2 . Finalement, on ajoute les éléments de $\mathcal{C}_{nl}(q)$ à la fin de π_1 et π_2 en prenant soin de les ranger dans le même ordre relatif.

□

Proposition 4.8 Si G' est un graphe de permutation et si le noeud d'insertion q de T^m est non coupé et possède un unique fils mixte f_1 , alors H'_q est un graphe de permutation qui admet un réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 . *Preuve :*

D'après la discussion ci-dessus, l'ensemble $A_1 = \{f \in \mathcal{C}(q'_s) \mid F \in \mathcal{MUM}(G[F_1])\}$ est un intervalle commun du réalisateur R et A_1 entoure x dans un des deux ordres linéaires de R' , disons π'_1 . Sans perte de généralité on peut supposer que x est antérieur aux éléments de A_2 dans π'_2 (sinon, on renverse les deux ordres). On distingue différents cas selon l'étiquette de q .

- Si q est un noeud premier, alors $Q = Q_s$ et $\mathcal{MUM}(G[Q_s]) = \mathcal{MUM}(G[F_1]) \cup (\mathcal{MFM}(G[Q]) \setminus \{F_1\})$. Le réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ de H'_q s'obtient comme suit. On retire x de π'_1 et π'_2 . On remplace les éléments de A_1 par un unique élément f_1 . Finalement, on réinsère x juste après f_1 dans π_1 et à sa place d'origine dans π_2 .
- Si q est un noeud parallèle (le cas où q est un noeud série est similaire), alors par définition $Q \neq Q_s$ et $\mathcal{MUM}(G[Q_s]) = \mathcal{MUM}(G[F_1]) \cup \{F_l\}$ (voir notation 4.2). Pour obtenir le réalisateur $R_{H'_q} = (\pi_1, \pi_2)$ de H'_q , on procède comme dans le cas où q est premier. Puis on remplace l'élément f_l représentant F_l par l'ensemble $\mathcal{C}_l(q)$, avec le même ordre relatif dans π_1 et π_2 . Finalement, on ajoute les éléments de $\mathcal{C}_{nl}(q)$ à la fin de π_1 et π_2 en prenant soin de les ranger dans le même ordre relatif.

□

Proposition 4.9 Si G' est un graphe de permutation et si le noeud d'insertion q de T^m est non coupé et si $p \neq q$ est un noeud de T_q qui possède un unique fils mixte f_1 , alors H'_p est un graphe de permutation qui admet un réalisateur $R_{H'_p} = (\pi_1, \pi_2)$ tel que x et f_1 sont consécutifs dans π_1 , et x est le premier élément de π_2 . *Preuve :* Soit $R' = (\pi'_1, \pi'_2)$ un

réaliseur de $G'[Q']$. En retirant x de R' on obtient un réalisateur R de $G[Q]$. Soit q_1 le fils de q qui est ancêtre de p . Comme Q_1 , P et F_1 sont des modules forts de $G[Q]$, alors Q_1 , P et F_1 sont des intervalles communs de R (voir théorème 1.29 page 84). Comme F_1 est mixte, F_1 (et a fortiori P et Q_1) entoure x dans un des deux ordres linéaires de R' , disons π'_1 . Supposons que $P \cup \{x\}$ est un intervalle de π'_2 . Alors $Q_1 \cup \{x\}$ est un intervalle de π'_2 et est donc un module de $G'[Q']$. Ainsi q ne peut avoir aucun fils mixte différent de q_1 et q ne peut pas être non propre : absurde. $P \cup \{x\}$ n'est pas un intervalle de π'_2 .

Soit p_u un fils de p différent de f_1 . Comme P n'entoure pas x dans π'_2 , P_u n'entoure pas x dans π'_2 . Par conséquent, comme P_u est uniforme rel. à x , P_u ne peut pas entourer x dans π'_1 . De plus, comme P_u est un module fort de $G[Q]$, alors P_u est un intervalle commun de $R = R'[V \setminus x]$. Il s'ensuit, comme P_u n'entoure x ni dans π'_1 ni dans π'_2 , que P_u est un intervalle commun de R' .

Voyons comment obtenir $R_{H'_p} = (\pi_1, \pi_2)$. Tout d'abord, en restreignant R' aux sommets de $P \cup \{x\}$ on obtient un réalisateur R'_p de $G'[P \cup \{x\}]$ tel que x est le premier ou le dernier élément de la restriction de π'_2 , sans perte de généralité on peut supposer que c'est le premier, et tel que $F_1 \cup \{x\}$ est un intervalle de la restriction de π'_1 . On peut déplacer x dans la restriction de π'_1 pour le placer juste avant F_1 (F_1 devient un intervalle commun). Ensuite, on remplace l'intervalle commun F_1 par un unique élément f_1 , et on remplace, pour tout fils p_u de p différent de f_1 , l'intervalle commun P_u par un unique élément p_u , et on obtient ainsi un réalisateur de H'_p qui satisfait les hypothèses du théorème 4.17. $\square$

$\Leftarrow$. D'après le lemme 4.13, pour conclure que G' est un graphe de permutation, il suffit de prouver que $G'[Q']$ est un graphe de permutation. Si q est coupé, alors aucun nouveau noeud premier n'est introduit et les réalisateurs de ceux qui restent sont inchangés (voir la discussion qui suit la définition 4.10 page 162). Donc G' est un graphe de permutation. On traite maintenant le cas où q est non coupé.

Pour commencer, on montre par induction que pour tout noeud $p \neq q$ de T_q^m , $G'[P']$ est un graphe de permutation qui admet un réalisateur $R_{G'[P']} = (\pi'_1, \pi'_2)$ tel que x est le premier élément de π'_2 . Si P est uniforme rel. à x , la propriété est vérifiée. Ainsi, les feuilles satisfont l'hypothèse d'induction.

Soit p un noeud mixte dont les fils satisfont l'hypothèse d'induction. Pour obtenir le réalisateur $R_{G'[P']}$, on procède comme suit. Dans $R_{H'_p} = (\pi_1, \pi_2)$, pour chaque $f \in \mathcal{C}(p)$ tel que f est uniforme, on substitue le réalisateur $R_{G'[F]}$ à f (voir la définition 1.40 de composition des réalisateurs, page 82).

Si p n'a pas de fils mixte, alors le théorème 4.17 (condition 2b) garantit que le réalisateur obtenu a la propriété souhaitée : x est le premier élément du second ordre linéaire.

Considérons donc le cas où p a un unique fils mixte f_1 . D'après l'hypothèse d'induction, $G'[F'_1]$ admet un réalisateur $R_{F'_1} = (\tau_1, \tau_2)$ tel que x est le premier élément de τ_2 . Ainsi, en substituant, dans π_1 , τ_1 à l'intervalle $\{x, f_1\}$ et en substituant, dans π_2 , $\tau_2[F_1]$ à l'élément f_1 , on obtient $R_{G'[P']}$. En effet, d'après le théorème 4.17 (condition 2a), x est le premier élément de π_2 et reste, après les manipulations décrites ci-dessus, le premier élément de π'_2 .

Intéressons nous maintenant au noeud d'insertion q .

Si q a au plus un fils mixte, alors la discussion précédente sur les descendants p de q

s'applique et montre comment construire un réalisateur de $G'[Q']$. $G'[Q']$ est donc un graphe de permutation.

Si q a deux fils mixtes f_1 et f_2 , nous avons prouvé que $G'[F'_1]$ et $G'[F'_2]$ ont respectivement un réalisateur $R_1 = (\tau_1, \tau_2)$ et $R_2 = (\sigma_1, \sigma_2)$ tel que x est le premier élément de τ_2 et σ_2 . Par hypothèse, q satisfait la condition 1a du théorème 4.17. Supposons sans perte de généralité que dans $R_{H'_q} = (\pi_1, \pi_2)$, f_1 se trouve avant f_2 dans π_1 . Si f_1 n'est pas adjacent à f_2 dans H'_q , alors f_1 se trouve avant f_2 dans π_2 , et après f_2 dans le cas contraire. On ne traite que le cas où f_1 et f_2 sont non adjacents, l'autre est similaire. Un réalisateur $R_{G'[Q']}$ peut s'obtenir de la manière suivante. Dans $R_{H'_q}$, pour chaque $f \in \mathcal{C}(p)$ tel que f n'est pas mixte, on substitue le réalisateur $R_{G[F]}$ à f . Ensuite, dans π_1 , on substitue $\bar{\tau}_1$ à l'intervalle $\{x, f_1\}$, et $\sigma_2[F_2]$ à f_2 ; et dans π_2 , on substitue σ_1 à l'intervalle $\{x, f_2\}$, et $\bar{\tau}_2[F_1]$ à f_1 . ■

Algorithme et complexité

Structure de donnée.

Pour implémenter la MD -représentation $MD(G) = (T^m(G), \mathcal{R}_G)$ d'un graphe de permutation G , on doit d'abord représenter l'arbre de décomposition modulaire. Chaque noeud maintient un pointeur vers son père et une liste de pointeurs vers ses fils. Chaque noeud reçoit une étiquette (premier, parallèle ou série) et à chaque noeud premier p est associé le réalisateur R_p de son graphe quotient G_p . Le réalisateur $R_p = (\pi_1, \pi_2)$ sera stocké dans deux listes doublement chaînées représentant les deux ordres linéaires π_1 et π_2 . Chaque cellule d'une liste représente un fils c de p et est reliée par un double pointeur à c . De plus, chaque cellule contient son rang dans la liste ($\pi_1(c)$ ou $\pi_2(c)$), ce qui permet de répondre à la requête d'adjacence dans les quotients en temps constant. Enfin, la structure de donnée contient aussi un réalisateur R_G de G tel que les cellules de ses deux listes sont reliées par un double pointeur aux feuilles de $T^m(G)$.

La procédure de marquage.

Cette procédure est exactement le procédé de marquage de [MS89], où les marques 1, -1 et 0 ont été remplacées par les types *plein*, *creux* et *mixte*. Chaque noeud reçoit son type au cours d'un parcours de l'arbre du bas vers le haut. Une feuille l_y de $T^m(G)$ est typée *plein* si $y \in N(x)$ et *creux* sinon. Chaque noeud fait suivre son type à son père. Si un noeud p réceptionne le même type de la part de tous ses fils, p se voit attribuer ce type. Sinon, p est typé *mixte*. La procédure se termine lorsque la racine reçoit un type. Il est clair que les noeuds typés *mixte* sont exactement les noeuds mixtes, les noeuds typés *plein* sont les noeuds pleins rel. à x et les noeuds typés *creux* sont les noeuds creux rel. à x . Comme le nombre de feuilles de $T^m(G)$ est $O(n)$ et comme chaque arête de $T^m(G)$ n'est traversée qu'une fois, la procédure de marquage s'exécute en temps $O(n)$.

Trouver le noeud d'insertion q .

Le but de cette étape est de localiser le noeud d'insertion q , dans le cas où la racine r de $T^m(G)$ est typée *mixte*. Par définition, q est le *ppca* des noeuds non propres de $T^m(G)$. Tout noeud p de l'unique chemin entre r et q est *mixte*, et propre si $p \neq q$. Comme, d'après la définition 4.8, tout noeud mixte propre possède un unique fils mixte, on peut trouver le noeud d'insertion par une recherche de haut en bas dans $T^m(G)$ qui suit le chemin de r à q , en choisissant comme noeud suivant l'unique fils mixte du noeud courant. La recherche s'arrête lorsque le noeud examiné p est non propre, ce qui peut être testé comme suit. Si p est un noeud série (resp. parallèle), alors p est propre ssi tous ses fils sauf un sont typés *plein* (resp. *creux*) et le fils restant est typé *mixte*. Si p est un noeud premier, p est propre ssi x a un jumeau dans le quotient de p , ce qui peut être testé par la routine *InsPrime* décrite en section 3.1. Dans les deux cas, tester si p est un noeud propre peut être fait en temps $O(|\mathcal{C}(p)|)$. Comme $T^m(G)$ contient $O(n)$ noeuds, la procédure trouve le noeud d'insertion q en temps $O(n)$.

Maintenir la MD -représentation.

Expliquons maintenant comment tester si $G'[Q']$ est un graphe de permutation ou non, et comment mettre à jour la MD -représentation le cas échéant. Considérons d'abord deux cas simples :

- Si le noeud d'insertion q a strictement plus de deux fils mixtes, d'après le lemme 4.17, $G'[Q']$ n'est pas un graphe de permutation : l'algorithme s'arrête. Ce test peut clairement être fait en temps $O(|\mathcal{C}(q)|)$.
- Si q est coupé, alors $G + x$ est un graphe de permutation (voir théorème 4.17) et les réalisateurs des graphes quotients des noeuds premiers restent inchangés. D'après la définition 4.10, q est soit un noeud dégénéré sans fils mixte mais des fils uniformes des deux types, soit un noeud premier sans fils mixte dont un est un jumeau de x dans G_q . Les deux cas peuvent être reconnus en temps $O(|\mathcal{C}(q)|)$. Cela est clair en ce qui concerne le cas dégénéré. Le cas premier peut être testé par un appel à la routine *InsPrime* de la section 3.1. Dans les deux cas, la mise à jour de $T^m(G)$ en $T^m(G')$ se fait, comme décrit à la page 162, en temps $O(|\mathcal{C}(q)|)$.

A partir de maintenant, on s'intéresse au cas où le noeud d'insertion q est non coupé. Avant tout, remarquons que d'après le lemme 4.12, les changements dans T^m résultant de l'insertion de x sont localisés dans le sous arbre T_q^m . La discussion ci-dessous s'attache à calculer $MD(G'[Q']) = (T^m(G'[Q']), \mathcal{R}')$. On pose $\tilde{T}' = T^m(G'[Q'])$.

Par souci de simplicité, l'algorithme est présenté en trois étapes (en pratique, ces trois étapes peuvent être effectuées simultanément) : 1) calculer l'arbre de décomposition modulaire T' ; 2) déterminer si $G'[Q']$ est un graphe de permutation ; et 3) si c'est le cas, calculer les réalisateurs de $\mathcal{R}'$. La figure 4.30 donne une vue générale de l'algorithme de mise à jour de la MD -représentation lorsque le graphe augmenté est un graphe de permutation.

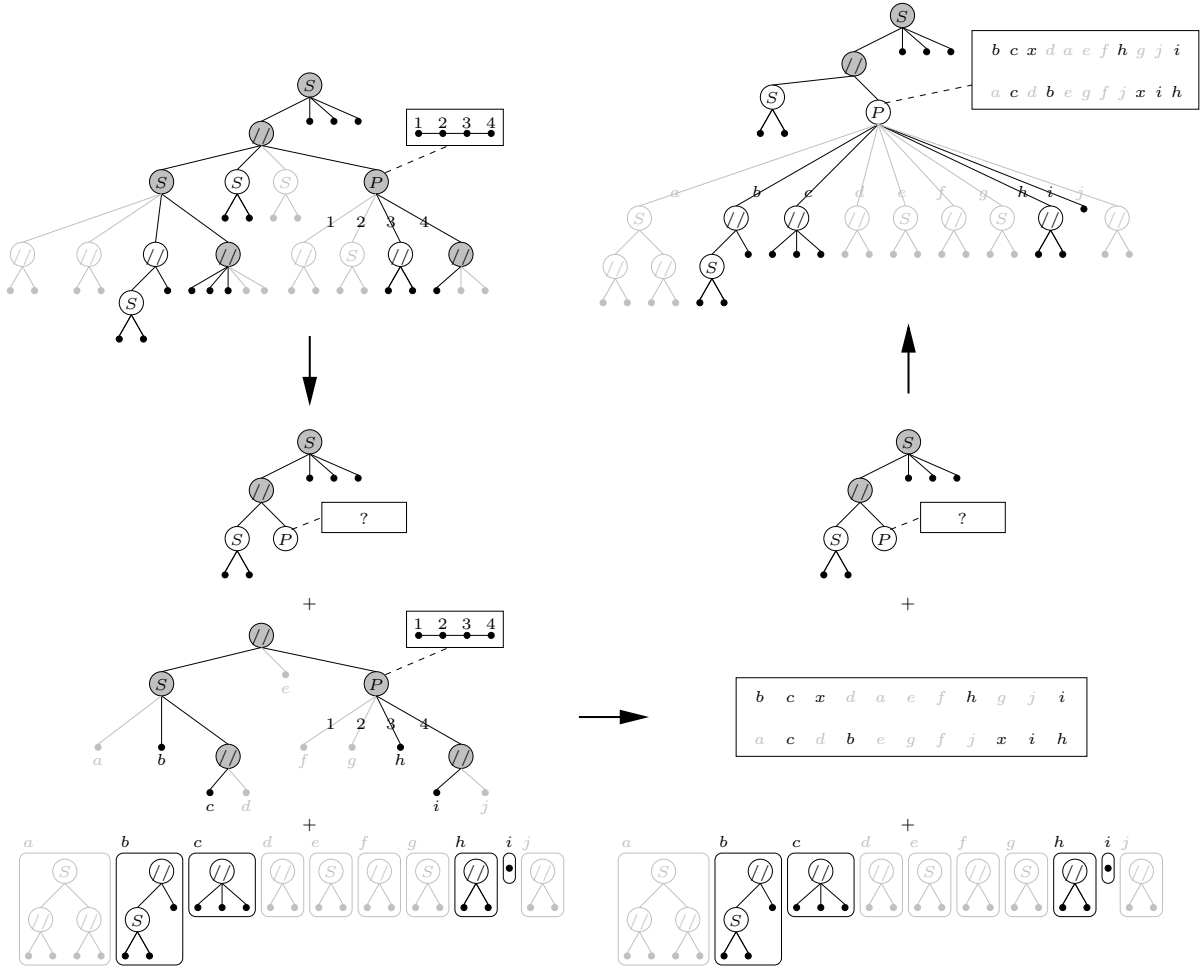


FIG. 4.30 – Mise à jour de la MD -représentation lors de l'insertion d'un sommet x . Les feuilles grises sont adjacentes à x , les feuilles noires sont non adjacentes à x . Les noeuds au contour gris sont pleins, les noeuds au contour noir sont vides et les noeuds au contour noir remplis en gris sont mixtes.

Calcul de T' . La discussion qui suit la preuve du théorème 4.16 explique comment obtenir T' une fois que l'arbre de décomposition modulaire a été typé par la procédure de marquage. La tâche principale est de déterminer les nouveaux modules forts qui apparaissent avec l'insertion de x . D'après le théorème 4.16 ces modules forts sont les modules uniformes maximaux. Ils peuvent être trouvés en temps $O(n)$ par une recherche dans T_{q_s} (voir notation 4.3), car M est un module uniforme maximal ssi il existe un noeud mixte p descendant de q_s tel que :

- p est dégénéré et $M = F_l(p)$ ou $M = F_{nl}(p)$; ou bien
- p est premier et M est l'ensemble de sommets représenté par un fils uniforme de p .

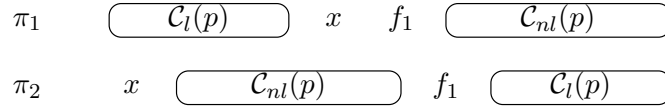


FIG. 4.31 – L’unique réalisateur de H'_p (si p est un noeud série) qui remplit la condition 2a du théorème 4.17. Pour un noeud p parallèle, $\mathcal{C}_{nl}(p)$ et $\mathcal{C}_l(p)$ doivent être échangés dans π_2 .

Tester si G' est un graphe de permutation. Considérons l’ensemble Q'_s défini dans la notation 4.3. Comme pour tout module uniforme maximal M de $G'[Q'_s]$, l’arbre de décomposition modulaire de $G'[M]$ s’obtient à partir de $T^m(G)$ sans ajouter de noeud premier, alors $G'[Q'_s]$ est un graphe de permutation ssi le graphe quotient $G'_{q'_s}$ en est un. Remarquons que $G'_{q'_s} = G[Q_s]/\mathcal{MUM}(Q_s) + x$ et que $MD(G[Q_s]/\mathcal{MUM}(Q_s)) = (\tilde{T}, \tilde{R})$ peut s’extraire de $MD(G)$ en temps $O(n)$. Il faut vérifier que chaque noeud p de $\tilde{T}$ remplit les conditions du théorème 4.17.

Comme q est non coupé, la racine de $\tilde{T}$ l’est aussi. Si p est un noeud dégénéré qui a le bon nombre de fils mixtes (0, 1 ou 2 selon si p est la racine ou non), alors H'_p possède toujours un réalisateur qui satisfait le théorème 4.17 (voir figure 4.31). Si p est premier, en utilisant la routine *InsPrime* (qui s’exécute en temps $O(|\mathcal{C}(p)|)$), on insère x dans $\tilde{R}_p$ en rendant x adjacent à $\mathcal{C}_l(p)$ et non adjacent à $\mathcal{C}_m(p) \cup \mathcal{C}_{nl}(p)$. Il peut y avoir deux positions d’insertion différentes pour x (seulement si x a un sommet jumeau). Ensuite on teste si au moins une des positions possibles remplit les conditions du théorème 4.17 ce qui consiste simplement à tester la position de x dans le réalisateur renvoyé par *InsPrime* : x est-il un élément extrémal dans un des ordres et/ou est-il consécutif avec le fils mixte ? Ce test se fait en temps constant. Comme on ne manipule que les graphes quotients des noeuds premiers de $\tilde{T}$, chacun étant traité en temps $O(|\mathcal{C}(p)|)$, la complexité de la deuxième étape est $O(n)$.

Calcul de $\mathcal{R}'$. On se place maintenant dans le cas où $G' = G + x$ est un graphe de permutation. Souvenons nous que le graphe quotient $G'_{q'_s}$ est le graphe $G[Q_s]/\mathcal{MUM}(Q_s) + x$ (voir théorème 4.16). Comme pour la deuxième étape, l’algorithme utilise l’identité $MD(G[Q_s]/\mathcal{MUM}(Q_s)) = (\tilde{T}, \tilde{R})$. Pour des raisons de complexité, les rangs des cellules dans les listes constituant les réalisateurs intermédiaires calculés au cours de la procédure ne sont pas maintenus.

Obtenir $\mathcal{R}'$ se résume à calculer $R_{G'[Q'_s]}$ qui est le seul nouveau réalisateur introduit dans $MD(G'[Q'])$. A cette fin, on effectue le parcours de $MD(G[Q_s]/\mathcal{MUM}(Q_s))$ décrit dans la preuve de l’implication réciproque du théorème 4.17. Pour un noeud premier mixte p de $\tilde{T}$, le réalisateur de H'_p est fourni par la routine *InsPrime*. Pour un noeud dégénéré p de $\tilde{T}$, le réalisateur de H'_p est celui représenté sur la figure 4.31. Comme les réalisateurs sont implémentés par des paires de listes doublement chaînées, l’opération de substitution utilisée dans la preuve du théorème 4.17 se fait en temps constant. Donc, durant le parcours, chaque noeud p est traité en temps $O(|\mathcal{C}(p)|)$. Il suit que le réalisateur $R_{G'[Q'_s]}$ est calculé en temps $O(n)$.

Finalement, pour maintenir la totalité de la structure de donnée, un parcours des listes de $R_{G'[Q'_s]}$ permet de numéroter les cellules par leur rang.

Théorème 4.18 *Mettre à jour la MD-représentation d'un graphe de permutation lors de l'insertion d'un sommet coûte un temps $O(n)$.*

4.6 Conclusion

Nous l'avons vu dans les sections précédentes de ce chapitre, lorsque l'on considère des classes de graphes pour lesquels la décomposition modulaire est suffisamment contrainte, il est possible de l'entretenir entièrement dynamiquement à la fois sur les arêtes et sur les sommets avec un coût par modification nettement inférieur au coût $O(n + m)$ du calcul statique de la décomposition modulaire pour un graphe quelconque. À la lumière de la série d'exemples de classes de graphes qui ont été traitées de cette manière, les contraintes qui apparaissent déterminantes pour la complexité sont celles s'appliquant sur les noeuds premiers.

Nous avons rencontré plusieurs cas de figure. Le premier cas est celui des familles de graphes qui ne contiennent pas de noeuds premier dans leur arbre de décomposition. Pour les graphes non orientés, il s'agit des cographes pour lesquels [CPS85, SS04] montrent que l'on peut traiter les modifications de sommet de degrés d en $O(d)$ et les modifications d'arête en $O(1)$.

Nous avons pu étendre ce résultat à la famille des cographes orientés [CP06], montrant par là que l'orientation ne constituait pas un obstacle majeur, et renforçant l'idée que la difficulté de l'entretien dynamique de la décomposition modulaire réside principalement dans les noeuds premiers. Idée que corrobore le résultat de [NPP06] sur les P_4 -sparse.

Sur cette famille dont les quotients des noeuds premiers sont représentables en espace $O(1)$, [NPP06] obtient les mêmes complexités de mise à jour que pour les cographes et les cographes orientés. Faisons remarquer que cette possibilité ne repose pas uniquement sur la bonne représentation des noeuds premiers des graphes P_4 -sparse, elle repose aussi sur la structure générale de l'arbre qui n'est pas entièrement libre : les noeuds premiers ne peuvent avoir qu'au plus un fils non feuille. Par contre, la hauteur de l'arbre n'est pas bornée et les noeuds premiers peuvent être ancêtres les uns des autres.

La famille des graphes P_4 -sparses est très éloignée de celle des graphes en général. C'est sur ce point que la famille des graphes de permutation est remarquable. Pour cette famille de graphes, tout se passe comme dans le cas des graphes quelconques, excepté le fait que les quotients des noeuds premiers doivent être et rester des graphes de permutation. Cela vient du fait que la famille des graphes de permutation est proprement décomposée par la décomposition modulaire (voir section 1.4.3). C'est à dire qu'un graphe est de permutation ssi les quotients des noeuds premiers de son arbre de décomposition modulaire le sont. Or, pour tout entier n , il existe des graphes de permutation premiers à n sommets, par exemple le chemin à n sommets. On en déduit le théorème suivant.

Théorème 4.19 *Tout arbre de décomposition modulaire est l'arbre de décomposition d'un graphe de permutation.*

Cette propriété fait des graphes de permutation une excellente famille à étudier pour s'approcher du problème sur les graphes quelconques. Ce qui ressort de l'algorithme proposé en section 4.5, c'est que même sans contrainte sur la forme de l'arbre de décomposition modulaire, si les quotients des noeuds premiers ont une représentation efficace, alors il est possible d'entretenir l'arbre de décomposition dynamiquement en temps raisonnable. Nous avons vu que c'est le cas des graphes de permutation. Mais cela pose aussi la question pour les graphes quelconques : existe-t-il une représentation des graphes premiers en général qui permette de traiter efficacement l'entretien dynamique de l'arbre de décomposition modulaire ?

Ce qui suit essaye de déterminer quelles sont les exigences que devrait satisfaire une telle représentation. Pour commencer, il convient de préciser un objectif de complexité à atteindre pour les modifications élémentaires d'arête et de sommet : visons une complexité de $O(n)$ par modification d'arête ou de sommet.

Pourquoi $O(n)$? Remarquons tout d'abord qu'une telle complexité serait un gain très appréciable par rapport au temps $O(n + m)$ pris par le calcul statique de l'arbre de décomposition modulaire. On pourrait chercher à atteindre une meilleure complexité, à l'instar de ce qui est réalisé pour les cographes, les cographes orientés et les graphes P_4 -sparses. Cependant, il me paraît difficile d'atteindre une complexité plus basse que $O(n)$ pour les graphes en général comme pour les graphes de permutation. Si on cherche à obtenir mieux, d'une certaine manière, le problème change de nature.

En effet, il existe des modifications d'arête, insertion comme suppression, et des suppression de sommets qui font passer d'un graphe premier (quelconque ou de permutation), ayant un seul noeud interne dans son arbre de décomposition, à un cographe ayant $\Omega(n)$ noeuds dans son arbre de décomposition, pour des entiers n arbitrairement grands. Cela signifie que si on veut atteindre pour ces modifications une complexité meilleure que $O(n)$, on ne peut pas calculer entièrement la modification de l'arbre. Intuitivement, il faudrait que la structure de donnée employée pour représenter un graphe premier soit assez proche d'un coarbre pour qu'on puisse l'obtenir en un temps de calcul moindre que la taille du coarbre à produire. Cela paraît une exigence très forte. Je ne sais pas si cela est possible, mais cet objectif est laissé de côté dans ce mémoire. Ici, on s'autorise un calcul de longueur $O(n)$ à chaque mise à jour, ce qui laisse possible la construction de bout en bout du coarbre résultant des transformations précitées. Remarquons que pour cette complexité, on peut toujours supposer que l'on entretient les listes d'adjacence du graphe (voir section 2.4). L'algorithme peut donc à tout moment parcourir la liste des voisins d'un sommet x quelconque en $O(d(x))$.

Pour obtenir une complexité de $O(n)$ pour les quatre opérations élémentaires, traiter les modifications de sommet dans cette complexité est suffisant. En effet, comme remarqué en section 4.5, toute modification d'arête se ramène à deux modifications de sommet.

Les exigences à satisfaire par la représentation des quotients des noeuds premiers pour traiter les insertions de sommet en temps $O(n)$ sont au nombre de deux :

- pouvoir déterminer si le sommet x à insérer a un jumeau dans un graphe premier, et le trouver si c'est le cas, en temps $O(n)$;
- lorsqu'un graphe quelconque G devient un graphe premier G' après l'insertion d'un sommet, pouvoir calculer la représentation de G' en temps $O(n)$ à partir de l'arbre de décomposition modulaire de G et des représentations des quotients de ses noeuds premiers.

Ces exigences sont légitimes car il faut bien traiter ces deux cas particuliers dans la complexité désirée. Le fait qu'elles soient suffisantes découle de l'algorithme pour les graphes de permutations décrit en section 4.5, qui peut être réécrit en utilisant que ces deux primitives et sans jamais supposer que l'on manipule un graphe de permutation.

En ce qui concerne l'opération de retrait de sommet, comme montré en section 4.5, le retrait d'un sommet dont la feuille correspondante est fils d'un noeud dégénéré ne pose pas de problème. Ainsi, la seule exigence à satisfaire par la représentation des quotients des noeuds premiers pour traiter les suppression de sommet en temps $O(n)$ est :

- lorsqu'un sommet x est retiré d'un graphe premier $G = (V, E)$, pouvoir calculer l'arbre de décomposition modulaire de $G' = G[V \setminus \{x\}]$ avec les représentations des quotients de ses noeuds premiers en temps $O(n)$.

Aucune de ces trois conditions ne paraît simple à obtenir, et il faut les satisfaire toutes les trois pour avoir un algorithme entièrement dynamique avec la complexité souhaitée de $O(n)$ par modification.

La N -représentation de [MS89] remplit les deux conditions relatives à l'ajout de sommet si on possède par ailleurs une structure permettant de tester l'adjacence en temps constant, ce qui est déjà un obstacle majeur pour un algorithme dynamique. De plus, il lui manque de satisfaire la condition pour le retrait de sommet qui n'est pas du tout évidente à obtenir. En conclusion, l'approche de [MS89] est remarquable et très encourageante. Mais le travail restant à accomplir pour arriver à un algorithme entièrement dynamique est conséquent. Dans la perspective de concevoir un tel algorithme, une des contributions de ce manuscrit est de :

- localiser le noyau de la difficulté, la représentation des quotients premiers, et identifier les principales contraintes à remplir par cette représentation ;
- fournir, par l'algorithme pour les graphes de permutation, un cadre générique pouvant servir de support à des algorithmes aussi bien sur des classes de graphes particulières que sur les graphes quelconques.

4.7 Perspectives

Nous venons de voir que la décomposition modulaire est un outil performant pour traiter la reconnaissance dynamique de plusieurs classes de graphes. Dans cette section nous envisageons d'autres classes de graphes dont la reconnaissance dynamique a été étudiée et nous examinons le rôle des décompositions de graphes dans ce contexte. Les classes considérées sont celles des graphes d'intervalles unitaires, des graphes distance héréditaires et des graphes triangulés. Pour ces classes, on se pose les questions : les techniques utilisées

font elles appel à des décompositions de graphes ? quelles décompositions peuvent aider à la reconnaissance dynamique et de quelle manière ?

4.7.1 Les graphes d'intervalles unitaires

Le cas des graphes distance héréditaires est très intéressant. A première vue, la représentation utilisée par l'algorithme de reconnaissance de [HSS02] ne fait usage d'aucune décomposition de graphe. Pourtant, en y regardant de plus près, on s'aperçoit que cette représentation est très fortement reliée à la décomposition modulaire. Il serait intéressant de pousser cette voie plus loin et de voir s'il est possible de réécrire entièrement l'algorithme de [HSS02] en termes de décomposition modulaire. Ici, nous nous contentons de montrer en quoi la représentation utilisée s'apparente à la décomposition modulaire.

La classe des graphes d'intervalles unitaires a été présentée en section 1.3.1. La représentation de la classe utilisée par [HSS02] est basée sur le théorème 1.17 qui est présenté dans une version reformulée et enrichie ici (théorème 4.20).

La relation $\mathcal{R}$ sur les sommets d'un graphe G définie par $x\mathcal{R}y$ ssi $N[x] = N[y]$, où $N[x]$ désigne le voisinage fermé de x (voir les généralités communes au mémoire), est à l'évidence une relation d'équivalence. La classe d'équivalence de x par cette relation est constituée de x et de l'ensemble de ses vrais jumeaux (voir section 1.2.1). En raccourcissant le vocabulaire, on parle de classes de vrais jumeaux. Si A et B sont deux classes de vrais jumeaux, alors soit les sommets de A voient tous les sommets de B , soit aucun sommet de A ne voit un sommet de B . En effet, les classes de vrais jumeaux sont des modules, et l'ensemble de ces classes est donc une partition de congruence. On peut donc définir le graphe quotient de G par ses classes de vrais jumeaux, aussi appelé graphe des classes de vrais jumeaux.

Définition 4.11 *Un ordre de contiguïté fermé d'un graphe $G = (V, E)$ est un ordre linéaire ϕ sur les sommets de G tel que pour tout $x \in V$, $N[x]$ est un intervalle de ϕ .*

La première partie du théorème qui suit est une formulation équivalente du théorème 1.17.

Théorème 4.20 [DHH96] *Un graphe G est un graphe d'intervalles unitaires ssi le graphe de ses classes de vrais jumeaux admet un ordre de contiguïté fermé. De plus, si G est un graphe d'intervalles unitaires connexe, alors cet ordre est unique à renversement près.*

[HSS02] entretient les deux ordres de contiguïté des classes de vrais jumeaux de chaque composante connexe du graphe d'intervalles unitaires considéré G . En fait, cela peut être vu comme l'entretien d'une représentation des noeuds premiers de l'arbre de décomposition modulaire de G . En effet, l'arbre de décomposition modulaire d'un graphe d'intervalles unitaires a une structure très simple, sa hauteur est bornée et l'arbre de décomposition de chacune de ses composantes connexes ne peut contenir qu'au plus un noeud premier. L'ordre de contiguïté entretenu par [HSS02] est une représentation de ce noeud premier.

Théorème 4.21 *Soit G un graphe d'intervalles unitaires connexe et soit T^m son arbre de décomposition modulaire. Si p est un noeud parallèle ou premier de T^m , alors tous les noeuds internes de T_p différents de p sont séries. De plus, si p est parallèle il a au plus deux fils.*

Preuve : Soit p un noeud parallèle de T^m . Comme G est connexe, la racine r de T^m n'est pas un noeud parallèle. Il existe donc un fils r_1 de r qui n'est pas ancêtre de p tel que les sommets de R_1 sont tous adjacents aux sommets de P . Soit $u \in R_1$. Supposons que p ait au moins trois fils, alors il existe trois sommets $x, y, z \in P$ qui sont indépendants. x, y, z, u induisent une griffe : absurde. Supposons que T_p contienne un noeud q qui ne soit pas série. Il existe deux sommets $x, y \in Q$ qui sont indépendants. Soit p_1 un fils de p qui n'est pas ancêtre de q . Soit $z \in P_1$, les sommets x, y, z, u induisent une griffe : absurde.

Soit maintenant p un noeud premier de T^m . Supposons que T_p contienne un noeud q qui ne soit pas série. Il existe deux sommets $x, y \in Q$ qui sont indépendants. Soit p_1 le fils de p qui est ancêtre de q . D'après le lemme 1.3, dans G_p , p_1 a au moins un non voisin p_2 et un voisin p_3 qui sont adjacents. Soit $z \in P_2$ et $z' \in P_3$, les sommets x, y, z, z' induisent une griffe : absurde. ■

Grâce au théorème 4.21 ci-dessus et au théorème 5.3 de la page 189 qui caractérise l'arbre de décomposition modulaire des graphes d'intervalles, on peut entièrement caractériser celui des graphes d'intervalles unitaires.

Comme pour tous les graphes, l'arbre de décomposition T^m d'un graphe d'intervalle unitaire a pour racine un noeud parallèle ssi le graphe est non connexe. Si la racine de T^m est un noeud parallèle alors ses fils sont les racines respectives des arbres de décompositions des restrictions du graphe à une de ses composantes connexes. Ainsi, caractériser l'arbre de décomposition modulaire d'une classe de graphe se ramène à caractériser celui des éléments de la classe qui sont connexes. Si dans le cas général, cela ne fait pas une grosse différence, la restriction est simplificatrice pour les graphes d'intervalles unitaires car on peut énoncer le résultat suivant :

Corollaire 4.3 *Soit G un graphe d'intervalles unitaires connexe et T^m son arbre de décomposition modulaire. Tous les noeuds internes de T^m sont étiquetés série sauf éventuellement un.*

Preuve : Supposons que T^m contienne au moins deux noeuds non séries. D'après le théorème 4.21, aucun couple de tels noeuds ne peut être comparable par la relation «être ancêtre de». De plus, l'ensemble des ancêtres de ces noeuds ne contient nécessairement que des noeuds séries. Comme des noeuds séries ne peuvent pas être liés par la relation de parenté, on en déduit que cet ensemble est réduit à un unique élément, qui est la racine de T^m . Ce qui veut dire que tous les noeuds non séries de T^m sont fils de la racine. Or d'après le théorème 5.3, dans l'arbre de décomposition d'un graphe d'intervalle, les noeuds série ont au plus un fils qui est un noeud interne. Ce qui conclut à une absurdité et prouve le corollaire. ■

Finalement, on tire les conclusions suivantes :

1. si l'arbre T^m de décomposition d'un graphe d'intervalles unitaires connexe G a pour racine r un noeud série. Alors r n'a que des fils feuilles sauf au plus un p .
 - (a) Si p est parallèle, alors p a deux fils et chacun d'eux est soit une feuille, soit un noeud série n'ayant que des fils feuilles.
 - (b) Si p est premier, alors nécessairement le graphe quotient associé G_p est un graphe d'intervalles unitaires (par hérédité) mais ne contient pas trois sommets indépendants, sinon G contiendrait une griffe (en ajoutant un fils feuille de r à ces trois sommets indépendants). De plus, comme G est un graphe d'intervalles, d'après le théorème 5.3, seuls les fils de p correspondant à des sommets simpliciaux de G_p peuvent être des noeuds internes. D'après le théorème 4.21, ces noeuds sont séries et tous leurs fils sont feuilles.
2. Si r est un noeud premier, alors G_r peut être n'importe quel graphe d'intervalles unitaires premier, et comme dans le cas précédant, seuls ses fils correspondant à des sommets simpliciaux de G_p peuvent être des noeuds internes, et ce sont des noeuds séries dont tous les fils sont feuilles.

Remarque 4.4 Remarquons que les conditions ci-dessus impliquent que la hauteur de l'arbre de décomposition modulaire d'un graphe d'intervalles unitaires est au plus de cinq.

Si la décomposition modulaire d'un graphe G vérifie les conditions ci-dessus, alors G est un graphe d'intervalles unitaires. D'après le théorème 5.3, G est clairement un graphe d'intervalles. Il reste à vérifier que G ne contient pas de griffe. Dans le cas 1, il n'est pas dur de vérifier que G ne contient même pas de S_3 (trois sommets indépendants). Par conséquent il ne contient pas de griffe. Dans le cas 2, supposons qu'il existe quatre sommets de G induisant une griffe. Au moins deux d'entre eux, notés a et b , sont dans un même module fort maximal de G , sinon G_r contiendrait lui-même une griffe et ne serait pas d'intervalles unitaires comme exigé par le condition 2. Comme les modules forts maximaux de G sont des cliques, a et b sont adjacents, et l'ensemble des sommets de la griffe qui se trouvent dans le même module fort maximal qu'eux (y compris eux) est un module de la griffe. Or le seul module de la griffe contenant une arête interne est le module trivial du graphe entier. Donc tous les sommets de la griffe sont dans le même module fort maximal, et par conséquent, la griffe est une clique : absurde.

Comme un graphe est un graphe d'intervalles unitaires ssi les restrictions respectives de ce graphe à une de ses composantes connexes le sont, on a complètement caractérisé l'arbre de décomposition modulaire des graphe d'intervalles unitaires.

Examinons maintenant la représentation d'un graphe d'intervalles unitaires G adoptée par [HSS02]. Elle est basée sur l'ordre de contiguïté du graphe des classes de vrais jumeaux. Or les classes de vrais jumeaux sont facilement lisibles sur l'arbre de décomposition T^m de G . On exclut d'emblée le cas trivial où G est une clique : il y a une seule classe de vrais jumeaux et T^m ne comporte qu'un noeud interne. Dans le cas 1a de la discussion ci dessus, lorsque T^m comporte un noeud parallèle p , il y a trois classes de vrais jumeaux qui sont l'ensemble des fils feuilles des trois noeuds séries de T^m (disant cela, on suppose

que les deux fils de p sont des noeuds internes, si ce sont des feuilles, la classe de vrais jumeaux correspondante est simplement réduite à un sommet). L'unique, à renversement près, ordre de contiguïté est facile à trouver : il faut que la classe des fils feuilles de la racine soit entre celles des deux fils de p . Dans ce cas, la représentation par ordre de contiguïté et la décomposition modulaire sont équivalentes : les deux définissent la même tripartition des sommets du graphe. Dans les cas 1b et 2, lorsque T^m comporte un noeud premier p , les classes de vrais jumeaux sont les ensembles respectifs des feuilles de chaque sous-arbre de T^m enraciné en un fils de p , plus l'ensemble des fils feuilles de la racine de T^m lorsque celle-ci est série. Le graphe quotient G_p associé à p est un graphe d'intervalles unitaires connexe, il admet donc un ordre de contiguïté Ψ . Ψ est précisément l'ordre de contiguïté Φ des classes de vrais jumeaux dans lequel on a pris soin de remplacer chaque classe par le sommet de G_p correspondant et de retirer la classe des sommets universels lorsqu'il y en a une. On peut représenter, dans l'arbre de décomposition modulaire, les quotients G_p des noeuds premiers par leur ordre de contiguïté avec pour chaque sommet de G_p un pointeur vers le premier et le dernier de ses voisins dans l'ordre de contiguïté. On obtient ainsi une représentation complète des quotients et donc du graphe entier lui-même. Cette représentation basée sur la décomposition modulaire est équivalente à celle de [HSS02].

Ainsi, on peut voir l'algorithme de [HSS02] comme réalisant l'entretien de la décomposition modulaire des graphes d'intervalles unitaires en prenant pour représentation des quotients des noeuds premiers un ordre de contiguïté.

4.7.2 Graphes distance héréditaires

Les graphes distance héréditaires sont ceux pour lesquels la distance entre toute paire de sommets dans un sous-graphe induit connexe est la même que dans le graphe entier. Cette classe de graphes est fortement liée à la décomposition en coupes. En effet, les graphes distance héréditaires sont exactement les graphes entièrement décomposables pour la décomposition en coupe (voir section 1.2.2). Ils jouent donc le rôle des cographes pour la décomposition modulaire et leur reconnaissance dynamique est un premier pas vers le maintien dynamique de la décomposition en coupes pour des classes de graphes particulières ou pour les graphes quelconques.

Il existe à ma connaissance deux algorithmes de reconnaissance dynamique des graphes distance héréditaires. Le premier [CT07] ne fonctionne que sur les arêtes. Il n'est pas basé sur la décomposition en coupes mais sur une autre caractérisation de la classe. Cette caractérisation repose sur les graphes induits par les couches de distance à partir d'un sommet donné. Ces graphes sont nécessairement des cographes et les liaisons entre couches de distance vérifie des relations assez contraignantes. En s'appuyant sur ce fait, [CT07] traite les ajouts et suppressions d'arête en temps constant.

[GP07] montre que l'ajout et la suppression de sommet peuvent se traiter de manière optimale en $O(d)$ par opération, où d est le degré du sommet modifié, ce qui égale la complexité atteinte pour les cographes [CPS85, SS04]. Leur algorithme est basé sur la décomposition en coupes et montre de fortes analogie avec le cas des cographes. Lors de la suppression d'un sommet, il est très aisé d'entretenir l'arbre de décomposition. Pour traiter l'ajout,

[GP07] utilise un procédé de marquage de l'arbre de décomposition similaire à celui utilisé par [CPS85] pour les cographes, à la différence près que dans le cas de la décomposition en coupes, l'arbre n'est pas enraciné. Basé sur les informations fournies par ce procédé de marquage, [GP07] donne une caractérisation des cas dans lesquels l'insertion débouche sur un graphe distance héréditaire.

Ceci constitue une première étape vers un entretien dynamique de la décomposition en coupes pour d'autres classes de graphes. En particulier, il est naturel de songer à la prolonger pour les graphes de cordes pour lesquels nous avons vu en section 1.4.4 que la décomposition en coupes représente l'ensemble de leurs modèles géométriques.

4.7.3 Graphes triangulés

Le cas des graphes triangulés n'est pas directement relié à une décomposition mais utilise une représentation arborescente à degrés de liberté pour décrire l'ensemble des modèles géométriques du graphe (intersection de sous-arbres d'un arbre). Il n'est cependant pas à exclure qu'une décomposition de graphes soit présente derrière cette structure : on ne peut qu'être frappé par les connexions fortes entre l'arbre des cliques d'un graphe triangulé et sa décomposition par séparateurs complets (section 1.2.4).

L'algorithme dynamique de reconnaissance des graphes triangulés que nous présentons très succinctement ici est du à Ibarra [Iba99, Iba00]. Cet algorithme traite uniquement les insertions et suppressions d'arêtes. L'auteur donne plusieurs implémentations possibles avantagant certaines opérations ou requêtes. Dans l'implémentation la plus équilibrée, les deux opérations d'insertion et de suppression d'arêtes prennent un temps $O(n)$.

L'algorithme d'Ibarra entretient un arbre des cliques du graphe G considéré. Comme le montre les chapitres 3 et 4, il n'est pas toujours très difficile de décider si un modèle d'intersection donné est compatible avec une modification du graphe. Cependant, pour pouvoir traiter une modification, il ne suffit pas de considérer un modèle d'intersection, mais il faut être capable de tester la possibilité de modification dans tous les modèles possibles pour G . Nous avons vu comment les décompositions de graphes pouvaient apporter une réponse à ce problème en représentant l'ensemble des modèles possibles pour un graphe. Dans le cas des graphes triangulés, ce rôle n'est pas joué par une décomposition mais par le graphe W_G des cliques maximales de G (voir section 1.3.5). Pour ma part, il me semble assez remarquable qu'une structure aussi simple suffise à manier tous les réalisateurs en vue de l'entretien dynamique. Cela peut paraître étonnant aussi si on considère que la façon de représenter un modèle dans la structure, un arbre couvrant de poids minimum, ne paraît pas à première vue facile à manier algorithmiquement.

[Iba00] donne les théorèmes de structure qui montrent comment utiliser le graphe des cliques maximales et un arbre de clique pour déterminer si une suppression ou une insertion d'arête est possible. Pour deux cliques maximales K_1 et K_2 de G , on note $w(K_1, K_2)$ le poids de l'arête K_1K_2 dans W_G , c'est à dire

Théorème 4.22 [Iba00] *Soit G un graphe triangulé comportant l'arête xy . $G - xy$ est triangulé ssi G a une unique clique maximale contenant x et y .*

Lorsque c'est le cas, [Iba00] montre comment actualiser la structure de donnée. Pour l'ajout, il ne suffit pas de connaître un arbre de cliques pour déterminer si l'ajout est possible, c'est ce que montre le théorème suivant.

Théorème 4.23 [Iba00] *Soit G un graphe triangulé ne comportant pas l'arête xy . $G + xy$ est triangulé ssi il existe un arbre de cliques T de G tels que deux noeuds voisins de T contenant respectivement x et y .*

Il faut donc être capable de considérer l'ensemble des arbres de cliques pour vérifier si un d'eux vérifie la propriété. Comme annoncé, ce test peut être fait assez simplement sur le graphe des cliques maximales de G .

Théorème 4.24 [Iba00] *Soit G un graphe triangulé ne comportant pas l'arête xy . Soit T un arbre de cliques de G et soit K_1 et K_2 les deux noeuds de T les plus proches tels que $x \in K_1$ et $y \in K_2$. Il existe un arbre de cliques T' de G tels que deux noeuds voisins de T' contiennent respectivement x et y ssi l'arête e de poids minimum sur le chemin de K_1 à K_2 a précisément pour poids $w(K_1, K_2)$.*

Beaucoup de questions peuvent être posées en observant les résultats de [Iba00], parmi lesquelles on trouve les suivantes.

Question ouverte 4.2 *Comment traiter les cas de suppression et d'insertion de sommet ? Existe-t-il une meilleure représentation de l'ensemble des modèles d'intersection d'un graphe triangulé, qui permettrait par exemple de traiter l'entretien dynamique en des temps plus efficaces que $O(n)$? Quels sont les liens entre l'arbre des cliques et la décomposition par cliques séparantes d'un graphe triangulé ? Quel est le comportement dynamique de cette dernière décomposition ?*

Chapitre 5

Représentations et dynamicité des graphes d'intervalles

Ce chapitre est consacré au problème de la reconnaissance dynamique des graphes d'intervalles. Il y a déjà eu plusieurs travaux traitant du sujet [BL76, Kor87, KM89, Hsu96, Iba01].

Commençons par remarquer que les algorithmes de [BL76, KM89] ne sont pas incrémentaux au sens strict du terme.

Celui de [BL76] est incrémental sur les contraintes de consécutivité, mais pas sur les sommets. En effet, [BL76] effectue un calcul préliminaire pour déterminer les cliques maximales du graphe G . La donnée de départ de la partie incrémentale de l'algorithme est un PQ -arbre ayant un unique noeud interne qui est dégénéré et les cliques maximales de G comme feuilles. Ensuite, la contrainte de consécutivité correspondant à chaque sommet est insérée et le PQ -arbre est modifié en intégrant cette nouvelle contrainte. Le calcul préliminaire des cliques suppose la connaissance préalable du graphe, ce qui fait que l'algorithme n'est pas purement incrémental.

On pourrait songer à calculer dynamiquement les cliques créées par l'insertion d'un nouveau sommet. Pour cela, il faut parcourir toutes les cliques maximales existantes. Avec la structure classique du PQ -arbre, qui stocke les cliques en extension, cela résulte en une complexité de $O(n + m)$. Cette complexité n'est pas acceptable car c'est celle d'un algorithme statique de reconnaissance. C'est ce qui pousse [KM89] à adopter une structure de donnée très proche du PQ -arbre mais ayant une taille de $O(n)$. Cette structure, appelée MPQ -arbre est présentée dans [KM85]. La structure que nous utilisons ici, la PQ -représentation, est la même à des détails d'implémentation près (voir section 5.2). Grâce au MPQ -arbre, le calcul des cliques créées par l'insertion d'un sommet peut être fait en temps $O(n)$.

Une autre raison pour laquelle nous n'utiliserons pas l'algorithme de [BL76], même dans sa partie incrémentale, est que celui-ci est difficile d'analyse car il met peu en avant la structure du problème. Cet algorithme consiste en l'application de 16 cas de figures génériques de bas en haut de l'arbre, qui lui a souvent valu d'être décrié. De plus, comme le souligne [KM89], cela rend l'analyse de complexité assez difficile.

L'algorithme de [KM89] est nettement plus simple. Malheureusement, il n'est pas non plus purement dynamique car les sommets doivent être ajoutés dans un ordre prédéfini qui est calculé statiquement, ce qui oblige à connaître le graphe à l'avance.

Dans [Hsu96], est présenté un algorithme incrémental dont la complexité de calcul sur tout le graphe est $O(n \log n + m)$. Cet algorithme utilise une structure de donnée qui est une modification de la décomposition modulaire du graphe : les ensembles utilisés pour décomposer le graphe sont les modules connexes. Comme le souligne [Hsu96], cette structure ne peut supporter les suppressions de sommet.

Enfin, pour être complets, signalons que [Iba01] propose un algorithme entièrement dynamique sur les arêtes qui traite une opération en temps $O(n \log n)$. Cet algorithme est basé sur l'ordre d'inclusion des séparateurs minimaux et cliques maximales du graphe. En effectuant l'ajout et le retrait de sommet en temps $O(n)$, nous améliorons cette complexité en traitant une opération d'arête par une suppression puis une insertion d'un de ses sommets extrémités.

Dans la thèse de Korte [Kor87], on trouve un algorithme purement incrémental de reconnaissance des graphes d'intervalles qui est basé sur le *MPQ*-arbre. Bien qu'ayant été développé complètement indépendamment, l'algorithme présenté dans ce manuscrit reprend l'approche de [Kor87], en y apportant des éléments nouveaux. En effet, dans [KM89], l'algorithme de [Kor87] est cité comme étant assez complexe. L'algorithme que nous présentons ici se veut de compréhension plus simple.

Un des points essentiels qui en facilite la compréhension est le rapprochement fort qui est fait entre l'arbre de décomposition modulaire et la *PQ*-représentation (équivalente au *MPQ*-arbre). On voit ainsi un parallèle très fort se dessiner entre la modification de la *PQ*-représentation lors de l'ajout d'un sommet [Kor87], la modification de l'arbre de décomposition modulaire des graphes de permutation (voir section 4.5) et des graphes en général [MS89]. Grâce aux relations établies entre la *PQ*-représentation et l'arbre de décomposition modulaire, notamment le théorème 5.6, et grâce au travail fait dans la partie sur les graphes de permutation (théorème 4.16), on est en mesure de déduire la forme de la *PQ*-représentation du graphe modifié sans travail supplémentaire.

Nous obtenons aussi une caractérisation purement mathématique des cas où le graphe augmenté reste un graphe d'intervalles (théorème 5.8), ce qui permet de mieux saisir la structure d'une insertion que par un algorithme. Par contre, malgré les efforts apportés, l'énoncé du théorème 5.8 reste technique et la construction du modèle d'intervalles du noeud premier qui se crée lors de l'insertion d'un nouveau sommet également. La question se pose alors naturellement : est-il possible de trouver un niveau d'abstraction supérieur qui permettrait d'obtenir une simplification plus importante de l'algorithme de [Kor87] ? C'est une des perspectives laissées par le travail que nous présentons ici.

Par ailleurs, mentionnons que les lignes qui suivent traitent pour la première fois, à ma connaissance, le problème de la suppression d'un sommet dans un graphe d'intervalles.

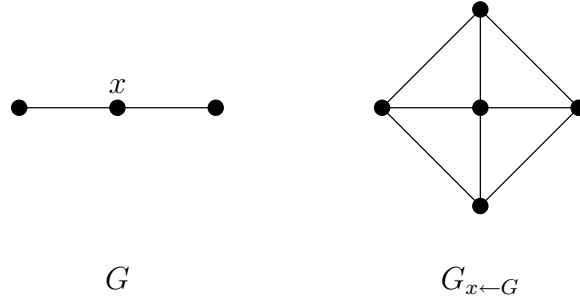


FIG. 5.1 – La composition de deux graphes d'intervalles n'est pas toujours un graphe d'intervalles. G est un graphe d'intervalles mais $G_{x \leftarrow G}$ possède un C_4 .

5.1 Décomposition modulaire d'un graphe d'intervalles

Contrairement à ce qui se passe pour les graphes de permutation, il ne suffit pas que les noeuds premiers de l'arbre de décomposition modulaire soit des graphes d'intervalles pour que le graphe lui-même en soit un. Cette section détermine les autres contraintes que doit satisfaire l'arbre et aboutit à une caractérisation de la décomposition modulaire d'un graphe d'intervalles (théorème 5.3).

5.1.1 Comportement des graphes d'intervalles vis à vis du quotient et de la substitution d'un graphe à un sommet

Comme un graphe quotient est un graphe induit, pour toute famille de graphes héréditaire $\mathcal{F}$, pour tout graphe $G \in \mathcal{F}$ et pour toute partition de congruence $\mathcal{P}$ de G , $G/\mathcal{P} \in \mathcal{F}$. En particulier, c'est vrai pour la famille des graphes d'intervalles qui est héréditaire.

Le fait que la caractérisation de la décomposition modulaire d'un graphe d'intervalles ne soit pas aussi simple que celle d'un graphe de permutation vient du fait que la substitution d'un graphe d'intervalles à un sommet d'un graphe d'intervalles ne résulte pas toujours en un graphe de la classe (voir figure 5.1). Le théorème 5.2 explicite les conditions supplémentaires à satisfaire pour que ce soit le cas.

On notera $T^m(G)$ l'arbre de décomposition modulaire d'un graphe d'intervalles G .

Définition 5.1 *Un sommet simplicial d'un graphe G est un sommet dont le voisinage est une clique.*

En d'autres termes, un sommet simplicial est un sommet présent dans une unique clique maximale de G .

Théorème 5.1 [Hsu93] *Pour tout module M d'un graphe triangulé G , M est une clique ou son voisinage est une clique.*

Théorème 5.2 [Hsu96] *Soit G un graphe d'intervalles, x un sommet de G , et H un graphe quelconque. $G_{x \leftarrow H}$ est un graphe d'intervalles ssi : (i) H est une clique ; ou (ii) x est simplicial et H est un graphe d'intervalles.*

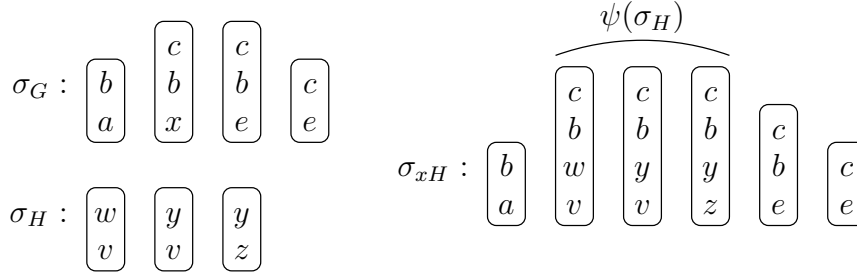


FIG. 5.2 – Composition des modèles d'intervalles lors de la substitution à un sommet simplicial.

Preuve : $\Rightarrow$. Si $G_{x \leftarrow H}$ est un graphe d'intervalles, alors $G_{x \leftarrow H}$ est triangulé (voir [LB62]). Comme, d'après la section 1.2.1 sur la décomposition modulaire, $V(H)$ est un module de $G_{x \leftarrow H}$, alors le théorème 5.1 implique que $V(H)$ est une clique ou son voisinage est une clique. Dans le premier cas, la condition (i) du théorème est vérifiée. Dans le second cas, comme le voisinage de $V(H)$ dans $G_{x \leftarrow H}$ est exactement le voisinage de x dans G , alors $N_G(x)$ est une clique. Par conséquent, x est simplicial. Comme la famille des graphes d'intervalles est héréditaire, alors $G[V(H)] = H$ est un graphe d'intervalles.

$\Leftarrow$. (i) H est une clique. Dans ce cas, les cliques maximales de $G_{x \leftarrow H}$ sont les cliques de G dans lesquelles on remplace x par $V(H)$. Il est clair qu'en effectuant ce remplacement dans un ordre consécutif de $\mathcal{K}(G)$, on obtient un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$.

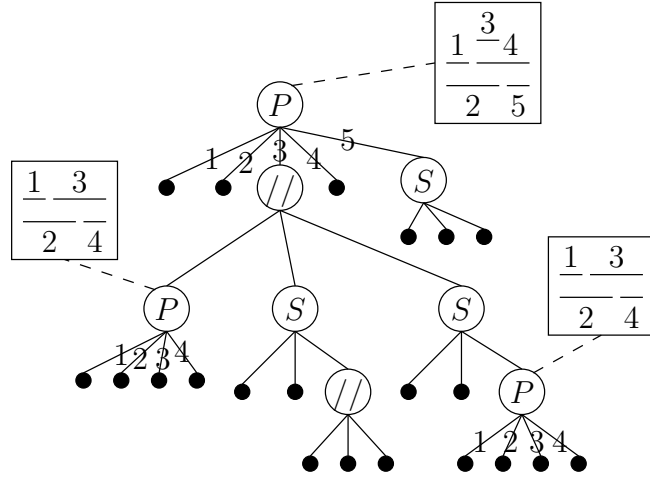
(ii) x est simplicial et H est un graphe d'intervalles. Soit K_x l'unique clique maximale de G contenant x . On note $\mathcal{K}_{xH} = \{K \cup (K_x \setminus \{x\}) \mid K \in \mathcal{K}(H)\}$. L'ensemble des cliques maximales de $G_{x \leftarrow H}$ est :
 $\mathcal{K}(G_{x \leftarrow H}) = (\mathcal{K}(G) \setminus \{K_x\}) \cup \mathcal{K}_{xH}$.

Soit σ_G un ordre consécutif de $\mathcal{K}(G)$ et soit σ_H un ordre consécutif de $\mathcal{K}(H)$. En remplaçant dans σ_G la clique K_x par σ_H dans lequel on a ajouté les sommets de $K_x \setminus \{x\}$, on obtient un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$ (voir figure 5.2). ■

5.1.2 MD-représentation d'un graphe d'intervalles

Dans la MD-représentation d'un graphe d'intervalles G , le graphe quotient G_u d'un noeud premier u est représenté par un modèle d'intervalles minimal, et les sommets de G_u sont en correspondance bijective avec les fils de u dans T^m . Comme toujours dans les arbres de décomposition modulaire, les feuilles de $T^m(G)$ correspondent aux sommets de G . Le but recherché ici est de caractériser la MD-représentation d'un graphe d'intervalles (théorème 5.3). Un exemple est donné sur la figure 5.3.

Lemme 5.1 *Les sommets simpliciaux d'un graphe premier forment un stable.*

FIG. 5.3 – Un exemple de MD -représentation d'un graphe d'intervalles.

Preuve : Soit G un graphe et x, y deux de ses sommets qui sont simpliciaux. Si x et y sont adjacents, ils sont présent dans une même clique maximale de G . Or comme ils sont présents chacun dans une unique clique maximale de G , ils sont vrais jumeaux. Or un graphe premier ne comporte pas de sommets jumeaux. ■

Théorème 5.3 *Un graphe $G = (V, E)$ est un graphe d'intervalles ssi les trois conditions suivantes sont vérifiées :*

1. *tout noeud série de $T^m(G)$ a au plus un fils non feuille ; et*
2. *pour tout noeud premier u de $T^m(G)$, les fils de u correspondant aux sommets non simpliciaux de G_u sont des feuilles ou des noeuds séries dont tous les fils sont des feuilles ; et*
3. *pour tout noeud premier u de $T^m(G)$, G_u est un graphe d'intervalles.*

Preuve : $\Rightarrow$. Comme la classe des graphes d'intervalles est héréditaire et comme un quotient d'un graphe en est un sous graphe, alors nécessairement tous les quotients des noeuds premiers de $T^m(G)$ sont des graphes d'intervalles : la condition 3 est vérifiée.

Si un graphe G_1 est tel que $T^m(G_1)$ possède un noeud série u ayant au moins deux fils u_1 et u_2 qui ne sont pas des feuilles, alors G_1 possède un C_4 induit. En effet, nécessairement, il existe $x_1, x_2 \in U_1$ et $y_1, y_2 \in U_2$ tels que x_1 et x_2 ne sont pas adjacents et y_1 et y_2 ne sont pas adjacents. Comme u est série, x_1 et x_2 sont tous deux adjacents à y_1 et y_2 . Donc $x_1 y_1 x_2 y_2$ est un C_4 . Comme d'après le théorème 1.15 page 67, G ne possède pas de C_4 , alors la condition 1 est vérifiée.

Soit u un noeud premier de $T^m(G)$ et u_1 un fils de u . U_1 est un module de $G[U]$, donc d'après le théorème 5.1, si $G[U_1]$ n'est pas une clique, alors le voisinage de U_1 dans $G[U]$ est une clique. Par conséquent, le sommet de G_u correspondant à u_1 est simplicial dans

G_u : la condition 2 est vérifiée.

$\Leftarrow$.

On commence par montrer que si les trois conditions du théorème sont vérifiées, alors G ne contient pas de C_4 . Supposons que G possède un C_4 formé des sommets $abcd$. Considérons le *ppca* u de ces quatre sommets. u n'est pas parallèle car un C_4 est connexe. u n'est pas série, car sinon, comme les noeuds série de $T^m(G)$ ne possède qu'au plus un fils non feuille, il y aurait un sommet universel dans le C_4 . u est donc premier. Comme il n'y a pas de sommet universel ni isolé dans un C_4 , il existe nécessairement $u_1, u_2 \in \mathcal{C}(u)$ tels que $|U_1 \cap \{a, b, c, d\}| = 2$ et $|U_2 \cap \{a, b, c, d\}| = 2$. On note $U_1 \cap \{a, b, c, d\} = \{x_1, x_2\}$ et $U_2 \cap \{a, b, c, d\} = \{y_1, y_2\}$. Comme le C_4 ne possède pas de couple de sommets vrais jumeaux, alors x_1 et x_2 ne sont pas adjacents et y_1 et y_2 non plus. Par contre x_1 et x_2 sont tous deux adjacents à y_1 et y_2 . Ce qui implique que les sommets z_1, z_2 de G_u correspondant respectivement à u_1 et u_2 sont adjacents dans G_u . D'où, d'après le lemme 5.1, z_1 et z_2 ne sont pas tous deux simpliciaux. Donc u ne vérifie pas la condition 2 : absurde. G ne possède donc pas de C_4 .

Comme les quotients des noeuds premiers de $T^m(G)$ sont de co-comparabilité, G est lui-même de co-comparabilité. D'après le théorème 1.15 page 67, G qui est sans C_4 et de co-comparabilité est un graphe d'intervalles. ■

5.2 PQ -représentation d'un graphe d'intervalles

Le PQ -arbre des cliques maximales d'un graphe d'intervalles est une représentation complète du graphe à condition de stocker la clique maximale correspondant à chacune des feuilles de l'arbre. Si ces cliques sont stockées en extension, c'est à dire par la liste des sommets qu'elles contiennent, la taille de la représentation peut atteindre $\Omega(n + m)$.

Cette représentation peu efficace des cliques maximales freine considérablement les possibilités algorithmiques de cette représentation. C'est pourquoi [KM85] introduit la PQ -représentation (sous le nom de "*MPQ-tree*"). Dans la PQ -représentation, au lieu de stocker un sommet dans toutes les feuilles de l'arbre auxquelles il appartient, on le stocke dans un unique (parfois deux) noeud de l'arbre. Il en résulte une représentation dont la taille est $O(n)$.

Nous utiliserons la PQ -représentation pour réaliser la reconnaissance dynamique des graphes d'intervalles. Notre approche repose sur une investigation approfondie de la structure de cette représentation, présentée dans cette section. Pour dégager ces nouvelles propriétés, nous avons éprouvé le besoin d'un formalisme différent de celui de [KM85]. C'est pourquoi nous présentons la PQ -représentation depuis sa définition.

5.2.1 Définition de la PQ-représentation

Dans cette section on introduit la définition de la PQ-représentation ainsi que les notions et notations de base dont nous aurons besoin pour en parler.

On note $T^c(G)$ le PQ-arbre des cliques maximales d'un graphe d'intervalles G . On notera simplement T^c lorsqu'il n'y a pas d'ambiguïté sur le graphe d'intervalles auquel on se réfère. On adopte les notations suivantes.

Notation 5.1 *Tout noeud premier $q \in T^c(G)$ est doté de deux ordres linéaires, inverses l'un de l'autre, sur l'ensemble de ses fils, notés σ_q et $\bar{\sigma}_q$. Le premier élément de σ_q est noté $C_{\min}(q)$ et le dernier $C_{\max}(q)$.*

Notation 5.2 $\mathcal{K}(G)$ désigne l'ensemble des cliques maximales de G . Pour un noeud u de $T^c(G)$, on note $\mathcal{K}_{T^c(G)}(u)$ l'ensemble des cliques maximales de G qui sont des feuilles de $T^c_u(G)$. Enfin, pour un sous-ensemble $S \subseteq V(G)$ de sommets, on note $\mathcal{K}_G(S)$ l'ensemble des cliques maximales de G dont l'intersection avec S est non creux. Lorsque S est réduit à un singleton $\{x\}$, on notera $\mathcal{K}_G(x) = \mathcal{K}_G(\{x\})$.

Remarque 5.1 Dans les notations qui précèdent, on omettra souvent de préciser le graphe G auquel on se réfère, lorsqu'il n'y a pas d'ambiguïté. On utilisera les notations $\mathcal{K}_{T^c}(u)$, $\mathcal{K}(S)$ et $\mathcal{K}(x)$.

Notation 5.3 Soit $x \in V(G)$, on note e_x le plus petit ancêtre commun des feuilles de $T^c(G)$ correspondant aux cliques maximales de G contenant x .

Le lemme suivant est à la base de la définition de la PQ-représentation : il donne le ou les noeuds de l'arbre que l'on va associer à chaque sommet.

Lemme 5.2 (voir figure 5.4) *Pour tout sommet x d'un graphe d'intervalles G , une et une seule des deux conditions suivantes est vérifiée :*

1. $\mathcal{K}(x) = \mathcal{K}_{T^c}(e_x)$, ou
2. e_x est un noeud premier et $\exists u_1, u_2 \in \mathcal{C}(e_x)$ tels que $u_1 \neq u_2$ et $\{u_1, u_2\} \neq \{C_{\min}(e_x), C_{\max}(e_x)\}$ et $\mathcal{K}(x) = \bigcup_{u_1 \leq_{\sigma_{e_x}} v \leq_{\sigma_{e_x}} u_2} \mathcal{K}_{T^c}(v)$.

Preuve :

- Si e_x est dégénéré, on montre par l'absurde que $\forall K \in \mathcal{K}_{T^c}(e_x), x \in K$. Supposons $\exists K_1 \in \mathcal{K}_{T^c}(e_x), x \notin K_1$. Soit l_1 la feuille de T^c correspondant à K_1 . Soit $u_1 \in \mathcal{C}(e_x) \cap \text{Anc}(l_1)$.
- Si $\forall K \in \mathcal{K}_{T^c}(u_1), x \notin K$, alors comme e_x est le plus petit ancêtre commun des feuilles de $\mathcal{K}(x)$, il existe au moins deux fils distincts u_2 et u_3 de e_x , différents de u_1 , tels que $\mathcal{K}_{T^c}(u_2)$ et $\mathcal{K}_{T^c}(u_3)$ contiennent des cliques contenant x . Une solidification (voir définition 1.38, page 76) s de T^c telle que $\Pi_{e_x}^s$ (voir notation 1.2, page 77) place u_1 entre u_2 et u_3 donne un ordre consécutif dans lequel les cliques de $\mathcal{K}(x)$ ne forment pas un intervalle : absurde.

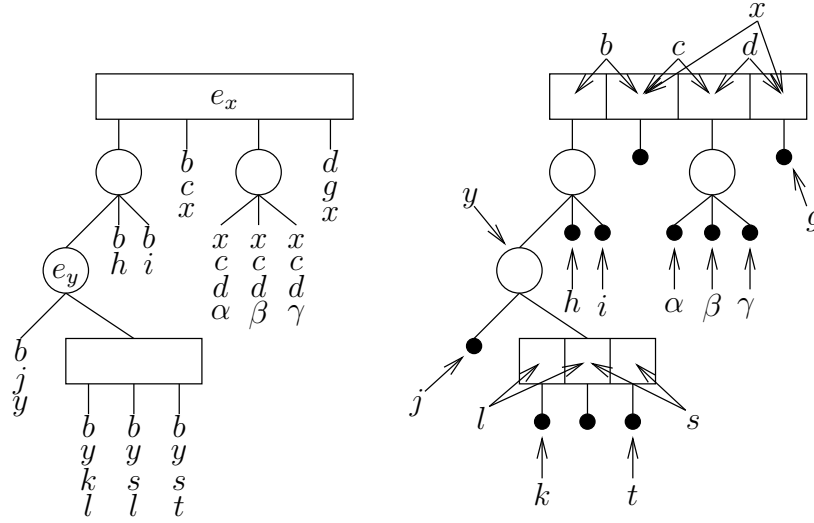


FIG. 5.4 – À gauche, le PQ -arbre des cliques maximales d'un graphe d'intervalles ; à droite, sa PQ -représentation. x vérifie la condition 2 du lemme 5.2 et y la condition 1.

- Si $\exists K'_1 \in \mathcal{K}_{T^c}(u_1), x \in K'_1$, alors, toujours par définition de e_x , $\exists u_2 \in \mathcal{C}(e_x), \exists K'_2 \in \mathcal{K}_{T^c}(u_2), u_2 \neq u_1$ et $x \in K'_2$. Soit s une solidification de T^c et $\sigma = \text{front}(s)$. Soit s' la solidification de T^c obtenue en gardant pour tous les noeuds p de T^c différents de u_1 le même ordre Π_p^s que celui utilisé pour s , et en prenant pour ordre des fils de u_1 l'inverse de celui utilisé pour s , $\Pi_{u_1}^{s'} = \overline{\Pi_{u_1}^s}$. Posons $\sigma' = \text{front}(s')$. σ et σ' sont deux ordres consécutifs. Pourtant, dans un des deux, K_1 est entre K'_1 et K'_2 . Or $x \in K'_1 \cap K'_2$ et $x \notin K_1$: absurde.
- Si e_x est premier, soient $u_1 = \min_{\sigma_{e_x}} \{u \in \mathcal{C}(e_x) \mid \mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \neq \emptyset\}$ et $u_2 = \max_{\sigma_{e_x}} \{u \in \mathcal{C}(e_x) \mid \mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \neq \emptyset\}$. Par définition de e_x , $u_1 \neq u_2$. Ainsi, supposons que $\exists v \in \mathcal{C}(e_x), \exists K \in \mathcal{K}_{T^c}(v), u_1 \leq_{\sigma_{e_x}} v \leq_{\sigma_{e_x}} u_2$ et $x \notin K$, alors en considérant une solidification de T^c et en y renversant l'ordre des fils de v , on conclut à une absurdité comme précédemment. Donc, $\forall w \in \mathcal{C}(e_x), (u_1 \leq_{\sigma_{e_x}} w \leq_{\sigma_{e_x}} u_2) \Rightarrow \forall K \in \mathcal{K}_{T^c}(w), x \in K$. Si u_1 est le minimum de σ_{e_x} et u_2 le maximum alors la condition 1 du lemme est vérifiée, sinon la condition 2 est vérifiée. ■

Notation 5.4 Les deux conditions du Lemme 5.2 sont mutuellement exclusives. Quand la condition 2 du Lemme 5.2 est satisfaite, on note $e_x^1 = \min_{\sigma_{e_x}} \{u \in \mathcal{C}(e_x) \mid \mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \neq \emptyset\}$ et $e_x^2 = \max_{\sigma_{e_x}} \{u \in \mathcal{C}(e_x) \mid \mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \neq \emptyset\}$.

La **PQ -représentation** d'un graphe d'intervalles G , notée $PQ(G)$, est composée de $T^c(G)$ et de l'ensemble des sommets de G . Chaque sommet x de G est doté d'un **pointeur principal** vers e_x ; et lorsque x ne satisfait pas la condition 1 du Lemme 5.2, x se voit

attribuer deux **pointeurs secondaires** vers respectivement e_x^1 et e_x^2 (voir figure 5.4). Réciproquement, chaque noeud u de $T^c(G)$ stocke l'ensemble des sommets x de G tels que $u = e_x$, l'ensemble des sommets x tels que $u = e_x^1$ et celui des sommets x tels que $u = e_x^2$.

Remarque 5.2 Sur les figures, les pointeurs principaux des sommets ayant des pointeurs secondaires ne seront pas représentés.

Notation 5.5 On note $PQ_u(G)$ la partie de $PQ(G)$ enracinée en u , c'est à dire $T_u^c(G)$ avec les pointeurs des sommets de u^* vers $T_u^c(G)$.

Remarque 5.3 Comme le montrera le lemme 5.12, $PQ_u(G)$ est en fait $PQ(G[u^*])$.

Nous aurons besoin de distinguer plusieurs ensembles de sommets pour décrire aisément les propriétés de la PQ-représentation (voir figure 5.5).

Notation 5.6 Soit u un noeud de $T^c(G)$ différent de la racine, et $v = \text{parent}(u)$, on note :

$X_u = \{y \in V(G) \mid e_y = u \text{ et } y \text{ ne possède pas de pointeurs secondaires}\}$

$Y_u = \{y \in V(G) \mid e_y = u \text{ et } y \text{ possède des pointeurs secondaires sur les fils de } u\}$

$\Delta_u = \{y \in Y_v \mid e_y^1 \leq_{\sigma_v} u \leq_{\sigma_v} e_y^2\}$

$\Gamma_u = X_u \cup \Delta_u$

$u^* = \{y \in V(G) \mid e_y \text{ est un noeud de } T_u^c\}$

Pour la racine r de $T^c(G)$, on adopte les mêmes notations avec, par convention, $\Delta_r = \emptyset$.

Remarque 5.4 Par définition, si u est dégénéré alors $Y_u = \emptyset$, et par conséquence, si $\text{parent}(u)$ est dégénéré alors $\Delta_u = \emptyset$.

La branche d'un noeud u est l'ensemble des sommets présents dans toutes les cliques maximales qui sont dans le sous-arbre enraciné en u .

Définition 5.2 On appelle **branche** d'un noeud $u \in T^c$ le sous-ensemble de sommets de G noté B_u et défini par $B_u = \bigcup_{v \in \text{Anc}(u)} \Gamma_v$.

Remarque 5.5 B_u est l'ensemble des sommets de G présents dans toutes les cliques de $\mathcal{K}_{T^c}(u)$. Seule la branche de la racine r de T^c peut être creuse. D'après la définition de la PQ-représentation de G , $\mathcal{K}(G) = \{B_l \mid l \text{ est une feuille de } T^c(G)\}$.

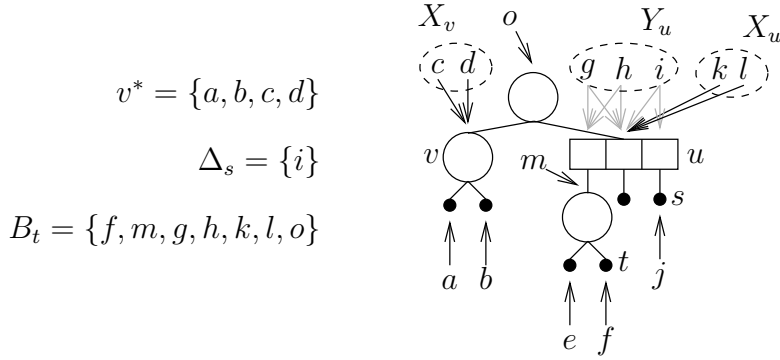
Dans le but d'alléger les preuves, on adopte le vocabulaire et les notations suivants.

Définition 5.3 Pour un sommet $y \in G$ et un noeud u de T^c , si $y \in \Delta_u$, on dit que y **couvre** u . Pour un noeud premier $q \in T^c$ et un sommet $z \in Y_q$, on note $C_\Delta(z)$ l'intervalle de σ_q formé par les fils de q couverts par z . $\overline{C_\Delta(z)}$ désigne le complémentaire de $C_\Delta(z)$ dans $\mathcal{C}(q)$.

Définition 5.4 Un noeud u de T^c tel que $u^* = \emptyset$ est dit **désert**.

Remarque 5.6 Si un noeud u de T^c est non désert, alors $\mathcal{K}(u^*) = \mathcal{K}_{T^c}(u)$.

Les notions ci-dessus sont illustrées sur la figure 5.5.

FIG. 5.5 – Ensembles associés à la PQ -représentation.

5.2.2 Structure des pointeurs de la PQ -représentation

Dans cette section, on présente quelques propriétés et contraintes satisfaites par les pointeurs sur la PQ -représentation. Par exemple, certains noeuds dégénérés doivent nécessairement être pointés par des pointeurs principaux ; seules certaines feuilles de l'arbre peuvent être désertes. On démontre aussi des propriétés des pointeurs secondaires pointant les fils d'un noeud premier. Ces propriétés nous seront très utiles dans la suite car c'est là qu'est concentrée toute la complexité de la structure.

Le lemme qui suit énonce des propriétés élémentaires des pointeurs secondaires vers les fils des noeuds premiers.

Lemme 5.3 *Soit u un noeud premier de T^c et soit $v \in \mathcal{C}(u)$. On a les propriétés suivantes :*

1. $\Delta_v \neq \emptyset$;
2. si $v \neq C_{Max}(u)$ et si $\{y \in Y_u \mid v = e_y^2\} = \emptyset$, alors $v^* \neq \emptyset$; de même, si $v \neq C_{min}(u)$ et si $\{y \in Y_u \mid v = e_y^1\} = \emptyset$, alors $v^* \neq \emptyset$;
3. $C_{min}(u)$ et $C_{Max}(u)$ sont non déserts.

Preuve : 1 Quitte à renverser σ_u , on peut supposer que v n'est pas le premier élément de σ_u . Soit une solidification s de T^c telle que $\Pi_u^s = \sigma_u$. Supposons que $\Delta_v = \emptyset$. En déplaçant l'intervalle $\mathcal{K}_{T^c}(v)$ dans σ pour le placer au début de l'intervalle de $\mathcal{K}_{T^c}(u)$, sans changer l'ordre relatif des autres cliques maximales de G ni l'ordre relatif des cliques de $\mathcal{K}_{T^c}(v)$, on obtient un ordre σ' sur $\mathcal{K}(G)$ qui respecte toutes les contraintes de consécutivité. Or $\sigma' \notin \text{Consistant}(T^c(G))$: absurde.

2 Si $v \neq C_{Max}(u)$ et si $\{y \in Y_u \mid v = e_y^2\} = \emptyset$, on note v_s le successeur de v dans σ_u . On a $\Delta_v \subseteq \Delta_{v_s}$. Soient $K_v \in \mathcal{K}_{T^c}(v)$ et $K_s \in \mathcal{K}_{T^c}(v_s)$. On a l'inclusion $K_v \subseteq v^* \cup \Delta_v \cup B_u$. Comme $\Delta_v \subseteq \Delta_{v_s}$ et $\Delta_{v_s} \cup B_u \subseteq K_s$, alors $\Delta_v \cup B_u \subseteq K_s$. Or $K_v \setminus K_s \neq \emptyset$, d'où $v^* \neq \emptyset$. Le cas où $v \neq C_{min}(u)$ et $\{y \in Y_u \mid v = e_y^1\} = \emptyset$ se démontre de manière similaire.

3 $C_{min}(u) \neq C_{Max}(u)$ et $\{y \in Y_u \mid C_{min}(u) = e_y^2\} = \emptyset$; $C_{Max}(u) \neq C_{min}(u)$ et $\{y \in Y_u \mid C_{Max}(u) = e_y^1\} = \emptyset$. La propriété 2 du présent lemme conclut. ■

Le lemme suivant montre qu'un noeud dégénéré dont le père est dégénéré est nécessairement pointée par un sommet. Cela implique que les pointeurs sont répartis dans toute la PQ-représentation : il ne peut pas y avoir de longue portion d'une branche qui ne soit pas pointée.

Lemme 5.4 *Soit u un noeud dégénéré de T^c ou une feuille, qui ne soit pas la racine, et dont le père est un noeud dégénéré, alors $X_u \neq \emptyset$.*

Preuve : Si u est une feuille, supposons que $X_u = \emptyset$. On note K_u la clique maximale de G correspondant à u . Soit $x \in K_u$, comme $X_u = \emptyset$ alors $e_x \in \text{Anc}(u) \setminus \{u\}$ et par conséquent $\forall K \in \mathcal{K}_{T^c}(\text{parent}(u)), x \in K$. Comme cela est vrai pour tout $x \in K_u$, K_u n'est pas maximale : absurde. Pour traiter le cas où u est un noeud dégénéré, on commence par établir les deux propositions suivantes.

Proposition 5.1 *Soit u un noeud dégénéré de T^c et x un sommet de G , alors $\mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) = \emptyset$ ou $\mathcal{K}_{T^c}(u) \subseteq \mathcal{K}(x)$ ou $\exists! v \in \mathcal{C}(u), \mathcal{K}_{T^c}(v) \cap \mathcal{K}(x) \neq \emptyset$.*

Preuve : Si $e_x \in T^c \setminus (\text{Anc}(u) \cup \text{Desc}(u))$, alors $\mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) = \emptyset$. Si $e_x \in \text{Anc}(u)$, alors $\mathcal{K}_{T^c}(u) \subseteq \mathcal{K}(x)$. Si $e_x \in \text{Desc}(u) \setminus \{u\}$, alors $\exists! v \in \mathcal{C}(u), \mathcal{K}_{T^c}(v) \cap \mathcal{K}(x) \neq \emptyset$. $\square$

Proposition 5.2 *Si u est un noeud dégénéré de T^c tel que $v = \text{parent}(u)$ est dégénéré et $X_u = \emptyset$, alors $\forall x \in V(G), (\exists u_1, u_2 \in \mathcal{C}(u), u_1 \neq u_2 \text{ et } \mathcal{K}_{T^c}(u_1) \cap \mathcal{K}(x) \neq \emptyset \text{ et } \mathcal{K}_{T^c}(u_2) \cap \mathcal{K}(x) \neq \emptyset) \Rightarrow \mathcal{K}_{T^c}(v) \subseteq \mathcal{K}(x)$.*

Preuve : Soit un tel x . Comme $u_1 \neq u_2$, et par définition de e_x , on a $e_x \in \text{Anc}(u)$. Or $X_u = \emptyset$, d'où $e_x \in \text{Anc}(v)$, et comme v est dégénéré, $\mathcal{K}_{T^c}(v) \subseteq \mathcal{K}(x)$. $\square$

Supposons que u , qui est un noeud dégénéré, est tel que $v = \text{parent}(u)$ est dégénéré et $X_u = \emptyset$. Soit $v_1 \in \mathcal{C}(v)$ tel que $v_1 \neq u$. Soit s une solidification de T^c . Soient u_1 et u_2 deux fils de u distincts qui sont consécutifs dans Π_u^s . D'après la proposition 5.2, les sommets présents à la fois dans des cliques de $\mathcal{K}_{T^c}(u_1)$ et dans des cliques de $\mathcal{K}_{T^c}(u_2)$ sont présents dans toutes les cliques de $\mathcal{K}_{T^c}(v)$. De plus, d'après la proposition 5.1, les sommets présents dans les cliques de $\mathcal{K}_{T^c}(v_1)$ sont soit présents uniquement dans des cliques de $\mathcal{K}_{T^c}(v_1)$ soit présents dans toutes les cliques de $\mathcal{K}_{T^c}(v)$. Ainsi, en déplaçant dans σ l'intervalle formé par les cliques de $\mathcal{K}_{T^c}(v_1)$ entre les deux intervalles formés respectivement par les cliques de $\mathcal{K}_{T^c}(u_1)$ et celles de $\mathcal{K}_{T^c}(u_2)$ on obtient un ordre consécutif σ' de $\mathcal{K}(G)$. Or σ' ne correspond à aucune solidification de T^c : absurde. ■

Les quelques propriétés déjà énoncées impliquent que les noeuds internes de l'arbre ne sont pas déserts, seules les feuilles qui sont fils d'un noeud premier peuvent l'être.

Lemme 5.5 *Les noeuds déserts sont nécessairement des feuilles dont le père est un noeud premier.*

Preuve : Soit $u \in T^c$. Si $\text{Desc}(u) \neq \emptyset$, alors T_u^c contient un noeud premier v ou deux noeuds dégénérés u_1 et u_2 tels que u_1 est le père de u_2 . Or d'après le lemme 5.3, $Y_v \neq \emptyset$ car v est premier. Et d'après le lemme 5.4, $X_{u_2} \neq \emptyset$. Ainsi, dans les deux cas, $u^* \neq \emptyset$. Donc si u est désert, u est une feuille. De plus, d'après le lemme 5.4, $\text{parent}(u)$ est nécessairement premier. ■

La propriété qu'on démontre maintenant est une "petite" propriété, mais elle est fondamentale dans la structure des noeuds premiers de la PQ -représentation. On la retrouvera souvent dans les preuves.

Lemme 5.6 *Soit u un noeud premier d'un PQ -arbre et soit I un intervalle de σ_u tel que $1 < |I| < |\mathcal{C}(u)|$. Il existe $y \in Y_u$ tel que $C_\Delta(y)$ chevauche I .*

Preuve : Si ce n'était pas le cas, on pourrait renverser l'ordre des éléments de I dans $\mathcal{C}(u)$ sans violer les contraintes de consécutivité. Ce qui est impossible car $|I| > 1$ et $I \neq \mathcal{C}(u)$. ■

Le lemme qui suit est un lemme technique. Il explique la chose suivante. Si, pour un noeud q ayant un fils q_1 , on considère l'intersection des intervalles que couvrent les sommets de Y_q qui couvrent q_1 , alors on obtient un intervalle qui contient q_1 . On peut récursivement grignoter cet intervalle en retirant l'intervalle couvert par un sommet de Y_q ne couvrant pas q_1 . Ainsi, à chaque étape, on obtient un intervalle qui contient toujours q_1 . Ce que dit le lemme 5.7, c'est qu'on peut itérer ce procédé jusqu'à ne laisser que q_1 dans l'intervalle considéré.

De ce lemme découle directement le corollaire 5.1 qui montre que l'ensemble des sommets de Y_q qui couvrent un fils de q caractérise ce fils de manière uivoque. Mais ce résultat peut se montrer beaucoup plus simplement. La raison pour laquelle on passe par le lemme 5.7 est qu'il sera utile d'autre manière encore.

Lemme 5.7 *Soit $G = (V, E)$ un graphe et T^c son PQ -arbre. Soit $q \in T^c$ un noeud premier et $q_1 \in \mathcal{C}(q)$. $\exists k \in \mathbb{N}, \exists y_1, \dots, y_k \in Y_q \setminus \Delta_{q_1}$,*

$$\left\{ \begin{array}{l} \forall j \in \llbracket 1, k \rrbracket, C_\Delta(y_j) \supset \left(\bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{C_\Delta(y_i)} \right) \\ \text{et} \\ \bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \bigcap_{1 \leq i \leq k} \overline{C_\Delta(y_i)} = q_1 \end{array} \right.$$

avec par convention, $\llbracket 1, 0 \rrbracket = \emptyset$ et $\bigcap_{1 \leq i \leq 0} \overline{C_\Delta(y_i)} = \mathcal{C}(q)$.

Preuve : $\bigcap_{z \in \Delta_{q_1}} C_\Delta(z)$ est un intervalle de σ_q qui n'est pas $\mathcal{C}(q)$ lui-même, car aucun sommet de Y_{q_1} ne couvre tous les noeuds de $\mathcal{C}(q_1)$, et $q_1 \in \bigcap_{z \in \Delta_{q_1}} C_\Delta(z)$. Donc, si $|\bigcap_{z \in \Delta_{q_1}} C_\Delta(z)| > 1$, alors, d'après le lemme 5.6, il existe $y \in Y_q \setminus \Delta_{q_1}$ tel que $C_\Delta(y)$ chevauche $\bigcap_{z \in \Delta_{q_1}} C_\Delta(z)$. D'où, $\bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \overline{C_\Delta(y)}$ est un intervalle de σ_q et $q_1 \in$

$\bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \overline{C_\Delta(y)}$ et $|\bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \overline{C_\Delta(y)}| < |\bigcap_{z \in \Delta_{q_1}} C_\Delta(z)|$. Si $|\bigcap_{z \in \Delta_{q_1}} C_\Delta(z) \cap \overline{C_\Delta(y)}| > 1$, on reconduit le procédé et, par récurrence, on montre le résultat escompté. ■

Le corollaire 5.1 en découle directement.

Corollaire 5.1 *Soit $G = (V, E)$ un graphe et T^c son PQ-arbre. Soit $q \in T^c$ un noeud premier et $q_1, q_2 \in \mathcal{C}(q)$. Alors $\Delta_{q_1} \neq \Delta_{q_2}$.*

5.2.3 Intervalles des arrangements consécutifs et connexité

La notion de connexité joue un rôle essentiel dans la théorie de la décomposition modulaire (voir théorème 1.3). Les résultats établis dans cette section serviront à faire le lien entre la PQ-représentation et la MD-représentation. La plupart ont déjà été remarqués dans [KM85], sans forcément en fournir de démonstration. J'ai jugé bon d'en fournir dans ce manuscrit.

Le premier lemme de cette section est une généralisation du lemme 5.2. Son intérêt est de montrer comment se localisent, dans la PQ-représentation, les intervalles communs à tous les ordres consécutifs.

Lemme 5.8 *Soit G un graphe d'intervalles et T^c son PQ-arbre. Soit $I \subseteq \mathcal{K}(G)$ tel que pour tout ordre consécutif σ de $\mathcal{K}(G)$, I est un intervalle de σ . Une et une seule des deux propositions suivantes est vraie :*

1. $\exists u \in T^c, I = \mathcal{K}_{T^c}(u)$, ou
2. $\exists u \in T^c, \exists u_1, u_2 \in \mathcal{C}(u), u$ est premier et $u_1 <_{\sigma_u} u_2$ et $\{u_1, u_2\} \neq \{C_{\min}(u), C_{\max}(u)\}$ et $I = \bigcup_{u_1 \leq_{\sigma_u} v \leq_{\sigma_u} u_2} \mathcal{K}_{T^c}(v)$.

Preuve : Soit u le ppca dans T^c des cliques de I .

1. Si u est dégénéré, alors supposons qu'il existe $u_1 \in \mathcal{C}(u)$ tel que $\mathcal{K}_{T^c}(u_1) \setminus I \neq \emptyset$.

Si $\mathcal{K}_{T^c}(u_1) \cap I \neq \emptyset$, alors en considérant une solidification s de T^c et en y renversant l'ordre des fils de u_1 , on obtient une autre solidification s' . Une de s ou s' donne lieu à un ordre consécutif de $\mathcal{K}(G)$ dans lequel les cliques de I ne forment pas un intervalle : absurde.

Si $\mathcal{K}(u_1) \cap I = \emptyset$, alors, par définition du ppca, $\exists u_2, u_3 \in \mathcal{C}(u), u_2 \neq u_3$ et $\mathcal{K}(u_2) \cap I \neq \emptyset$ et $\mathcal{K}(u_3) \cap I \neq \emptyset$. Une solidification de T^c dans laquelle u_1 est entre u_2 et u_3 donne lieu à un ordre consécutif dans lequel les cliques de I ne forment pas un intervalle : absurde.

En conclusion, $\forall u_1 \in \mathcal{C}(u), \mathcal{K}_{T^c}(u_1) \subseteq I$. Comme $I \subseteq \mathcal{K}_{T^c}(v)$, on conclut que $I = \mathcal{K}_{T^c}(v)$.

2. Si u est premier, nécessairement, $S_I = \{v \in \mathcal{C}(u) \mid \mathcal{K}_{T^c}(v) \cap I \neq \emptyset\}$ est un intervalle de $\mathcal{C}(u)$. Supposons que $\exists u_1 \in S_I, \mathcal{K}(u_1) \setminus I \neq \emptyset$. Comme précédemment, en considérant une solidification de T^c et en y renversant l'ordre des fils de u_1 , on conclut à une absurdité. Le résultat suit.

■

Le lemme suivant fait partie du folklore, il a été remarqué dans plusieurs contextes, notamment dans la théorie de la "pathwidth".

Lemme 5.9 *Soit G un graphe d'intervalles et $S \subseteq V(G)$ tel que $G[S]$ est connexe, alors $\mathcal{K}(S)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G)$.*

Les énoncés qui suivent montrent des propriétés de connexité des graphes induits par les ensembles associés aux noeuds de la PQ -représentation.

Lemme 5.10 *[KM85] Soit v un noeud premier de T^c , alors $G[v^* \setminus X_v]$ est connexe.*

Preuve : D'après le lemme 5.3, $\Delta_{C_{min}(v)} \neq \emptyset$. Soit $x_1 \in \Delta_{C_{min}(v)}$, $C_\Delta(x_1)$ est un intervalle de σ_v qui n'est pas $\mathcal{C}(v)$ lui-même. D'où, d'après le lemme 5.6, $\exists x_2 \in Y_v, C_\Delta(x_2) \otimes C_\Delta(x_1)$. Par conséquent, $|C_\Delta(x_2) \cup C_\Delta(x_1)| > |C_\Delta(x_1)|$. Si $C_\Delta(x_2) \cup C_\Delta(x_1) \neq \mathcal{C}(v)$, on recommence. On montre ainsi par récurrence que $\exists k \in \mathbb{N}^*, \exists x_1, \dots, x_k \in Y_v, \bigcup_{1 \leq i \leq k} C_\Delta(x_i) = \mathcal{C}(v)$ et $\forall j \in \llbracket 2, k \rrbracket, C_\Delta(x_j) \otimes \bigcup_{1 \leq i \leq j-1} C_\Delta(x_i)$ (avec par convention $\llbracket 2, 1 \rrbracket = \emptyset$).

Ainsi, $\forall K \in \mathcal{K}_{T^c}(v), \exists i \in \llbracket 1, k \rrbracket, x_i \in K$. De plus, comme $G[\{x_1, \dots, x_k\}]$ est connexe, alors $G[v^* \setminus X_v]$ est connexe. ■

Lemme 5.11 *[KM85] Soit G un graphe d'intervalles muni de sa PQ -représentation. Soit u un noeud de $T^c(G)$, il vérifie toutes les propriétés suivantes :*

1. Si $X_u \neq \emptyset$, alors $G[u^*]$ est non co-connexe.
2. Si u est dégénéré, alors $G[u^* \setminus X_u]$ est non connexe.
3. Si u est premier, alors $G[u^* \setminus X_u]$ est connexe et co-connexe.

Preuve :

1. Les sommets de X_u sont universels dans $G[u^*]$, donc $G[u^*]$ est non co-connexe.
2. Soient u_1, u_2 deux fils de u . D'après le lemme 5.4, $u_1^* \neq \emptyset$ et $u_2^* \neq \emptyset$. Or aucun sommet de u_1^* n'est présent dans une même clique maximale qu'un sommet de u_2^* . Donc tous les sommets de u_1^* sont non adjacents à tous les sommets de u_2^* . Par conséquent, $G[u^* \setminus X_u]$ est non connexe.
3. D'après le lemme 5.10, $G[u^* \setminus X_u]$ est connexe.

Pour montrer que $G[u^* \setminus X_u]$ est co-connexe, on commence par remarquer que comme dans le cas où u est dégénéré, si u_1 et u_2 sont deux fils de u tels que $u_1^* \neq \emptyset$ et $u_2^* \neq \emptyset$, alors tous les sommets de u_1^* sont non adjacents à tous les sommets de u_2^* . Ainsi, $G[u^* \setminus (X_u \cup Y_u)]$ est co-connexe. Par définition, $\forall y \in Y_u, \{C_{min}(\sigma_u), C_{Max}(\sigma_u)\} \not\subseteq C_\Delta(y)$. Or d'après le lemme 5.3, $C_{min}(\sigma_u)^* \neq \emptyset$ et $C_{Max}(\sigma_u)^* \neq \emptyset$. Donc y est non adjacent à au moins un sommet de $u^* \setminus (X_u \cup Y_u)$. Comme cela est vrai pour tout $y \in Y_u$, $G[u^* \setminus X_u]$ est co-connexe.

■

Corollaire 5.2 *Soit u un noeud dégénéré d'un PQ-arbre et $v \in \mathcal{C}(u)$, alors $G[v^*]$ est connexe.*

Preuve : Si v est dégénéré, alors, d'après le lemme 5.4, $X_v \neq \emptyset$. Le lemme 5.11 conclut. ■

5.2.4 Restriction, quotient et composition de PQ-représentations

Comme nous le verrons par la suite, la PQ-représentation est fortement liée à la décomposition modulaire. Dans cette section, on montre comment effectuer les opérations de base de la décomposition modulaire sur la PQ-représentation : restriction, quotient et composition.

Le premier lemme concerne l'opération de restriction à l'ensemble u^* d'un noeud u de la PQ-représentation.

Lemme 5.12 *Soit u un noeud de T^c , $T_u^c = T^c(G[u^*])$.*

Preuve :

Pour démontrer le lemme, on a besoin d'établir en premier lieu les deux propriétés suivantes.

Proposition 5.3 *Soit $x \in V(G) \setminus u^*$, si $\mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \neq \emptyset$, alors $\mathcal{K}_{T^c}(u) \subseteq \mathcal{K}(x)$.*

Preuve : Il suffit de remarquer que nécessairement e_x est un ancêtre strict de u . □

Proposition 5.4 *Soit $K \in \mathcal{K}_{T^c}(u)$, alors $K' = K \cap u^* \in \mathcal{K}(G[u^*])$.*

Preuve : $K' \subseteq u^*$ et K' est une clique. Il suffit de montrer que K' est maximale dans $G[u^*]$. Supposons qu'il existe K'' une clique de $G[u^*]$ telle que $K' \subsetneq K''$. Comme K'' est aussi une clique de G , $\exists K_1 \in \mathcal{K}(G)$, $K'' \subseteq K_1$ et comme $K'' \subseteq u^*$, $K'' \subseteq K_1 \cap u^*$. D'où $K \cap u^* \subsetneq K_1 \cap u^*$. Comme $K_1 \cap u^* \neq \emptyset$, $K_1 \in \mathcal{K}_{T^c}(u)$. Or, d'après la proposition 5.3, les cliques de $\mathcal{K}_{T^c}(u)$ contiennent toutes les mêmes sommets de $V(G) \setminus u^*$, d'où $K \subsetneq K_1$: absurde. □

Ainsi, on peut définir l'application :

$$\begin{aligned} \phi : \mathcal{K}_{T^c}(u) &\rightarrow \mathcal{K}(G[u^*]) \\ K &\mapsto K \cap u^* \end{aligned}$$

On montre que ϕ est une bijection. Soient $K_1, K_2 \in \mathcal{K}_{T^c}(u)$. Si $\phi(K_1) = \phi(K_2)$, alors $K_1 \cap u^* = K_2 \cap u^* \neq \emptyset$. Comme d'après la proposition 5.3 les sommets de $K_1 \setminus u^*$ et ceux de $K_2 \setminus u^*$ sont les mêmes, alors $K_1 = K_2$. Donc, ϕ est injective. Pour montrer que ϕ est

surjective, considérons une clique $K' \in \mathcal{K}(G[u^*])$. Comme K' est une clique de G , alors $\exists K \in \mathcal{K}(G)$, $K' \subseteq K$. $K' \subseteq \phi(K)$ et $\phi(K)$ est une clique de $G[u^*]$, d'où par maximalité de K' , $\phi(K) = K'$. Donc ϕ est surjective.

On pose $S = \{\mathcal{K}_{T^c}(u) \cap \mathcal{K}(x) \mid x \in V(G)\}$ et $S' = \{\mathcal{K}_{G[u^*]}(x) \mid x \in u^*\}$. Considérons les problèmes d'arrangement consécutif $A = \mathcal{A}_1(\mathcal{K}_{T^c}(u), S)$ et $A' = \mathcal{A}_1(\mathcal{K}(G[u^*]), S')$ (voir section 1.4.1 pour les notations). Comme les sommets de $V(G) \setminus u^*$ sont présents dans aucune ou toutes les cliques de $\mathcal{K}_{T^c}(u)$, il est clair que $\phi(\sigma) \in \Pi(A')$ ssi $\sigma \in \Pi(A)$. Donc les PQ -arbres de A et A' sont les mêmes. Or le PQ -arbre de A' est celui du graphe d'intervalles $G[u^*]$. Il reste à montrer que le PQ -arbre de A est T_u^c .

Il est clair que tous les ordres linéaires de $\mathcal{K}_{T^c}(u)$ consistants (voir définition 1.38, page 76) avec T_u^c sont cohérents avec A (voir page 76). Réciproquement, soit un ordre σ cohérent avec A , et soit π consistant avec $T^c(G)$. En réordonnant dans π l'intervalle $\mathcal{K}_{T^c}(u)$ par σ , on obtient encore un ordre consécutif de G . La solidification de T^c permettant de l'obtenir restreinte aux noeuds de T_u^c montre que σ est consistant avec T_u^c . Donc, le PQ -arbre de A est bien T_u^c . ■

Le deuxième lemme concerne le quotient par l'ensemble u^* d'un noeud u non désert.

Lemme 5.13 *Soit u un noeud non désert de $T^c(G)$. $PQ(G/\{u^*\})$ s'obtient en remplaçant le noeud u de $T^c(G)$ par une feuille sur laquelle pointe le représentant a de u^* (figure 5.6).*

Preuve : Commençons par remarquer que d'après le théorème 5.6 page 209, u^* est un module de G , ce qui justifie la validité du quotient par u^* .

On note T^{quo} l'arbre obtenu en remplaçant le noeud u de $T^c(G)$ par une feuille sur laquelle pointe le représentant a de u^* .

Soit $s \in \text{sol}(T^{quo})$ et $\sigma = \text{front}(s)$. Soit $s_G \in \text{sol}(T^c(G))$ telle que pour tout noeud interne $p \in T^c(G) \setminus \text{Desc}_{T^c(G)}(u)$, $\Pi_p^{s_G} = \Pi_p^s$. On note $\sigma_G = \text{front}(s_G)$. Soit $K \in \mathcal{K}_G(a)$. Il est clair qu'en retirant de σ_G les cliques de $\mathcal{K}_G(u^*) \setminus K$ et en retirant de K les sommets de $u^* \setminus \{a\}$, on obtient σ . Or retirer des cliques ne viole aucune contrainte de consécuité et retirer des sommets diminue ces contraintes. Ainsi, σ est cohérent avec le problème d'arrangement consécutif des cliques maximales de $G/\{u^*\}$.

Réciproquement, soit σ un ordre consécutif de $\mathcal{K}(G/\{u^*\})$. Comme le voisinage de u^* dans G est une clique, a est simplicial dans $G/\{u^*\}$. Soit K_a l'unique clique maximale de $G/\{u^*\}$ contenant a . Soit σ_u un ordre consécutif de $\mathcal{K}(G[u^*])$. On note $\mathcal{K}_u = \{K \cup (K_a \setminus \{a\}) \mid K \in \mathcal{K}(G[u^*])\}$, et on note σ'_u l'ordre obtenu sur $\mathcal{K}_u$ en ajoutant les sommets de $K_a \setminus \{a\}$ à toutes les cliques de σ_u . En remplaçant, dans σ , K_a par σ'_u on obtient un ordre consécutif σ' de $\mathcal{K}(G)$. Soit $s' \in \text{sol}(T^c(G))$ telle que $\text{front}(s') = \sigma'$. Soit $s \in \text{sol}(T^{quo})$ telle que pour tout noeud interne $p \in T^{quo}$, $\Pi_p^s = \Pi_p^{s'}$. Alors $\text{front}(s) = \sigma$ et par conséquent $\sigma \in \text{consistent}(T^{quo})$. ■

Le théorème suivant montre comment obtenir la PQ représentation de $G_{x \leftarrow H}$ à partir de celles de G et H . Les figures 5.7, 5.8 et 5.9 donnent des exemples des différents cas du théorème.

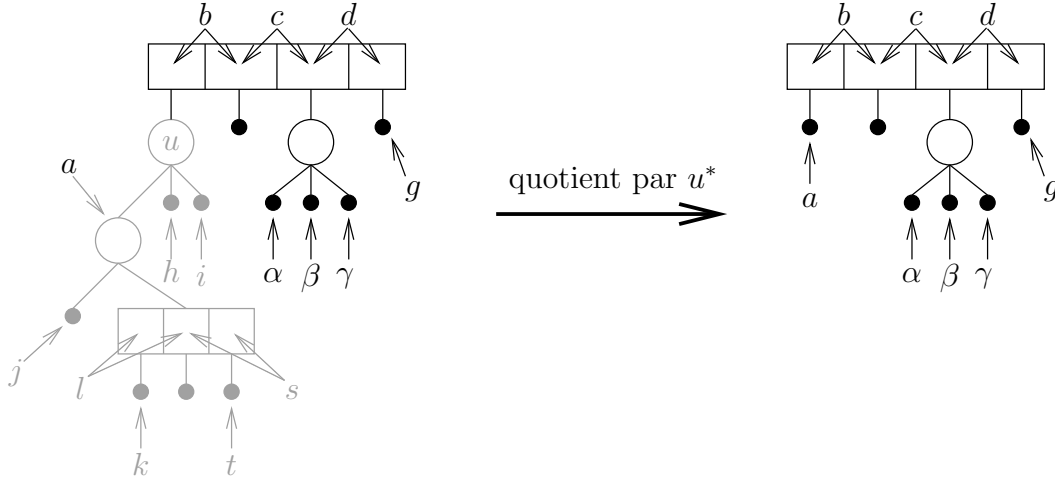


FIG. 5.6 – Quotient de PQ-représentation.

Théorème 5.4 Soient G et H deux graphes d'intervalles et x un sommet de G . Soit r_H la racine de $T^c(H)$. Si $G_{x \leftarrow H}$ est un graphe d'intervalles alors $PQ(G_{x \leftarrow H})$ s'obtient comme suit.

1. Si H est une clique, $PQ(G_{x \leftarrow H})$ s'obtient à partir de $PQ(G)$ en attribuant aux sommets de $V(H)$ les pointeurs de x .
2. Si H n'est pas une clique et si x est simplicial, soit K_x l'unique clique maximale de G contenant x , soit l sa feuille correspondante dans $T^c(G)$ et $u = \text{parent}(l)$:
 - (a) Si u est un noeud premier ou si $K_x \setminus \{x\}$ contient un sommet simplicial ou si $X_{r_H} \neq \emptyset$ ou si r_H est premier, alors $PQ(G_{x \leftarrow H})$ s'obtient en substituant r_H à l dans $T^c(G)$ et en faisant pointer les sommets de $X_l \setminus \{x\}$ vers r_H .
 - (b) Sinon, $PQ(G_{x \leftarrow H})$ s'obtient en retirant l de $T^c(G)$ et en attribuant les fils de r_H à u .

Preuve :

Avant toute chose, établissons quelques propositions dont nous ferons usage dans la preuve.

Proposition 5.5 $\forall K \in \mathcal{K}(G), \exists K' \in \mathcal{K}(G_{x \leftarrow H}), K' \cap V(G) = K \setminus \{x\}$

Preuve : Soit K une clique de G . Si $x \notin K$, alors $K \in \mathcal{K}(G_{x \leftarrow H})$. Si $x \in K$, soit $K_1 \in \mathcal{K}(H)$, alors $(K \setminus \{x\}) \cup K_1 \in \mathcal{K}(G_{x \leftarrow H})$. $\square$

Proposition 5.6 Soit u un noeud premier de T^c tel que $x \notin Y_u$ et soit $v \in \mathcal{C}(u)$. Soit $j \geq 2$ un entier et soient $y_1, \dots, y_j \in Y_u$. Soit les deux propositions suivantes :

$$(A) : C_\Delta(y_j) \oslash \left(\bigcap_{z \in \Delta_v} C_\Delta(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{C_\Delta(y_i)} \right)$$

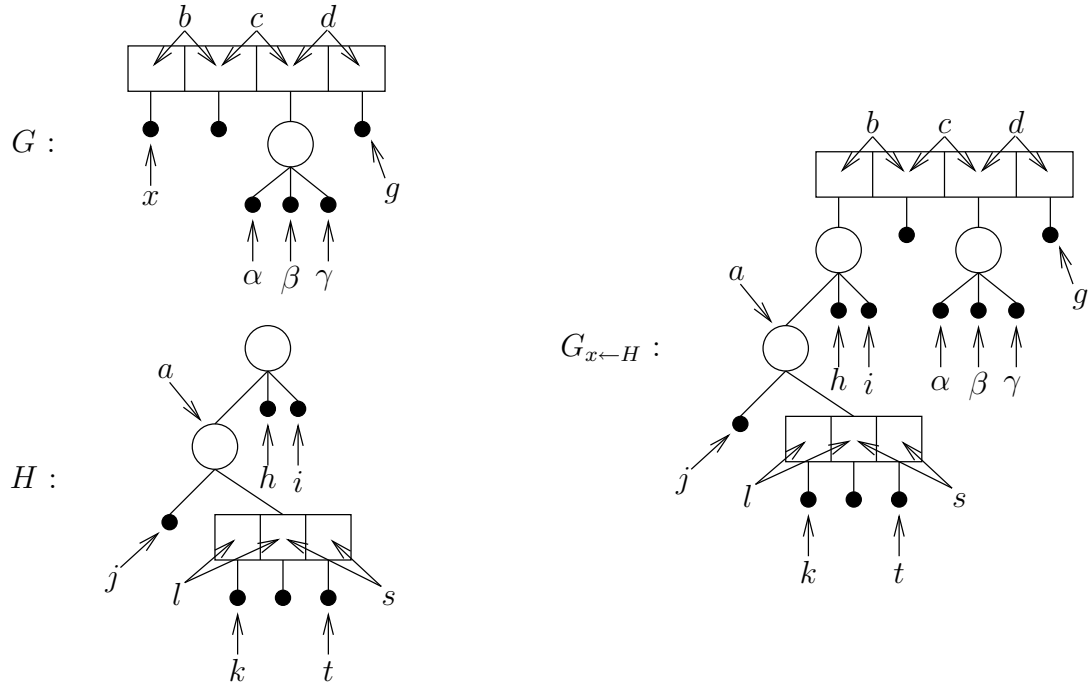


FIG. 5.7 – Illustration du cas 2a du théorème 5.4.

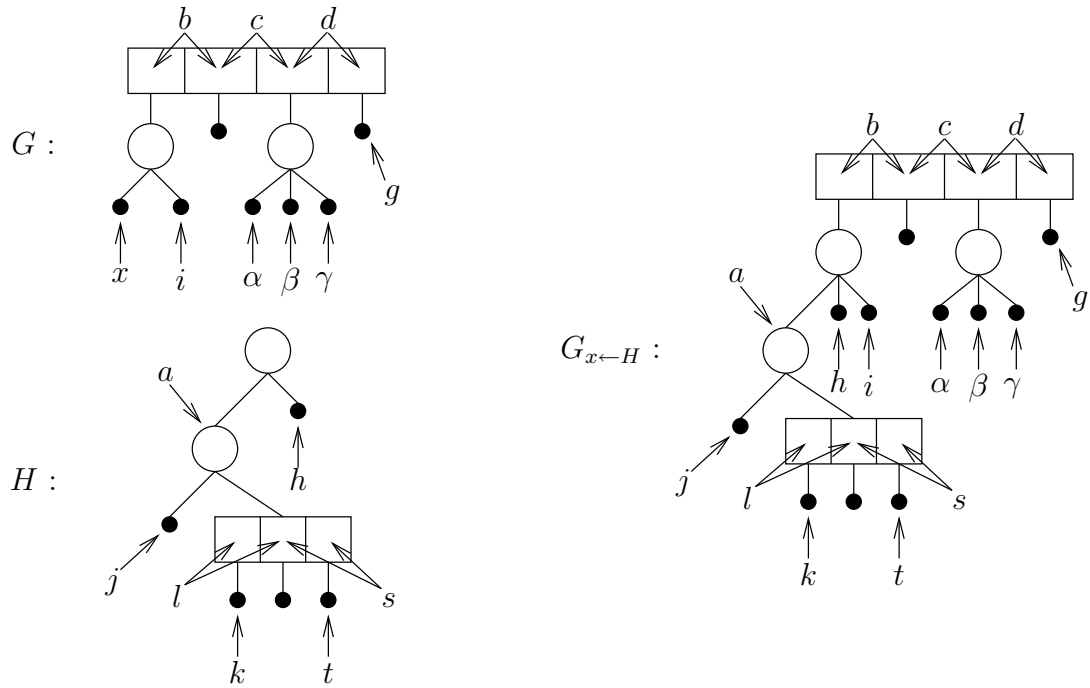


FIG. 5.8 – Illustration du cas 2b du théorème 5.4.

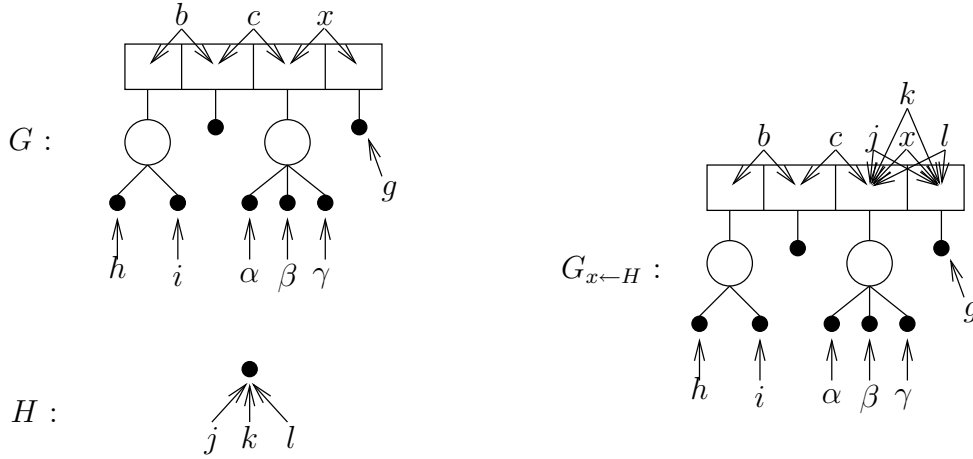


FIG. 5.9 – Illustration du cas 1 du théorème 5.4.

$$(B) : \mathcal{K}_{G_{x \leftarrow H}}(y_j) \cap \left(\bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} \right)$$

On a (A) implique (B).

Preuve :

- Si (A) alors $\exists K \in \mathcal{K}(G), \Delta_v \cup y_j \subseteq K$ et $K \cap \{y_i \mid 1 \leq i \leq j-1\} = \emptyset$. D'après la proposition 5.5, il existe $K' \in \mathcal{K}(G_{x \leftarrow H})$ vérifiant les mêmes propriétés, d'où

$$\mathcal{K}_{G_{x \leftarrow H}}(y_j) \cap \left(\bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} \right) \neq \emptyset.$$

- Si (A) alors $\exists K \in \mathcal{K}(G), y_j \in K$ et $(\Delta_v \setminus K \neq \emptyset$ ou $K \cap \{y_i \mid 1 \leq i \leq j-1\} \neq \emptyset)$. D'après la proposition 5.5, il existe $K' \in \mathcal{K}(G_{x \leftarrow H})$ vérifiant les mêmes propriétés,

$$\text{d'où } \mathcal{K}_{G_{x \leftarrow H}}(y_j) \setminus \left(\bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} \right) \neq \emptyset.$$

- Si (A) alors $\exists K \in \mathcal{K}(G), \Delta_v \subseteq K$ et $K \cap \{y_i \mid 1 \leq i \leq j\} = \emptyset$.

D'après la proposition 5.5, il existe $K' \in \mathcal{K}(G_{x \leftarrow H})$ vérifiant les mêmes propriétés,

$$\text{d'où } \left(\bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} \right) \setminus \mathcal{K}_{G_{x \leftarrow H}}(y_j) \neq \emptyset.$$

On en conclut que (B) est vérifiée.

□

Proposition 5.7 Soit T un PQ-arbre et soit V un ensemble. Pour tout $x \in V$, on attribue à x un pointeur principal vers un noeud de T noté e_x et éventuellement deux pointeurs secondaires vers deux fils e_x^1 et e_x^2 de e_x lorsque e_x est premier. Soit F l'ensemble des feuilles de T . On pose $A = \mathcal{A}_2(\{B_f \mid f \in F\}, V)$. Si $\forall l_1, l_2 \in F, B_{l_1} \neq B_{l_2}$ (voir définition 5.2), alors $\text{consistant}(T) \subseteq \Pi(A)$.

Preuve : Soit un sommet $x \in V$. Si x n'a pas de pointeurs secondaires, alors x est présent

dans les branches des feuilles $f \in \text{Desc}(e_x)$ et seulement dans celles-ci. Si x a deux pointeurs secondaires, alors x est présent dans les branches des feuilles $f \in \bigcup_{e_x^1 < \sigma_u v < \sigma_u e_x^2} \text{Desc}(v)$ et seulement dans celles-ci. Soit s une solidification de T et σ sa frontière, par définition l'ensemble $\{B_f \mid f \in F \text{ et } x \in B_f\}$ est un intervalle de σ . $\square$

1. Si H est un graphe d'intervalles et si x est simplicial, on pose $\mathcal{K}_{xH} = \{K \cup (K_x \setminus \{x\}) \mid K \in \mathcal{K}(H)\}$. Les cliques maximales de $G_{x \leftarrow H}$ sont $\mathcal{K}(G_{x \leftarrow H}) = (\mathcal{K}(G) \setminus \{K_x\}) \cup \mathcal{K}_{xH}$ et $\mathcal{K}_{xH} = \mathcal{K}_{G_{x \leftarrow H}}(V(H))$.

- (a) Si u est un noeud premier ou si il existe un sommet de $K_x \setminus \{x\}$ qui est simplicial ou si H est connexe, montrons que $\mathcal{K}_{xH}$ est un intervalle de tout ordre consécutif σ_{xH} de $\mathcal{K}(G_{x \leftarrow H})$.
 - Si H est connexe, comme $\mathcal{K}_{xH} = \mathcal{K}_{G_{x \leftarrow H}}(V(H))$, alors d'après le lemme 5.9 le résultat est acquis.
 - Si $K_x \setminus \{x\}$ contient un sommet simplicial y , alors $\mathcal{K}(xH) = \mathcal{K}(y)$ est un intervalle de σ_{xH} .
 - Examinons le cas où u est un noeud premier. D'après le lemme 5.7 et la proposition 5.6,

$$\left\{ \begin{array}{l} \forall j \in \llbracket 1, k \rrbracket, \mathcal{K}_{G_{x \leftarrow H}}(y_j) \supset \left(\bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq j-1} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} \right) \\ \text{et} \\ \bigcap_{z \in \Delta_v} \mathcal{K}_{G_{x \leftarrow H}}(z) \cap \bigcap_{1 \leq i \leq k} \overline{\mathcal{K}_{G_{x \leftarrow H}}(y_i)} = \mathcal{K}_{xH} \end{array} \right.$$

d'où $\mathcal{K}_{xH}$ est un intervalle de σ_{xH} .

On note T^{xH} le PQ -arbre obtenu en substituant r_H à l dans $T^c(G)$ et en faisant pointer les sommets de $X_l \setminus \{x\}$ vers r_H . L'application qui à un sommet u fait correspondre B_u est une bijection des feuilles de T^{xH} vers $\mathcal{K}(G_{x \leftarrow H})$.

D'après la proposition 5.7, $\forall \sigma \in \text{consistent}(T^{xH})$, σ est un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$.

Réciproquement, soit σ un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$. On a montré que $\mathcal{K}_{xH}$ est un intervalle de σ . Ainsi, en remplaçant $\mathcal{K}_{xH}$ par K_x , on obtient un ordre consécutif σ_G de $\mathcal{K}(G)$, on note s_G la solidification de $T^c(G)$ telle que $\sigma_G = \text{front}(s_G)$. En supprimant les sommets de $K_x \setminus \{x\}$ des cliques de $\mathcal{K}_{xH}$, on obtient un ordre consécutif σ_H de $\mathcal{K}(H)$, on note s_H la solidification de $T^c(H)$ telle que $\sigma_H = \text{front}(s_H)$. Soit s la solidification de T^{xH} telle que pour tout noeud interne $p \in \text{Desc}_{T^{xH}}(r_H)$, $\Pi_p^s = \Pi_p^{s_H}$ et pour tout noeud interne $p \in T^{xH} \setminus \text{Desc}_{T^{xH}}(r_H)$, $\Pi_p^s = \Pi_p^{s_G}$. On a $\sigma = \text{front}(s)$, donc $\sigma \in \text{consistent}(T^{xH})$.

- (b) Si u est un noeud dégénéré et $K_x \setminus \{x\}$ ne contient aucun sommet simplicial et H n'est pas connexe, alors montrons que $(\mathcal{K}_{T^c(G)}(u) \setminus \{K_x\}) \cup \mathcal{K}_{xH}$ est un intervalle de tout ordre consécutif σ_{xH} de $\mathcal{K}(G_{x \leftarrow H})$.

- Si $\text{parent}(u)$ est dégénéré, alors, d'après le lemme 5.4, $X_u \neq \emptyset$. Soit $y \in X_u$, $(\mathcal{K}(u) \setminus \{K_x\}) \cup \mathcal{K}_{xH} = \mathcal{K}_{G_{x \leftarrow H}}(y)$ est un intervalle de σ_{xH} .
- Examinons le cas où $X_u = \emptyset$, qui ne se produit, d'après le lemme 5.4, que si $v = \text{parent}(u)$ est premier. En utilisant le lemme 5.7 et la proposition 5.6, on montre comme précédemment pour $\mathcal{K}_{xH}$, que $(\mathcal{K}(u) \setminus \{K_x\}) \cup \mathcal{K}_{xH}$ est un intervalle de σ_{xH} .

On note T^{xH} le PQ-arbre obtenu en retirant l de $T^c(G)$ et en attribuant les fils de r_H à u . L'application qui à un sommet u fait correspondre B_u est une bijection des feuilles de T^{xH} vers $\mathcal{K}(G_{x \leftarrow H})$.

D'après la proposition 5.7, $\forall \sigma \in \text{consistant}(T^{xH})$, σ est un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$.

Réciproquement, soit σ un ordre consécutif de $\mathcal{K}(G_{x \leftarrow H})$. Comme nous l'avons montré, $(\mathcal{K}(u) \setminus \{K_x\}) \cup \mathcal{K}_{xH}$ est un intervalle de σ .

Soit $v \in \mathcal{C}_{T^c(G)}(u) \setminus \{l\}$. Comme u est dégénéré, $G[v^*]$ est connexe d'après le corollaire 5.2. D'où, d'après le lemme 5.9, $\mathcal{K}_{G_{x \leftarrow H}}(v^*)$ est un intervalle de σ . Or $\mathcal{K}_{G_{x \leftarrow H}}(v^*) = \mathcal{K}_{T^c(G)}(v)$. Ainsi, pour tout $v \in \mathcal{C}_{T^c(G)}(u) \setminus \{l\}$, $\mathcal{K}_{T^c(G)}(v)$ est un intervalle de σ . Remarquons que $\mathcal{S}_G = \{\mathcal{K}_G(v) \mid v \in \mathcal{C}_{T^c(G)}(u) \setminus \{l\}\}$ est une partition de $\mathcal{K}_{T^c(G)}(u) \setminus \{K_x\}$.

De même, on montre que pour tout $w \in \mathcal{C}_{T^c(H)}(r_H)$, $\mathcal{K}_{G_{x \leftarrow H}}(w^*)$ est un intervalle de σ . On note $\mathcal{K}'_H(w) = \{K \cup (K_x \setminus \{x\}) \mid K \in \mathcal{K}_{T^c(H)}(w)\}$. On a $\forall w \in \mathcal{C}_{T^c(H)}(r_H)$, $\mathcal{K}_{G_{x \leftarrow H}}(w^*)$. Et $\mathcal{S}_H = \{\mathcal{K}'_H(w) \mid w \in \mathcal{C}_{T^c(H)}(r_H)\}$ est une partition de $\mathcal{K}_{xH}$. On note ϕ l'application suivante :

$$\begin{aligned} \phi : (\mathcal{C}_{T^c(G)}(u) \setminus \{l\}) \cup \mathcal{C}_{T^c(H)}(r_H) &\rightarrow \mathcal{S}_G \cup \mathcal{S}_H \\ p &\mapsto \begin{cases} \mathcal{K}_{T^c(G)}(p), & \text{si } p \in \mathcal{C}_{T^c(G)}(u) \setminus \{l\} \\ \text{et} \\ \mathcal{K}'_H(p), & \text{si } p \in \mathcal{C}_{T^c(H)}(r_H) \end{cases} \end{aligned}$$

On note π_u l'ordre linéaire sur $(\mathcal{C}_{T^c(G)}(u) \setminus \{l\}) \cup \mathcal{C}_{T^c(H)}(r_H)$ défini par : $v_1 \leq_{\pi_u} v_2 \Leftrightarrow$ les cliques de $\phi(v_1)$ précèdent celles de $\phi(v_2)$ dans σ .

Soit $K \in \mathcal{K}_{xH}$. Si on supprime de σ toutes les cliques de $\mathcal{K}_{xH} \setminus \{K\}$ et qu'on remplace K par $(K \setminus V(H)) \cup \{x\}$, on obtient un ordre consécutif σ_G de $\mathcal{K}(G)$. On note s_G la solidification de $< T^c(G)$ telle que $\sigma_G = \text{front}(s_G)$.

Pour $v \in \mathcal{C}_{T^c(G)}(u) \setminus \{l\}$, $\sigma[\mathcal{K}_{T^c(G)}(v)]$ est un ordre consécutif de $\mathcal{K}_{T^c(G)}(v)$. Soit s_v la solidification de $T_v^c(G)$ telle que $\text{front}(s_v) = \sigma[\mathcal{K}_G(v)]$.

Pour $w \in \mathcal{C}_{T^c(H)}(r_H)$, en enlevant les sommets de $K_x \setminus \{x\}$ de toutes les cliques de $\sigma[\mathcal{K}'_H(w)]$, on obtient un ordre consécutif σ_w de $\mathcal{K}_{T^c(H)}(w)$. Soit s_w la solidification de $T_w^c(H)$ telle que $\text{front}(s_w) = \sigma_w$.

On considère la solidification s de T^{xH} telle que pour tout noeud interne $p \in T^{xH} \setminus \text{Desc}(u)$, $\Pi_p^s = \Pi_p^{s_G}$ et $\Pi_u^s = \pi_u$ et $\forall v \in \mathcal{C}_{T^{xH}}(u) \cap \mathcal{C}_{T^c(G)}(u)$, pour tout noeud interne $p \in \text{Desc}_{T^{xH}}(v)$, $\Pi_p^s = \Pi_p^{s_v}$ et $\forall w \in \mathcal{C}_{T^{xH}}(u) \cap \mathcal{C}_{T^c(H)}(r_H)$, pour tout noeud interne $q \in \text{Desc}_{T^{xH}}(w)$, $\Pi_q^s = \Pi_q^{s_w}$. On a $\text{front}(s) = \sigma$. Donc $\sigma \in \text{consistant}(T^{xH})$.

2. Si H est une clique, les cliques maximales de $G_{x \leftarrow H}$ sont les cliques maximales de G dans lesquelles on remplace x par $V(H)$. Les ordres consécutifs sont donc les mêmes dans les deux cas : $T^c(G_{x \leftarrow H}) = T^c(G)$.

■

5.2.5 Retrait d'un sommet dans un ordre consécutif

Afin de cerner le comportement dynamique des graphes d'intervalles, aussi bien lors de l'ajout que lors du retrait d'un sommet, il est indispensable d'observer les effets d'un retrait de sommet dans un ordre consécutif. En particulier, on voit que l'on perd au plus deux cliques maximales du graphe lors d'une telle opération. Cela aura une influence majeure sur la façon dont se modifie la PQ -représentation.

Lemme 5.14 *Soit G un graphe d'intervalles et $x \in V(G)$. Soit σ un ordre consécutif de $\mathcal{K}(G)$. On note K_1 (resp. K_2) la première (resp. dernière) clique maximale de G contenant x . $\forall K \in \mathcal{K}(G) \setminus \{K_1, K_2\}$, $K \setminus \{x\} \in \mathcal{K}(G - x)$. De plus, $K_1 \setminus \{x\}$ (resp. $K_2 \setminus \{x\}$) n'est pas une clique maximale de $G - x$ ssi K_1 (resp. K_2) a un prédécesseur K_p (resp. un successeur K_s) dans σ et $K_1 \setminus \{x\} \subseteq K_p$ (resp. $K_2 \setminus \{x\} \subseteq K_s$).*

Preuve :

Pour démontrer le lemme, commençons par établir les deux faits suivants.

Proposition 5.8 Dans un ordre consécutif σ , si il existe deux cliques K_1 et K_2 telles que $K_1 \subseteq K_2$ et $K_1 <_\sigma K_2$, alors $K_1 \subseteq \text{succ}(K_1)$. Si $K_1 \subseteq K_2$ et $K_2 <_\sigma K_1$, alors $K_1 \subseteq \text{pred}(K_1)$.

Preuve : Soit $x \in K_1$, alors $x \in K_2$. Comme σ est un ordre consécutif, toutes les cliques entre K_1 et K_2 dans σ contiennent x , donc en particulier $\text{succ}(K_1)$ si $K_1 <_\sigma K_2$, et $\text{pred}(K_1)$ si $K_2 <_\sigma K_1$. □

Proposition 5.9 Soit $K \in \mathcal{K}(G - x)$. $K \in \mathcal{K}(G)$ ou $K \cup \{x\} \in \mathcal{K}(G)$.

Preuve : Si x est adjacent à tous les sommets de K , alors $K \cup \{x\}$ est une clique. Elle est maximale dans G car aucun sommet de $V \setminus (K \cup \{x\})$ n'est adjacent à tous les sommets de K (K est maximale dans $G - x$).

Si x n'est pas adjacent à tous les sommets de K , alors $K \cup \{x\}$ n'est pas une clique. Comme K est maximale dans $G - x$, alors K est maximale dans G . □

On note ϕ l'application suivante :

$$\begin{array}{ccc} \phi : \mathcal{K}(G) & \rightarrow & \text{Cliques}(G - x) \\ & & K \mapsto K \setminus \{x\} \end{array}$$

D'après la proposition 5.9, $\mathcal{K}(G - x) \subseteq \phi(\mathcal{K}(G))$.

Soit $\sigma = K_1, \dots, K_{|\mathcal{K}(G)|}$ un ordre consécutif de $\mathcal{K}(G)$. Soient respectivement K_f et K_d la première et dernière clique de σ contenant x . D'après ce qui précède, on déduit que

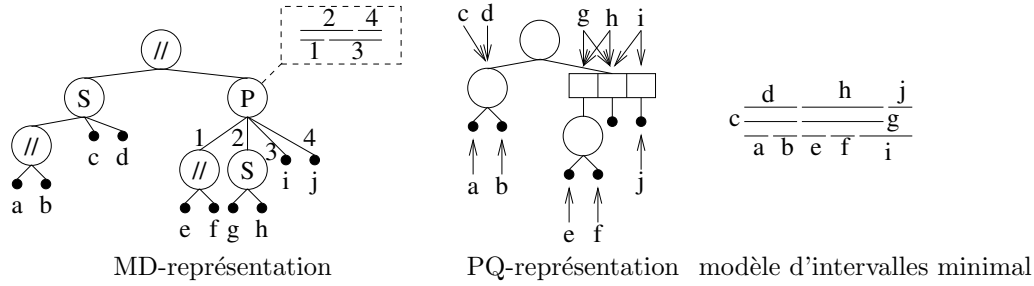


FIG. 5.10 – *MD*-représentation, *PQ*-représentation et modèle d'intervalles d'un graphe d'intervalles.

$\sigma' = \phi(K_1), \dots, \phi(K_{|\mathcal{K}(G)|}) = K_1, \dots, K_{f-1}, K_f \setminus \{x\}, \dots, K_d \setminus \{x\}, K_{d+1}, \dots, K_l$ est un ordre contenant toutes les cliques maximales de $G - x$ et satisfaisant toutes les contraintes de consécutivité imposées par les sommets de $V(G - x)$.

Soit $K_a \in \mathcal{K}(G)$. Si $\phi(K_a)$ n'est pas maximale dans $G - x$, alors $\exists K_b \in \mathcal{K}(G), \phi(K_a) \subseteq \phi(K_b)$. Nécessairement, $a \in \llbracket p, d \rrbracket$, sinon K_a serait une clique de $\mathcal{K}(G)$ qui est incluse dans K_b , une autre clique de G . Nécessairement, $b \notin \llbracket p, \dots, d \rrbracket$, sinon K_a serait incluse dans K_b qui est une autre clique de $\mathcal{K}(G)$. D'où, grâce à la proposition 5.8, $K_a = K_f$ et $\phi(K_a) \subseteq \text{pred}(K_f)$, ou $K_a = K_d$ et $\phi(K_a) \subseteq \text{succ}(K_d)$.

Ce qui achève la preuve. ■

Corollaire 5.3 Soit $G = (V, E)$ un graphe d'intervalles et $x \in V$. $|\mathcal{K}(G - x)| \geq |\mathcal{K}(G)| - 2$.

5.3 Equivalence linéaire de trois représentations des graphes d'intervalles

Dans cette section, on montre que la *PQ*-représentation, la *MD*-représentation et un modèle minimal d'un graphe d'intervalles (voir figure 5.10) sont trois représentations algorithmiquement équivalentes. Plus précisément, on montre que l'on peut passer d'une quelconque de ces représentations à une autre en temps linéaire, c'est à dire en temps $O(n)$ car toutes sont de taille $\Omega(n)$.

Mais la contribution essentielle de cette section est le rapprochement mathématique qui est fait entre les structures de la *PQ*-représentation et de la *MD*-représentation. Ce rapprochement nous permettra, lors de l'entretien dynamique des graphes d'intervalles, d'utiliser des théorèmes démontrés pour l'entretien dynamique de la *MD*-représentation des graphes de permutation. Cela dresse un parallèle saisissant entre les deux problèmes.

5.3.1 Implémentations

Cette section a pour but de préciser l'implémentation que nous adoptons pour chacune des trois représentations qui nous occupent.

Modèles d'intervalles minimaux

Un modèle d'intervalles minimal d'un graphe d'intervalles G est implémenté par une liste L des cliques maximales de G telle que l'ordre σ des cellules de L est un ordre consécutif. Chaque cellule contient son rang dans L et chaque sommet de G possède deux pointeurs vers les cellules de L (éventuellement identiques) qui représentent la première et la dernière clique maximale de G contenant x dans l'ordre consécutif σ .

PQ -représentation

La PQ -représentation d'un graphe d'intervalle G est formée du PQ -arbre $T^c(G)$ des cliques maximales de G (voir section 1.3.1) accompagné d'un tableau des sommets de G dans lequel chaque sommet x de G est muni de un ou trois pointeurs vers $T^c(G)$: un pointeur principal vers e_x , et éventuellement deux pointeurs secondaires vers respectivement e_x^1 et e_x^2 (voir lemme 5.2). La liste des fils d'un noeud premier q de $T^c(G)$ est doublement chaînée et rangée dans l'ordre de σ_q . Le noeud q possède un pointeur vers le début et un pointeur vers la fin de la liste de ses fils, ce qui permet de choisir de manipuler σ_q ou $\bar{\sigma}_q$.

MD -représentation

La MD -représentation d'un graphe d'intervalles G est formée de son arbre de décomposition modulaire $T^m(G)$ accompagné des graphes quotients des noeuds premiers (voir section 1.2.1). Le graphe quotient G_u d'un noeud premier u est représenté par un modèle d'intervalles minimal, implémenté comme décrit ci-dessus, et les sommets de G_u sont en correspondance bijective avec les fils de u dans T^m . Comme toujours dans les arbres de décomposition modulaire, les feuilles de $T^m(G)$ correspondent aux sommets de G .

5.3.2 Modèle d'intervalles minimal et PQ -représentation

Etant donnée la PQ -représentation d'un graphe G , on obtient facilement un modèle d'intervalles minimal de G . Un simple parcours en profondeur du PQ -arbre fournit un ordre consécutif σ des cliques maximales de G . En même temps, on peut attribuer à chaque noeud u du PQ -arbre deux pointeurs respectivement vers la première et la dernière clique de σ dont la feuille correspondante appartient à T_u^c . Ainsi, grâce aux pointeurs d'un sommet x vers e_x , e_x^1 et e_x^2 , on peut attribuer à x ses deux pointeurs vers la première et la dernière clique le contenant dans l'ordre consécutif. Enfin, un parcours de la liste implémentant l'ordre consécutif permet de numéroter ses cellules par leur rang. Comme le PQ -arbre a une taille $O(n)$, on obtient le modèle d'intervalles en temps $O(n)$.

Il est beaucoup plus difficile de calculer le PQ -arbre des cliques maximales à partir d'un modèle d'intervalles minimal. Par bonheur, [MdM05] montre le résultat suivant.

Théorème 5.5 [MdM05] *Il existe un algorithme qui, étant donné un modèle d'intervalles comprenant n intervalles dont les bornes sont des entiers compris entre 1 et s , calcule le PQ -arbre du modèle d'intervalles en temps $O(s + n)$.*

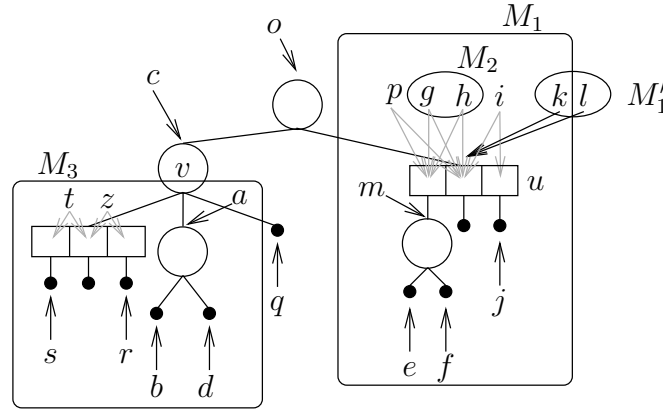


FIG. 5.11 – $M_1 = \{e, f, j, m, p, g, h, i, k\}$ et $M'_1 = \{k, l\}$ vérifient la condition 1 du théorème 5.6 avec le noeud u ; $M_2 = \{g, h\}$ vérifie la condition 2 avec le noeud u ; et $M_3 = \{s, r, t, z, a, b, d\}$ vérifie la condition 3 avec le noeud v . Les pointeurs principaux sont représentés en noir, les secondaires en gris.

L'algorithme de [MdM05] peut être modifié pour calculer les pointeurs des sommets x (correspondant aux intervalles I) vers e_x , e_x^1 et e_x^2 dans la même complexité. Pour un modèle d'intervalles minimal, s est le nombre de cliques maximales, qui est $O(n)$. Il en découle que l'on peut calculer la PQ -représentation à partir de n'importe quel modèle d'intervalles minimal en temps $O(n)$.

5.3.3 PQ -représentation et MD -représentation

Lecture des modules sur la PQ -représentation

Une étape importante en vue d'expliciter les relations entre PQ -représentation et MD -représentation des graphes d'intervalles est la lecture des modules sur la PQ -représentation (voir figure 5.11). Le fait que les deux représentations soient liées découle du fait que les modules respectent la structure de la PQ -représentation. Le théorème 5.6 complète, corrige et démontre le théorème 4.2 de [KM85]. Le corollaire 5.4 en découle directement, il permet d'identifier les modules forts sur la PQ -représentation.

Théorème 5.6 *Soit G un graphe d'intervalles muni de sa PQ -représentation. M est un module de G ssi il existe un noeud $u \in T^c(G)$ qui vérifie une des conditions suivantes :*

1. $M \subseteq X_u$ ou $u^* \setminus X_u \subseteq M \subseteq u^*$; ou
2. u est premier et $\exists u_1, u_2 \in \mathcal{C}(u), M \subseteq \{y \in Y_u \mid e_y^1 = u_1 \text{ et } e_y^2 = u_2\}$; ou
3. u est dégénéré et $\exists k \in \llbracket 2, |\mathcal{C}(u)| - 1 \rrbracket, \exists u_1, \dots, u_k \in \mathcal{C}(u), M = \bigcup_{1 \leq i \leq k} u_i^*$.

Preuve : $\Leftarrow$. Si tous les sommets de M ont les mêmes pointeurs (principaux et secondaires) vers $T^c(G)$, alors leurs voisinages fermés sont égaux et M est un module.

Ainsi, si M vérifie la condition 2, alors M est bien un module de G .

De même, si $M \subseteq X_u$, alors M est un module. En outre, si $u^* \setminus X_u \subseteq M \subseteq u^*$, considérons un sommet $y \in M$. On a $X_u \cup (B_u \setminus X_u) \subseteq N[y] \subseteq (u^* \setminus X_u) \cup X_u \cup (B_u \setminus X_u)$. D'où $N(y) \setminus M = (X_u \setminus M) \cup (B_u \setminus X_u)$. Comme cela est vrai pour tout $y \in M$, M est un module. Finalement M vérifie la condition 1, alors M est un module de G .

Si M vérifie la condition 3, considérons un sommet $y \in M$. Alors $\exists i \in \llbracket 2, |\mathcal{C}(u)| - 1 \rrbracket, y \in u_i^*$. $B_u \subseteq N[y] \subseteq B_u \cup u_i^*$ d'où $N(y) \setminus M = B_u$. Comme cela ne dépend pas du sommet y choisi dans M , alors M est un module de G .

$\implies$. Soit M un module de G . Posons $u = ppca\{e_x \mid x \in M\}$. On a $M \subseteq u^*$. Si $M \subseteq X_u$ ou $u^* \setminus X_u \subseteq M$, alors M vérifie la condition 1. Sinon, $\exists x, y \in u^* \setminus X_u$ tels que $x \in M$ et $y \notin M$.

D'après le lemme 5.11, $G[u^* \setminus X_u]$ est co-connexe. Ainsi, il existe $z_1 \in (u^* \setminus X_u) \cap M$ et $z_2 \in (u^* \setminus X_u) \setminus M$ tels que z_1 et z_2 sont non adjacents. Par conséquent, comme M est un module et comme les sommets de X_u sont adjacents à tous les sommets de u^* , $M \cap X_u = \emptyset$.

Distinguons deux cas :

- u est dégénéré. Dans ce cas, par définition du *ppca*, il existe au moins deux fils u_1, u_2 de u tels que $u_1^* \cap M \neq \emptyset$ et $u_2^* \cap M \neq \emptyset$. Soit v un fils de u tel que $v^* \cap M \neq \emptyset$. D'après le corollaire 5.2, $G[v^*]$ est connexe. Supposons que $v^* \setminus M \neq \emptyset$. Alors, comme $G[v^*]$ est connexe, $\exists x_v, y_v \in v^*$ tels que x_v et y_v sont adjacents et $x_v \in M$ et $y_v \notin M$. Comme il existe $v_2 \in \mathcal{C}(u)$ tel que $v_2^* \cap M \neq \emptyset$ et comme les sommets de v^* sont non adjacents à tous les sommets de v_2^* alors $y_v \notin M$ est adjacent à au moins un sommet de M et non adjacent à au moins un sommet de M : absurde. Donc $v^* \subseteq M$. Enfin, comme $u^* \setminus X_u \not\subseteq M$, M vérifie la condition 3.
- u est premier. Supposons que tous les fils de u soient couverts par un sommet de M . Alors, tous les sommets de $u^* \setminus (X_u \cup M)$ sont adjacents à tous les sommets de M . Ce qui est absurde car $G[u^* \setminus X_u]$ est co-connexe d'après le lemme 5.11. Supposons que $Y_u \cap M = \emptyset$, alors par définition du *ppca*, il existe $u_1, u_2 \in \mathcal{C}(u)$ tels que $u_1 <_{\sigma_u} u_2$ et $u_1^* \cap M \neq \emptyset$ et $u_2^* \cap M \neq \emptyset$. D'après le lemme 5.6, il existe $y \in Y_u$ tel que $C_\Delta(y)$ chevauche $\llbracket u_1, u_2 \rrbracket$. y est adjacent à au moins un sommet de M et non adjacent à au moins un sommet de M , donc $y \in M$: absurde. En conclusion, $Y_u \cap M \neq \emptyset$ et $\exists v \in \mathcal{C}(u)$ tel que v n'est couvert par aucun sommet de M .

Considérons un intervalle maximal I de σ_u tels que tous les noeuds de I sont couverts par un sommet de M . D'après ce qui précède, $I \neq \llbracket C_{min}(u), C_{Max}(u) \rrbracket$ et $I \neq \emptyset$. D'où, d'après le lemme 5.6, il existe $y \in Y_u$ tel que $C_\Delta(y)$ chevauche I . Comme I est maximal, $y \notin M$. Comme M est un module et comme y est adjacent à au moins un sommet de M , alors y est adjacent à tous les sommets de M . Ainsi, s'il existe $u_1 \in \mathcal{C}(u)$, $u_1^* \cap M \neq \emptyset$ alors $u_1 \in C_\Delta(y)$. De plus, $\forall a \in M \cap Y_u, e_a^2 \geq e_y^1$. Quitte à renverser σ , on suppose que $e_y^2 \notin I$. Notons e_I la borne gauche de I et e_{I+1} son suivant dans σ_u . Comme les cliques de $\mathcal{K}_{T^c}(e_I)$ ne sont pas incluse dans celles de $\mathcal{K}_{T^c}(e_{I+1})$, alors $e_I^* \neq \emptyset$ ou $\Delta_{e_I} \setminus \Delta_{e_{I+1}} \neq \emptyset$. Soit $x \in e_I^* \cup (\Delta_{e_I} \setminus \Delta_{e_{I+1}})$. Comme y et x ne sont pas adjacents, $x \notin M$. Or x est adjacent à au moins un sommet de M , donc x est adjacent à tous les sommets de M . Ce qui implique que $\forall u_1 \in \mathcal{C}(u) \setminus \{e_I\}, u_1^* \cap M = \emptyset$ et $\forall a \in M, e_a^1 \leq e_I$. Or y est adjacent à tous les sommets de M , donc $e_I \cap M = \emptyset$. Finalement, $\forall u_1 \in \mathcal{C}(u), u_1^* \cap M = \emptyset$.

Supposons qu'il existe $a, b \in M$ tels que $e_a^2 \neq e_b^2$. On considère $e_M = \max\{e_a^2 \mid a \in M\}$ et $e_m = \max\{e_a^2 \mid a \in M\} \setminus \{e_M\}$. On a $e_I < e_y^1 \leq e_m < e_M$. Soient $a, b \in M$ tel que $e_a^2 = e_m$ et $e_b^2 = e_M$. Comme les cliques de $\mathcal{K}_{T^c}(e_M)$ ne sont pas incluse dans celles de $\mathcal{K}_{T^c}(e_m)$, alors $e_M^* \neq \emptyset$ ou $\Delta_{e_M} \setminus \Delta_{e_m} \neq \emptyset$. Soit $x \in e_M^* \cup (\Delta_{e_M} \setminus \Delta_{e_m})$. Si $x \in \Delta_{e_M} \setminus \Delta_{e_m}$ alors $e_I < e_m < e_x^1$, donc $x \notin M$ (car $\forall x \in M, e_x^1 \leq e_I$). Comme $e_M^* \cap M = \emptyset$, alors dans tous les cas $x \notin M$. Or x est adjacent à $b \in M$ et pas à $a \in M$: absurde. Donc, $\forall a, b \in M, e_a^2 = e_b^2$. Une démonstration similaire prouve que $\forall a, b \in M, e_a^1 = e_b^1$. Par conséquent, M satisfait la condition 2 du théorème. ■

On en déduit le corollaire suivant.

Corollaire 5.4 *Soit G un graphe d'intervalles muni de sa PQ -représentation. M est un module fort non trivial de G ssi $|M| > 1$ et il existe un noeud $u \in T^c(G)$ qui vérifie une des deux conditions suivantes :*

1. $M = u^*$ ou $M = u^* \setminus X_u$; ou
2. u est premier et $\exists u_1, u_2 \in \mathcal{C}(u), M = \{y \in Y_u \mid e_y^1 = u_1 \text{ et } e_y^2 = u_2\}$.

Idée de la preuve. Les modules non triviaux M qui vérifient la condition 1 du théorème 5.6 et qui sont tels que $M \subseteq X_u$ sont chevauchés par d'autres modules. Ceux qui vérifient $u^* \setminus X_u \subsetneq M \subsetneq u^*$ sont également chevauchés par d'autres modules. Par contre, ceux qui vérifient $M = u^*$ ou $M = u^* \setminus X_u$ ne sont chevauchés par aucun autre module.

Les modules non triviaux qui vérifient la condition 2 sont chevauchés si l'inclusion est stricte et ne le sont pas s'il y a égalité.

Enfin, les modules non triviaux de G qui vérifient la condition 3 sont chevauchés par d'autres modules. □

De la PQ -représentation à la MD -représentation

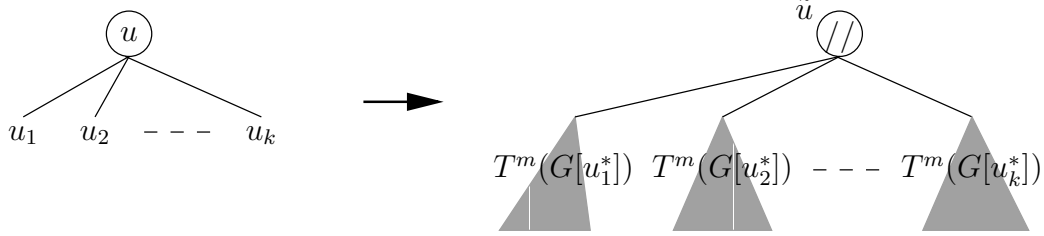
Pour passer de la PQ -représentation à la MD -représentation, de bas en haut dans l'arbre, pour chaque noeud u de $T^c(G)$, on calcule $T^m(G[u^*])$ (voir notation 5.6). Le procédé se termine par le calcul de $T^m(G[r^*]) = T^m(G)$, où r est la racine de $T^c(G)$.

Pour toute feuille l de $T^c(G)$ telle que $l^* \neq \emptyset$, comme l^* est une clique, $T^m(G[l^*])$ est formé d'un noeud série (ou aucun si $|l^*| = 1$) qui possède un fils feuille pour chaque sommet de l^* .

Pour un noeud interne u de $T^c(G)$, dont l'ensemble des fils est $\{u_1, \dots, u_k\}$, où $k \in \mathbb{N} \setminus \{0, 1\}$, nous sommes amenés à distinguer deux cas :

1. u est un noeud dégénéré (voir figure 5.12). D'après le lemme 5.4, $\forall i \in \llbracket 1, k \rrbracket, u_i^* \neq \emptyset$. Grâce au corollaire 5.4, on peut identifier les modules forts maximaux de $G[u^*]$. Si $X_u = \emptyset$, les modules forts maximaux sont les u_i^* , avec $i \in \llbracket 1, k \rrbracket$. On construit $T^m(G[u^*])$ en introduisant un nouveau noeud parallèle $\tilde{u}$ dont les fils sont les racines des arbres $\{T^m(u_1^*), \dots, T^m(u_k^*)\}$.

Si $X_u = \emptyset$



Si $X_u \neq \emptyset$

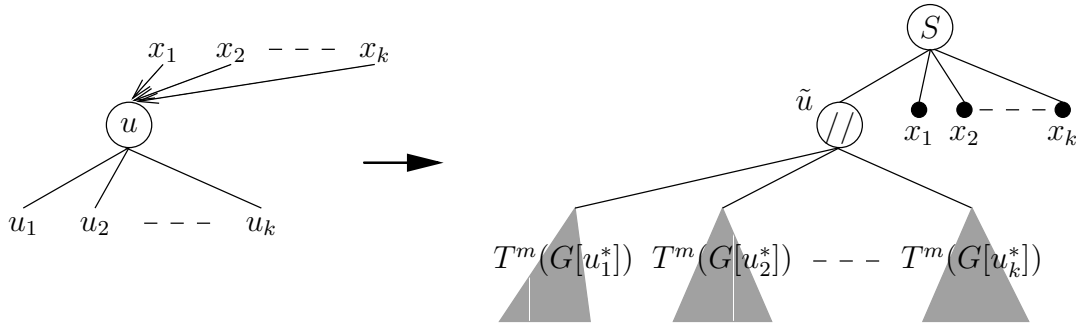


FIG. 5.12 – De la *PQ*-représentation à la *MD*-représentation : traiter un noeud dégénéré u .

Si $X_u \neq \emptyset$, les modules forts maximaux de $G[u^*]$ sont les singletons $\{x\}$, avec $x \in X_u$, et $u^* \setminus X_u$. Comme précédemment, les modules forts maximaux de $G[u^* \setminus X_u]$ sont les u_i^* , avec $i \in \llbracket 1, k \rrbracket$. Il faut donc introduire un nouveau noeud supplémentaire $\tilde{u}_2$ en plus de $\tilde{u}$ par rapport au cas où $X_u = \emptyset$. $\tilde{u}_2$ est série et possède pour fils $\tilde{u}$ et les feuilles représentant les sommets de X_u . $\tilde{u}$ se voit attribuer les mêmes fils que dans le cas où $X_u = \emptyset$.

2. u est un noeud premier.

Si $X_u = \emptyset$ (voir figure 5.13), d'après le lemme 5.11, $G[u^*]$ est connexe et co-connexe. Par conséquent, le théorème 1.3 donne que la racine de $T^m(G[u^*])$ est un noeud premier. Le corollaire 5.4 nous fournit les modules forts maximaux de $G[u^*]$: les u_i^* qui sont non creux, avec $i \in \llbracket 1, k \rrbracket$, et les ensembles $\{y \in Y_u \mid e_y^1 = u_1 \text{ et } e_y^2 = u_2\}$ qui sont non creux, avec $u_1, u_2 \in \mathcal{C}(u)$.

On construit $T^m(G[u^*])$ en introduisant un nouveau noeud premier $\tilde{u}$. Le modèle d'intervalles associé à $\tilde{u}$ a pour support la liste Z des fils de u , ordonnée selon σ_u . Reste à attribuer ses fils à $\tilde{u}$ et les pointeurs des sommets correspondants de $G_{\tilde{u}}$ vers Z .

Commençons par les sommets simpliciaux de $G_{\tilde{u}}$. Cet ensemble est $\mathcal{S} = \{v \in \mathcal{C}(u) \mid v^* \neq \emptyset\}$. Pour tout $v \in \mathcal{S}$, $T^m(G[v^*])$ a déjà été construit précédemment par l'algorithme. La racine de $T^m(G[v^*])$ devient un fils de $\tilde{u}$ et son sommet corres-

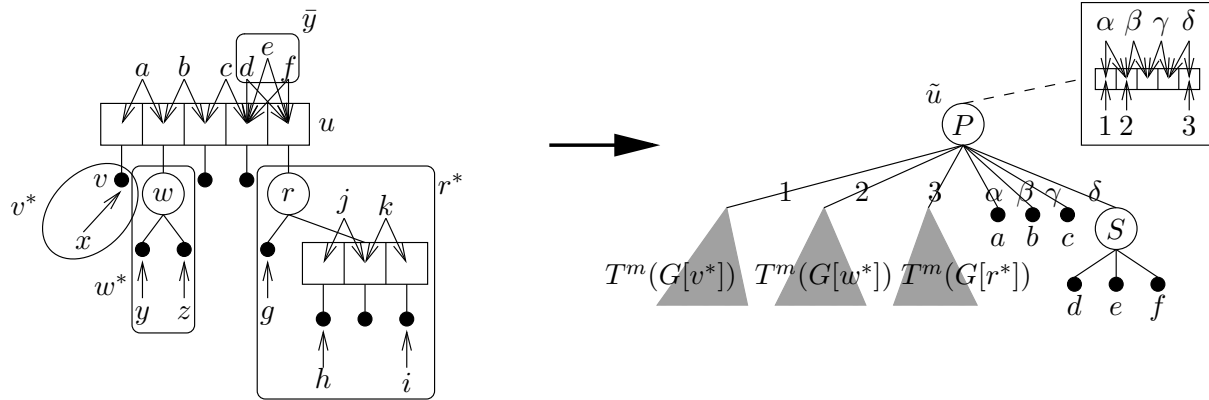


FIG. 5.13 – De la PQ -représentation à la MD -représentation : traiter un noeud premier u lorsque $X_u = \emptyset$.

pendant dans $G_{\tilde{u}}$ possède deux pointeurs vers la cellule de Z correspondant à v . Les sommets non simpliciaux de $G_{\tilde{u}}$ sont les classes $\bar{y}$ de sommets $y \in Y_u$ qui ont les mêmes pointeurs secondaires. Soit $\tilde{v}$ le sommet non simplicial de $G_{\tilde{u}}$ correspondant à $\bar{y}$. Le fils de $\tilde{u}$ correspondant à $\tilde{v}$ est un noeud série (ou une feuille si $|\bar{y}| = 1$) dont les fils sont les sommets de $\bar{y}$. Les pointeurs de $\tilde{v}$ vers Z sont les mêmes que les pointeurs de n'importe quel $y \in \bar{y}$.

Si $X_u \neq \emptyset$, les modules forts maximaux de $G[u^*]$ sont les singletons $\{x\}$, $x \in X_u$, et l'ensemble $u^* \setminus X_u$. Comme dans le cas où u est dégénéré, on introduit un noeud série supplémentaire par rapport au cas où $X_u = \emptyset$. Le reste de la construction est identique au cas où u est premier et $X_u = \emptyset$.

Le traitement des feuilles de $T^c(G)$ prend un temps $O(n)$. Il est immédiat de voir que le traitement d'un noeud dégénéré u prend un temps $O(|X_u| + |\mathcal{C}(u)|)$. Dans le traitement d'un noeud premier, l'opération difficile est de trouver les classes d'équivalences $\bar{y}$ dans Y_u . A cette fin, on peut trier les sommets $y \in Y_u$ dans l'ordre lexicographique avec le rang de e_y^1 dans Z en clef primaire, et le rang de e_y^2 dans Z en clef secondaire. Comme on trie $|Y_u|$ éléments prenant des valeurs entre 1 et $|\mathcal{C}(u)|$, on peut le faire en temps $O(|Y_u| + |\mathcal{C}(u)|)$ grâce à un tableau de taille $|\mathcal{C}(u)|$. Il s'ensuit que le temps de traitement d'un noeud premier est $O(|X_u| + |Y_u| + |\mathcal{C}(u)|)$. Ainsi, le temps total de calcul est $O(n + N)$, où N est le nombre de noeuds dans $T^c(G)$. Or le nombre de noeuds dans $T^c(G)$ est en $O(n)$. Le temps total de calcul de $T^m(G)$ est donc $O(n)$.

De la MD -représentation à la PQ -représentation

Pour calculer la PQ -représentation à partir de la MD -représentation, on procède à un parcours ascendant de T^m similaire à celui de T^c dans le cas précédent. Au cours de cette ascension, on calcule $PQ(G[U])$ pour chaque noeud u de T^m . Le procédé se termine par le calcul de $PQ(G[R]) = PQ(G)$, avec r qui est la racine de $T^m(G)$.

Pour une feuille l_x de $T^m(G)$ qui correspond au sommet x , la PQ -représentation de

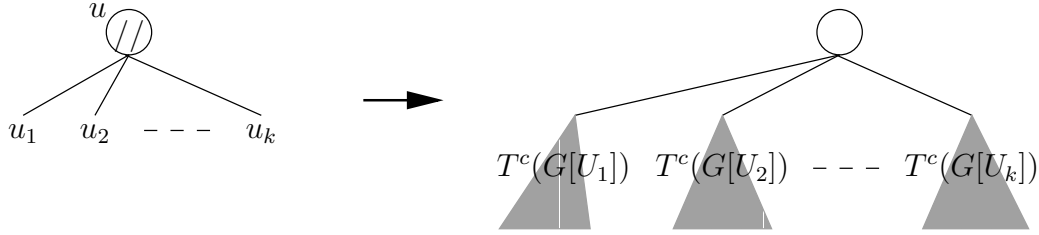


FIG. 5.14 – De la *MD*-représentation à la *PQ*-représentation : traiter un noeud parallèle u .

G restreint à x est triviale. Elle ne comporte qu'une feuille et aucun noeud interne. Le pointeur principal du sommet x pointe sur cette feuille.

Pour un noeud interne u de $T^m(G)$, dont l'ensemble des fils est $\{u_1, \dots, u_k\}$, où $k \in \mathbb{N} \setminus \{0, 1\}$, on distingue trois cas.

1. u est parallèle (voir figure 5.14). Les cliques maximales de $G[U]$ ne sont autres que les cliques maximales des $G[U_i]$, avec $i \in \llbracket 1, k \rrbracket$. Comme u est un noeud parallèle, pour tout $i \in \llbracket 1, k \rrbracket$, $G[U_i]$ est connexe. D'où, d'après le lemme 5.9, les cliques de $\mathcal{K}(U_i)$ sont un intervalle de tout ordre consécutif de $G[U]$. Or, pour $i, j \in \llbracket 1, k \rrbracket$ avec $i \neq j$, les cliques de $\mathcal{K}(U_i)$ n'ont aucun sommet en commun avec les cliques de $\mathcal{K}(U_j)$. Ainsi, un ordre σ sur $\bigcup_{i \in \llbracket 1, k \rrbracket} \mathcal{K}(U_i)$ est un ordre consécutif ssi chaque $\mathcal{K}(U_i)$, pour $i \in \llbracket 1, k \rrbracket$ est un intervalle de σ et $\sigma[\mathcal{K}(U_i)]$ est un ordre consécutif. Ce qui justifie la construction suivante de $T^c(G[U])$.

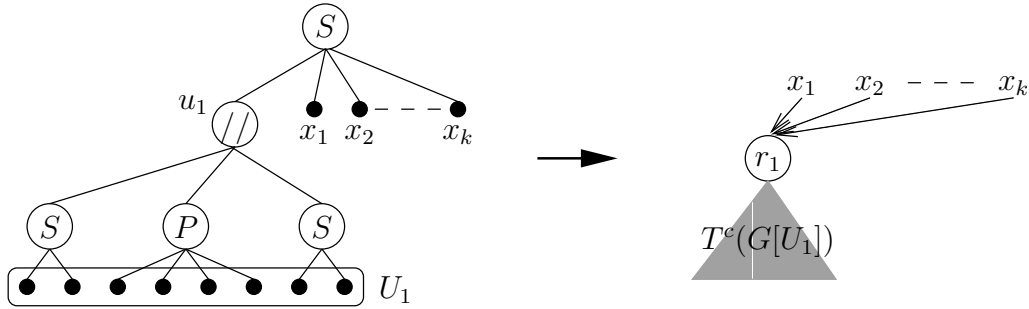
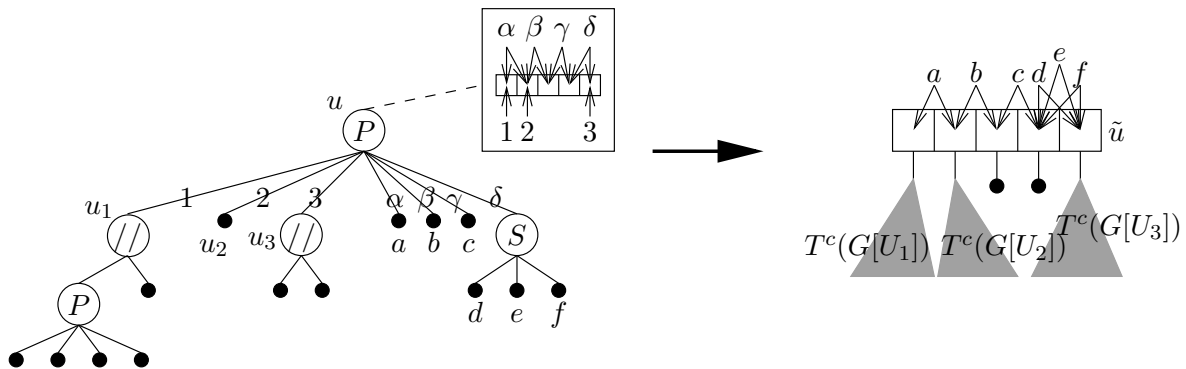
$T^c(G[U])$ s'obtient en introduisant un nouveau noeud dégénéré dont les fils sont les racines des arbres $T^c(G[U_1]), \dots, T^c(G[U_k])$. Tous les sommets conservent leurs pointeurs vers leurs arbres $T^c(G[U_i])$ respectifs.

2. u est série (voir figure 5.15). D'après le théorème 5.3, u a au plus un fils non feuille. Si u n'a que des fils feuilles, alors $G[U]$ est une clique et sa *PQ*-représentation est composée d'une unique feuille et d'aucun noeud interne. Et tous les sommets de U pointent vers cette feuille.

Si u a un fils non feuille u_1 . Les cliques maximales de $G[U]$ sont les cliques maximales de $G[U_1]$ auxquelles on ajoute les sommets de G correspondant aux fils feuille de u . Les ordres consécutifs de $G[U]$ et de $G[U_1]$ sont les mêmes. Ce qui justifie la construction suivante de $T^c(G[U])$. $T^c(G[U])$ s'obtient à partir de $T^c(G[U_1])$ en attribuant à tous les sommets de $U \setminus U_1$ un pointeur principal vers la racine r_1 de $T^c(G[U_1])$.

3. u est premier (voir figure 5.16). D'après le théorème 1.3, $G[U]$ est connexe et co-connexe. De plus, $G[U]$ ne possède pas de sommets universels. Donc, la racine $\tilde{u}$ de $T^c(G[U])$ est telle que $X_{\tilde{u}} = \emptyset$ et, d'après le lemme 5.11, $\tilde{u}$ est un noeud premier. Nous avons montré précédemment comment obtenir $T^m(G[U])$ à partir de $T^c(G[\tilde{u}^*]) = T^c(G[U])$. Il s'agit ici de faire la transformation inverse.

Afin d'obtenir $T^c(G[U])$, un nouveau noeud premier $\tilde{u}$ est introduit. Les fils de $\tilde{u}$ correspondent aux cellules de la liste Z implémentant le modèle d'intervalles de G_u . $\sigma_{\tilde{u}}$ est précisément l'ordre des cellules de Z .

FIG. 5.15 – De la *MD*-représentation à la *PQ*-représentation : traiter un noeud série u .FIG. 5.16 – De la *MD*-représentation à la *PQ*-représentation : traiter un noeud premier u .

Pour une cellule c de Z dont la clique de G_u correspondante ne contient aucun sommet simplicial dans G_u , le fils de $\tilde{u}$ correspondant à c est une simple feuille.

Pour une cellule c de Z dont la clique de G_u correspondante contient un sommet x simplicial dans G_u , le fils de $\tilde{u}$ correspondant à c est la racine de l'arbre $T^c(G[U_x])$ où u_x désigne le fils de u correspondant à x dans $T^m(G)$. Les sommets de U_x conservent leurs pointeurs vers $T^c(G[U_x])$.

Soit $\mathcal{NS}$ l'ensemble des fils p de u tels que le sommet y correspondant à p dans G_u est non simplicial. On a $Y_{\tilde{u}} = \bigcup_{p \in \mathcal{NS}} P$. Pour $p \in \mathcal{NS}$, tous les sommets de P se voient attribuer un pointeur principal vers $\tilde{u}$ et deux pointeurs secondaires vers les fils de $\tilde{u}$ correspondant aux deux cellules de Z pointées par y .

Une feuille de $T^m(G)$ se traite en temps constant. Un noeud parallèle u , comme un noeud série, requiert un temps de traitement $O(|\mathcal{C}(u)|)$. Pour un noeud premier $u \in T^m(G)$, construire le noeud premier $\tilde{u}$ de $T^c(G[U])$ avec la liste de ses fils dans l'ordre de σ_u prend un temps $O(|Z|)$, où $|Z|$ désigne la longueur de la liste Z . Ensuite, il faut identifier quels sont les fils de $\tilde{u}$ qui correspondent à une cellule de Z dont la clique qu'elle représente contient un sommet simplicial. Cela peut se faire en parcourant les sommets de G_u et en testant si leurs deux pointeurs vers Z sont identiques. Ce parcours prend un temps $O(|\mathcal{C}(u)|)$. Durant ce même parcours, si les deux cellules c_1 et c_2 de Z pointées par un sommet y de G_u , correspondant au fils p de u , ne sont pas identiques, on assigne à tous

les sommets de P deux pointeurs secondaires vers les fils de $\tilde{u}$ correspondant à c_1 et c_2 . Cela se fait en temps $O(|P|)$ pour chaque sommet de G_u . Comme, d'après le théorème 5.3, p est soit une feuille, soit un noeud série n'ayant que des fils feuilles, un sommet x de G appartient à au plus un tel ensemble P . Ainsi, le coût de l'attribution de ses pointeurs à x est constant et est absorbé par le temps de traitement de la feuille correspondant à x . Ainsi, le coût d'attribution des pointeurs aux sommets de P n'est pas pris en compte dans le coût de traitement du noeud premeir u . Donc, le temps de traitement d'un noeud premier $u \in T^m(G)$ est $O(|\mathcal{C}(u)| + |Z|)$. Or G_u est un graphe d'intervalles ayant $|\mathcal{C}(u)|$ sommets, donc $|Z| = O(|\mathcal{C}(u)|)$. Finalement, le temps de traitement d'un noeud premier est $O(|\mathcal{C}(u)|)$, et le temps d'exécution total de l'algorithme construisant $T^c(G)$ est $O(n)$, car le nombre de noeuds de $T^m(G)$ est $O(n)$.

5.4 Maintien entièrement dynamique

Comme les trois représentations des graphes d'intervalles peuvent être obtenues de n'importe laquelle d'entre elles en temps $O(n)$, et comme nous voulons obtenir un algorithme en temps $O(n)$, nous pouvons choisir de maintenir une seule de ces trois représentations et en obtenir les autres dans la complexité souhaitée. J'ai choisis de maintenir le PQ -arbre des cliques maximales car je pense que c'est la plus pratique pour le problème traité. En effet, quand on veut déterminer si le graphe modifié G' admet un modèle d'intervalles, nous devons considérer tous les modèles du graphe précédent G . T^c est la structure naturelle pour le faire. Dans cette section, on montre comment traiter les opérations d'ajout et de retrait d'une arête ainsi que le retrait d'un sommet. L'ajout est l'opération la plus difficile et fait l'objet d'une étude séparée.

5.4.1 Opérations dynamiques sur les arêtes

Malheureusement, une modification d'arête peut générer $\Omega(n)$ changements dans le PQ -arbre des cliques maximales (voir figure 5.17). Comme nous proposons un algorithme en temps $O(n)$ pour l'insertion et la suppression de sommet, insérer ou supprimer une arête incidente à un sommet x sera effectué en supprimant d'abord x et en réinsérant x avec son voisinage mis à jour.

5.4.2 Suppression de sommet

Dans cette section, on considère le cas de la suppression d'un sommet $x \in V$ dans un graphe d'intervalles $G = (V, E)$. Comme la famille des graphes d'intervalles est héréditaire, le nouveau graphe $G' = G - x$ est toujours un graphe d'intervalles. L'algorithme doit donc mettre à jour la PQ -représentation du graphe, ce qui commence toujours par supprimer x de l'ensemble des sommets. Pour maintenir $T^c(G)$ et les pointeurs vers ses noeuds, on distingue trois cas qui dépendent de e_x . Dans cette section, on note $u = e_x$ et $v = \text{parent}(u)$

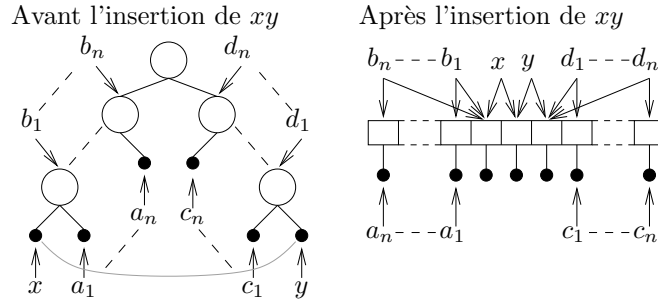


FIG. 5.17 – Une modification d'arête entraînant $\Omega(n)$ changements dans la PQ -représentation.

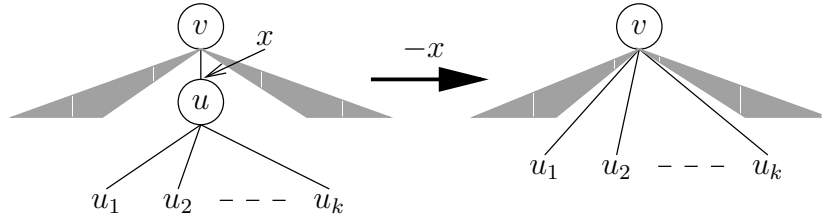
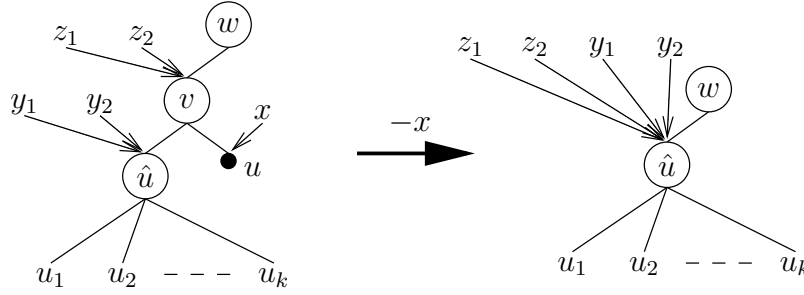


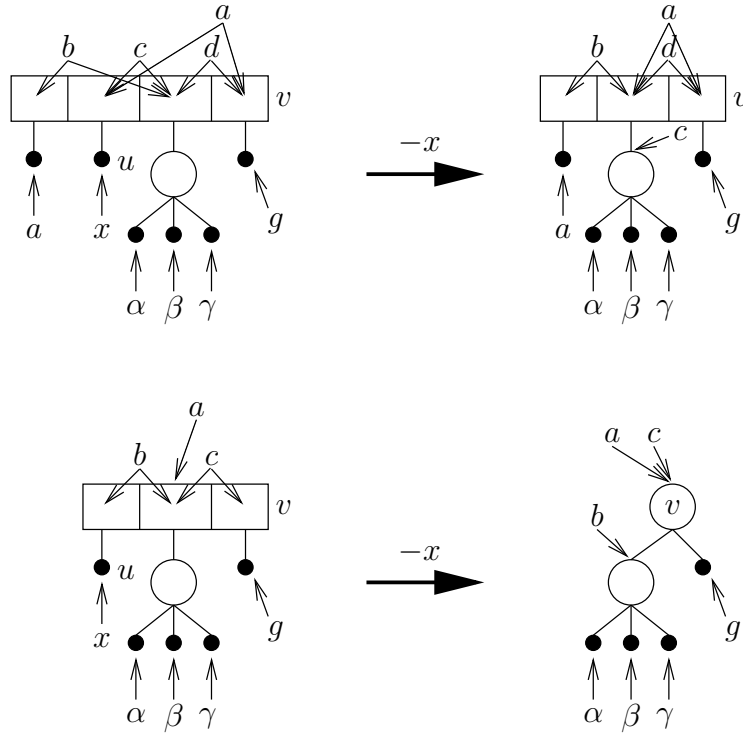
FIG. 5.18 – Suppression de x dans le cas 1a de la discussion.

lorsque u n'est pas la racine de $T^c(G)$. On mène la discussion de cas suivante selon la configuration des noeuds de T^c autour de e_x .

1. u est un noeud interne de $T^c(G)$ et x n'a pas de pointeurs secondaires. Remarquons que, en notant a le sommet représentatif de u^* dans $G/\{u^*\}$, $G-x = G/\{u^*\}_{a \leftarrow G[u^* \setminus \{x\}]}$. Ainsi, d'après le théorème 5.4, $T^c(G-x)$ s'obtient à partir de $T^c(G/\{u^*\})$ et $T^c(G[u^* \setminus \{x\}])$. D'après le lemme 5.13, $T^c(G/\{u^*\})$ s'obtient en remplaçant le noeud u de $T^c(G)$ par une feuille sur laquelle pointe a . D'après le lemme 5.12, $T^c(G[u^*]) = T_u^c(G)$. Comme x pointe sur la racine de $T_u^c(G)$, il s'ensuit que $PQ(G[u^* \setminus \{x\}])$ s'obtient à partir de $PQ(G[u^*])$ simplement en retirant x . L'application du théorème 5.4 mène aux conclusions suivantes :
 - (a) Si u n'est pas la racine et si u et v sont deux noeuds dégénérés et si $X_u = \{x\}$, alors $T^c(G)$ est mis à jour en supprimant u et en attribuant ses fils à v (voir figure 5.18). Les pointeurs des sommets de $V \setminus \{x\}$ vers l'arbre sont inchangés.
 - (b) Sinon, $T^c(G') = T^c(G)$ et les pointeurs des sommets de $V \setminus \{x\}$ vers l'arbre sont inchangés.
2. u est une feuille de $T^c(G)$. On note K_u la clique maximale de G correspondante.
 - (a) Si $X_u \setminus \{x\} \neq \emptyset$, alors $K_u \setminus \{x\}$ est une clique maximale de G' et $PQ(G)$ est inchangé.
 - (b) Si $X_u = \{x\}$ et v est dégénéré, alors $K_u \setminus \{x\}$ n'est pas une clique maximale de G' , car elle est incluse dans chaque clique de $\mathcal{K}_{T^c(G)}(v) \setminus \{K_u\}$.

FIG. 5.19 – Suppression de x dans le cas 2(b)i de la discussion.

- i. Si v n'a que deux fils u et $\hat{u}$ dans $T^c(G)$ (voir figure 5.19), on raisonne de la manière suivante. On a $G - x = G / \{v^*\}_{a \leftarrow G[v^* \setminus \{x\}]}$. Comme vu précédemment, on peut déduire $T^c(G - x)$ de $T^c(G / \{v^*\})$ et $T^c(G[v^* \setminus \{x\}])$. Dans le cas présent, il est clair que $PQ(G[v^* \setminus \{x\}])$ est obtenu à partir de $T_{\hat{u}}^c(G)$ en faisant pointer les sommets de X_v sur $\hat{u}$ et sans changer les pointeurs des sommets de $\hat{u}^*$. Comme v est dégénéré, d'après le lemme 5.4, $\text{parent}(v)$ est premier ou $X_v \neq \emptyset$. Le théorème 5.4, justifie la construction suivante.
On retire u et v de $T^c(G)$. Si v n'était pas la racine, alors $\hat{u}$ est substitué à v dans les fils de $\text{parent}(v)$, sinon $\hat{u}$ devient la racine. Dans tous les cas, on fait pointer sur $\hat{u}$ les sommets qui pointaient sur v .
- ii. Sinon, u est simplement supprimé de $T^c(G)$.
- (c) Si $X_u = \{x\}$ et v est premier.
 - i. Si $\exists a, b \in Y_v, e_a^1 = u$ et $e_b^2 = u$, alors $K_u \setminus \{x\}$ est une clique maximale de G' et $PQ(G)$ est inchangé.
 - ii. Sinon, $K_u \setminus \{x\}$ n'est pas une clique maximale de G' (voir figure 5.20). On supprime u des fils de v et de σ_u . Pour chaque sommet $y \in Y_v$, si e_y^1 valait u , il vaut maintenant $\text{succ}(u)$; et si e_y^2 valait u , il vaut maintenant $\text{pred}(u)$. Les sommets de Y_v dont les deux pointeurs secondaires pointent le même fils v_1 de v sont retirés de Y_v et ajoutés à X_{v_1} . Si après ces modifications v n'a plus que deux fils, v devient un noeud dégénéré et les sommets de Y_v qui couvrent les deux fils de v passent dans X_v .
3. Si u est un noeud premier et si x a des pointeurs secondaires, l'arbre peut subir de profondes modifications (voir figure 5.21). D'après le théorème 5.6 page 209, u^* est un module de G . Donc $u^* \setminus \{x\}$ est un module de G' . D'après le théorème 5.4, $T^c(G')$ s'obtient à partir de $T^c(G' / \{u^* \setminus \{x\}\})$ et $T^c(G'[u^* \setminus \{x\}])$.
Comme tout sommet représentatif de $u^* \setminus \{x\}$ est aussi représentatif de u^* , alors $T^c(G' / \{u^* \setminus \{x\}\}) = T^c(G / \{u^*\})$, qui, d'après le lemme 5.13, s'obtient de $T^c(G)$ en remplaçant le noeud u par une feuille et en faisant pointer le sommet représentatif de u^* sur cette feuille.
Il nous reste à calculer $T^c(G'[u^* \setminus \{x\}]) = T^c(G[u^* \setminus \{x\}])$. Soit $\{u_1, \dots, u_k\}$ l'ensemble des fils non déserts de u . D'après le théorème 5.4 appliqué pour chacun des

FIG. 5.20 – Suppression de x dans le cas 2(c)ii de la discussion.

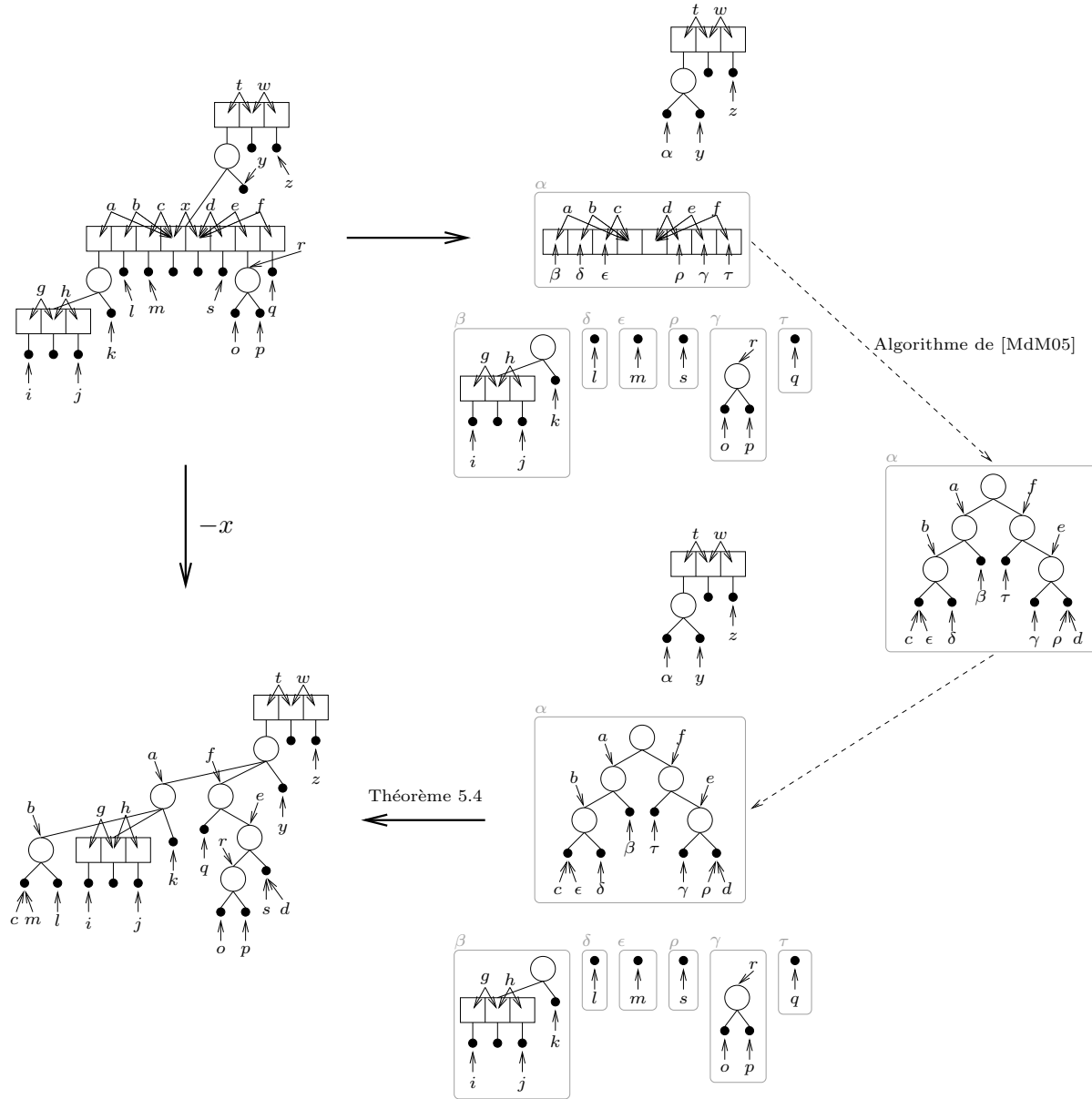
modules $\{u_1^*, \dots, u_k^*\}$ de $G[u^* \setminus \{x\}]$, on obtient $T^c(G[u^* \setminus \{x\}])$ à partir de $T^c(G[u^* \setminus \{x\}]/\{u_1^*, \dots, u_k^*\})$ et $\{T^c(G[u_1^*]), \dots, T^c(G[u_k^*])\}$. Et d'après le théorème 5.12, $\forall i \in \llbracket 1, k \rrbracket, T^c(G[u_i^*]) = T_{u_i}^c$. Comme $x \notin \bigcup_{1 \leq i \leq k} u_i^*$, alors $G[u^* \setminus \{x\}]/\{u_1^*, \dots, u_k^*\} = (G[u^*]/\{u_1^*, \dots, u_k^*\}) - x$.

Donc, le calcul de $T^c(G')$ se ramène à celui de $T^c((G[u^*]/\{u_1^*, \dots, u_k^*\}) - x)$. Pour ce faire, l'algorithme calcule d'abord un modèle d'intervalles de $(G[u^*]/\{u_1^*, \dots, u_k^*\}) - x$ et utilise l'algorithme de [MdM05] afin de calculer $T^c((G[u^*]/\{u_1^*, \dots, u_k^*\}) - x)$ à partir de ce modèle d'intervalles.

Pour obtenir un modèle d'intervalles de $(G[u^*]/\{u_1^*, \dots, u_k^*\}) - x$ à partir de la liste L des fils de u , on peut procéder ainsi :

- les pointeurs vers L attribués aux sommets de $Y_u \setminus \{x\}$ sont leurs pointeurs secondaires ;
- les sommets de X_u se voient attribuer deux pointeurs sur respectivement le premier et le dernier élément de L ;
- et enfin, pour tout $i \in \llbracket 1, k \rrbracket$, on attribue au sommet représentatif de u_i^* deux pointeurs identiques vers le fils de u correspondant.

Dans les cas 1 et 2, à l'exclusion du cas 2(c)ii, le calcul de $T^c(G')$ peut être effectué en temps $O(|\mathcal{C}(u)|)$. Dans le cas 2(c)ii, il faut examiner les sommets de Y_v est le temps de calcul est donc $O(|Y_v|)$. Dans le cas 3, on peut trouver un modèle d'intervalles de $(G[u^*]/\{u_1^*, \dots, u_k^*\}) - x$ en temps $O(|\mathcal{C}(u)| + |X_u| + |Y_u|)$. Ensuite, l'algorithme

FIG. 5.21 – Suppression de x dans le cas 3 de la discussion.

de [MdM05] s'exécute en temps $O(|\mathcal{C}(u)| + |X_u| + |Y_u|)$. Le calcul de $T^c(G')$ à partir de $T^c((G[u^*]/\{u_1^*, \dots, u_k^*\}) - x)$ et des $T^c(G[u_i^*])$, pour $i \in \llbracket 1, k \rrbracket$, qui est un simple recollement d'arbres, prend un temps $O(|\mathcal{C}(u)|)$. Le coût de la suppression de x lorsque x a des pointeurs secondaires est donc $O(|\mathcal{C}(u)| + |X_u| + |Y_u|)$.

Dans tous les cas sauf dans le cas 2(c)ii, où elle reste quand même en $O(n)$, la suppression du sommet x peut donc être traitée en $O(|\mathcal{C}(u)| + |X_u| + |Y_u|)$.

5.5 Insertion de sommet

L'opération d'insertion d'un sommet dans la PQ -représentation est de loin la plus difficile. Cette section est organisée de la manière suivante :

1. on commence par identifier les nouvelles cliques maximales créées par l'insertion de x ;
2. ensuite, on définit le noeud d'insertion w , qui est le noeud le plus bas dans l'arbre tel que son sous-arbre contient toutes les modifications engendrées par l'insertion de x ;
3. puis on caractérise, sur T_w^c , les cas d'insertions résultants en un graphe d'intervalles ;
4. enfin, on décrit les transformations de la PQ -représentation.

L'algorithme lui-même n'est décrit que dans la section suivante, en même temps que sa complexité est analysée.

Avant de commencer ce travail, on adopte un vocabulaire destiné à décrire l'adjacence entre x et les sommets d'un ensemble donné.

Définition 5.5 Soit $G = (V, E)$ un graphe d'intervalles et x un sommet à insérer dans G . Soit $Y \subseteq V$.

- Y est **lié** à x ssi il existe $y \in Y$ tel que y est adjacent à x .
- Y est **creux** relativement à x ssi aucun sommet de Y n'est adjacent à x .
- Y est **plein** relativement à x ssi tous les sommets de Y sont adjacents à x .
- Y est **mixte** relativement à x ssi Y n'est ni plein ni creux.

On utilisera également ce vocabulaire pour un noeud de $T^c(G)$ en faisant référence au sous-ensemble de sommets $u^* \subseteq V$. Pour toutes les notions ci-dessus, lorsqu'il n'y a pas de confusion possible, on omettra le sommet x auquel on se réfère.

Remarque 5.7 On emploiera le vocabulaire **lié** également pour les sommets. On dira qu'un sommet $y \in V(G)$ est lié ssi y est adjacent à x .

5.5.1 Grappes minces et cliques maximales additionnelles

Cette section vise à identifier sur la PQ -représentation les nouvelles cliques créées par l'insertion de x , les cliques maximales additionnelles.

Définition 5.6 Soit $G = (V, E)$ un graphe d'intervalles et x un sommet à insérer dans G . Soit u un noeud de T^c . La **trace** $Tr(B_u)$ d'une branche B_u est définie par $Tr(B_u) = B_u \cap N(x)$. Une branche B_u est **pleine** ssi $B_u \subseteq N(x)$. Une **grappe** est une trace de branche maximale pour l'inclusion parmi toutes les traces de branches. Une grappe A est **mince** ssi A n'est pas une clique maximale de G . Pour un graphe d'intervalles G et un sommet $x \notin V(G)$, on note $\mathcal{GM}(G, x)$ l'ensemble des grappes minces de G relativement à x .

Définition 5.7 On appelle **cliques maximales additionnelles** de $G+x$ le sous-ensemble des cliques maximales de $G+x$ noté $\mathcal{CMA}(G+x)$ et défini par $\mathcal{CMA}(G+x) = \{K \in \mathcal{K}(G+x) \mid K \setminus \{x\} \notin \mathcal{K}(G)\}$.

Lemme 5.15 L'application $\phi : \mathcal{GM}(G, x) \rightarrow \mathcal{CMA}(G+x)$ est bien définie et est une bijection.

$$A \mapsto A \cup \{x\}$$

Preuve : Pour montrer que Φ est bien définie, il suffit de montrer que pour toute grappe mince A de G , $A \cup \{x\}$ est une clique maximale additionnelle de $G+x$. Soit donc une grappe mince A de G , et soit K' une clique de $G+x$ contenant $A \cup \{x\}$. Soit $K = K' \setminus \{x\}$. Alors, $A \subseteq K$. Or K est une clique de G , $\exists v \in T^c(G), K \subseteq B_v$. Comme $A \subseteq N(x)$ et $K \subseteq N(x)$, alors $A \subseteq K \subseteq Tr(B_v)$. Or A est maximale parmi toutes les traces de branches, donc $A = Tr(B_v)$ et $A = K$. Par conséquent, $A \cup \{x\}$ est une clique maximale de $G+x$, elle est de surcroît additionnelle car A étant une grappe mince, A n'est pas une clique maximale de G .

L'injectivité de Φ est acquise par sa définition. Il suffit donc de montrer que Φ est surjective. Soit K' une clique maximale additionnelle de $G+x$. On pose $A = K' \setminus \{x\}$. A est une clique de G , donc il existe un noeud $u \in T^c(G)$ tel que $A \subseteq B_u$.

On montre que $A = Tr(B_u)$ et que A est maximale pour l'inclusion parmi toutes les traces de branches. En effet, $A \subseteq Tr(B_u)$. De plus, si $\exists v \in T^c(G), A \subseteq Tr(B_v)$, comme $Tr(B_v) \cup \{x\}$ est une clique de $G+x$ et comme $A \cup \{x\}$ est une clique maximale de $G+x$, alors $A = Tr(B_v)$. Ainsi, $A = Tr(B_u)$ et A est maximale parmi les traces de branches : A est une grappe. Comme $A \cup \{x\}$ est une clique maximale additionnelle de $G+x$, alors A est une grappe mince. ■

5.5.2 Noeuds propres et noeud d'insertion

Dans cette section, on introduit la notion de noeud propre. La définition en est très technique mais cette notion joue un rôle central dans l'analyse de la transformation subie par l'arbre lors de l'insertion d'un sommet. En particulier, le noeud d'insertion, qui est le noeud de l'arbre le plus bas dont le sous-arbre contient toutes les modifications résultant de l'insertion de x , est le plus petit ancêtre commun des noeuds non propres.

Cette partie peut bien entendu se lire en parallèle avec le même travail pour les graphes de permutation. La correspondance est forte. Pourtant, j'ai préféré ne pas utiliser la correspondance entre PQ -représentation et MD -représentation et les résultats déjà établis sur l'ajout d'un sommet dans la MD -représentation. Il est possible que cette approche permettrait d'éviter certaines preuves. Cependant, la PQ -représentation se prête beaucoup mieux que la MD -représentation à la caractérisation des insertions réussies. Aussi, les preuves qui suivent m'ont paru nécessaire à la compréhension de la manière dont une insertion de sommet touche la PQ -représentation.

J'aimerais attirer l'attention du lecteur sur le fait qu'il ne sert à rien de chercher à comprendre sur un noeud précis ce qu'apporte la propriété d'être propre : cette notion ne se comprend bien que pour les noeuds propres dont tous les ancêtres sont propres. En fait, les noeuds propres et non creux sont les noeuds du chemin entre la racine et le noeud d'insertion.

Définition 5.8 Soit r la racine de $T^c(G)$. Un noeud $u \in T^c(G)$ est **propre** ssi u vérifie une des deux propriétés suivantes :

1. $u \neq r$ et $(\Delta_u = \emptyset$ ou $\Delta_u \neq Y_{\text{parent}(u)} \cap N(x))$ et u est creux, ou
2. $(u = r$ ou $(u \neq r$ et $\Delta_u = Y_{\text{parent}(u)} \cap N(x)))$ et $X_u \subseteq N(x)$ et u vérifie une des deux conditions suivantes :
 - (a) u est premier et $\exists v \in \mathcal{C}(v), \Delta_v = Y_u \cap N(x)$ et $\forall v_1 \in \mathcal{C}(u) \setminus \{v\}, v_1$ est creux ; ou
 - (b) u est dégénéré et possède un unique fils non creux.

Les résultats qui suivent sont un travail préliminaire en vue de la démonstration du théorème 5.7. Ils précisent la répartition des noeuds propres dans l'arbre et en donne quelques propriétés.

Lemme 5.16 Un noeud propre u a au plus un fils non creux et au plus un fils non propre. De plus, si u a un fils non creux u_1 et un fils non propre u_2 , alors $u_1 = u_2$.

Preuve : Si u vérifie la condition 1 de la définition 5.8, alors u est creux et par conséquent ses fils sont creux et vérifient la condition 1 de la définition. Tous les fils de u sont propres et creux. Traitons le cas où u vérifie la condition 2 de la définition 5.8.

Si u est dégénéré, il suffit de remarquer que les fils creux de u sont propres car ils vérifient la condition 1. Comme u a au plus un fils non creux, le résultat suit.

Si u est premier, d'après la condition 2a de la définition, u a au plus un fils non creux u_1 . Soit $u_2 \in \mathcal{C}(u) \setminus \{u_1\}$. Comme u est propre, $\Delta_{u_1} = Y_u \cap N(x)$. Or d'après le corollaire 5.1, $\Delta_{u_2} \neq \Delta_{u_1}$. D'où $\Delta_{u_2} \neq Y_u \cap N(x)$, et comme u_2 est creux, alors u_2 est propre. ■

Lemme 5.17 Soit $u \in T^c(G)$ qui ne soit pas la racine. Si u est non propre et creux, alors $\text{parent}(u)$ est premier et non creux.

Preuve : Si u est non propre et creux et n'est pas la racine, alors, d'après la définition de propre, $\Delta_u \neq \emptyset$ et $\Delta_u = Y_{parent(u)} \cap N(x)$. Donc $parent(u)$ est premier, et comme $\Delta_u \subseteq N(x)$, $parent(u)$ est non creux. ■

Notation 5.7 Dans la suite, le plus petit ancêtre commun (ppca en abrégé) des noeuds non propres de $T^c(G)$ sera noté w .

Le lemme suivant montre le rôle joué par les noeuds propres et non creux.

Lemme 5.18 Soit G un graphe d'intervalles et x un sommet à insérer dans G . On a les trois propriétés suivantes :

1. w est non propre ; et
2. les noeuds de $Anc(w) \setminus \{w\}$ sont propres et non creux ; et
3. les noeuds de $T^c(G) \setminus (Desc(w) \cup Anc(w))$ sont propres et creux.

Preuve :

1. Pour montrer que w est non propre, il suffit de montrer que pour tous noeuds $u, v \in T^c(G)$, si u et v sont non propres, alors $ppca(u, v)$ est non propre. Le cas où u et v sont liés par la relation de descendance est immédiat. Examinons le cas contraire. D'après le lemme 5.17, si un noeud non propre est creux, alors son père est non creux. Comme les ancêtres d'un noeud non creux sont non creux et comme u et v sont non propres, $ppca(u, v)$ a deux fils distincts dont chacun est non propre ou non creux. D'après le lemme 5.16, $ppca(u, v)$ est non propre.
2. Soit $u \in Anc(w) \setminus \{w\}$, par définition de w , u est propre et comme w est non propre, d'après le lemme 5.17, $parent(w)$ est non creux. Or l'ancêtre d'un noeud non creux est non creux, donc u est propre et non creux.
3. Soit $u \in Anc(w) \setminus \{w\}$ et soit $\tilde{u} \in Anc(w) \cap \mathcal{C}(u)$. D'après ce qui précède, $\tilde{u}$ est non propre ou non creux. Comme u est propre, d'après le lemme 5.16, tous les fils de u différents de $\tilde{u}$ sont creux. Donc, si $v \in T^c \setminus (Desc(w) \cup Anc(w))$ alors v est creux. Et v est propre car $v \notin Desc(w)$. ■

On tire deux corollaires de ce lemme.

Corollaire 5.5 Les noeuds de $Anc(w)$ sont non propres ou non creux.

Corollaire 5.6 $B_w \setminus X_w \subseteq N(x)$

Preuve : Soit $u \in Anc(w) \setminus \{w\}$ et soit v son unique fils dans $\mathcal{C}(u) \cap Anc(w)$. Comme u est propre et non creux (lemme 5.18), alors $X_u \subseteq N(x)$. Si u est dégénéré, $\Delta_v = \emptyset \subseteq N(x)$. Si u est premier, on distingue deux cas.

- Tout d'abord, si u n'est pas le père de w , alors $v \in \text{Anc}(w) \setminus \{w\}$ et d'après le lemme 5.18, v est non creux. De même, u est non creux et propre. Ainsi, d'après la définition de propre, on en déduit que v est le fils de u tel que $\Delta_v = Y_u \cap N(x)$. Donc, $\Delta_v \subseteq N(x)$.
- Dans le cas où $u = \text{parent}(w)$, $v = w$. Si w est non creux, on peut reconduire le même raisonnement que précédemment pour aboutir à $\Delta_v \subseteq N(x)$. Par contre, si w est creux, comme w est propre, différent de la racine et tel que $\Delta_w \neq \emptyset$ (car $\text{parent}(w)$ est premier), alors d'après la définition de propre, il vient que $\Delta_w = Y_u \cap N(x)$. Donc $\Delta_v \subseteq N(x)$, car $v = w$.

Comme pour tout noeud $u \in \text{Anc}(w) \setminus \{w\}$ on a $X_u \cup \Delta_v \subseteq N(x)$, on en déduit que $B_w \setminus X_w \subseteq N(x)$. ■

La propriété qui suit est celle qui fait que les changements de l'arbre T^c sont situés en dessous du noeud w .

Lemme 5.19 $w^* \cup \{x\}$ est un module de G' .

Preuve : D'après le théorème 5.6, w^* est un module de G . Soit $y \in w^*$, alors $e_y \in T_w^c(G)$. Soit $z \in V(G) \setminus w^*$.

Si $z \notin B_w$ alors y et z ne sont pas adjacents car présents dans aucune clique maximale en commun. Comme $z \notin B_w$, $e_z \notin \text{Anc}(w)$ ou e_z est premier et $\Delta_z \cap \text{Anc}(w) = \emptyset$. Si $e_z \notin \text{Anc}(w)$, comme $e_z \notin \text{Desc}(u)$ et comme d'après le lemme 5.18, les noeuds de $T^c(G) \setminus (\text{Desc}(w) \cup \text{Anc}(w))$ sont creux, alors x et z ne sont pas adjacents. Si $e_z \in \text{Anc}(w)$ et e_z est premier et $\Delta_z \cap \text{Anc}(w) = \emptyset$, soit u l'unique élément de $\text{Anc}(w) \cap \mathcal{C}(e_z)$. Comme e_z est propre et comme, d'après le corollaire 5.5, u est non propre ou non creux, alors, d'après la définition d'un noeud propre et le lemme 5.16, $Y_{e_z} \cap N(x) = \Delta_u$. Comme $z \notin \Delta_u$, alors $z \notin N(x)$.

Si $z \in B_w$, alors $\forall K \in \mathcal{K}_{T^c(G)}(w), z \in K$ donc z est adjacent à y . Or, d'après le corollaire 5.6, z est adjacent à x .

Donc, $w^* \cup \{x\}$ est un module de G' . ■

On est maintenant en mesure d'établir le théorème recherché : lors d'une insertion, il suffit de considérer le sous-arbre T_w^c , le reste de la PQ -représentation ne change pas.

Théorème 5.7 Soit G un graphe d'intervalles. Soit w le ppca des noeuds non propres de $T^c(G)$. $G' = G + x$ est un graphe d'intervalles ssi $G[w^*] + x$ est un graphe d'intervalles. De plus, si G' est un graphe d'intervalles, alors $\mathcal{PQ}(G')$ s'obtient à partir de $\mathcal{PQ}(G)$ en remplaçant le sous-arbre T_w^c de $T^c(G)$ par $T^c(G[w^*] + x)$, et en assignant aux sommets de $w^* \cup \{x\}$ leurs pointeurs vers $T^c(G[w^*] + x)$.

Preuve : Si G' est un graphe d'intervalles, alors $G'[w^* \cup \{x\}]$ l'est aussi par hérédité de la classe.

Réciproquement, si $G'[w^* \cup \{x\}]$ est un graphe d'intervalles, d'après le lemme 5.19 $w^* \cup \{x\}$ est un module de G' . Soit a un sommet représentatif de $w^* \cup \{x\}$ dans $G'/\{w^* \cup \{x\}\}$.

$G' = G'/\{w^* \cup \{x\}\}_{a \leftarrow G'[w^* \cup \{x\}]}$. Comme a peut être choisi dans w^* , $G'/\{w^* \cup \{x\}\} = G/\{w^*\}$.

Comme w^* est un module de G qui est triangulé, alors, d'après le théorème 5.1, a est simplicial dans $G/\{w^*\}$. Ainsi, d'après le théorème 5.2, $G' = G/\{w^*\}_{w \leftarrow G'[w^* \cup \{x\}]}$ est un graphe d'intervalles et son PQ -arbre est donné par le théorème 5.4. Pour achever la preuve du lemme, il convient de remarquer que le cas 1.b du théorème 5.4 ne se produit pas ici. En effet, d'après le lemme 5.13, le PQ -arbre de $G/\{w^*\}$ est celui de G dans lequel on remplace w par une feuille. Comme w est non propre, d'après le lemme 5.17, w est non creux ou son père est premier. Dans le cas où w est non creux, si $G'[w^* \cup \{x\}]$ est non connexe, alors $G[w^*]$ est non connexe. Ce qui implique que w est dégénéré et $X_w = \emptyset$. D'après le lemme 5.4, $\text{parent}(w)$ est nécessairement premier. Finalement, dans tous les cas $\text{parent}(w)$ est premier, ce qui exclut le cas 1.b du théorème 5.4. ■

En conséquence, le problème d'insérer x dans G se réduit à celui d'insérer x dans $G[w^*]$, c'est à dire de déterminer si $G[w^*] + x$ est un graphe d'intervalles et de calculer $PQ(G[w^*] + x)$ si tel est le cas. C'est la raison pour laquelle w est appelé le **noeud d'insertion**.

5.5.3 Caractérisation dynamique des graphes d'intervalles

Afin de pouvoir démontrer le théorème qui caractérise les cas où l'insertion de x résulte en un graphe d'intervalles (théorème 5.8), on établit une série de lemmes techniques qui s'intéresse à savoir quels sont les intervalles communs à tous les ordres consécutifs qui ont été conservés lors de l'ajout du sommet x .

Les notations qui suivent alourdissent un peu la mémoire mais allègeront beaucoup les preuves.

Notation 5.8 Pour un noeud u de $T^c(G)$, on note $\mathcal{K}'_T(u) = \{K' \in \mathcal{K}(G') \mid K' \setminus \{x\} \in \mathcal{K}_{T^c(G)}(u)\}$. Pour un sous-ensemble de sommets $S \subseteq V(G')$, on note $\mathcal{K}'(S) = \mathcal{K}_{G'}(S)$. Lorsque S est réduit à un singleton $\{x\}$, on notera $\mathcal{K}'(x) = \mathcal{K}'(\{x\})$.

Lemme 5.20 Soit v un noeud non désert de $T^c_w(G) \setminus \{w\}$, $\mathcal{K}'(v^*)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G')$.

Preuve :

Soit σ' un ordre consécutif de $\mathcal{K}(G')$. En enlevant x dans toutes les cliques de σ' , on obtient un nouvel ordre σ'' dans lequel la propriété de consécutivité est conservé. σ'' est composé de toutes les cliques maximales de G , plus éventuellement d'autres cliques de G qui ne sont pas maximales. D'après le lemme 5.14 (et le corollaire 5.3), le nombre de cliques non maximales de G contenues dans σ'' est au plus deux. En les enlevant, on obtient un ordre consécutif σ de $\mathcal{K}(G)$ dans lequel $\mathcal{K}_G(v^*)$ est un intervalle. Soit K_d une clique non maximale de G présente dans σ'' . Pour montrer que $\mathcal{K}'(v^*)$ est un intervalle de σ' , il suffit de montrer que si, dans σ'' , K_d est entre deux cliques qui n'appartiennent pas à $\mathcal{K}_G(v^*)$ alors $K_d \cap v^* = \emptyset$, et que si K_d est entre deux cliques de $\mathcal{K}_G(v^*)$ alors $K_d \cap v^* \neq \emptyset$.

- Si K_d était entre deux cliques qui n'appartiennent pas à $\mathcal{K}_G(v^*)$, comme, d'après le lemme 5.14, K_d est incluse dans une de ses voisines dans σ'' qui est maximale dans G et comme les cliques de $\mathcal{K}_G(v^*)$ sont les seules à contenir des sommets de v^* , alors K_d ne contient aucun sommet de v^* .
- Considérons le cas où K_d était entre deux cliques de $\mathcal{K}(v^*)$. On veut montrer que $K_d \cap v^* \neq \emptyset$.

Supposons que K_d ne contient aucun sommet de v^* . Alors $\mathcal{K}(G'[v^*])$ n'est pas un intervalle de σ' . D'où, par contraposée du lemme 5.9, $G'[v^*] = G[v^*]$ n'est pas connexe. Comme $\forall K \in \mathcal{K}_G(v^*), K \subseteq B_v \cup v^*$, et comme K_d est incluse dans une de ses voisines dans σ'' (qui sont dans $\mathcal{K}_G(v^*)$), alors $K_d \subseteq B_v$. Comme $\forall K \in \mathcal{K}_G(v^*), B_v \subseteq K$ et comme σ'' a la propriété de consécutivité, $K_d = B_v$. Cela implique que K_d est incluse à la fois dans sa voisine de gauche et de droite, dans σ'' , et donc qu'aucune de ses deux voisines ne contenait x dans σ' . Donc $K' = B_v \cup \{x\}$ est l'unique clique maximale de G' contenant x . Autrement dit, $N(x) = B_v$.

Montrons que v est le *ppca* des noeuds non propres.

- Soit $u \in \text{Anc}(v) \setminus \{v\}$, montrons que u est propre. Comme $B_v = N(x)$ et d'après le lemme 5.4, u est non creux. Soit $\tilde{v} \in \text{Anc}(v) \cap \mathcal{C}(u)$, alors, comme $N(x) = B_v$, $\forall u_1 \in \mathcal{C}(u) \setminus \{\tilde{v}\}$, u_1 est creux. De plus si u est premier, comme $N(x) = B_v$, alors $\Delta_{\tilde{v}} = Y_u \cap N(x)$ et u est propre. Si u est dégénéré alors, d'après le lemme 5.4, $\tilde{v}$ est non creux, donc u est propre.
- Soit $u \in \text{Anc}(v) \setminus \{v\}$ et soit $u_1 \in \mathcal{C}(u) \setminus \{\tilde{v}\}$. On a montré ci-dessus que u_1 est creux. Si u est premier, comme $\Delta_{\tilde{v}} = Y_u \cap N(x)$, alors, d'après le corollaire 5.1, $\Delta_{u_1} \neq Y_u \cap N(x)$ et donc u_1 est propre. Si u est dégénéré, $\Delta_{u_1} = \emptyset$ et u_1 est creux, donc u_1 est propre. Comme les descendants stricts d'un noeud creux sont propres (ils vérifient la condition 1 de la définition 5.8), alors tous les noeuds de $T^c(G) \setminus (\text{Anc}(v) \cup \text{Desc}(v))$ sont propres.
- Reste à montrer que v est non propre. Comme v est dégénéré et comme $B_v = N(x)$, si $\Delta_v = \emptyset$, c'est à dire si $\text{parent}(v)$ est dégénéré, alors, d'après le lemme 5.4, v est non creux. De plus, si $\Delta_v \neq \emptyset$ alors $\Delta_v = Y_{\text{parent}(v)} \cap N(x)$. Donc v ne vérifie pas la condition 1 de la définition 5.8. Comme $B_v = N(x)$, v n'a que des fils creux. Donc v ne vérifie pas la condition 2a de la définition 5.8 : v est non propre. Finalement, $v = w$: absurde. Donc $K_d \cap v^* \neq \emptyset$.

Cela achève la preuve du lemme. ■

Corollaire 5.7 *Soit $v \in T_w^c$, $\forall u \in \mathcal{C}(v)$, si $u^* \neq \emptyset$ alors $\mathcal{K}'(u^*)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G')$. De plus, dans tout ordre consécutif de $\mathcal{K}(G')$, les intervalles $(\mathcal{K}'(u^*))_{u \in \mathcal{C}(v) \text{ et } u^* \neq \emptyset}$ sont deux à deux disjoints et ont le même ordre relatif, à renversement près, que les fils de v auxquels ils correspondent.*

Preuve : La première partie du corollaire découle directement du lemme 5.20. Le fait que les $\mathcal{K}'(u^*)$, pour $u \in \mathcal{C}(v)$ et, soient deux à deux disjoints provient du fait que $\forall u_1, u_2 \in \mathcal{C}_G(v)$, si u_1 et u_2 sont non déserts alors $\forall (y_1, y_2) \in u_1^* \times u_2^*, y_1$ et y_2 ne sont

pas adjacent. y_1 et y_2 ne peuvent donc pas être présents dans une même clique maximale de G' . Pour montrer que l'ordre relatif des intervalles $(\mathcal{K}'(u))_{u \in \mathcal{C}(v) \text{ et } u^* \neq \emptyset}$ est le même que celui des fils de v auxquels ils correspondent, considérons un ordre consécutif σ' de $\mathcal{K}(G')$. Faisons la même manipulation que dans la preuve du lemme 5.20 : on enlève x dans toutes les cliques et on retire les éventuelles cliques non maximales de G apparues pour obtenir un ordre consécutif σ de $\mathcal{K}(G)$. Dans σ , les $\mathcal{K}(u^*)$, pour $u \in \mathcal{C}(v)$ et $u^* \neq \emptyset$, sont des intervalles et apparaissent dans l'ordre relatif des fils de v . Comme les cliques de $\mathcal{K}(u^*)$ présentes dans σ proviennent de cliques de $\mathcal{K}'(u^*)$ dans σ' , cela prouve le corollaire. ■

Lemme 5.21 *Soit v un noeud premier de $T_w^c(G)$ et σ' un ordre consécutif de $\mathcal{K}(G')$. Quitte à renverser σ' ,*

$$\forall u_1, u_2 \in \mathcal{C}(v), u_1 <_{\sigma_v} u_2 \Rightarrow \forall (K_1, K_2) \in \mathcal{K}'_T(u_1) \times \mathcal{K}'_T(u_2), K_1 <_{\sigma'} K_2$$

Preuve : En retirant x des cliques de σ' et en supprimant les cliques non maximales de G obtenues (au plus deux), on obtient un ordre consécutif σ de $\mathcal{K}(G)$. Lors de cette transformation, aucune clique de $\mathcal{K}'_T(u)$, où $u \in \mathcal{C}(v)$, n'est supprimée et ces cliques deviennent les cliques de $\mathcal{K}_{T^c(G)}(u)$. Dans σ , les ensembles $\mathcal{K}_{T^c(G)}(u)$, pour $u \in \mathcal{C}(v)$, sont des intervalles et sont dans le même ordre relatif, à renversement près, que les fils de v auxquels ils correspondent. Cela prouve le lemme. ■

Lemme 5.22 *Soit v un noeud premier de $T_w^c(G)$ et σ' un ordre consécutif de $\mathcal{K}(G')$. Quitte à renverser σ' ,*

$$\forall u_1, u_2 \in \mathcal{C}(v), u_1 <_{\sigma_v} u_2 \Rightarrow \forall (K_1, K_2) \in \mathcal{K}'_T(u_1) \cup \mathcal{K}'(u_1^*) \times \mathcal{K}'_T(u_2) \cup \mathcal{K}'(u_2^*), K_1 <_{\sigma'} K_2$$

Preuve : Si $\forall i \in \{1, 2\}, u_i^* = \emptyset$, alors $\forall i \in \{1, 2\}, \mathcal{K}'_T(u_i) \cup \mathcal{K}'(u_i^*) = \mathcal{K}'_T(u_i)$ et le lemme 5.21 conclut. Si $\forall i \in \{1, 2\}, u_i^* \neq \emptyset$, alors $\forall i \in \{1, 2\}, \mathcal{K}'_T(u_i) \cup \mathcal{K}'(u_i^*) = \mathcal{K}'(u_i^*)$ et le corollaire 5.7 conclut. Enfin, si $\exists i \in \{1, 2\}, u_i^* \neq \emptyset$ et $\exists j \in \{1, 2\}, u_j^* = \emptyset$, alors $\mathcal{K}'_T(u_i) \cup \mathcal{K}'(u_i^*) = \mathcal{K}'(u_i^*)$ et $\mathcal{K}'_T(u_j) \cup \mathcal{K}'(u_j^*) = \mathcal{K}'_T(u_j)$. D'après le lemme 5.20, $\mathcal{K}'(u_i^*)$ est un intervalle de σ' . De plus, les cliques de $\mathcal{K}'_T(u_j)$ ne contiennent aucun sommet de u_i^* et il s'ensuit que $\mathcal{K}'_T(u_j) \cap \mathcal{K}'(u_i^*) = \emptyset$. Comme $\mathcal{K}'_T(u_i) \subseteq \mathcal{K}'(u_i^*)$ et comme d'après le lemme 5.21, toutes les cliques de $\mathcal{K}'_T(u_1)$ se trouvent avant toutes les cliques de $\mathcal{K}'_T(u_2)$ dans σ' , alors $K_1 <_{\sigma'} K_2$. ■

Les définitions et notations suivantes préparent l'énoncé du théorème qui suit.

Définition 5.9 *Soit G un graphe d'intervalles, u un noeud premier de $T^c(G)$ et v un fils de u . v **satisfait la propriété gauche (resp. droite)** ssi pour tout $y \in Y_u$, si y est lié et $e_y^2 \leq v$ (resp. $e_y^1 \geq v$), alors $e_y^2 = v$ (resp. $e_y^1 = v$). On dit que v **satisfait la propriété gauche (resp. droite) stricte** ssi v satisfait la propriété gauche (resp. droite) et $\exists y \in Y_u$ lié tel que $e_y^2 = v$ (resp. $e_y^1 = v$).*

Définition 5.10 *Un noeud $u \in T^c(G)$ est **saturé** ssi u est plein et B_u est pleine.*

Notation 5.9 *Soit u un noeud premier de $T^c(G)$. On note $S_\Delta(u) = \{v_1 \in \mathcal{C}(u) \mid Y_u \cap N(x) \subseteq \Delta_{v_1}\}$. Si les fils saturés de u forment un intervalle de σ_u , on note I_u cet intervalle, et on note f_u et l_u , si ces noeuds existent, respectivement le noeud précédent et le noeud suivant I_u dans σ_u .*

On est maintenant en mesure d'établir le théorème de caractérisation des insertions réussies, qui est le but de cette partie.

Théorème 5.8 *Soit $G = (V, E)$ un graphe d'intervalles et $x \notin V$ un sommet à insérer dans G . Soit w le noeud d'insertion. $G' = G + x$ est un graphe d'intervalles ssi les quatre conditions suivantes sont vérifiées :*

1. *Tout noeud $u \in T_w^c \setminus \{w\}$ a au plus un fils mixte, et w a au plus deux fils mixtes. De plus, pour tout noeud $u \in T_w^c$, si B_u n'est pas pleine alors u a au plus un fils non creux.*
2. *Pour tout noeud premier $u \in T_w^c$, l'ensemble des fils saturés de u est un intervalle I_u de σ_u . De plus, si $I_u \neq \emptyset$, alors tout noeud $v_1 \in \mathcal{C}(u) \setminus (I_u \cup \{f_u, l_u\})$ est creux.*
3. *Si w est un noeud premier, alors il vérifie une des conditions suivantes :*
 - (a) *B_w n'est pas pleine ; et à inversion près de σ_w , le dernier élément v de σ_w satisfait la propriété gauche et les noeuds appartenant à $\mathcal{C}(w) \setminus \{v\}$ sont creux.*
 - (b) *B_w est pleine et $I_w \neq \emptyset$; et f_w et l_w satisfont respectivement la propriété gauche et la propriété droite.*
 - (c) *B_w est pleine et $I_w = \emptyset$; et, une des deux sous-conditions suivantes est vérifiée :*
 - i. *à inversion près de σ_w , le dernier élément v de σ_w satisfait la propriété gauche et les noeuds appartenant à $\mathcal{C}(w) \setminus \{v\}$ sont creux, ou*
 - ii. *il existe deux éléments consécutifs f et l de σ_w , avec $f <_{\sigma_w} l$, qui satisfont respectivement la propriété gauche et la propriété droite, et les noeuds appartenant à $\mathcal{C}(w) \setminus \{f, l\}$ sont creux, et aucun sommet non lié de Y_w ne couvre à la fois f et l .*
4. *Tout noeud premier $u \neq w$ de T_w^c vérifie une des conditions suivantes :*
 - (a) *B_u n'est pas pleine, ou B_u est pleine et $I_u = \emptyset$; et, à inversion près de σ_u , le dernier élément v de σ_u satisfait la propriété gauche et les noeuds appartenant à $\mathcal{C}(u) \setminus \{v\}$ sont creux.*
 - (b) *B_u est pleine et $I_u \neq \emptyset$; et, à inversion près de σ_u , le dernier élément de σ_u appartient à I_u et f_u satisfait la propriété gauche.*

Preuve :

$\implies$. **Conditions nécessaires.**

On commence par établir deux propriétés dont nous ferons usage dans la preuve.

Proposition 5.10 Soit $u \in T_w^c \setminus \{w\}$. Si B_u est pleine et u est non creux, alors il existe une clique maximale de G' contenant x qui n'appartient pas à $\mathcal{K}'(u^*)$.

Preuve :

Soit $\tilde{u} \in \text{Anc}(u) \cap \mathcal{C}(w)$. $X_w \subseteq N(x)$, $\tilde{u}$ est non creux et $\Delta_{\tilde{u}} \subseteq N(x)$. D'où, comme w n'est pas propre, il existe $y \in (Y_w \setminus \Delta_{\tilde{u}}) \cup \bigcup_{v \in \mathcal{C}(\tilde{u}) \setminus \{\tilde{u}\}} v^*$ tel que y est adjacent à x . Il existe donc une clique maximale K_{xy} de G' contenant x et y . Comme $y \notin B_u \cup u^*$, alors les cliques maximales de G' contenant y n'appartiennent pas à $\mathcal{K}'(u^*)$, et par conséquent $K_{xy} \notin \mathcal{K}'(u^*)$. $\square$

Proposition 5.11 Soit $u \in T_w^c$. Si B_u n'est pas pleine et u est non creux, alors $\mathcal{K}'_T(u)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G')$ et il existe une clique maximale de G' contenant x qui n'appartient pas à $\mathcal{K}'_T(u)$.

Preuve : Comme u est non creux, $\mathcal{K}'(u^*)$ contient une clique contenant x , et comme B_u n'est pas pleine, cette clique n'appartient pas à $\mathcal{K}'_T(u)$, car $\forall K \in \mathcal{K}'_T(u), B_u \subseteq K$.

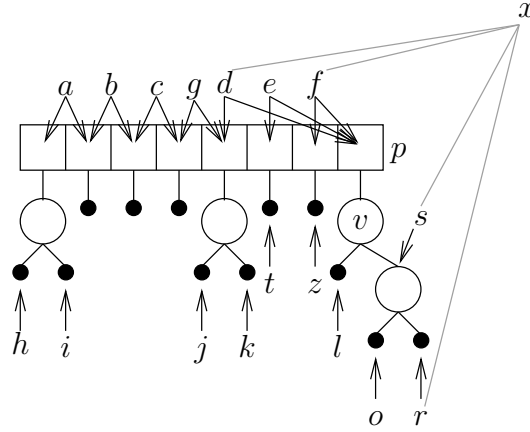
Soit σ' un ordre consécutif de $\mathcal{K}'(G)$. Pour montrer que $\mathcal{K}'_T(u)$ est un intervalle de σ' , supposons qu'il existe $K_1, K_2 \in \mathcal{K}'_T(u)$ et $K \notin \mathcal{K}'_T(u)$ telles que $K_1 <_{\sigma'} K <_{\sigma'} K_2$. En retirant x des cliques de σ' et en supprimant les cliques non maximales de G obtenues (au plus deux), on obtient un ordre consécutif σ de $\mathcal{K}(G)$. Lors de cette transformation, aucune clique de $\mathcal{K}'_T(u)$ n'est supprimée et ces cliques deviennent les cliques de $\mathcal{K}_{T^c(G)}(u)$. Or, dans σ les cliques de $\mathcal{K}_{T^c(G)}(u)$ forment un intervalle. On en déduit que K est nécessairement une clique maximale additionnelle et que $x \in K$. Comme $B_u \subseteq K_1 \cap K_2$ et comme σ' est un ordre consécutif, alors $B_u \subseteq K$. Donc B_u est pleine : absurde. $\mathcal{K}'_T(u)$ est un intervalle de σ' . $\square$

– Condition 1.

Supposons que w a au moins trois fils mixtes v_1, v_2, v_3 . Soit $v_i \in \{v_1, v_2, v_3\}$. Comme v_i est mixte, $\mathcal{K}'(v_i^*)$ contient au moins une clique contenant x et au moins une clique ne contenant pas x . Or, d'après le corollaire 5.7, $\mathcal{K}'(v_i^*)$ est un intervalle de $\mathcal{K}(G')$ disjoints des autres $\mathcal{K}(v_j^*)$, pour $v_j \in \{v_1, v_2, v_3\} \setminus \{v_i\}$. Donc, les cliques contenant x ne forment pas un intervalle de $\mathcal{K}(G')$: absurde.

Si B_u n'est pas pleine, supposons que u a au moins deux fils non creux u_1 et u_2 . $\exists y \in u_1^*$ et $\exists z \in u_2^*$ tels que y et z sont adjacents à x . Par définition, y et z ne sont pas adjacents entre eux. Comme B_u n'est pas pleine, $\exists x' \in B_u$ tel que x' n'est pas adjacent à x . Or x' est adjacent à y et z , d'où $xyzx'$ induit un C_4 dans G' : absurde. Donc si B_u n'est pas pleine, u a au plus un fils non creux.

Soit $u \neq w$ un noeud de T_w^c tel que B_u est pleine. Supposons que u ait au moins deux fils mixtes u_1 et u_2 . D'après la proposition 5.10, il existe une clique maximale K_{xy} de G' contenant x et n'appartenant pas à $\mathcal{K}'(u)$. D'après le lemme 5.20, $\mathcal{K}'(u^*)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G')$. Comme u_1 est mixte, $\mathcal{K}'(u_1^*)$ contient au moins une clique contenant x , et au moins une ne le contenant pas. De même

FIG. 5.22 – Le noeud p vérifie les conditions 4a, 3a et 3c du théorème 5.8.

pour $\mathcal{K}'(u_2^*)$. De plus $\mathcal{K}'(u_1^*) \subseteq \mathcal{K}'(u^*)$ et $\mathcal{K}'(u_2^*) \subseteq \mathcal{K}'(u^*)$. Comme $K_{xy} \notin \mathcal{K}'(u^*)$, dans n'importe quel ordre consécutif de $\mathcal{K}(G')$, les cliques contenant x ne peuvent pas former un intervalle : absurde. Donc u a au plus un fils mixte.

– Condition 2.

Si u ne possède aucun fils saturé, alors le résultat tient (l'ensemble vide est un intervalle) et $I_u = \emptyset$. Le cas intéressant est celui où u possède au moins un fils saturé. Dans ce cas, B_u est nécessairement pleine. On note p_1, p_2 respectivement le premier et dernier noeud de I_u dans σ_u (éventuellement $p_1 = p_2$).

Soit $v \in \mathcal{C}(u)$. Supposons $p_1 <_{\sigma_u} v <_{\sigma_u} p_2$ et v n'est pas saturé. Comme v n'est pas saturé, il existe une clique $K_v \in \mathcal{K}_{T^c(G)}(v)$ qui est maximale dans G' . Soit σ' un ordre consécutif de $\mathcal{K}(G')$. D'après le lemme 5.21, dans σ' , K_v est entre les cliques de $\mathcal{K}'_T(p_1)$ et celles de $\mathcal{K}'_T(p_2)$. $x \notin K_v$ et toutes les cliques de $\mathcal{K}'_T(p_1)$ et de $\mathcal{K}'_T(p_2)$ contiennent x . $\mathcal{K}'(x)$ n'est pas un intervalle de $\mathcal{K}(G')$: absurde. Donc, l'ensemble des fils saturés de u est un intervalle.

Montrons que si $I_u \neq \emptyset$, alors tout noeud $v_1 \in \mathcal{C}(u) \setminus (I_u \cup \{f_u, l_u\})$ est creux. Si au moins deux noeuds précèdent p_1 dans σ_u , alors soit $v_1 \in \mathcal{C}(u)$ tel que $v_1 <_{\sigma_u} f_u$. Supposons que v_1 est non creux, alors il existe une clique $K_x \in \mathcal{K}'(v_1^*)$ qui contient x . Comme f_u n'est pas saturé, il existe une clique $K \in \mathcal{K}_{T^c(G)}(f_u)$ qui est maximale dans G' . D'après le lemme 5.21, dans σ' , K est antérieur aux cliques de $\mathcal{K}'_T(p_1)$. D'après le lemme 5.22, K_x est antérieur à K dans σ' .

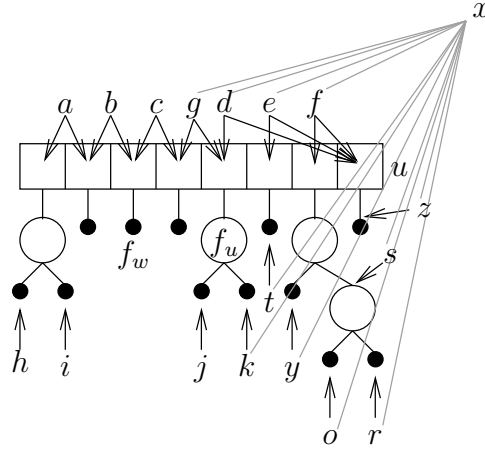
Donc, dans σ' , K est entre K_x et les cliques de $\mathcal{K}'_T(p_1)$. Or $x \notin K$ et toutes les cliques de $\mathcal{K}'_T(p_1)$ contiennent x . $\mathcal{K}'(x)$ n'est pas un intervalle de $\mathcal{K}(G')$: absurde. Donc v_1 est creux.

Le même raisonnement sur l_u termine la preuve.

– Condition 4.

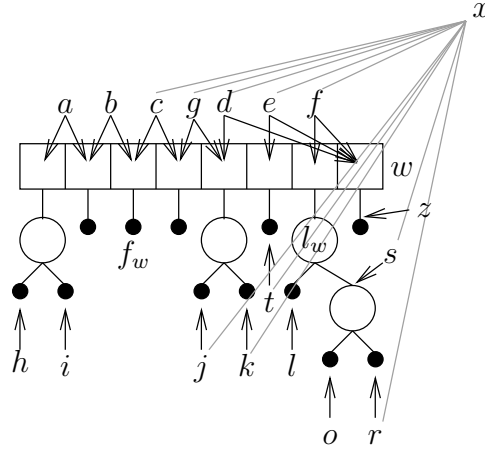
– Condition 4a (voir figure 5.22) : B_u n'est pas pleine, ou B_u est pleine et $I_u = \emptyset$.

Dans ce cas, aucun fils de u n'est saturé. Si u est creux, il satisfait la condition 4a. Examinons le cas où u est non creux.

FIG. 5.23 – Le noeud u vérifie les conditions 4b du théorème 5.8.

- Si B_u est pleine, comme u est non creux, d'après la proposition 5.10, il existe $K' \in \mathcal{K}(G')$ telle que $x \in K'$ et $K' \notin \mathcal{K}'(u^*)$. Soit σ' un ordre consécutif de $\mathcal{K}(G')$ dans lequel K' se trouve après $\mathcal{K}'(u^*)$.
Soit v le fils de u qui possède la clique de $\bigcup_{u_1 \in \mathcal{C}(u)} \mathcal{K}'_T(u_1)$ la plus proche de K' dans σ' . D'après le lemme 5.21, v est le dernier fils de u dans σ_u . Comme v n'est pas saturé (aucun fils de u ne l'est), il existe une clique de $\mathcal{K}_{T^c(G)}(v)$ qui est une clique maximale de G' . C'est à dire qu'il existe une clique K_v de $\mathcal{K}'_T(v)$ qui ne contient pas x .
Supposons qu'il existe $u_1 \in \mathcal{C}(u) \setminus \{v\}$ qui soit non creux. Alors, il existe une clique K_1 de $\mathcal{K}'(u_1^*)$ qui contient x . Or D'après le lemme 5.22, K_1 se trouve avant K_v dans σ' . D'où, $K_1 <_{\sigma'} K_v <_{\sigma'} K'$ et $x \in K_1 \cap K'$ et $x \notin K_v$: absurde. Donc tous les fils de u différent de v sont creux.
Montrons que v satisfait la propriété gauche. Supposons qu'il existe $y \in Y_u$ tel que y est lié et ne couvre pas v . Alors, il existe une clique K_{xy} de $\mathcal{K}'(u^*)$ qui contient x et y . Comme y ne couvre pas v , les cliques de $\mathcal{K}'_T(v)$ ne contiennent pas y . D'après le lemme 5.21, les cliques de $\mathcal{K}(G')$ contenant y se trouvent avant les cliques de $\mathcal{K}'_T(v)$ dans σ' . Ainsi, $K_{xy} <_{\sigma'} K_v <_{\sigma'} K'$ et $x \in K_{xy} \cap K'$ et $x \notin K_v$: absurde. Donc, v satisfait la propriété gauche.
- Si B_u n'est pas pleine, comme u est non creux, d'après la proposition 5.11, $\mathcal{K}'_T(u)$ est un intervalle de tout ordre consécutif de $\mathcal{K}(G')$ et il existe $K' \in \mathcal{K}(G')$ telle que $x \in K'$ et $K' \notin \mathcal{K}'_T(u)$. En considérant un ordre consécutif σ' de $\mathcal{K}(G')$ dans lequel K' se trouve après $\mathcal{K}'_T(u)$ et le fils v de u qui possède la clique de $\bigcup_{u_1 \in \mathcal{C}(u)} \mathcal{K}'_T(u_1)$ la plus proche de K' dans σ' , on reconduit les mêmes raisonnements que précédemment pour arriver aux mêmes conclusions.
- Condition 4b (voir figure 5.24) : B_u est pleine et $I_u \neq \emptyset$.

Comme u est premier et $I_u \neq \emptyset$, alors u est non creux, et d'après la proposition 5.10, il existe $K' \in \mathcal{K}(G')$ telle que $x \in K'$ et $K' \notin \mathcal{K}'(u^*)$. Soit σ' un

FIG. 5.24 – Le noeud w vérifie les conditions 3b du théorème 5.8.

ordre consécutif de $\mathcal{K}(G')$ dans lequel K' se trouve après $\mathcal{K}'(u^*)$. Quitte à renverser σ_u , on peut supposer que $\sigma_u = \Pi_u^{\sigma'}$. Soit $v \in I_u$. Comme v est saturé, $\mathcal{K}'_T(v)$ contient une clique K_x contenant x . On note l le dernier élément de σ_u . Supposons $l \notin I_u$. D'après le lemme 5.21, K_x précède les cliques de $\mathcal{K}'_T(l)$ dans σ' . Le fait que K' se trouve après $\mathcal{K}'_T(l)$ dans σ' implique donc que toutes les cliques de $\mathcal{K}'_T(l)$ contiennent x : l est saturé.

Comme f_u n'est pas saturé, il existe une clique $K_f \in \mathcal{K}_{T^c(G)}(f_u)$ qui est maximale dans G' . Pour montrer que f_u satisfait la propriété gauche, supposons qu'il existe $y \in Y_u$ lié tel que $e_y^2 < f_u$. Alors, il existe une clique K_{xy} de $\mathcal{K}'(u^*)$ qui contient x et y . Comme y ne couvre pas f_u , les cliques de $\mathcal{K}'_T(f_u)$ ne contiennent pas y . D'après le lemme 5.21, les cliques de $\mathcal{K}(G')$ contenant y se trouvent avant les cliques de $\mathcal{K}'_T(f_u)$ dans σ' . Ainsi, $K_{xy} <_{\sigma'} K_f <_{\sigma'} K'$ et $x \in K_{xy} \cap K'$ et $x \notin K_v$: absurde. Donc, f_u satisfait la propriété gauche.

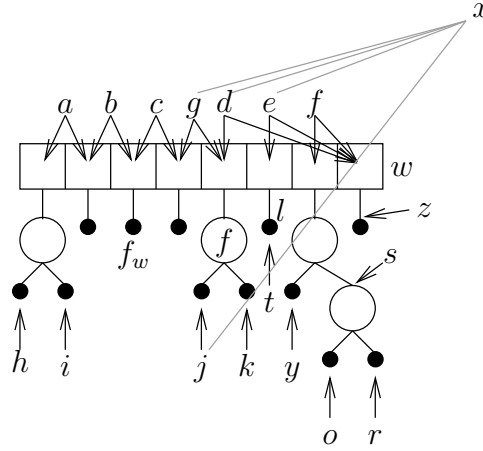
– Condition 3.

– Condition 3a (voir figure 5.22)

La démonstration du cas où B_u est non pleine dans la condition 4a s'applique ici et montre que w satisfait la condition 3a.

– Condition 3b (voir figure 5.24)

On montre que f_w satisfait la propriété gauche, cela implique, en renversant σ_u , que l_w satisfait la propriété droite. Soit $v \in I_w$. Comme v est saturé, il existe $K_v \in \mathcal{K}_{T^c(G)}(v)$ qui contient x . Comme f_w n'est pas saturé, il existe une clique $K_f \in \mathcal{K}_{T^c(G)}(f_w)$ qui est maximale dans G' . Pour montrer que f_w satisfait la propriété gauche, supposons qu'il existe $y \in Y_u$ lié tel que $e_y^2 < f_w$. Alors, il existe une clique K_{xy} de $\mathcal{K}'(u^*)$ qui contient x et y . Comme y ne couvre pas f_w , les cliques de $\mathcal{K}'_T(f_w)$ ne contiennent pas y . D'après le lemme 5.21, les cliques de $\mathcal{K}(G')$ contenant y se trouvent avant les cliques de $\mathcal{K}'_T(f_w)$ dans σ' , et les cliques de $\mathcal{K}'_T(v)$ se trouvent après toutes celles-ci. Ainsi, $K_{xy} <_{\sigma'} K_f <_{\sigma'} K'$ et $x \in K_{xy} \cap K'$ et $x \notin K_v$: absurde. Donc, f_w satisfait la propriété gauche.

FIG. 5.25 – Le noeud w vérifie la condition 3(c)ii du théorème 5.8.

- Condition 3c. Un exemple de noeud w vérifiant la condition 3(c)i et un exemple vérifiant la condition 3(c)ii sont donnés sur les figures 5.22 et 5.25.

Proposition 5.12 Soit u un noeud premier de $T^c(G)$ et v un fils non saturé de u , alors il existe une clique K' de $\mathcal{K}'_T(v)$ qui ne contient pas x .

Preuve : Les cliques de $\mathcal{K}_{T^c}(v)$ sont maximales dans G' . □

- Si w a deux fils mixtes v_1 et v_2 , avec $v_1 <_{\sigma_w} v_2$. Alors, il existe $K_1 \in \mathcal{K}'(v_1^*)$ et $K_2 \in \mathcal{K}'(v_2^*)$ telles que $x \in K_1 \cap K_2$. Supposons que v_1 et v_2 sont non consécutifs dans σ_u . Alors il existe un fils v de u non saturé tel que $v_1 <_{\sigma_w} v <_{\sigma_w} v_2$. D'après la proposition 5.12, il existe une clique K de $\mathcal{K}'_T(v)$ qui ne contient pas x . D'après le lemme 5.22, $K_1 <_{\sigma_w} K <_{\sigma_w} K_2$: absurde. v_1 et v_2 sont consécutifs.

Comme v_2 est mixte, il existe une clique de $\mathcal{K}'(v_2^*)$ qui contient x . Comme v_1 est mixte, on montre, comme on a montré que f_w satisfait la propriété gauche pour la condition 3b, que v_1 satisfait la propriété gauche. En renversant σ_w , on obtient que v_2 satisfait la propriété droite.

Comme v_1 et v_2 sont non creux, aucun sommet non lié de Y_w ne couvre à la fois v_1 et v_2 , sinon il y aurait un C_4 dans G' .

- Si w a un unique fils mixte v , soit σ' un ordre consécutif de G' . Quitte à renverser σ_w on peut supposer que $\sigma_w = \Pi_w^{\sigma'}$. Il existe une clique $K_v \in \mathcal{K}'_T(v)$ qui ne contient pas x . Pour montrer que v satisfait la propriété gauche ou droite, supposons qu'il n'en satisfasse aucune des deux. Alors il existe $y, z \in Y_u$ liés tels que $e_y^2 <_{\sigma_w} v <_{\sigma_w} e_z^1$. Il existe dans G' une clique maximale K_{xy} contenant x et y et une clique maximale K_{xz} contenant x et z . D'après le lemme 5.21, dans σ' , les cliques contenant y précèdent K_v et celles contenant z suivent K_v . On conclut à une absurdité. v satisfait la propriété gauche ou la propriété droite. Quitte à renverser σ_w , on peut considérer que v satisfait la propriété droite.

Si v ne satisfait pas la propriété gauche, alors il a un prédécesseur v_1 dans σ_w . Montrons que v_1 satisfait la propriété gauche. Comme v est mixte, il existe une

clique $K'_v \in \mathcal{K}'(v^*)$ qui contient x , et comme v_1 est non saturé, il existe une clique $K_1 \in \mathcal{K}'_T(v_1)$ qui ne contient pas x . De manière très similaire à la preuve de la condition 3b, on montre que v_1 satisfait la propriété gauche. Comme v ne la satisfait pas, alors il existe $z_1 \in Y_w$ lié tel que $e_{z_1}^2 = v_1$. Soit $z_2 \in v^*$ lié. Supposons qu'il existe $y \in Y_w$ qui couvre à la fois v_1 et v . Alors xz_1z_2y est un C_4 : absurde. Donc, $\Delta_{v_1} \cap \Delta_v \subseteq N(x)$.

Si v satisfait aussi la propriété gauche, alors, si v est le premier ou le dernier élément de σ_w , w satisfait la condition 3(c)i. Sinon, v a un prédécesseur v_1 et un successeur v_2 dans σ_w . v_1 satisfait nécessairement la propriété gauche car il est à gauche de v qui la satisfait. De même, v_2 satisfait nécessairement la propriété droite. Ainsi, pour montrer que w vérifie la condition 3(c)ii, il suffit de montrer que $\Delta_{v_1} \cap \Delta_v \subseteq N(x)$ ou $\Delta_v \cap \Delta_{v_2} \subseteq N(x)$. Supposons qu'il existe $y_1 \in (\Delta_{v_1} \cap \Delta_v) \setminus N(x)$ et $y_2 \in (\Delta_v \cap \Delta_{v_2}) \setminus N(x)$ tels que y_1 et y_2 sont tous les deux non liés. il se peut que $y_1 = y_2$, mais cela ne change pas le raisonnement suivant. Soit σ' un ordre consécutif de $\mathcal{K}(G')$. Comme v est mixte, il existe $a \in v^*$ tel qu'il existe une clique K de $\mathcal{K}(G')$ qui contient x et a et ne contient ni y_1 ni y_2 . D'après le lemme 5.21, dans σ' , à renversement près, on rencontre les cliques suivantes dans cet ordre : les cliques de $\mathcal{K}'_T(v_1)$ qui contiennent toutes y_1 et dont aucune ne contient a , puis les cliques de $\mathcal{K}'_T(v)$ qui contiennent toutes y_1 et y_2 et dont certaines contiennent a , puis enfin les cliques de $\mathcal{K}'_T(v_2)$ qui contiennent toutes y_2 et dont aucune ne contient a . Dans ces conditions, l'existence de la clique K rend impossible la consécutivité des cliques contenant a : absurde. $\Delta_{v_1} \cap \Delta_v \subseteq N(x)$ ou $\Delta_v \cap \Delta_{v_2} \subseteq N(x)$, et donc w vérifie la condition 3(c)ii.

- Si w n'a aucun fils mixte, considérons le plus petit élément v de σ_w qui satisfait la propriété droite. Si v est le premier élément de σ_w , alors en renversant σ_w , son dernier élément satisfait la propriété gauche, et donc w vérifie la condition 3(c)i. Sinon, c'est à dire si v a un prédécesseur dans σ_w , on distingue deux cas.

Le premier est celui dans lequel v ne satisfait pas la propriété gauche. On montre alors que le prédécesseur v_1 de v , lui, la satisfait. Par définition de v , il existe $z \in Y_u$ lié tel que $e_z^1 = v$. Il existe donc une clique maximale de G' qui contient x et z . Comme v_1 est non saturé, il existe une clique $K_1 \in \mathcal{K}'_T(v_1)$ qui ne contient pas x . La preuve que v_1 satisfait la propriété gauche se termine de la même façon que la preuve du fait que, dans le cas précédent, l'unique fils mixte de w satisfait la propriété gauche ou droite. Comme v satisfait la propriété droite et ne satisfait pas la gauche, et comme v_1 satisfait la gauche et pas la droite, alors $\Delta_{v_1} \cap \Delta_v \subseteq N(x)$, sinon il y aurait un C_4 dans G' .

Le second cas est celui dans lequel v satisfait aussi la propriété gauche. L'ensemble des fils de w qui satisfont à la fois la propriété gauche et la propriété droite, auquel v appartient, n'est autre que $S_\Delta(w)$ (voir notation page 229). Si $S_\Delta(w)$ contient le premier ou le dernier élément de σ_w , alors w vérifie la condition 3(c)i. Sinon, on note $S_\Delta(w) = \{v_i\}_{1 \leq i \leq k}$ et on note v_0 et v_{k+1} respectivement le prédécesseur et le successeur des noeuds de $S_\Delta(w)$ dans σ_w . v_0 satisfait la propriété gauche et v_{k+1} la droite. Pour montrer que w satisfait la condition 3c, il

suffit de montrer qu'il existe $i \in \llbracket 0, k \rrbracket$ tel que $\Delta_{v_i} \cap \Delta_{v_{i+1}} \subseteq N(x)$. Supposons que $\forall i \in \llbracket 0, k \rrbracket, \exists y_i \in (\Delta_{v_i} \cap \Delta_{v_{i+1}}) \setminus N(x)$. Il existe une clique de $\mathcal{K}(G')$ contenant x et tous les sommets de $Y_u \cap N(x)$ et ne contenant aucun des $\{y_i\}_{0 \leq i \leq k}$. En procédant de manière similaire à la preuve du fait que w vérifie la condition 3(c)ii dans le cas où w a un unique fils mixte v et où v satisfait à la fois la propriété gauche et droite, on conclut à une absurdité. Ainsi, dans le cas qui nous occupe, w vérifie aussi la condition 3(c)ii.

⇐ . Conditions suffisantes.

Cette partie de la preuve a pour objet de montrer que si les conditions du théorème 5.8 sont vérifiées, alors il est possible de construire un ordre consécutif de $G'[w^* \cup \{x\}]$. Afin de faciliter la lecture de la construction d'un tel ordre consécutif, on introduit les notations suivantes.

Notation 5.10 Soient σ_1 et σ_2 deux ordres linéaires respectivement sur les ensembles S_1 et S_2 . On note $\sigma_1 + \sigma_2$ l'ordre linéaire sur l'ensemble $S_1 \cup S_2$ défini par : Pour tous $x, y \in S_1 \cup S_2, x <_{\sigma_1 + \sigma_2} y$ ssi :

$$(x \in S_1 \text{ et } y \in S_2) \text{ ou } \exists i \in \{1, 2\}, x, y \in S_i \text{ et } x <_{\sigma_i} y$$

Remarque 5.8 L'opération $+$ ainsi définie est associative et admet un élément neutre qui est l'ordre linéaire dont l'ensemble de référence est creux. Nous ferons implicitement usage de ces deux faits.

Notation 5.11 Toujours dans le soucis d'alléger les constructions, on s'autorise, lorsque σ est un ordre linéaire ne possédant qu'un élément e , à noter $\sigma_1 + e$ à la place de $\sigma_1 + \sigma$. Enfin, on utilisera pour l'opération $+$ la notation classique : $\sum_{1 \leq i \leq k} \sigma_i = \sigma_1 + \dots + \sigma_k$.

Notation 5.12 Pour un ordre linéaire σ dont les éléments sont des ensembles de sommets, et pour un ensemble de sommets S , on note $\sigma_{[S]}$ l'ordre σ dans lequel on a ajouté les sommets de S à chacun des éléments qui composent σ . Si σ est l'ordre linéaire dont l'ensemble de référence est creux, alors $\sigma_{<S>}$ aussi.

Pour un noeud u de $T^c(G)$, on note σ^{u^*} un ordre consécutif quelconque de $G[u^*]$, en prenant comme convention que lorsque $u^* = \emptyset$, σ^{u^*} est un ordre linéaire sur un unique élément, cet élément étant l'ensemble vide.

Définition 5.11 Un noeud non creux $v \in T_w^c \setminus \{w\}$ est **prolongeable** ssi il satisfait les deux conditions suivantes :

- $G'[v^* \cup \{x\}]$ est un graphe d'intervalles et admet un ordre consécutif dans lequel la dernière clique contient x ; et
- si B_v est non pleine, alors x est simplicial dans $G'[v^* \cup \{x\}]$.

La preuve de la suffisance des conditions du théorème 5.8 que nous donnons ici est inductive. Elle part des noeuds pleins non déserts les plus hauts dans T_w^c et remonte vers le noeud w . A chaque étape on montre que le noeud u considéré est prolongeable, c'est à dire que $G'[u^* \cup \{x\}]$ admet un ordre consécutif et vérifie certaines propriétés qui nous permettront de montrer que son père aussi est prolongeable. L'initialisation de l'induction consiste à établir que les noeuds pleins non déserts sont prolongeables.

Proposition 5.13 *Tout noeud plein et non désert $v \in T_w^c \setminus \{w\}$ qui satisfait les conditions du théorème 5.8 est prolongeable.*

Preuve : Rajouter un sommet universel à un graphe d'intervalles résulte toujours en un graphe d'intervalles. Comme toutes les cliques de $G'[v^* \cup \{x\}]$ contiennent x , dans un ordre consécutif donné, la dernière, en particulier, contient x . Si B_v n'est pas pleine, comme v satisfait la condition 2, alors v a au plus un fils non creux. Or tous les fils de v sont pleins, ce qui implique qu'au plus un fils u de v est non désert. D'après le lemme 5.4, v ne peut pas être un noeud dégénéré. Supposons que v est premier, alors d'après la condition 4a, tous les sommets liés de Y_v couvrent un même fils de v . Or tous les sommets de Y_v sont liés puisque v est plein : absurde. Donc v n'est ni premier ni dégénéré : c'est une feuille. x est donc bien simplicial dans $G'[v^* \cup \{x\}]$. $\square$

Le lemme suivant sert à propager l'induction depuis les noeuds pleins non déserts jusqu'aux fils mixtes de w . Un troisième lemme sera dédié à la dernière étape d'induction, celle qui montre que $G'[w^* \cup \{x\}]$ admet un ordre consécutif.

Lemme 5.23 *Soit u un noeud mixte de $T_w^c \setminus \{w\}$. Si u vérifie les conditions du théorème 5.8 et si tous ses fils mixtes sont prolongeables, alors u est prolongeable.*

Preuve :

Tout d'abord, il convient de remarquer que comme u vérifie la condition 1, u a au plus un fils mixte.

1. Le cas où u est dégénéré est facile à traiter. D'après le lemme 5.4, tout fils v de u est tel que $v^* \neq \emptyset$. Ainsi, v ne peut pas être à la fois creux et plein. Les fils de u se partitionnent, au sens large, en l'ensemble $\mathcal{V} = \{v_i\}_{1 \leq i \leq k}$, avec $k \in \mathbb{N}$, de ceux qui sont creux, l'ensemble $\mathcal{P} = \{p_i\}_{1 \leq i \leq l}$, avec $l \in \mathbb{N}$, de ceux qui sont pleins, et enfin l'ensemble $\mathcal{M}$ de ceux qui sont mixtes.

- (a) Dans le cas où B_u est pleine, $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \bigcup_{v \in \mathcal{M}} \{K \cup X_u \mid K \in \mathcal{K}(G'[v^* \cup \{x\}])\} \cup \bigcup_{v \in \mathcal{P}} \{K \cup X_u \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\}$. Lorsque $\mathcal{M}$ possède un élément q , on note σ^q un ordre consécutif de $G'[q^* \cup \{x\}]$ dans lequel la dernière clique contient x . Sinon, c'est à dire lorsque $\mathcal{M} = \emptyset$, σ^q désigne l'ordre linéaire dont l'ensemble de référence est creux.

$\sigma' = \sum_{1 \leq i \leq k} \sigma_{<X_u>}^{v_i^*} + \sigma_{<X_u>}^q + \sum_{1 \leq i \leq l} \sigma_{<X_u \cup \{x\}>}^{p_i^*}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$. En effet, il est aisé de voir que les contraintes de consécuité imposées par les sommets de u^* sont respectées par σ' . Pour la consécuité

des cliques contenant x , elle est garantie par l'utilisation pour σ^q d'un ordre consécutif dans lequel la dernière clique contient x .

$G'[u^* \cup \{x\}]$ est donc un graphe d'intervalles et admet un ordre consécutif dans lequel la dernière clique contient x .

- (b) Dans le cas où B_u est non pleine, comme u satisfait la condition 1, u possède au plus un fils non creux.

- i. Si tous les fils de u sont creux, alors comme u est non creux, $X_u \cap N(x) \neq \emptyset$. On a $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{(X_u \cap N(x)) \cup \{x\}\}$. Posons $K_{ux} = (X_u \cap N(x)) \cup \{x\}$.

Il est immédiat de voir que $\sigma' = \sum_{1 \leq i \leq k} \sigma_{<X_u>}^{v_i^*} + K_{ux}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$. De plus, la dernière clique de σ' contient x et x est bien simplicial dans $G'[u^* \cup \{x\}]$.

- ii. Si u a un fils plein u_1 , comme B_{u_1} est non pleine, d'après la démonstration de la proposition 5.13, u_1 est une feuille.
- Si $X_u \setminus N(x) \neq \emptyset$, on a $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{X_u \cup X_{u_1}\} \cup \{(X_u \cap N(x)) \cup X_{u_1} \cup \{x\}\}$.
 - Si $X_u \subseteq N(x)$, on a $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{(X_u \cap N(x)) \cup X_{u_1} \cup \{x\}\}$.

On note $K_{ux} = (X_u \cap N(x)) \cup X_{u_1} \cup \{x\}$. Lorsque $X_u \setminus N(x) \neq \emptyset$, on note σ^1 l'ordre linéaire sur un unique élément $X_u \cup X_{u_1}$. Sinon, c'est à dire si $X_u \subseteq N(x)$, σ^1 désigne l'ordre linéaire dont l'ensemble de référence est creux.

Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{<X_u>}^{v_i^*} + \sigma^1 + K_{ux}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$. Il est assez simple de voir que les contraintes de consécuité imposées par les sommets de u^* sont respectées par σ' . x est bien présent dans la dernière clique de σ' , et x est simplicial dans $G'[u^* \cup \{x\}]$.

- iii. Si u a un fils mixte u_1 ,

$\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup X_u \mid K \in \mathcal{K}(G[u_1^* \cup \{x\}]) \text{ et } x \notin K\} \cup \{K \cup (X_u \cap N(x)) \mid K \in \mathcal{K}(G'[u_1^* \cup \{x\}]) \text{ et } x \in K\}$.

Comme u_1 est prolongeable, $\{K \in \mathcal{K}(G'[u_1^* \cup \{x\}]) \mid x \in K\}$ est de cardinal 1, on note K_x son unique élément. De plus, il existe un ordre consécutif σ^1 de $G'[u_1^* \cup \{x\}]$ tel que $\sigma^1 = \sigma^2 + K_x$, avec σ^2 tel qu'aucune clique qui le compose ne contient x . On pose $K_{ux} = K_x \cup (X_u \cap N(x))$. Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{<X_u>}^{v_i^*} + \sigma_{<X_u>}^2 + K_{ux}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$. De plus, x est présent seulement dans la dernière clique de σ' et est donc simplicial dans $G'[u^* \cup \{x\}]$.

Finalement, dans tous les cas, u est prolongeable.

2. Traitons le cas où u est premier.

Remarque 5.9 On considère, par convention, que $G[\emptyset] = (\emptyset, \emptyset)$ et que l'ensemble vide est l'unique clique maximale de $(\emptyset, \emptyset)$. Ainsi, $\mathcal{K}(G[\emptyset]) = \{\emptyset\}$. Dans ce qui

suit, on utilise ces conventions de manière avantageuse pour conserver l'unité des notations et l'intelligibilité des constructions.

(a) Dans le cas où B_u est pleine, comme u satisfait les conditions 2 et 4, à renversement près de σ_u , les fils non saturés de u précèdent les fils saturés, le dernier fils non saturé f_u vérifie la propriété gauche et les autres fils non saturés sont creux. On note $\mathcal{V} = \{v \in \mathcal{C}(u) \mid v <_{\sigma_u} f_u\}$, et $\mathcal{P} = \{v \in \mathcal{C}(u) \mid v >_{\sigma_u} f_u\}$. On pose $\mathcal{V} = \{v_i\}_{1 \leq i \leq k}$, avec $k \in \mathbb{N}$, et $\mathcal{P} = \{p_i\}_{1 \leq i \leq l}$, avec $l \in \mathbb{N}$. Notons que $\mathcal{P}$ peut être creux.

i. Si f_u est non creux, on distingue deux sous cas :

- B_{f_u} est pleine. Les cliques maximales de $G'[u^* \cup \{x\}]$ sont : $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_{f_u} \cup X_u \mid K \in \mathcal{K}(G'[f_u^* \cup \{x\}])\} \cup \bigcup_{v \in \mathcal{P}} \{K \cup \Delta_v \cup X_u \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\}$. Comme f_u est prolongeable, on note σ^f un ordre consécutif de $G'[f_u^* \cup \{x\}]$ dans lequel la dernière clique contient x . On a $\sigma^f = \sigma^{f1} + \sigma^{f2}$, avec σ^{f1} tel qu'aucune clique qui le compose ne contient x (lorsque f_u est plein, σ^{f1} est l'ordre linéaire dont l'ensemble de référence est creux) et avec σ^{f2} tel que toutes les cliques qui le composent contiennent x . Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{<\Delta_{v_i} \cup X_u>}^{v_i^*} + \sigma_{<\Delta_{f_u} \cup X_u>}^{f1} + \sigma_{<\Delta_{f_u} \cup X_u>}^{f2} + \sum_{1 \leq i \leq l} \sigma_{<\Delta_{p_i} \cup X_u \cup \{x\}>}^{p_i^*}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$.

Il ne fait aucun doute que σ' respecte les contraintes de consécutivité imposées par tous les sommets de $u^* \cup \{x\}$. u est donc prolongeable.

- B_{f_u} est non pleine. Dans ce cas, $\mathcal{K}(G'[f_u^* \cup \{x\}])$ comporte une unique clique maximale K_x contenant x . Les cliques maximales de $G'[u^* \cup \{x\}]$ sont : $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_{f_u} \cup X_u \mid K \in \mathcal{K}(G'[f_u^* \cup \{x\}]) \text{ et } x \notin K\} \cup \mathcal{K}_u \cup \{K_x \cup (\Delta_{f_u} \cap N(x)) \cup X_u\} \cup \bigcup_{v \in \mathcal{P}} \{K \cup \Delta_v \cup X_u \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\}$, avec $\mathcal{K}_u = \{(K_x \setminus \{x\}) \cup X_u \cup \Delta_{f_u}\}$ si $K_x \setminus \{x\}$ est maximale dans $G[f_u^*]$ et $\mathcal{K}_u = \emptyset$ sinon.

Comme f_u est prolongeable, on note σ^f un ordre consécutif de $G'[f_u^* \cup \{x\}]$ dans lequel la dernière clique est K_x . On a $\sigma^f = \sigma^{f1} + K_x$, avec σ^{f1} tel qu'aucune clique qui le compose ne contient x (lorsque f_u est plein, σ^{f1} est l'ordre linéaire dont l'ensemble de référence est creux). On note σ^1 l'ordre linéaire ayant pour unique élément $(K_x \setminus \{x\}) \cup \Delta_{f_u} \cup X_u$ si $\mathcal{K}_u \neq \emptyset$. Sinon, si $\mathcal{K}_u = \emptyset$, σ^1 est l'ordre linéaire dont l'ensemble de référence est creux. On note $K_{xu} = K_x \cup (\Delta_{f_u} \cap N(x)) \cup X_u$.

$\sigma' = \sum_{1 \leq i \leq k} \sigma_{<\Delta_{v_i} \cup X_u>}^{v_i^*} + \sigma_{<\Delta_{f_u} \cup X_u>}^{f1} + \sigma^1 + K_{xu} + \sum_{1 \leq i \leq l} \sigma_{<\Delta_{p_i} \cup X_u \cup \{x\}>}^{p_i^*}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$.

Il ne fait aucun doute que σ' respecte les contraintes de consécutivité imposées par les sommets de $\bigcup_{v \in \mathcal{C}(u) \setminus \{f_u\}} v^*$, par les sommets de X_u , par les sommets de $Y_u \setminus \Delta_{f_u}$ et par x . Il suffit donc de vérifier que σ' satisfait les contraintes de consécutivité imposées par les sommets de u_f^* et par

ceux de Δ_{f_u} . Le doute porte plus précisément sur les sommets de $\Delta_{f_u} \setminus N(x)$ car ils ne sont pas présents dans K_{x_u} . Mais comme les sommets de $\Delta_{f_u} \setminus N(x)$ ne couvrent pas le suivant de f_u , qui est saturé, alors σ' satisfait les contraintes de consécuitivité imposées par les sommets de $\Delta_{f_u} \setminus N(x)$. Pour les sommets de f_u^* , le doute provient de l'insertion de σ^1 , lorsque celui-ci n'est pas creux, qui sépare K_x des autres cliques de $\mathcal{K}(G[f_u^* \cup \{x\}])$. Heureusement, x n'est pas présent dans ces autres cliques, et l'unique clique de σ^1 contient tous les sommets de $K_x \setminus \{x\}$. u est donc prolongeable.

ii. Si f_u est creux,

$\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_{f_u} \cup X_u \mid K \in \mathcal{K}(G[f_u^*])\} \cup \mathcal{K}\mathcal{F}\mathcal{X} \cup \bigcup_{v \in \mathcal{P}} \{K \cup \Delta_v \cup X_u \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\}$, avec $\mathcal{K}\mathcal{F}\mathcal{X} = \{(\Delta_{f_u} \cap N(x)) \cup X_u \cup \{x\}\}$ si f_u satisfait la propriété gauche stricte et $\mathcal{K}\mathcal{F}\mathcal{X} = \emptyset$ sinon.

Si f_u satisfait la propriété gauche stricte, on note σ^{fx} l'ordre linéaire sur un unique élément qui est précisément $(\Delta_{f_u} \cap N(x)) \cup X_u \cup \{x\}$. Sinon, σ^{fx} désigne l'ordre linéaire dont l'ensemble de référence est creux. Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{<\Delta_{v_i} \cup X_u>}^{v_i^*} + \sigma_{<\Delta_{f_u} \cup X_u>}^{f_u^*} + \sigma^{fx} + \sum_{1 \leq i \leq l} \sigma_{<\Delta_{p_i} \cup X_u \cup \{x\}>}^{p_i}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$.

Encore une fois, il ne fait aucun doute que σ' satisfait les contraintes de consécuitivité imposées par les sommets de $\bigcup_{v \in \mathcal{C}(u)} v^*$, par les sommets de X_u , par les sommets de $Y_u \setminus \Delta_{f_u}$, par les sommets de $\Delta_{f_u} \cap N(x)$ et par x . Par contre, il convient d'examiner avec un peu plus d'attention le cas des sommets de $\Delta_{f_u} \setminus N(x)$ lorsque $\mathcal{K}\mathcal{F}\mathcal{X} \neq \emptyset$. En effet, les sommets de $\Delta_{f_u} \setminus N(x)$ sont absents de $\mathcal{K}\mathcal{F}\mathcal{X}$. Pour que les contraintes de consécuitivité soient respectées pour ces sommets, il faut qu'aucun ne couvre le suivant de f_u : c'est le cas car les sommets de $\Delta_{f_u} \setminus N(x)$ ne sont pas liés et le suivant de f_u dans σ_u est saturé. Donc u est prolongeable.

(b) Dans le cas où B_u est non pleine, comme u satisfait la condition 4a, tous ses fils, sauf éventuellement le dernier f , sont creux et f satisfait la propriété gauche. On note $\mathcal{V} = \{v \in \mathcal{C}(u) \mid v <_{\sigma_u} f\}$, et on pose $\mathcal{V} = \{v_i\}_{1 \leq i \leq k}$, avec $k \in \mathbb{N}$. Comme B_u est non pleine, B_f est non pleine également. Ainsi, $\mathcal{K}(G[f^*])$ comporte une unique clique maximale K_x contenant x .

i. Si f est non creux, les cliques maximales de $G'[u^* \cup \{x\}]$ sont :

$\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_f \cup X_u \mid K \in \mathcal{K}(G'[f^* \cup \{x\}]) \text{ et } x \notin K\} \cup \mathcal{K}_u \cup \{K_x \cup ((\Delta_f \cup X_u) \cap N(x))\}$, avec $\mathcal{K}_u = \{(K_x \setminus \{x\}) \cup \Delta_f \cup X_u\}$ si $K_x \setminus \{x\}$ est maximale dans $G[f^*]$ et $(\Delta_f \cup X_u) \setminus N(x) \neq \emptyset$.

Comme f est prolongeable, il existe un ordre de consécuitivité σ^f de $G'[f^* \cup \{x\}]$ qui s'écrit $\sigma^{f1} + K_x$, avec σ^{f1} tel qu'aucune des cliques qui le composent ne contient x . Lorsque f est plein, σ^{f1} désigne l'ordre linéaire dont

l'ensemble de référence est creux. On pose $K_{xf} = K_x \cup ((\Delta_f \cup X_u) \cap N(x))$. On note σ^1 l'ordre linéaire ayant $(K_x \setminus \{x\}) \cup \Delta_f \cup X_u$ pour seul élément si $\mathcal{K}_u \neq \emptyset$. σ^1 désigne l'ordre linéaire dont l'ensemble de référence est creux si $\mathcal{K}_u = \emptyset$.

Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{\langle \Delta_{v_i} \cup X_u \rangle}^{v_i^*} + \sigma_{\langle \Delta_f \cup X_u \rangle}^{f1} + \sigma^1 + K_{xf}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$.

En effet, comme l'unique clique de σ^1 contient tous les sommets de $K_x \setminus \{x\}$ et comme x n'est présent que dans la dernière clique de σ' , alors la consécutivité des cliques contenant un sommet donné de f^* n'est pas menacé. σ' respecte toutes les contraintes de consécutivité imposées par les sommets de u^* et par x . Donc u est prolongeable.

ii. Si f est creux, on distingue deux sous cas :

- $f^* \neq \emptyset$. Les cliques maximales de $G'[u^* \cup \{x\}]$ sont :
 $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_f \cup X_u \mid K \in \mathcal{K}(G[f^*])\} \cup \{((\Delta_f \cup X_u) \cap N(x)) \cup \{x\}\}$.
- $f^* = \emptyset$. Les cliques maximales de $G'[u^* \cup \{x\}]$ sont :
 $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup \Delta_v \cup X_u \mid K \in \mathcal{K}(G[v^*])\} \cup \mathcal{K}_u \cup \{(\Delta_f \cup X_u) \cap N(x) \cup \{x\}\}$, avec $\mathcal{K}_u = \{\Delta_f \cup X_u\}$ si $(\Delta_f \cup X_u) \setminus N(x) \neq \emptyset$, et $\mathcal{K}_u = \emptyset$ sinon.

Lorsque $f^* \neq \emptyset$, on note $\sigma^f = \sigma_{[\Delta_f \cup X_u]}^{f^*}$. Si $f^* = \emptyset$ et $(\Delta_f \cup X_u) \setminus N(x) \neq \emptyset$, σ^f désigne l'ordre linéaire ayant $\Delta_f \cup X_u$ pour seul élément. Si $f^* = \emptyset$ et $\Delta_f \cup X_u \subseteq N(x)$, σ^f désigne l'ordre linéaire dont l'ensemble de référence est creux. On pose $K_{xf} = ((\Delta_f \cup X_u) \cap N(x)) \cup \{x\}$.

Ainsi, $\sigma' = \sum_{1 \leq i \leq k} \sigma_{[\Delta_{v_i} \cup X_u]}^{v_i^*} + \sigma^f + K_{xf}$ est un ordre consécutif de $G'[u^* \cup \{x\}]$.

En effet, il est clair que σ' respecte toutes les contraintes de consécutivité imposées par les sommets de u^* et par x . De plus, seule la dernière clique de σ' contient x . Donc u est prolongeable. ■

Grâce à la proposition et au lemme précédents, on a obtenu que les noeuds pleins non déserts et les fils mixtes de w sont prolongeables. Il ne reste plus qu'à montrer que sous ces conditions, $G'[w^* \cup \{x\}]$ admet un ordre consécutif.

Lemme 5.24 *Si w satisfait les conditions du théorème 5.8 et si tous les fils non creux de w sont prolongeables, alors $G'[w^* \cup \{x\}]$ est un graphe d'intervalles.*

Preuve :

D'après la condition 1, w a au plus deux fils mixtes.

1. Traitons d'abord le cas où w est dégénéré. D'après le lemme 5.4, tout fils v de w est tel que $v^* \neq \emptyset$. Ainsi, v ne peut pas être à la fois creux et plein. Les fils de w se

partitionnent, au sens large, en l'ensemble $\mathcal{V} = \{v_i\}_{1 \leq i \leq k}$, avec $k \in \mathbb{N}$, de ceux qui sont creux, l'ensemble $\mathcal{P} = \{p_i\}_{1 \leq i \leq l}$, avec $l \in \mathbb{N}$, de ceux qui sont pleins, et enfin l'ensemble $\mathcal{M}$ de ceux qui sont mixtes.

- (a) Dans le cas où B_w est pleine, $\mathcal{K}(G'[w^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}} \{K \cup X_w \mid K \in \mathcal{K}(G[v^*])\} \cup \bigcup_{v \in \mathcal{M}} \{K \cup X_w \mid K \in \mathcal{K}(G'[v^* \cup \{x\}])\} \cup \bigcup_{v \in \mathcal{P}} \{K \cup X_w \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\}$

Si $|\mathcal{M}| = 1$, on note $\mathcal{M} = \{q_1\}$. Si $|\mathcal{M}| = 2$, on note $\mathcal{M} = \{q_1, q_2\}$. Lorsque $|\mathcal{M}| \geq 1$, on note σ^{q_1} un ordre consécutif de $G'[q_1^* \cup \{x\}]$ dans lequel la dernière clique contient x , et lorsque $|\mathcal{M}| \geq 2$, on note σ^{q_2} un ordre consécutif de $G'[q_2^* \cup \{x\}]$ dans lequel la dernière clique contient x . Si $\mathcal{M} = \emptyset$, σ^{q_1} désigne l'ordre linéaire dont l'ensemble de référence est creux ; de même pour σ^{q_2} lorsque $|\mathcal{M}| \leq 1$.

$\sigma' = \sum_{1 \leq i \leq k} \sigma_{<X_w>}^{v_i^*} + \sigma_{<X_w>}^{q_1} + \sum_{1 \leq i \leq l} \sigma_{<X_w>}^{p_i^*} + \bar{\sigma}_{<X_w>}^{q_2}$ est un ordre consécutif de $G'[w^* \cup \{x\}]$.

En effet, il est aisé de voir que les contraintes de consécutivité imposées par les sommets de w^* sont respectées par σ' . Pour la consécutivité des cliques contenant x , elle est garantie par l'utilisation pour σ^{q_1} et σ^{q_2} d'ordres consécutifs dans lesquels la dernière clique contient x et par l'utilisation de l'ordre $\bar{\sigma}^{q_2}$ inverse de σ^{q_2} . Donc, $G'[w^* \cup \{x\}]$ est un graphe d'intervalles.

- (b) Le cas où B_w est non pleine est en tous points similaire à celui de la preuve du lemme 5.23 dans lequel u est un noeud dégénéré de $T_w^c \setminus \{w\}$ tel que B_u est non pleine. De la même façon, on montre que $G'[w^* \cup \{x\}]$ est un graphe d'intervalles.

2. Traitons maintenant le cas où w est premier.

- (a) Si B_w est pleine et $I_w \neq \emptyset$, on est dans un cas très proche du cas 2a de la disjonction de cas du lemme 5.23 où u est un noeud de $T_w^c \setminus w$ tel que B_u est pleine. La différence essentielle entre les deux cas est que dans le cas de w , I_w ne contient pas nécessairement un des éléments extrémaux de σ_w . Ainsi, le cas général est celui où I_w est précédé et suivi dans σ_w par des noeuds non saturés, alors que dans le cas de u , les noeuds non saturés se situent tous avant I_u dans σ_u . De la même façon que, dans le cas 2a de la preuve du lemme 5.23, f_u satisfait la propriété gauche, dans le cas de w , d'après la condition 3b du théorème, f_w et l_w satisfont respectivement la propriété gauche et la propriété droite. En fait, on peut dire que le cas qui nous occupe ici est le cas symétrisé de celui déjà traité pour u . Ainsi, la discussion tenue sur f_u se reconduit à l'identique sur f_w dans le cas présent et se reconduit par symétrie sur l_w . Les disjonctions de cas sur f_w et l_w , selon qu'ils sont pleins et non creux, mixtes ou creux, selon qu'ils vérifient respectivement les propriétés gauche et droite strictes ou non, sont indépendantes et donnent lieu dans le cas qui nous occupe à de très nombreux sous-cas qui ne font que reprendre le contenu du cas 2a de la preuve du lemme 5.23. Cela explique au lecteur que cette disjonction de cas

soit absente de ce manuscrit. A sa place, nous prendrons comme exemple un de ses sous cas "assez représentatif" de tous les autres.

Ce cas est celui où I_w ne contient aucun sommet extrémal de σ_w , où f_w et l_w sont mixtes et où B_{f_w} et B_{l_w} sont pleines. On note $\mathcal{V}_1 = \{v \in \mathcal{C}(u) \mid v <_{\sigma_u} f_w\}$, $\mathcal{P} = \{v \in \mathcal{C}(u) \mid f_u <_{\sigma_u} v <_{\sigma_u} l_w\}$, et $\mathcal{V}_2 = \{v \in \mathcal{C}(u) \mid v >_{\sigma_u} l_w\}$. On pose $\mathcal{V}_1 = \{v_{1,i}\}_{1 \leq i \leq k_1}$, avec $k_1 \in \mathbb{N}$, $\mathcal{V}_2 = \{v_{2,i}\}_{1 \leq i \leq k_2}$, avec $k_2 \in \mathbb{N}$, et $\mathcal{P} = \{p_i\}_{1 \leq i \leq l}$, avec $l \in \mathbb{N}$.

Dans ce cas, on a $\mathcal{K}(G'[u^* \cup \{x\}]) = \bigcup_{v \in \mathcal{V}_1} \{K \cup \Delta_v \cup X_w \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup \Delta_{f_w} \cup X_w \mid K \in \mathcal{K}(G'[f_w^* \cup \{x\}]) \text{ et } x \notin K\} \cup \{K \cup (\Delta_{f_w} \cap N(x)) \cup X_w \mid K \in \mathcal{K}(G'[f_w^* \cup \{x\}]) \text{ et } x \in K\} \cup \bigcup_{v \in \mathcal{P}} \{K \cup \Delta_v \cup X_w \cup \{x\} \mid K \in \mathcal{K}(G[v^*])\} \cup \{K \cup (\Delta_{l_w} \cap N(x)) \cup X_w \mid K \in \mathcal{K}(G'[l_w^* \cup \{x\}]) \text{ et } x \in K\} \cup \{K \cup \Delta_{l_w} \cup X_w \mid K \in \mathcal{K}(G'[l_w^* \cup \{x\}]) \text{ et } x \notin K\} \cup \bigcup_{v \in \mathcal{V}_2} \{K \cup \Delta_v \cup X_w \mid K \in \mathcal{K}(G[v^*])\}$. Comme f_w (resp. l_w) est prolongeable, on note σ^f (resp. σ^l) un ordre consécutif de $G'[f_w^* \cup \{x\}]$ (resp. $G'[l_w^* \cup \{x\}]$) dans lequel la dernière clique contient x . On a $\sigma^f = \sigma^{f1} + \sigma^{f2}$, avec σ^{f1} tel qu'aucune clique qui le compose ne contient x et avec σ^{f2} tel que toutes les cliques qui le composent contiennent x . De même, on a $\sigma^l = \sigma^{l1} + \sigma^{l2}$. Ainsi,

$$\sigma' = \sum_{1 \leq i \leq k_1} \sigma_{[\Delta_{v_{1,i}} \cup X_w]}^{v_{1,i}^*} + \sigma_{[\Delta_{f_w} \cup X_w]}^{f1} + \sigma_{[(\Delta_{f_w} \cap N(x)) \cup X_w]}^{f2} + \sum_{1 \leq i \leq l} \sigma_{[\Delta_{p_i} \cup X_w \cup \{x\}]}^{p_i} + \bar{\sigma}_{[(\Delta_{l_w} \cap N(x)) \cup X_w]}^{l2} + \bar{\sigma}_{[\Delta_{l_w} \cup X_w]}^{l1} + \sum_{1 \leq i \leq k_2} \sigma_{[\Delta_{v_{2,i}} \cup X_w]}^{v_{2,i}^*} \text{ est un ordre consécutif de } G'[u^* \cup \{x\}].$$

Notez que comme dans le cas où w est dégénéré, il convient d'inverser l'ordre consécutif pris pour le fils mixte suivant les fils saturés. La vérification du fait que σ' ainsi défini respecte les contraintes de consécutivité est similaire à celle du cas 2a de la preuve du lemme 5.23, en tenant compte de la symétrie du cas présent. $G'[w^* \cup \{x\}]$ est bien un graphe d'intervalles.

- (b) Le cas où B_w est pleine et $I_w = \emptyset$ s'obtient directement à partir de cas déjà traités.

Le cas où w vérifie la condition 3(c)i du théorème se traite de manière identique au cas d'un noeud $u \in T_w^c \setminus \{w\}$ tel que B_u est pleine et $I_u = \emptyset$ (condition 4a du théorème). Ce cas est celui traité au 2a de la disjonction de cas de la preuve du lemme 5.23.

Le cas où w vérifie la condition 3(c)ii, s'obtient simplement comme cas particulier du cas où B_w est pleine et $I_w \neq \emptyset$ en remplaçant f_w et l_w par f et l qui sont consécutifs, puisque justement, dans le cas qui nous occupe, $I_w = \emptyset$.

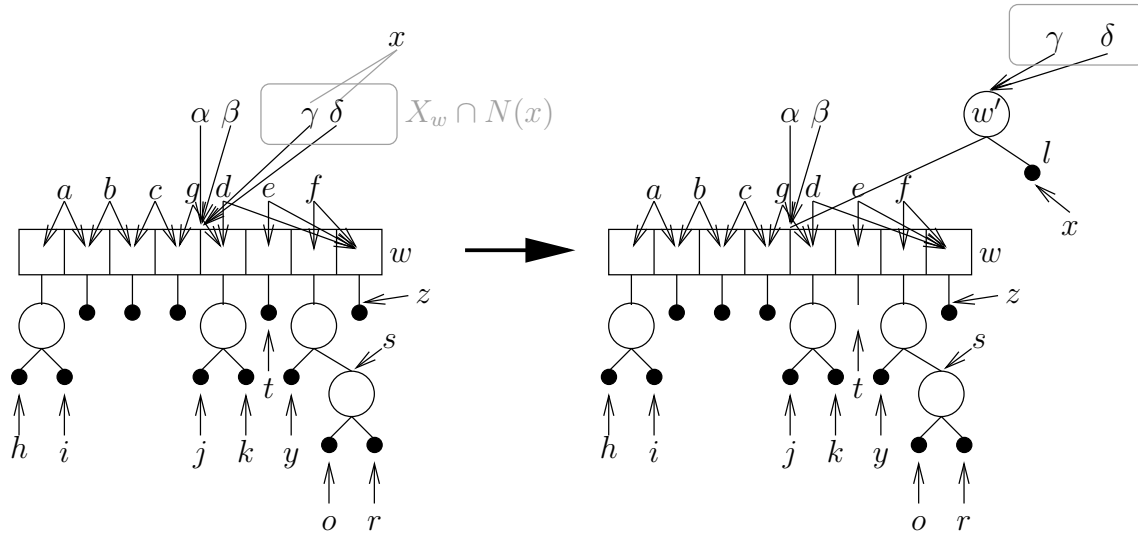
- (c) Si B_w n'est pas pleine, la preuve du cas 2b de la disjonction de cas du lemme 5.23, dans lequel u est un noeud premier de $T_w^c \setminus \{w\}$ tel que B_u est non pleine (condition 4a du théorème), s'applique parfaitement dans le cas présent et montre que $G'[w^* \cup \{x\}]$ est un graphe d'intervalles. ■

Les lemmes 5.23 et 5.24 constituent le moteur d'une induction de bas en haut dans l'arbre. Cette induction part des noeuds mixtes sans fils mixte et grâce au lemme 5.23 conclut que tous les fils non creux de w sont prolongeables. Le lemme 5.24 implique alors que $G'[w^* \cup \{x\}]$ est un graphe d'intervalles. Le théorème 5.7 donne que G' lui-même est un graphe d'intervalles. Ce qui achève la preuve du sens réciproque du théorème 5.8. ■

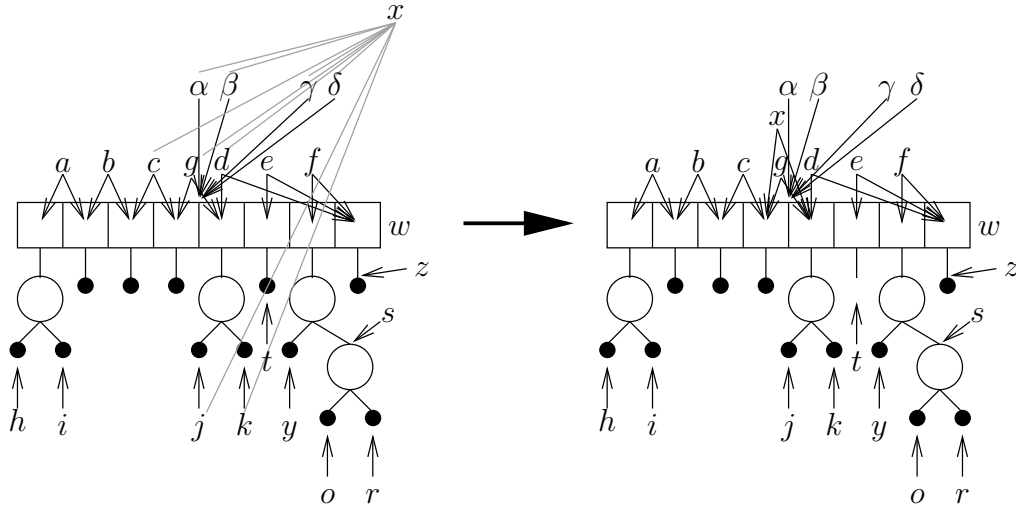
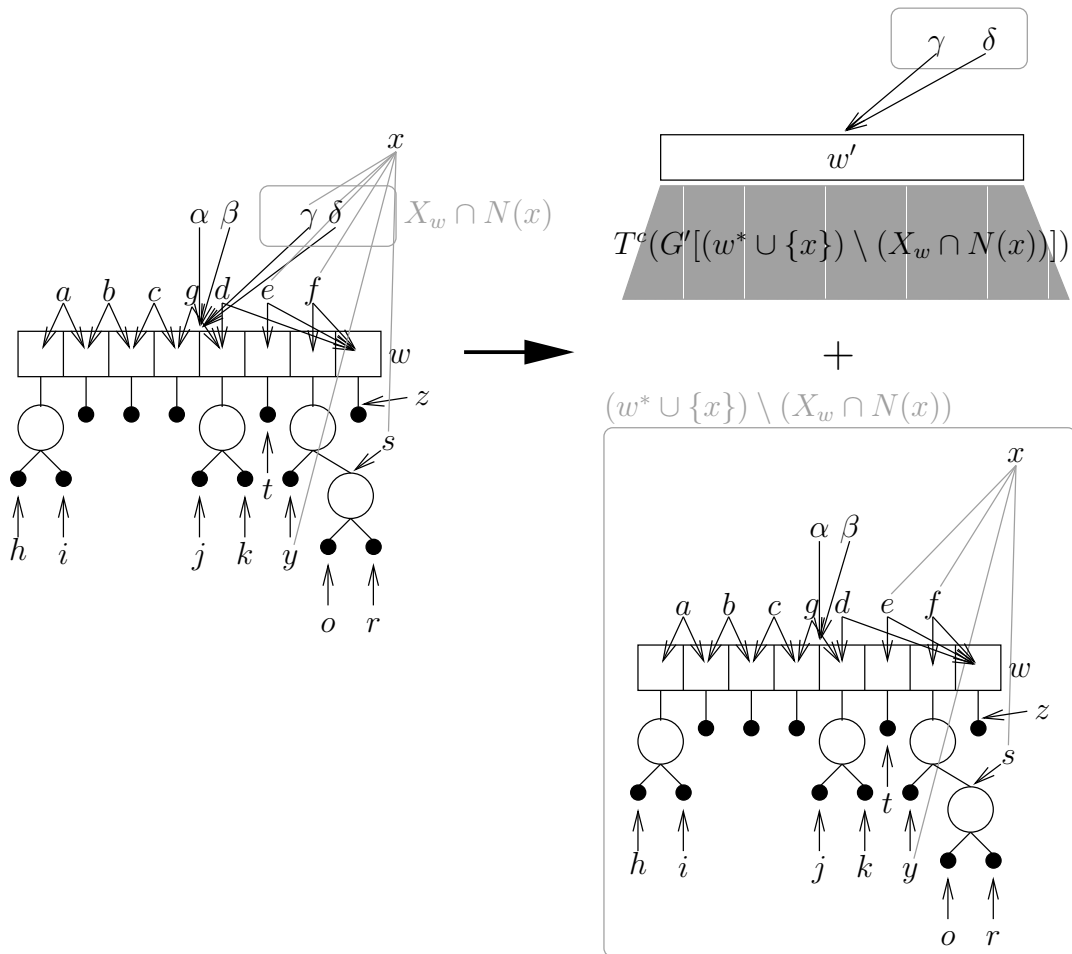
5.5.4 Déterminer $PQ(G + x)$

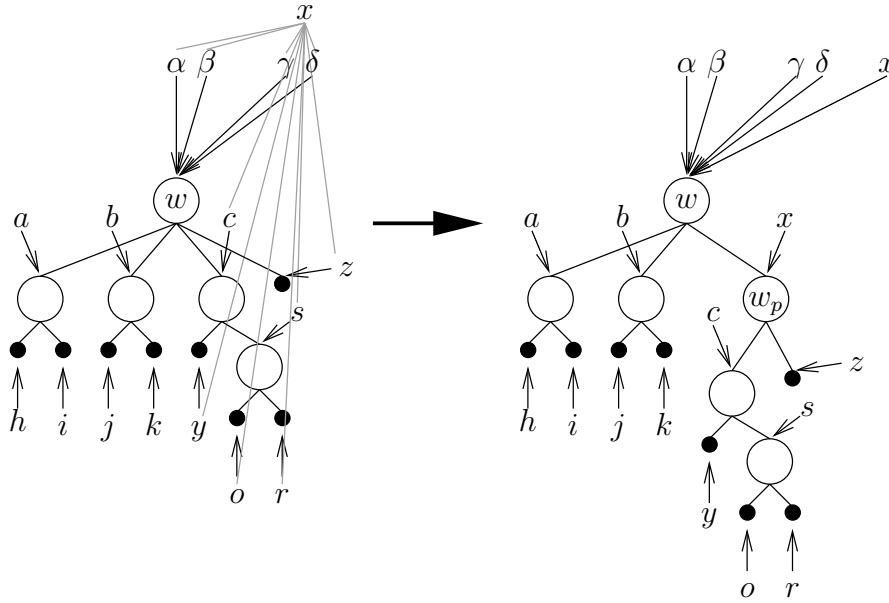
Maintenant qu'on sait discerner les cas où $G + x$ est un graphe d'intervalles des cas où il ne l'est pas (théorème 5.8), il nous reste à déterminer $PQ(G + x)$ lorsque $G + x$ est bien un graphe d'intervalles. Le théorème 5.7 montre qu'il suffit pour cela de déterminer $PQ(G'[w^* \cup \{x\}])$. C'est l'objet de la disjonction de cas qui suit. Dans la plupart des cas de cette discussion, les modifications de $PQ(G[w^*])$ sont simples. Par contre, dans trois de ces cas, il y a contraction d'une partie de l'arbre en un noeud premier. Pour ces cas, on ne détermine ici que la forme de l'arbre extérieure au sous arbre enraciné en ce nouveau noeud premier. Déterminer entièrement ce sous-arbre sera fait ultérieurement.

1. Si w est creux, on obtient $PQ(G'[w^* \cup \{x\}])$ très simplement. La racine de $PQ(G'[w^* \cup \{x\}])$ est un nouveau noeud dégénéré w' qui possède deux fils : le premier est le noeud w de $PQ(G)$ et le second est une feuille l . x se voit attribuer un pointeur vers l et les sommets de w^* conservent leurs pointeurs vers $T_w^c(G)$.
2. Si w est plein, la construction de $PQ(G'[w^* \cup \{x\}])$ est encore plus simple. Il suffit de reprendre $T_w^c(G)$, sans changer les pointeurs des sommets de w^* , et d'assigner à x un pointeur vers w .
3. Si w est mixte, les sommets universels de $G'[w^* \cup \{x\}]$ sont les sommets de $X_w \cap N(x)$. Le cas où w est dégénéré et celui où il est premier se traitent de manières très différentes.
 - (a) Commençons par écarter le cas trivial où w est une feuille. Dans ce cas, on remplace w par un noeud dégénéré ayant deux fils feuilles w_l et w_{nl} . x et les sommets de $X_w \cap N(x)$ pointent sur w_l , et les sommets de $X_w \setminus N(x)$ pointent sur w_{nl} .
 - (b) Si w est premier, il convient de remarquer que $w^* \setminus X_w$ n'est pas plein. En effet, s'il l'était, w ne satisfairait pas la condition 3a du théorème 5.8 et donc B_w serait pleine. Ainsi, w serait plein. Or, dans le cas présent, w est mixte. On distingue les sous-cas suivants.
 - i. Si $w^* \setminus X_w$ est creux (voir figure 5.26), alors on obtient $T^c(G'[w^* \cup \{x\}])$ en introduisant un nouveau noeud dégénéré w' auquel on attribue deux fils : w et une nouvelle feuille l . Les sommets de $w^* \setminus (X_w \cap N(x))$ conservent leurs pointeurs et les sommets de $X_w \cap N(x)$ pointent vers w' . x pointe vers l .
 - ii. Si $w^* \setminus X_w$ est mixte, on sépare deux cas.

FIG. 5.26 – Insertion de x dans le cas 3(b)i de la discussion.

- A. S'il existe un sommet $y \in w^* \setminus X_w$ qui est un vrai jumeau de x dans $G'[w^* \cup \{x\}]$ (voir figure 5.27), alors nécessairement $y \in Y_u$, sinon w serait propre. On obtient $PQ(G'[w^* \cup \{x\}])$ à partir de $T_w^c(G)$ en laissant leurs pointeurs à tous les sommets de w^* et en attribuant à x les mêmes pointeurs que y .
- B. Aucun sommet de $w^* \setminus X_w$ n'est vrai jumeau de x dans $G'[w^* \cup \{x\}]$ (voir figure 5.28). Les sommets universels de $G'[w^* \cup \{x\}]$ sont les sommets de $X_w \cap N(x)$. Comme w est premier et comme $w^* \setminus X_w$ est mixte, alors $G'[(w^* \cup \{x\}) \setminus (X_w \cap N(x))]$ est connexe. Donc, la racine w' de $T^c(G'[w^* \cup \{x\}])$ est un noeud premier et les sommets de $X_w \cap N(x)$ ont un pointeur principal vers w' et pas de pointeurs secondaires. Nous déterminons le reste de la PQ -représentation de $G'[w^* \cup \{x\}]$ ultérieurement. Nous ferons de même pour deux des cas suivants qui feront l'objet avec le cas présent d'une étude approfondie postérieure à cette discussion.
- (c) Si w est dégénéré, on distingue trois sous-cas. Rappelons que, par le lemme 5.4, aucun fils de w ne peut être à la fois creux et plein.
- Si $X_w \subseteq N(x)$ et w n'a aucun fils mixte (voir figure 5.29), on obtient $T^c(G'[w^* \cup \{x\}])$ à partir de $T_w^c(G)$ de la manière suivante. Comme w est mixte, il a au moins un fils creux. On retire les fils pleins de w de la liste des fils de w et on les rend fils d'un nouveau noeud dégénéré w_p . w_p devient un fils de w . Les pointeurs des sommets de w^* sont inchangés et x pointe sur w_p .
 - Si $X_w \subseteq N(x)$ et w a au moins un fils mixte (voir figure 5.30), on note $\mathcal{P}$ l'ensemble des fils de w qui sont pleins ou mixtes, et $P = \bigcup_{v \in \mathcal{P}} v^*$. A l'évidence, le graphe $G'[P \cup \{x\}]$ ne possède aucun sommet universel. Or,

FIG. 5.27 – Insertion de x dans le cas 3(b)iiA de la discussion.FIG. 5.28 – Ramener le cas 3(b)iiB de la discussion au cas où w vérifie la condition 1 de la définition 5.12.

FIG. 5.29 – Insertion de x dans le cas 3(c)i de la discussion.

comme w est dégénéré, pour tout $v \in \mathcal{P}$, $G[v^*]$ est connexe. Et comme x est adjacent à au moins un sommet de chacun des ensembles v^* , pour $v \in \mathcal{P}$, alors $G'[P \cup \{x\}]$ est connexe. Comme $G'[P \cup \{x\}]$ n'a aucun sommet universel, la racine de $T^c(G'[P \cup \{x\}])$ est un noeud premier. Les sommets de X_w sont les sommets universels de $G'[w^* \cup \{x\}]$ et aucun sommet de $w^* \setminus P$ n'est adjacent à un sommet de $P \cup \{x\}$. Ainsi, on obtient $T^c(G'[w^* \cup \{x\}])$ en retirant les noeuds de $\mathcal{P}$ des fils de w , et en les remplaçant par un nouveau noeud premier w_1 que l'on fait fils de w , et qui est tel que $w_1^* = P \cup \{x\}$. Les sommets de $(w^* \setminus P)$ conservent leurs pointeurs. $T^c(G'[P \cup \{x\}])$ ainsi que les pointeurs des sommets de $P \cup \{x\}$ seront déterminés ultérieurement.

- iii. Si $X_w \setminus N(x) \neq \emptyset$ (voir figure 5.31), les sommets universels de $G'[w^* \cup \{x\}]$ sont les sommets de $X_w \cap N(x)$. Soit $a \in X_w \setminus N(x)$. a est adjacent à tous les sommets de $w^* \setminus (X_w \cap N(x))$. Comme x est adjacent à au moins un sommet de $w^* \setminus X_w$, alors $G'[(w^* \cup \{x\}) \setminus (X_w \cap N(x))]$ est connexe. On en conclut que la racine w' de $T^c(G'[w^* \cup \{x\}])$ est un noeud premier. Les sommets de $X_w \cap N(x)$ ont un pointeur principal vers w' et pas de pointeurs secondaires. Le reste de la PQ -représentation de $G'[w^* \cup \{x\}]$ est déterminé dans ce qui suit, en même temps que les cas précédents laissés inachevés.

On introduit un vocabulaire spécial pour désigner les trois configurations dans lesquelles il y a création d'un noeud premier : on dit que w est sec.

Définition 5.12 Soit G un graphe d'intervalles et x un sommet à insérer dans G . Un noeud $u \in T^c(G)$ est **sec** ssi u vérifie une des trois conditions suivantes :

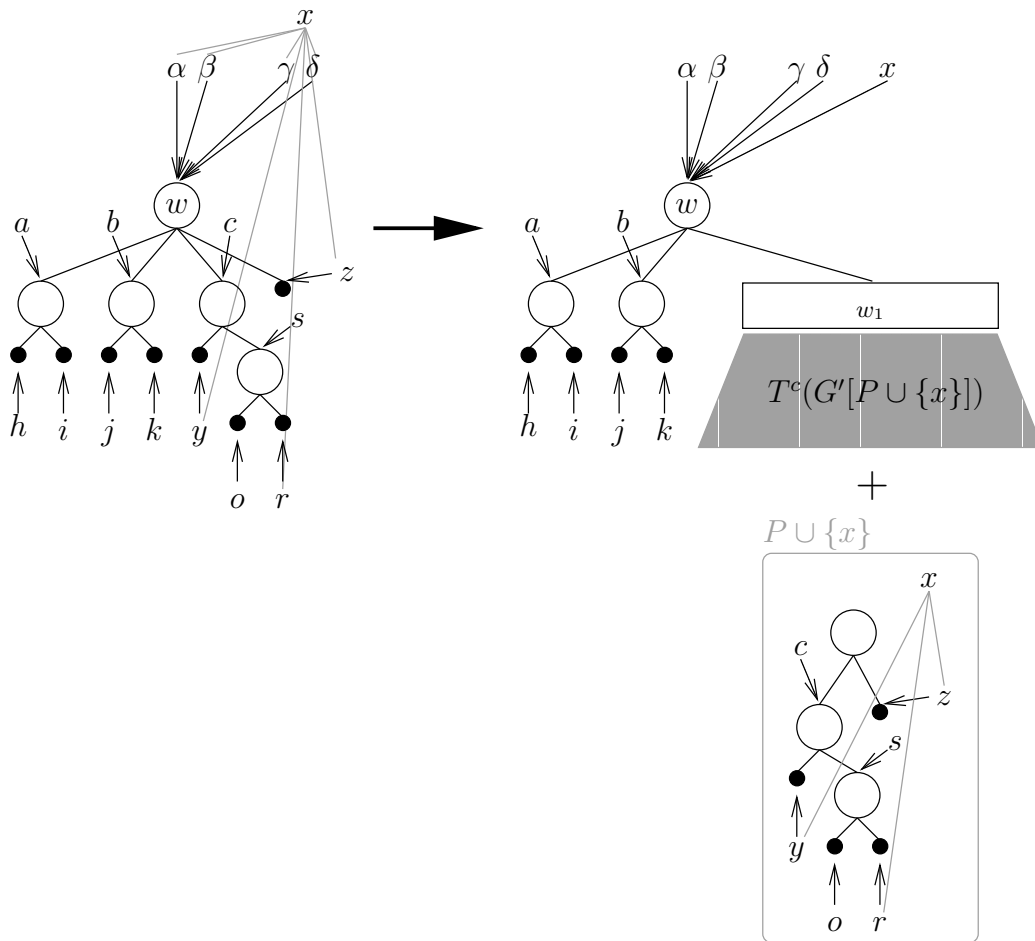


FIG. 5.30 – Ramener le cas 3(c)ii de la discussion au cas où w vérifie la condition 2 de la définition 5.12.

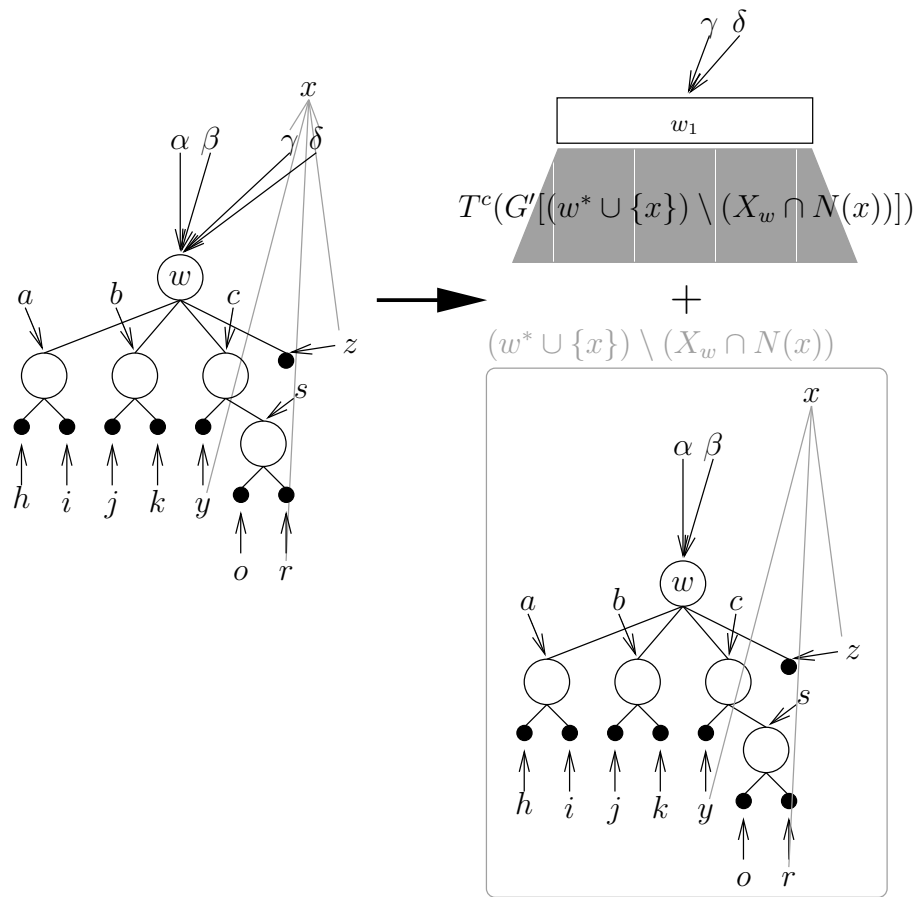


FIG. 5.31 – Ramener le cas 3(c)iii de la discussion au cas où w vérifie la condition 3 de la définition 5.12.

1. u est premier et $X_u \cap N(x) = \emptyset$ et $u^* \setminus X_u$ est mixte et x n'a pas de vrai jumeau dans $G'[(u^* \setminus X_u) \cup \{x\}]$; ou
2. u est dégénéré et $X_u = \emptyset$ et u n'a aucun fils creux; ou
3. u est dégénéré et $X_u \neq \emptyset$ et $X_u \cap N(x) = \emptyset$.

La discussion précédente a ramené le cas 3(b)iiB au cas où w vérifie la condition 1 de la définition d'un noeud sec. Le cas 3(c)ii de cette même discussion a été ramené au cas où w vérifie la condition 2 de la définition d'un noeud sec. Enfin, on a montré que l'étude du cas 3(c)iii se réduit à celle du cas où w vérifie la condition 3 de la définition d'un noeud sec.

La suite est dédiée à déterminer $PQ(G'[w^* \cup \{x\}])$ dans les cas où w est sec.

Traiter les cas où w est sec

Les cas d'insertions les plus délicats sont ceux où w est sec. Dans ces cas, il y a effondrement d'une partie de l'arbre en un nouveau noeud premier. Pour traiter ces cas difficiles nous allons tirer grand parti de la correspondance entre PQ -représentation et MD -représentation pour récupérer les résultats sur la modification de la décomposition modulaire d'un graphe quelconque lors de l'insertion d'un sommet. Ces résultats ont été démontré dans la section 4.5.2 pour servir les besoins de l'entretien dynamique des graphes de permutation, le théorème majeur étant le théorème 4.16. Ils s'appliquent tout autant aux graphes d'intervalles et vont nous permettre de déduire presque entièrement la PQ -représentation du nouveau graphe.

Le lemme qui suit montre que, dans le cas d'une insertion réussie, lorsque w est sec, les conditions d'applications du théorème 4.16 sont remplies.

Lemme 5.25 *Si w est sec et vérifie les conditions du théorème 5.8, alors la racine r de $T^m(G[w^*])$ est non coupée et $R'_s = w^* \cup \{x\}$ (voir définition 4.10 et notation 4.3 page 162).*

Preuve :

Au cours de la preuve, nous aurons besoin de la propriété suivante.

Proposition 5.14 *Si le noeud d'insertion w vérifie les conditions du théorème 5.8, alors x n'a pas de faux jumeau dans $G'[(w^* \setminus X_w) \cup \{x\}]$.*

Preuve : Supposons que x a un jumeau $y \in Y_w$. Comme $C_\Delta(y) \cap I_w = \emptyset$ et comme w vérifie la condition 3 du théorème 5.8, alors, à renversement près de σ , il existe un noeud $v \in \mathcal{C}(w)$ tel que $v > e_y^2$ et v vérifie la propriété gauche et $\forall u \in \mathcal{C}(w), u < v \rightarrow u$ est creux. D'après le lemme 5.3, $e_y^{1*} \neq \emptyset$ ou $\exists y_2 \in Y_w, e_{y_2}^2 = e_y^{1*}$. On note z un élément de e_y^{1*} ou un sommet de Y_w tel que $e_z^2 = e_y^{1*}$. Dans tous les cas, z n'est pas adjacent à x et est adjacent à y : absurde. $\square$

1. w est premier et $X_w \cap N(x) = \emptyset$ et $w^* \setminus X_w$ est mixte, et x n'a pas de vrai jumeau dans $G'[(w^* \setminus X_w) \cup \{x\}]$. D'après la proposition 5.14, x n'a pas de faux jumeau dans $G'[(w^* \setminus X_w) \cup \{x\}]$.

- Si $X_w = \emptyset$, alors, d'après le lemme 5.11, r est un noeud premier. Si r a au moins un fils mixte, alors r est coupé. Si r n'a pas de fils mixte, comme x n'a pas de jumeau dans $G'[(w^* \setminus X_w) \cup \{x\}]$, alors x n'a pas de jumeau dans G_r , et r est donc non coupé. Comme r est un noeud premier, $R'_s = w^* \cup \{x\}$.
- Si $X_w \neq \emptyset$, alors, d'après le lemme 5.11, r est un noeud série dont les fils sont les feuilles correspondant aux sommets de X_w et un noeud premier u tel que $U = w^* \setminus X_w$. Comme $X_w \subseteq \bar{N}(x)$, tous les fils de r sont creux sauf u qui est mixte. Donc r est non coupé et $R'_s = w^* \cup \{x\}$.
- 2. w est dégénéré et $X_w = \emptyset$ et w n'a aucun fils creux. Dans ce cas, r est un noeud parallèle n'ayant aucun fils creux. r est donc non coupé et $R'_s = w^* \cup \{x\}$.
- 3. w est dégénéré et $X_w \neq \emptyset$ et $X_w \cap N(x) = \emptyset$. Comme B_w n'est pas pleine, w vérifie la condition 3a du théorème 5.8. Donc $w^* \setminus X_w$ n'est pas plein. Dans ce cas r est un noeud série dont les fils sont les feuilles correspondant aux sommets de X_w et un noeud u tel que $U = w^* \setminus X_w$. Donc aucun fils de r n'est plein : r est non coupé et $R'_s = w^* \cup \{x\}$.

■

Du lemme 5.25 et du théorème 4.16 on déduit que lorsque w est sec, la racine w' de $T^c(G'[w^* \cup \{x\}])$ est un noeud premier tel que $X_{w'} = \emptyset$. Pour déterminer entièrement $T^c(G'[w^* \cup \{x\}])$, il suffit de déterminer les fils v_i de w' , les $PQ(G[v_i^*])$, les sommets de $Y_{w'}$ et leurs pointeurs secondaires vers les fils de w' . Il convient d'abord d'identifier les ensembles v_i^* ainsi que les classes de vrais jumeaux des sommets de $Y_{w'}$, c'est à dire les sous-ensembles de sommets de $Y_{w'}$ qui ont les mêmes pointeurs secondaires. En fait, d'après le théorème 5.6, les v_i^* et les classes de vrais jumeaux des sommets de $Y_{w'}$ sont exactement les modules forts maximaux de $G'[w^* \cup \{x\}]$. Comme w est sec, d'après le lemme 5.25, la racine r de $T^m(G[w^*])$ est non coupée et $R'_s = w^* \cup \{x\}$, et le théorème 4.16 nous donne que les modules forts maximaux de $G'[w^* \cup \{x\}]$ sont le singleton $\{x\}$ et les modules uniformes maximaux de $G[w^*]$.

Modules uniformes maximaux de $G[w^*]$

Les deux graphes de la notation suivante jouent un rôle central.

Notation 5.13 On définit les deux graphes $\tilde{G}'_w = G'[w^* \cup \{x\}] / (\mathcal{MUM}(G[w^*]) \cup \{\{x\}\})$ et $\tilde{G}_w = G[w^*] / \mathcal{MUM}(G[w^*])$.

Pour construire $PQ(G'[w^* \cup \{x\}])$, en tenant compte de ce qu'on vient de voir et en utilisant le théorème de composition des PQ -représentations (théorème 5.4 page 201), il nous suffit de connaître $PQ(\tilde{G}'_w)$ ainsi que tous les modules uniformes maximaux M de $G[w^*]$ et pour chacun d'eux $PQ(G[M])$. Nous verrons plus loin comment obtenir $PQ(\tilde{G}'_w)$, nous aurons besoin à cette fin de $PQ(\tilde{G}_w)$.

Dans ce qui suit, on s'attache donc à trouver les modules uniformes maximaux M de $G[w^*]$ et à obtenir pour chacun d'eux $PQ(G[M])$ et $PQ(G[w^*]/\{M\})$. Comme les modules

uniformes maximaux sont deux à deux disjoints, en itérant ces opérations de quotients sur les PQ -représentations obtenues, on finit par obtenir $PQ(G[w^*]/\mathcal{MUM}(G[w^*]))$. En même temps, au cours du processus on a trouvé tous les modules uniformes maximaux M et leurs $PQ(G[M])$.

Concentrons nous sur une étape qui consiste à identifier un module uniforme maximal M de $G[w^*]$, $PQ(G[M])$ et $PQ(G[w^*]/\{M\})$. Le lemme 5.26, qui se déduit très facilement du théorème 5.6, montre comment identifier les modules uniformes maximaux sur la PQ -représentation.

Notation 5.14 Pour un noeud u de $T^c(G)$, on note $\mathcal{S}_l(u) = \{v \in \mathcal{C}(u) \mid v^* \text{ est plein}\}$ et $\mathcal{S}_{nl}(u) = \{v \in \mathcal{C}(u) \mid v^* \text{ est creux}\}$.

Lemme 5.26 Soit G un graphe d'intervalles et x un sommet à y insérer. M est un module uniforme maximal de G ssi il existe un noeud u de $T^c(G)$ qui satisfait une des conditions suivantes :

1. $\text{parent}(u)$ est premier et mixte et u^* est uniforme et $M = u^*$; ou
2. u est premier et mixte et $\exists v_1, v_2 \in \mathcal{C}(u), M = N(x) \cap \{y \in Y_u \mid e_y^1 = v_1 \text{ et } e_y^2 = v_2\}$ ou $M = \overline{N}(x) \cap \{y \in Y_u \mid e_y^1 = v_1 \text{ et } e_y^2 = v_2\}$; ou
3. u est dégénéré et (u possède des fils non pleins et $M = \bigcup_{v \in \mathcal{S}_l(u)} v^*$) ou (u possède des fils non creux et $M = \bigcup_{v \in \mathcal{S}_{nl}(u)} v^*$) ; ou
4. X_u n'est pas uniforme et ($(u^* \setminus X_u \text{ est plein et } M = (X_u \cap N(x)) \cup (u^* \setminus X_u))$ ou $(u^* \setminus X_u \text{ est creux et } M = (X_u \cap \overline{N}(x)) \cup (u^* \setminus X_u))$) ; ou
5. $u^* \setminus X_u$ n'est pas plein et $M = X_u \cap N(x)$, ou $u^* \setminus X_u$ n'est pas creux et $M = X_u \cap \overline{N}(x)$.

Pour chaque condition, un exemple de module uniforme maximal la satisfaisant est donné sur la figure 5.32.

Remarque 5.10 Les conditions du lemme 5.26 sont mutuellement exclusives. C'est à dire que pour un module M donné, il n'existent pas deux noeuds u et v tels que u et M vérifie une condition et v et M en vérifient une autre.

On montre maintenant, pour un module uniforme maximal M donné, comment obtenir $PQ(G[M])$ et $PQ(G[w^*]/\{M\})$. On désigne par a le sommet représentatif de M dans $G[w^*]/\{M\}$. La numérotation des cas reprend celle du lemme 5.26.

1. $PQ(G[M]) = PQ_u(G)$ et on obtient $PQ(G[w^*]/\{M\})$ en remplaçant, dans $PQ_w(G)$, u par une feuille sur laquelle pointe a .
2. M étant une clique, $PQ(G[M])$ est réduit à une feuille sur la quelle pointent tous les sommets de M . Pour obtenir $PQ(G[w^*]/\{M\})$ il suffit de supprimer de $PQ_w(G)$ tous les sommets de $M \setminus \{a\}$.
3. Si $M = \bigcup_{v \in \mathcal{S}_l(u)} v^*$, on distingue deux sous-cas.
 - Si $|\mathcal{S}_l(u)| = 1$, alors en notant v l'unique élément de $\mathcal{S}_l(u)$, on a $PQ(G[M]) = PQ_v(G)$. On obtient $PQ(G[w^*]/\{M\})$ en remplaçant, dans $PQ_w(G)$, v par une feuille sur laquelle pointe a .

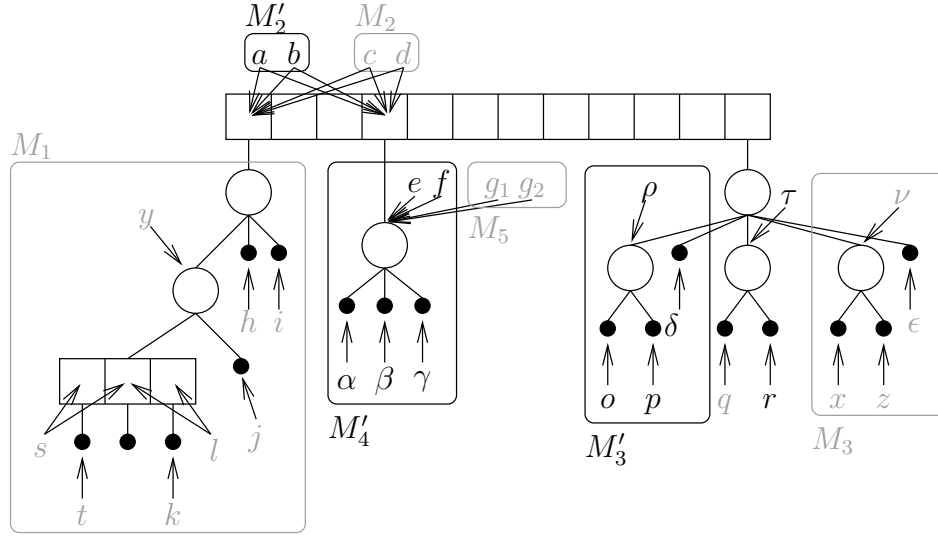


FIG. 5.32 – Les différentes conditions du lemme 5.26. Chaque ensemble M_i ou M'_i , pour $i \in \llbracket 1, 5 \rrbracket$, est un module uniforme maximal satisfaisant la condition i du lemme. Les sommets en gris sont adjacents à x , ceux en noir ne le sont pas. La PQ -représentation n'est que partiellement dessinée.

- Si $|\mathcal{S}_l(u)| > 1$, alors on obtient $PQ(G[M])$ à partir de $PQ_u(G)$, en retirant les fils de u (et leurs sous-arbres) qui appartiennent à $\mathcal{S}_{nl}(u)$ et les sommets de X_u . Pour obtenir $PQ(G[w^*]/\{M\})$ à partir de $PQ_w(G)$, on remplace dans $\mathcal{C}(u)$ les noeuds de $\mathcal{S}_{nl}(u)$ par une feuille sur laquelle pointe a .

Le cas où $M = \bigcup_{v \in \mathcal{S}_l(u)} v^*$ se traite de manière identique.

4. Si $u^* \setminus X_u$ est plein (resp. creux), on obtient $PQ(G[M])$ à partir de $PQ(G[u^*])$ simplement en retirant les sommets de $X_u \cap \overline{N}(x)$ (resp. $X_u \cap N(x)$). Pour obtenir $PQ(G[w^*]/\{M\})$ à partir de $PQ_w(G)$, on remplace u par une feuille sur laquelle pointe a et les sommets de $X_u \cap \overline{N}(x)$ (resp. $X_u \cap N(x)$).
5. M étant une clique, $PQ(G[M])$ est réduit à une feuille sur la quelle pointent tous les sommets de M . Pour obtenir $PQ(G[w^*]/\{M\})$ il suffit de supprimer de $PQ_w(G)$ tous les sommets de $M \setminus \{a\}$.

Déterminer $PQ(\tilde{G}'_w)$

On s'intéresse maintenant à l'obtention de $PQ(\tilde{G}'_w)$. Il est facile de vérifier dans la discussion qui précède que, pour un module uniforme M , si $T^c(G)$ vérifie les conditions du théorème 5.8, alors $T^c(G/\{M\})$ les vérifie également. Donc, si $T^c_w(G)$ vérifie les conditions du théorème 5.8, $T^c(\tilde{G}_w)$ également. Remarquons que le quotient par un module uniforme est compatible avec le voisinage de x . On peut donc considérer l'insertion de x dans $\tilde{G}_w$, et on a $(G[w^*]/\text{MUM}(G[w^*])) + x = G[w^* \cup \{x\}]/\text{MUM}(G[w^*])$, c'est à dire $\tilde{G}_w + x = \tilde{G}'_w$.

La preuve de la suffisance des conditions du théorème 5.8 est constructive, en l'appli-

quant à $T^c(\tilde{G}_w)$ on obtient un modèle d'intervalles de $\tilde{G}_w + x = \tilde{G}'_w$. D'après le théorème 4.16, $\tilde{G}'_w$ est un graphe premier. La MD -représentation d'un graphe d'intervalles premier consiste en un unique noeud interne premier dont les fils sont les feuilles correspondant aux sommets du graphe, et dont le modèle d'intervalle associé est précisément le modèle d'intervalle de G . On connaît donc la MD -représentation de $\tilde{G}'_w$ et grâce à la section 5.3.3, on sait en déduire sa PQ -représentation $PQ(\tilde{G}'_w)$.

Enfin, grâce au théorème 5.4, en recollant les $PQ(G[M])$, pour $M \in \mathcal{MUM}(G[w^*])$, et $PQ(\tilde{G}'_w)$ on obtient $PQ(G'[w^* \cup \{x\}])$. Il nous reste à vérifier que l'on peut faire tous les calculs dans la complexité désirée de $O(n)$: c'est l'objet de la section suivante.

5.5.5 Algorithme et complexité

La première étape de notre algorithme détermine le noeud d'insertion w et rassemble certaines informations sur $T^c(G)$. La deuxième étape vérifie si $T^c(G)$ satisfait les conditions du théorème 5.8. La troisième et dernière étape met $T^c(G)$ à jour en construisant $T^c(G')$.

Pour l'analyse de complexité, l'argument principalement utilisé est que le nombre de noeuds de $T^c(G)$ est $O(n)$, car tout noeud interne a au moins deux fils, et que $\sum_{u \in T^c(G)} |X_u| + |Y_u| = O(n)$, car un sommet x de G est présent dans exactement un X_u et au plus un Y_u . Ainsi, un parcours de $T^c(G)$ qui traite une seule fois chaque noeud u et en temps $O(|\mathcal{C}(u)| + |X_u| + |Y_u|)$, a une complexité totale de $O(n)$.

Première étape.

On détermine d'abord pour chaque noeud de T^c s'il est plein, mixte ou creux par un procédé de marquage classique de bas en haut dans l'arbre, qui est le pendant du procédé de marquage de [MS89] de l'arbre de décomposition modulaire, en temps $O(n)$. Pour initialiser le procédé, on parcourt les sommets du graphe et on détermine pour chaque noeud u de T^c si X_u est plein et non vide, si X_u est creux et non vide, si X_u est mixte ou si X_u est vide. De plus, on attribue à chaque feuille l de T^c un type :

- *plein* si X_l est plein et non vide,
- *mixte* si X_l est mixte,
- *creux* si X_l est creux et non vide,
- *desert* si X_l est vide.

A partir de là commence le parcours de bas en haut dans l'arbre. Chaque noeud communique son type à son père. En fonction du type de ses fils, un noeud u reçoit lui même un type :

1. *plein* si $X_u \cup Y_u$ est plein et si ses fils sont tous typés *plein* ou *desert*,
2. *creux* si $X_u \cup Y_u$ est creux et si ses fils sont tous typés *creux* ou *desert*,
3. *mixte* dans tous les autres cas.

Cela prend un temps $O(n)$.

Ensuite, on détermine le noeud d'insertion w en suivant un chemin de la racine à w . Tant que le noeud u visité est propre, on visite son unique fils non creux si u est dégénéré

ou son unique fils v tel que $\Delta_v = Y_u \cap N(x)$ si u est premier. Le premier noeud non propre rencontré est w . Pour que la localisation de w coûte un temps $O(n)$, il suffit de pouvoir déterminer si un noeud est propre et trouver, le cas échéant, son fils qui est le prochain noeud à examiner en temps $O(|X_u \cup Y_u| + |\mathcal{C}(u)|)$. Lors du déroulement de l'algorithme de recherche de w , le noeud u examiné est la racine ou son père est propre. Ainsi, il suffit de vérifier d'une part que X_u est plein (cette information a déjà été calculé dans le procédé de marquage) et d'autre part que :

- u est dégénéré et u possède un unique fils non creux ; ou
- u est premier et u possède un fils v tel que $\Delta_v = Y_u \cap N(x)$ et les autres fils de u sont creux.

Si u est dégénéré, on parcourt la liste des fils de u pour vérifier qu'ils sont tous creux sauf un, ce qui permet au passage de déterminer le prochain noeud à examiner si u est propre, cela prend un temps $O(|\mathcal{C}(u)|)$. Dans le cas où u est premier, pour vérifier l'existence de v , on parcourt la liste des sommets liés de Y_u (déterminée lors du procédé de marquage) en calculant l'intersection de leurs intervalles dans σ_u . Le résultat est positif ssi l'intersection finale est exactement un fils de u : c'est le sommet v cherché. Cela prend un temps $O(|Y_u|)$, et en $O(|\mathcal{C}(u)|)$, on vérifie que les autres fils de u sont creux.

Ainsi, dans tous les cas, l'examen de u peut se faire en temps $O(|Y_u| + |\mathcal{C}(u)|)$ et on trouve donc w en temps $O(n)$.

Enfin, un simple parcours de l'arbre de haut en bas permet de déterminer pour chaque noeud u si B_u est pleine, mixte ou aucun des deux. Dans la suite, on fait référence à cette information comme étant l'**état de branche** du noeud u .

Le temps total de l'exécution de la première étape de l'algorithme est donc $O(n)$.

Deuxième étape.

Grâce aux types et aux états de branche attribués aux noeuds de T^c lors de la première étape, on peut vérifier si T^c satisfait les conditions du théorème 5.8 en temps $O(n)$. Pour cela, on montre que pour un noeud $u \in T_w^c$, on peut vérifier chacune des conditions du théorème en temps $O(|X_u| + |Y_u| + |\mathcal{C}(u)|)$.

1. Pour vérifier la condition 1, il suffit de consulter l'état de branche de u et le type des fils de u , cela prend un temps $O(|\mathcal{C}(u)|)$.
2. Pour tester la condition 2, le plus difficile est de vérifier que l'ensemble des fils saturés de u forme un intervalle. Dans ce but, on détermine l'ensemble S_u des fils v de u tels que $\Delta_v \subseteq N(x)$. Par définition, $S_u = \{v \in \mathcal{C}(u) \mid \forall y \in Y_u \cap \overline{N}(x), v \notin \llbracket e_y^1, e_y^2 \rrbracket\}$. Donc, en posant $\mathcal{I} = \{\llbracket e_y^1, e_y^2 \rrbracket \mid y \in Y_u \cap \overline{N}(x)\}$, on a $S_u = \mathcal{NC}(\sigma, \mathcal{I})$ (voir notation 3.3 page 113). Ainsi, en utilisant la procédure *PartNonCouv* (voir page 113), on obtient la partition intervallaire (définition 3.6 page 113) de S_u .

L'ensemble des fils saturés de u sont les noeuds pleins de S_u . Une fois déterminé S_u , il faut donc vérifier que les noeuds pleins de S_u forment un intervalle, ce qui se fait en temps $O(|S_u|) = O(|\mathcal{C}(u)|)$. S'ils ne forment pas un intervalle, u ne vérifie pas la condition 2. S'ils forment un intervalle, alors on vient d'identifier I_u et il suffit de

parcourir les fils de u pour vérifier que les noeuds de $I_u \setminus \{f_u, l_u\}$ sont creux, ce qui prend un temps $O(|\mathcal{C}(u)|)$.

La condition 2 peut donc être testée en temps $O(|\mathcal{C}(u)| + |Y_u|)$.

3. Tester si un fils de w satisfait la propriété gauche ou droite est très facile en examinant les pointeurs de tous les sommets de Y_w , cela prend un temps $O(|Y_w|)$, ce qui est acceptable pour nous. Par conséquent, une fois déterminé I_w , ce qui a été fait en vérifiant que w satisfait la condition 2, on peut tester si w satisfait les conditions 3a, 3b et 3(c)i en temps $O(|\mathcal{C}(w)| + |Y_w|)$.

Tester si w vérifie la condition 3(c)ii est plus délicat et demande un examen particulier. Ce qui suit est parfaitement similaire au test de la condition 2 du théorème 3.2 qui est décrit page 114. Pour éviter au lecteur la gymnastique d'un changement de contexte et de formalisme, on retranscrit ci-dessous le raisonnement dans le cadre qui nous occupe.

Remarquons d'abord que si un fils w_1 de w vérifie la propriété gauche, alors $\forall v \in \mathcal{C}(w), v \leq_{\sigma_w} w_1 \Rightarrow v$ vérifie la propriété gauche. De plus, si w_1 vérifie la propriété gauche stricte (voir définition 5.9), alors $\forall v \in \mathcal{C}(w), w_1 <_{\sigma_w} v \Rightarrow v$ ne vérifie pas la propriété gauche. Par symétrie, on a les assertions correspondantes pour la propriété droite. On note $w_f = \min\{e_y^2 \mid y \in Y_w \text{ et } y \text{ est lié}\}$, et $w_l = \max\{e_y^1 \mid y \in Y_w \text{ et } y \text{ est lié}\}$. D'après ce qui précède, l'ensemble des fils de u vérifiant la propriété gauche est $\{v \in \mathcal{C}(w) \mid v \leq_{\sigma_w} w_f\}$ et l'ensemble des fils de w vérifiant la propriété droite est $\{v \in \mathcal{C}(w) \mid w_l \leq_{\sigma_w} v\}$. Trouver w_f et w_l se fait en parcourant les sommets de Y_w en temps $O(|\mathcal{C}(w)|)$.

- Examinons le cas où $w_f < w_l$. Si w_f n'est pas le prédécesseur de w_l , alors w ne satisfait pas la condition 3(c)ii. Si au contraire w_f est le prédécesseur de w_l , alors on teste si w satisfait la condition 3(c)ii en parcourant les sommets de Y_w et en vérifiant qu'aucun sommet non lié ne couvre w_f et w_l simultanément, puis en parcourant $\mathcal{C}(w)$ pour vérifier que les noeuds de $\mathcal{C}(w) \setminus \{w_f, w_l\}$ sont creux.
- Dans le cas où $w_l \leq w_f$, il peut y avoir plusieurs couples de sommets successeurs prétendant à être le couple (f, l) . On note k la longueur de l'intervalle $\llbracket w_l, w_f \rrbracket$ et $\{w_i\}_{1 \leq i \leq k}$, avec $w_l = w_1 <_{\sigma_w} w_2 <_{\sigma_w} \dots <_{\sigma_w} w_k = w_f$. Lorsqu'ils existent, on note w_0 le prédécesseur de w_l et w_{k+1} le successeur de w_f . Les couples possibles pour (f, l) sont donc les couples (w_i, w_{i+1}) pour $0 \leq i \leq k$. Si w satisfait la condition 3(c)ii alors au moins un de ces couples n'est couvert par aucun sommet non lié de Y_w .

Ici, on veut déterminer l'ensemble R_w des noeuds $v \in \llbracket w_0, w_k \rrbracket$ tels que v et son successeur ne sont simultanément couverts par aucun sommet non lié de Y_w . En posant $\mathcal{I} = \{\llbracket e_y^1, \text{pred}(e_y^2) \rrbracket \mid y \in Y_u \cap \overline{N}(x)\}$, on a $R_w = \mathcal{NC}(\llbracket w_0, w_k \rrbracket, \mathcal{I})$. Un appel à *PartNonCouv* $(\llbracket w_0, w_k \rrbracket, \mathcal{I})$ détermine la partition intervallaire de $\mathcal{NC}(\llbracket w_0, w_k \rrbracket, \mathcal{I})$ en temps $O(k + |Y_w|)$. Si R_w est creux, alors w ne vérifie pas la condition 3(c)ii. Dans le cas contraire, R_w nous décrit l'ensemble des couples (f, l) pouvant satisfaire la condition 3(c)ii. Il suffit alors de parcourir les fils de w pour vérifier qu'il en possède au plus deux non creux.

- Si w a exactement deux fils non creux, on vérifie qu'ils sont consécutifs dans σ_w et que le premier appartient à R_w .
- Si w a un unique fils v non creux, on vérifie que v ou son prédécesseur appartient à R_w .
- Si w n'a que des fils creux, il vérifie la condition 3(c)ii.

Ainsi, le temps total de calcul du test de la condition 3(c)ii est donc $O(|\mathcal{C}(w)| + |Y_w|)$.

4. Tester la condition 4 n'est pas plus difficile que tester les conditions 3a, 3b et 3(c)i. Un temps $O(|Y_u| + |\mathcal{C}(u)|)$ suffit.

Finalement, comme toutes les conditions peuvent être testées pour un noeud u en $O(|Y_u| + |\mathcal{C}(u)|)$, on peut déterminer si $G + x$ est un graphe d'intervalles en temps $O(n)$.

Troisième étape.

Le but de cette étape est la construction de $T^c(G')$. La discussion tenue en page 244 a montré qu'il existe essentiellement deux cas de figure selon la configuration de w et de ses fils :

- d'un côté, il y a les cas dans lesquels on peut construire $T^c(G')$ par quelques manipulations assez minimales sur $T^c(G)$. Ces cas sont les cas 1, 2, 3a, 3(b)i, 3(b)ii, 3(b)iiA et 3(c)i. Pour pouvoir déceler ces cas et les traiter, il faut avoir la liste des sommets de $X_w \cap N(x)$ et de ceux de $X_w \cap \overline{N}(x)$. Ces listes ont été obtenues lors du procédé de marquage dans la première étape de l'algorithme. Il faut aussi parcourir les fils de w en inspectant leur type, cela prend un temps $O(|\mathcal{C}(w)|)$. Il faut également pouvoir déterminer si x possède un jumeau parmi les sommets de Y_w . Pour cela, on parcourt les fils de w dans l'ordre de σ_w en vérifiant que les fils pleins forment un intervalle et que les autres sont creux. Ensuite on cherche s'il existe un sommet de Y_w qui pointe sur le premier et le dernier fils plein de w . Cela demande un temps $O(|\mathcal{C}(w)| + |Y_w|)$. Donc, tous les cas précités peuvent se traiter dans cette complexité de $O(|\mathcal{C}(w)| + |Y_w|)$.
- de l'autre côté, se trouvent les cas dans lesquels $T^c(G)$ est profondément affecté par l'insertion de x . Dans ces cas, qui sont les cas 3(b)iiB, 3(c)ii et 3(c)iii de la discussion page 244, une partie de $T^c(G)$ se contracte en un noeud premier que l'on note w' dans $T^c(G')$. On a montré en page 244 comment obtenir la partie de $T^c(G')$ autre que $T_{w'}^c$. Cela se fait par quelques manipulations similaires à celles vues ci-dessus pour lesquelles il est aisé de vérifier qu'elles prennent un temps total $O(|\mathcal{C}(w)| + |Y_w|)$. On se ramène ainsi à étudier les cas où w est sec. Lorsque w est sec, nous avons vu que les modules forts maximaux de $G'[w^* \cup \{x\}]$ sont les modules uniformes maximaux de G . Nous avons vu comment les reconnaître et comment pour un de ces modules M obtenir $PQ(G[M])$. Grâce aux types attribués aux noeuds et aux listes des sommets de $X_u \cap N(x)$ calculées pour chaque noeud $u \in T^c(G)$ lors de la première étape, il est aisé d'effectuer un parcours de $T_w^c(G)$ où chaque noeud u est traité en temps $O(|\mathcal{C}(u)| + |Y_u|)$ pour :
 - trouver tous les modules uniformes maximaux M de G , et

- calculer $PQ(G[M])$ pour chacun d'eux, et
- obtenir $PQ(\tilde{G}_w)$.

Ce parcours prend donc un temps total $O(n)$.

Comme remarqué page 253, si $T_w^c(G)$ vérifie les conditions du théorème 5.8, alors $T^c(\tilde{G}_w)$ les vérifie aussi. En appliquant à $T^c(\tilde{G}_w)$ la construction de la preuve de la suffisance des conditions du théorème 5.8, on obtient un modèle d'intervalles de $\tilde{G}_w + x = \tilde{G}'_w$. Comme expliqué page 253, il est très facile d'obtenir la MD -représentation d'un graphe premier à partir d'un de ses modèles d'intervalles. Ainsi, on obtient $MD(\tilde{G}'_w)$ en temps $O(|V(\tilde{G}'_w)| = O(n))$ à partir de son modèle d'intervalles. Or, grâce à la section 5.3.3, on sait obtenir la PQ -représentation d'un graphe d'intervalles à partir de sa MD -représentation. Cette transformation appliquée à $MD(\tilde{G}'_w)$ prend un temps $O(|V(\tilde{G}'_w)| = O(n))$. Enfin, d'après le théorème 5.4, on peut construire $PQ(G'[w^* \cup \{x\}])$ à partir de $PQ(G'[w^* \cup \{x\}]/\mathcal{MUM}(G'[w^*])) = PQ(\tilde{G}'_w)$ et des $PQ(G[M])$, pour $M \in \mathcal{MUM}(G[w^*])$. Cette construction prend un temps $O(|\mathcal{C}(r)| + \sum_{M \in \mathcal{MUM}(G[w^*])} |\mathcal{C}(r_M)|)$, où r désigne la racine de $T^c(\tilde{G}'_w)$ et r_M celle de $T^c(G[M])$. Cette dernière étape de calcul est donc bornée en temps par $O(n)$.

On le voit, la complexité finale de $O(n)$ repose entièrement sur le fait qu'on puisse calculer un modèle d'intervalles de $\tilde{G}_w + x$ dans cette complexité.

Le but que nous poursuivons maintenant est donc de calculer un modèle d'intervalles de $\tilde{G}_w + x$ en temps $O(n)$. Comme annoncé, nous utilisons la construction de la preuve de la suffisance des conditions du théorème 5.8. Tout l'enjeu est donc de vérifier que l'on peut implémenter cette construction en temps $O(|w^*|)$, ce qui donne une complexité de $O(|V(\tilde{G}_w)|) = O(n)$ lorsqu'on l'applique à $\tilde{G}_w$ plutôt qu'à $G[w^*]$.

Le processus de construction d'un modèle d'intervalle de $G[w^*] + x$ est un parcours de T_w^c de bas en haut. On peut y discerner trois phases :

1. la construction du modèle d'intervalles de $G[w^*]$ pour chaque noeud $u \in T_w^c$ qui est uniforme et dont le père est mixte,
2. la construction du modèle d'intervalles de $G[u^*] + x$ pour chaque noeud mixte $u \in T_w^c \setminus \{w\}$ en partant des noeuds mixtes les plus bas dans l'arbre et en remontant vers w ,
3. la construction de $G[w^*] + x$.

Détaillons pour chacune de ces étapes les opérations à réaliser et le temps de calcul nécessaire.

1. pour un noeud uniforme $u \in T_w^c$ dont le père est mixte, on peut obtenir un modèle d'intervalles de $G[u^*]$ en parcourant T_u^c , comme expliqué en section 5.3.2.
2. pour un noeud mixte $u \in T_w^c \setminus \{w\}$, la construction du modèle d'intervalles de $G[u^*] + x$ repose sur les opérations suivantes :
 - (a) dans le cas où u est dégénéré :
 - trouver les fils creux, les fils pleins et le fils mixte de u . Cela prend un temps $O(|\mathcal{C}(u)|)$. Rappelons que les modèles d'intervalles de $G[v^*]$ pour chacun des

filis v de u ont été calculés lors de la première phase pour les noeuds uniformes et au cours de la deuxième phase, mais antérieurement, pour le fils mixte.

- mettre les modèles des fils de u bout à bout. Cela prend encore un temps $O(|\mathcal{C}(u)|)$. Il n'est pas nécessaire de changer les pointeurs des sommets de $u^* \setminus X_u$.
- attribuer les pointeurs appropriés aux sommets de $X_u \cup \{x\}$. Cela requiert un temps $O(|X_u|)$. En effet, on peut attribuer ses pointeurs à un sommet $z \in X_u \cup \{x\}$, en temps constant. Pour cela, il suffit d'attribuer à chaque fils v de u , lorsqu'on met leurs modèles d'intervalles respectifs bout à bout, deux pointeurs vers la première et la dernière clique du modèle de $G[v^*]$.

Le temps total de traitement d'un noeud dégénéré est donc $O(|\mathcal{C}(u)| + |X_u|)$.

(b) dans le cas où u est premier :

- identifier I_u et f_u . Ce travail a déjà été fait lors de la deuxième étape de l'algorithme d'insertion pour savoir si le nouveau graphe est un graphe d'intervalles.
- mettre bout à bout les modèles d'intervalles des fils de u . Cela prend un temps $O(|\mathcal{C}(u)|)$. Il n'est pas nécessaire de changer les pointeurs des sommets de $u^* \setminus (X_u \cup Y_u)$.
- décider de l'insertion de nouvelles cellules dans le modèle d'intervalle. Pour cela on a besoin de tester les conditions suivantes, avec les notations du théorème 5.8 :
 - f_u satisfait la propriété gauche stricte : un parcours de Y_u suffit à le déterminer.
 - $\Delta_f \cup X_u \subseteq N(x)$: comme on connaît depuis la première étape de l'algorithme les ensembles $X_u \cap N(x)$ et $X_u \setminus N(x)$, il suffit de déterminer si $\Delta_f \subseteq N(x)$. Un parcours de Y_u répond à la question.
 - $f^* = \emptyset$.
 - $K_x \setminus \{x\}$ est maximale dans $G[f^*]$ (resp. $G[f_u^*]$), dans le cas où f (resp. f_u) est mixte. Cette information peut être précalculée lors de la première étape de l'algorithme. En effet, pour un noeud mixte $u \in T_w^c(G)$ tel que B_u est non pleine, $K_x \setminus \{x\}$ est maximale dans $G[u^*]$ ssi il existe une feuille l de $T_u^c(G)$ telle que la branche de l dans $T_u^c(G)$ est pleine. Ainsi, dans la première étape on peut effectuer un parcours de bas en haut de $T^c(G)$ en partant des feuilles marquées *plein* et en remontant leur branche tant qu'elle est pleine. Cela prend un temps $O(n)$ pour tout l'arbre.
- attribuer les pointeurs appropriés aux sommets de $X_u \cup Y_u \cup \{x\}$. Cela peut se faire en temps $O(|X_u| + |Y_u|)$. Par un parcours des sommets de Y_u , on peut tester en temps constant si le sommet y considéré appartient à un des ensembles $\Delta_{f_u} \cap N(x)$, $\Delta_{f_u} \setminus N(x)$, $\Delta_f \cap N(x)$, $\Delta_f \setminus N(x)$ (utilisés dans la preuve du théorème 5.8) et choisir les pointeurs à attribuer à y en conséquence.

3. en ce qui concerne la construction de $G[w^*] + x$, les méthodes décrites ci-dessus pour le cas d'un noeud $u \neq w$ fonctionnent encore. Le fait que w puisse avoir deux fils

mixtes ne change ni la façon de procéder ni la complexité. On traite w en temps $O(|\mathcal{C}(w)| + |X_w| + |Y_w|)$.

De cette manière, le parcours de T_w^c de bas en haut permettant d'obtenir le modèle d'intervalles de $G[w^*] + x$ prend un temps $O(|w^*|) = O(n)$. Il est essentiel de remarquer que les structures représentant les différents modèles d'intervalles obtenus le long du parcours de T_w^c sont des listes dont les cellules ne contiennent pas leur rang dans la liste. L'utilisation de listes permet de faire l'opération de mise bout à bout de deux modèles en temps constant. Le fait que l'on ne numérote pas les cellules dans les modèles d'intervalles intermédiaires est absolument nécessaire pour garantir la complexité : si on re-numérote les cellules dans chaque modèle d'intervalles intermédiaire, on obtient une complexité quadratique ! Par contre, une fois calculée la liste implémentant le modèle d'intervalles de $G[w^*] + x$, on peut la parcourir pour numéroter ses cellules et cela prend un temps $O(n)$ seulement.

Perspectives

Les contributions de ce mémoire ont été pointées en introduction et développées dans les conclusions de chacun des chapitres et sections correspondant. Je ne vais pas y revenir ici, cette conclusion générale s'attachera surtout à développer quelques perspectives ouvertes par ce travail de thèse.

Le chapitre 4 poursuivait le but d'entretenir dynamiquement l'arbre de décomposition modulaire avec une complexité de $O(n)$ par modification. Cet objectif n'a pu être atteint. Cependant, les résultats obtenus sont encourageants et mériteraient approfondissement. En particulier, le travail sur les graphes de permutation a montré qu'on peut traiter les noeuds premiers efficacement si on en possède une bonne représentation. Nous avons précisé, en section 4.6, ce qu'il fallait entendre par bonne représentation en donnant trois conditions nécessaires et suffisantes pour atteindre la complexité visée de $O(n)$. La question reste donc ouverte de trouver une représentation satisfaisant ces exigences pour les graphes quelconques.

Une étape intermédiaire intéressante serait de trouver une telle représentation pour les graphes de comparabilité de dimension bornée, puis pour les graphes de comparabilité en général. Cela étendrait le résultat sur les graphes de permutation qui sont les graphes de comparabilité de dimension 2. Cet espoir semble raisonnable au vu des liens très forts entre décomposition modulaire et orientation transitive [Gal67]. De plus, de même que les graphes de permutation premiers admettent un unique réalisateur, les graphes de comparabilité premiers admettent une unique orientation transitive. Cette observation est positive car cette propriété de l'unicité de la représentation d'un noeud premier était présente dans toutes les représentations arborescentes à degrés de liberté utilisées dans le manuscrit ; et on peut penser qu'elle joue un rôle important dans l'entretien dynamique de ces structures.

La question de l'entretien dynamique de la décomposition modulaire des graphes en général rejoint une autre question posée dans [Spi03] : celle de trouver un algorithme linéaire simple de décomposition modulaire. En effet, les algorithmes linéaires existants [MS99, CH94, DGM01] sont souvent jugés très complexes à analyser. Je suis convaincu qu'une compréhension de la structure des noeuds premiers suffisante pour permettre l'entretien entièrement dynamique de la décomposition modulaire en $O(n)$ par modification déboucherait sur une simplification des algorithmes statiques existants.

Une perspective motivante est de réaliser pour l'entretien dynamique de la décomposition en coupes un travail similaire à celui fait pour la décomposition modulaire. Cela a été commencé par [GP07] qui développe un algorithme, entièrement dynamique sur les

sommets, de reconnaissance de la classe des graphes distance héréditaires. Cette classe de graphe est celle des graphes entièrement décomposables par la décomposition en coupes. L'algorithme de [GP07] est basé sur l'arbre de décomposition en coupes et ressemble beaucoup à celui de [CPS85] pour les cographes. Se pose alors naturellement la question de l'entretien dynamique des graphes de cordes pour lesquels la décomposition en coupes joue le même rôle que la décomposition modulaire pour les graphes de permutation, au sens où elle en représente l'ensemble des modèles géométriques. Comment entretenir l'arbre de décomposition en coupes d'un graphe de cordes ? L'algorithme est-il beaucoup plus complexe que celui pour les permutations ? Peut-on obtenir l'algorithme des permutations comme cas particulier de celui des graphes de cordes (les graphes de cordes contiennent la classe des graphes de permutation) ?

En section 1.5, nous avons soulevé la question de savoir si le *PC*-arbre d'un graphe d'arcs de cercle de Helly a un rapport avec son arbre de décomposition en coupes. Que la réponse soit positive ou négative, il n'en reste pas moins qu'il serait intéressant de voir si les techniques employées pour le maintien dynamique du *PQ*-arbre d'un graphe d'intervalles peuvent se transposer sur le *PC*-arbre d'un graphe d'arcs de cercle.

Une problématique générale soulevée par le mémoire est celle de trouver des représentations arborescentes par degrés de liberté pour d'autres classes de graphes définies de manière géométrique et de voir si ces structures peuvent améliorer les performances algorithmiques atteintes sur ces classes. En particulier ces représentations sont naturellement désignées pour les problèmes d'optimisation sur l'ensemble des modèles géométriques d'un graphe. Par exemple, [GMW05] s'intéresse à trouver le dessin avec le plus petit nombre de croisement d'arêtes d'un graphe planaire augmenté d'une arête. L'algorithme est basé sur les *SPQR*-arbres qui sont une représentation de tous les plongements d'un graphe planaire. Un autre exemple similaire dans les graphes planaires est celui de [BM90] qui veut trouver un plongement qui minimise certaines distances.

Dans le chapitre 1, il a été fait un gros effort de présentation des décompositions modulaire, en coupes et en composantes triconnexes. Pour ces trois décompositions, on a montré qu'on pouvait définir leurs décompositions canoniques de manière très similaire. Cette définition repose essentiellement sur l'existence d'éléments de décomposition forts, c'est à dire qui ne sont pas sensibles à l'utilisation d'autres éléments de décomposition. Il serait très intéressant de pousser cette analogie jusqu'à définir un cadre formel d'un niveau d'abstraction supérieur dans le quel on pourrait définir une décomposition canonique et en démontrer certaines propriétés, et qui s'appliquerait aux trois décompositions précitées et à d'autres.

Bibliographie

- [BB97] Anne Berry and Jean-Paul Bordat. Decomposition by clique minimal separators. Research Report 97-213, LIRMM, 1997.
- [BCdMR05] A. Bergeron, C. Chauve, F. de Montgolfier, and M. Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In *13th European Symposium on Algorithm (ESA05)*, number 3669 in Lecture Notes in Computer Science, pages 779–790, 2005.
- [BEY98] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University press, 1998.
- [BG81] Philip A. Bernstein and Nathan Goodman. Power of natural semijoins. *SIAM J. Comput.*, 10(4) :751–771, 1981.
- [BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13(3) :335–379, 1976.
- [BLS99] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes : a Survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.
- [BM86] H.-J. Bandelt and H.M. Mulder. Distance hereditary graphs. *Journal of Combinatorial Theory Series B*, 41 :182–208, 1986.
- [BM88] Daniel Bienstock and Clyde L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM J. Comput.*, 17(1) :53–76, 1988.
- [BM90] Daniel Bienstock and Clyde L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1) :93–109, 1990.
- [Bou87] André Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7(3) :243–254, 1987.
- [BT96] Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5) :956–997, 1996.
- [Bun74] Peter Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9 :205–212, 1974.
- [BXHP05] B.M. Bui-Xuan, M. Habib, and C. Paul. Revisiting uno and yagiura’s algorithm. In *16th International Symposium on Algorithms and Computation*

- (ISAAC05), number 3827 in Lecture Notes in Computer Science, pages 146–155, 2005.
- [Cap97] C. Capelle. *DÃ©compositions de graphes et permutations factorisantes*. ThÃ©se de doctorat, UniversitÃ© Montpellier 2, janvier 1997.
- [CE80] W. H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3) :734–765, 1980.
- [CH94] Alain Cournier and Michel Habib. A new linear algorithm for modular decomposition. In *CAAP*, pages 68–84, 1994.
- [CHM81] Michel Chein, Michel Habib, and Marie-Catherine Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37 :35–50, 1981.
- [CI98] Alain Cournier and Pierre Ille. Minimal indecomposable graphs. *Discrete Mathematics*, 183(1-3) :61–80, 1998.
- [CLR94] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction Ã l’algorithmique*. Dunod, 1994.
- [CLSB81] D.G. Corneil, H. Lerchs, and L. Stewart-Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3 :163–174, 1981.
- [CO00] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3) :77–114, 2000.
- [Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.
- [Cou06] Bruno Courcelle. The monadic second-order logic of graphs xvi : Canonical graph decompositions. *Logical Methods in Computer Science*, 2(2), 2006.
- [Cou07] Bruno Courcelle. Circle graphs and monadic second-order logic. *Discrete Appl. Math.*, 2007. To appear in Journal of Applied Logic.
- [CP04] C. Crespelle and C. Paul. Fully dynamic recognition algorithm and certificate for directed cographs. In *30th Int. Workshop on Graph Theoretical Concepts in Computer Science (WG04)*, number 3353 in Lecture Notes in Computer Science, pages 93–104, 2004.
- [CP05] C. Crespelle and C. Paul. Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. In *31th Int. Workshop on Graph Theoretical Concepts in Computer Science (WG05)*, number 3787 in Lecture Notes in Computer Science, pages 38–48, 2005.
- [CP06] C. Crespelle and C. Paul. Fully dynamic recognition algorithm and certificate for directed cographs. *Discrete Applied Mathematics*, 154(12) :1722–1741, 2006.
- [CPS85] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear time recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4) :926–934, 1985.
- [CS87] V. Chvatal and N. Shibi. Bull-free berge graphs are perfect. *Graphs and Combinatorics*, 3 :127–139, 1987.

- [CSRT02] Maria Chudnovsky, Paul Seymour, Neil Robertson, and Robin Thomas. The strong perfect graph theorem. Manuscript, 2002.
- [CT07] D. Corneil and M. Tedder. An optimal, edges-only fully dynamic algorithm for distance-hereditary graphs. In *24th International Symposium on Theoretical Aspects of Computer Science (STACS)*, Lecture Notes in Computer Science, 2007.
- [Cun82] William H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algorithmic Discrete Methods*, 3(2) :214–228, 1982.
- [DGM01] Elias Dahlhaus, Jens Gustedt, and Ross M. McConnell. Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, 41(2) :360–387, 2001.
- [DHH96] Xiaotie Deng, Pavol Hell, and Jing Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM J. Comput.*, 25(2) :390–403, 1996.
- [dM03] Fabien de Montgolfier. *Décomposition modulaire des graphes - Théorie, extensions et algorithmes*. PhD thesis, Université Montpellier II, 2003.
- [EGIS98] D. Eppstein, Z. Galil, G. F. Italiano, and T. H. Spencer. Separator based sparsification ii : edge and vertex connectivity. *SIAM J. Comput.*, 28(1) :341–381, 1998.
- [EHR99] A. Ehrenfeucht, T. Harju, and G. Rozenberg. *The theory of 2-structures. A framework for decomposition and transformation of graphs*. World Scientific, 1999.
- [EIT⁺92] David Eppstein, Giuseppe F. Italiano, Roberto Tamassia, Robert Endre Tarjan, Jeffery Westbrook, and Moti Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1) :33–54, 1992.
- [EPL72] Shimon Even, Amir Pnueli, and Abraham Lempel. Permutation graphs and transitive graphs. *J. ACM*, 19(3) :400–410, 1972.
- [ER90a] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Primitivity is hereditary for 2-structures. *Theoretical Computer Science*, 70(3) :343–358, 1990.
- [ER90b] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Theory of 2-structures, part i : Clans, basic subclasses, and morphisms. *Theoretical Computer Science*, 70(3) :277–303, 1990.
- [FS89] Michael L. Fredman and Michael E. Saks. The cell probe complexity of dynamic data structures. In *STOC*, pages 345–354, 1989.
- [FW98] A. Fiat and G. J. Woeginger. *Online algorithmes : The state of the art*. Number 1442 in LNCS State of the art survey. Springer, 1998.
- [Gal67] T. Gallai. Transitiv orientierbare graphen. *Acta Math. Acad. Sci. Hungar.*, 18 :25–66, 1967.
- [Gav74a] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4 :357–369, 1974.

- [Gav74b] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory (B)*, 16 :47–56, 1974.
- [GH64] Paul C. Gilmore and Alan J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16 :539–548, 1964.
- [GI91] Zvi Galil and Giuseppe F. Italiano. Maintaining biconnected components of dynamic planar graphs. In *ICALP*, pages 339–350, 1991.
- [GI96] Dora Giammarresi and Giuseppe F. Italiano. Decremental 2- and 3-connectivity on planar graphs. *Algorithmica*, 16(3) :263–287, 1996.
- [GIS92] Zvi Galil, Giuseppe F. Italiano, and Neil Sarnak. Fully dynamic planarity testing. In *STOC*, pages 495–506, 1992.
- [GIS99] Zvi Galil, Giuseppe F. Italiano, and Neil Sarnak. Fully dynamic planarity testing with applications. *J. ACM*, 46(1) :28–91, 1999.
- [GMW05] Carsten Gutwenger, Petra Mutzel, and René Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4) :289–308, 2005.
- [Gol04] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier, second edition, 2004.
- [GP07] E. Gioan and C. Paul. Dynamic distance hereditary graphs using split decomposition. Technical Report 07007, LIRMM, March 2007.
- [GSH89] Csaba P. Gabor, Kenneth J. Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3) :435–473, 1989.
- [Haj57] G. Hajös. Über eine art von graphen. *Inter. Math. Nachr.*, 11, 1957.
- [HdLT98] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. In *STOC*, pages 79–89, 1998.
- [HF98] Monika Rauch Henzinger and Michael L. Fredman. Lower bounds for fully dynamic connectivity problems in graphs. *Algorithmica*, 22(3) :351–362, 1998.
- [HM90] P. Hammer and F. Maffray. Completely separable graphs. *Discrete Applied Mathematics*, 27(1-2) :85–99, 1990.
- [HM03] Wen-Lian Hsu and Ross M. McConnell. Pc trees and circular-ones arrangements. *Theoretical Computer Science*, 296(1) :99–116, 2003.
- [HoÃ83] C.T. HoÃ ng. A class of perfect graphs. M.sc. thesis, School of Computer Science, McGill University, Montreal, 1983.
- [HRS94] John Hershberger, Monika Rauch, and Subhash Suri. Data structures for two-edge connectivity in planar graphs. *Theor. Comput. Sci.*, 130(1) :139–161, 1994.
- [HSS02] P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM Journal on Computing*, 31(1) :289–305, 2002.

- [Hsu93] Wen-Lian Hsu. A simple test for interval graphs. In *18th International Workshop on Graph-Theoretic Concept in Computer Science (WG92)*, number 657 in Lecture Notes in Computer Science, pages 11–16, 1993.
- [Hsu96] Wen-Lian Hsu. On-line recognition of interval graphs in $o(m+n \log n)$ time. In *Combinatorics and Computer Science*, pages 27–38, 1996.
- [Hsu01] Wen-Lian Hsu. Pc-trees vs. pq-trees. In Jie Wang, editor, *COCOON*, volume 2108 of *Lecture Notes in Computer Science*, pages 207–217, 2001.
- [HT73] John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3) :135–158, 1973.
- [Iba99] L. Ibarra. Fully dynamic algorithms for chordal graphs. In *10th ACM-SIAM Annual Symposium on Discrete Algorithm (SODA '99)*, pages 923–924, 1999.
- [Iba00] L. Ibarra. Fully dynamic algorithms for chordal graphs and split graphs. Technical Report DCS-262-IR, University of Victoria, 2000.
- [Iba01] Louis Ibarra. A fully dynamic algorithm for recognizing interval graphs using the clique-separator graph. Technical Report DCS-263-IR, Dept. of Computer Science, University of Victoria, 2001.
- [Ill97] Pierre Ille. Indecomposable graphs. *Discrete Mathematics*, 173(1-3) :71–78, 1997.
- [iO05a] Sang il Oum. *Graphs of Bounded Rank-width*. Ph.d. thesis, Princeton University, 2005.
- [iO05b] Sang il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1) :79–100, 2005.
- [JO92] B. Jamison and S. Olariu. A tree representation for p_4 -sparse graphs. *Discrete Appl. Math.*, 35 :115–129, 1992.
- [JW95] Bill Jackson and Nicholas C. Wormald. Long cycles and 3-connected spanning subgraphs of bounded degree in 3-connected $_1$ -free graphs. *J. Comb. Theory, Ser. B*, 63(2) :163–169, 1995.
- [KM85] N. Korte and R. Möhring. Transitive orientation of graphs with side constraints. In *WG*, pages 143–160, 1985.
- [KM89] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18 :68–81, 1989.
- [Kor87] Norbert Korte. *Intervallgraphen und MPQ-Bäume : Algorithmen und Datenstrukturen zur Manipulation von Intervallgraphen und der Lösung von Seriationsproblemen*. Ph.d. thesis, report 87461, Institute for Operations Research, Universität Bonn, 1987.
- [Lan01] Jean-Marc Lanlignel. *Autour de la décomposition en coupes*. PhD thesis, Université Montpellier II, 2001.
- [LB62] C. G. Lekkerkerker and J. C. Boland. Representation of finite graphs by a set of intervals on the real line. *Fund. Math.*, 51 :45–64, 1962.

- [Lov72] L. Lovász. A characterization of perfect graphs. *J. Combinatorial Theory Ser. B*, 13 :95–98, 1972.
- [MdM05] Ross M. McConnell and Fabien de Montgolfier. Algebraic operations on pq trees and modular decomposition trees. In *31th Int. Workshop on Graph Theoretical Concepts in Computer Science (WG05)*, number 3787 in Lecture Notes in Computer Science, pages 421–432, 2005.
- [MM99] Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. SIAM, 1999.
- [Möh85] R.H. Möhring. Algorithmic aspect of the substitution decomposition in optimization over relations, set systems and boolean functions. *Annals of Operations Research*, 4 :195–225, 1985.
- [MP01] Frederic Maffray and Myriam Preissmann. A translation of tiber galleri's paper : Transitive orientable graphs. In J.L. Ramirez-Alfonsin and B.A. Reed, editors, *Perfect graphs*, pages 25–66. J. Wiley, 2001.
- [MR84] R.H. Möhring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19 :257–356, 1984.
- [MS89] J.H. Muller and J.P. Spinrad. Incremental modular decomposition algorithm. *Journal of the Association for Computing Machinery*, 36(1) :1–19, 1989.
- [MS99] Ross M. McConnell and Jeremy Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3) :189–241, 1999.
- [MT01] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.
- [NC88] T. Nishizeki and N. Chiba. *Planar Graphs : Theory and Algorithms*, volume 140 of *North-Holland Mathematics Studies*. Elsevier, 1988.
- [NPP06] Stavros D. Nikolopoulos, Leonidas Palios, and Charis Papadopoulos. A fully dynamic algorithm for the recognition of p_4 -sparse graphs. In *32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG06)*, Lecture Notes in Computer Science, 2006. To appear.
- [Ric04] R. Bruce Richter. Decomposing infinite 2-connected graphs into 3-connected components. *The electronic journal of combinatorics*, 11(1), 2004.
- [Rob69] Fred S. Roberts. Indifference graphs. In F. Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, New York, 1969.
- [Sei74] D. Seinsche. On a property of the class of n -colorable graphs. *J. Comb. Theory B*, 16 :191–193, 1974.
- [SH99] Wei Kuan Shih and Wen-Lian Hsu. A new planarity test. *Theor. Comput. Sci.*, 223(1-2) :179–191, 1999.
- [Spi94] J. Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16 :264–282, 1994.

- [Spi03] Jeremy P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute Monographs*. American Mathematical Society, 2003.
- [SS04] R. Shamir and R. Sharan. A fully dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Applied Mathematics*, 136(2-3) :329–340, 2004.
- [Tam88] Roberto Tamassia. A dynamic data structure for planar graph embedding. In *ICALP*, pages 576–590, 1988.
- [Tar85] Robert Endre Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55 :221–232, 1985.
- [Tro92] William T. Trotter. *Combinatorics and Partially Ordered Sets : Dimension Theory*. The Johns Hopkins University Press, 1992.
- [Tuc71] Alan Tucker. Matrix characterizations of circular-arc graphs. *Pacific J. Math.*, 39(2) :535–545, 1971.
- [Tut66] W.T. Tutte. *Connectivity in graphs*. Number 15 in Mathematical Expositions. University of Toronto Press, Toronto, Ont. ; Oxford University Press, London, 1966.
- [UY00] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2) :290–309, 2000.
- [Wal72] J. Walter. *Representations of rigid cycle graphs*. PhD thesis, Wayne State University, USA, 1972.
- [Wes00] Douglas Brent West. *Introduction to Graph Theory (2nd edition)*. Prentice Hall, 2000.
- [Whi32] H. Whitney. Non separable and planar graph. *Trans. Amer. Math. Soc.*, 34 :339–362, 1932.
- [Whi33] H. Whitney. 2-isomorphic graphs. *Amer. J. Math.*, 55 :245–254, 1933.
- [Yao81] Andrew Chi-Chih Yao. Should tables be sorted? *Journal of the ACM*, 28(3) :615–628, 1981.

Résumé

Ce travail de thèse traite du maintien dynamique de représentations géométriques de graphes. Le manuscrit met en avant des connexions fortes entre trois types de représentation de graphes : les décompositions de graphes, les modèles géométriques et les représentations arborescentes à degrés de liberté (PQ -arbres, PC -arbres et autres structures du même type). De nouvelles relations entre ces objets sont mises en évidence et d'autres déjà connues sont approfondies. Notamment, il est établi une équivalence mathématique et algorithmique entre la décomposition modulaire des graphes d'intervalles et le PQ -arbre de leurs cliques maximales.

Les connexions entre les trois types de représentation précités sont exploitées pour la conception d'algorithmes de reconnaissance entièrement dynamiques pour les cographes orientés, les graphes de permutation et les graphes d'intervalles. Pour les cographes orientés, l'algorithme présenté est de complexité optimale, il traite les modifications de sommet en temps $O(d)$, où d est le degré du sommet en question, et les modifications d'arête en temps constant. Les algorithmes pour les graphes de permutation et les graphes d'intervalles ont la même complexité : les modifications d'arête et de sommet sont traitées en temps $O(n)$, où n est le nombre de sommets du graphe. Une des contributions du mémoire est de mettre en lumière des similarités très fortes entre les opérations d'ajout d'un sommet dans un graphe de permutation et dans un graphe d'intervalles. L'approche mise en oeuvre dans ce mémoire est assez générale pour laisser entrevoir les mêmes possibilités algorithmiques pour d'autres classes de graphes définies géométriquement.

Mots clés : algorithmes dynamiques, représentations de graphes, décompositions de graphes, graphes de permutation, graphes d'intervalles, PQ -arbres

Abstract

This work deals with dynamic maintain of geometric graph representations. It points out strong connections between three kinds of graph representations : graph decompositions, geometric models and tree-like representations with degrees of freedom (PQ -trees, PC -trees and other structures of the same type). New relationships between these objects are enlightened and some other known relationships are pushed further. In particular, a mathematical and algorithmic equivalence between the modular decomposition of interval graphs and the PQ -tree of their maximal cliques is established.

The connections between the three kinds of representations referred above are used for designing fully dynamic recognition algorithms for the classes of directed cographs, permutation graphs and interval graphs. The algorithm for directed cographs has an optimal complexity, it handles vertex modifications in $O(d)$ time, where d is the degree of the modified vertex, and edge modifications in constant time. The algorithms for permutation graphs and interval graphs have the same complexity : edge and vertex modifications are treated in $O(n)$ time, where n is the number of vertices of the graph. It is worth mentioning that very strong similarities between vertex insertion operations in permutation graphs and interval graphs are pointed out. The approach used here seems general enough to be applied to other geometrically defined graph classes.

Keywords : dynamic algorithms, graph representations, graph decompositions, permutation graphs, interval graphs, PQ -trees