Recherche de régularités et représentations succinctes de graphes

François Pitois

THESE DE DOCTORAT DE L'ETABLISSEMENT UNIVERSITE BOURGOGNE FRANCHE-COMTE

PREPAREE A L'UNIVERSITE DE BOURGOGNE

Ecole doctorale nº 37

SPIM (Science Pour l'Ingénieur et Microtechniques)

Doctorat d'informatique

Par

M. Pitois François

Recherche de régularités et représentations succinctes de graphes

Thèse présentée et soutenue à Dijon, le 10 juillet 2024

Composition du Jury :

M. Crespelle Christophe	Professeur à l'Université Côte d'Azur	Président
M. Gavoille Cyril	Professeur à l'Université de Bordeaux	Rapporteur
M. Paul Christophe	Directeur de recherche CNRS au LIRMM	Rapporteur
M. De Montgolfier Fabien	Maître de conférences à l'Université Paris Cité	Examinateur
Mme Parreau Aline	Chargée de recherche CNRS au LIRIS	Examinatrice
M. Togni Olivier	Professeur à l'Université de Bourgogne	Directeur de thèse
Mme Seba Hamida	Professeure à l'Université Lyon 1	Codirectrice de thèse
M. Haddad Mohammed	Maître de conférences à l'Université Lyon 1	Co-encadrant de thèse

Remerciements

Je voudrais tout d'abord remercier l'intégralité des membres du jury. Je remercie Christophe Paul pour avoir accepté d'être rapporteur de ma thèse. Je le remercie aussi pour avoir souvent assisté aux exposés de mes travaux en cours et pour m'avoir aiguillé avec ses questions. Merci également à Cyril Gavoille pour son rôle de rapporteur, pour avoir participé à mon comité de suivi de thèse ainsi que pour ses remarques et ses corrections. Je remercie Fabien De Montgolfier, que j'ai pu croiser à l'IRIF et qui a toujours été très accueillant, ainsi que Christophe Crespelle, membre de mon comité de suivi de thèse, pour avoir tous deux rejoint mon jury. Enfin, je remercie Aline Parreau pour son aide et son soutien tout au long des quatre ans de thèse au sein de l'équipe GOAL.

Je voudrais évidemment remercier mes encadrants de thèse, Olivier Togni, Hamida Seba et Mohammed Haddad pour m'avoir laissé travailler sur des sujets qui me plaisaient, pour m'avoir laissé explorer des directions nouvelles et pour m'avoir mis en relation avec d'excellents chercheurs qui m'ont permis d'avancer.

Je voudrais ensuite remercier les personnes avec qui j'ai pu travailler. Tout d'abord, un grand merci à Michel Habib pour nos échanges, pour m'avoir accueilli à l'IRIF, pour m'avoir fait aimer l'algorithmique (notamment linéaire), pour m'avoir fait découvrir les motifs ordonnés et les Lex-BFS, et pour son enthousiasme constant. Merci à Laurent Feuilloley pour ses nombreux conseils et pour son optimisme. Je voudrais aussi remercier Mamadou M. Kanté pour son aide avec les r-splits en me donnant de précieux conseils et pour avoir pris le temps de travailler à distance et en présentiel avec moi.

Je voudrais aussi remercier mes collègues lyonnais pour m'avoir accueilli. Merci à Quentin Deschamps, mon co-bureau pendant ces quatre années, pour avoir suivi mon parcours. Merci à Éric Duchêne pour m'avoir intégré dans l'équipe GOAL. Merci à Théo Pierron pour ses précieux retours au cours de ses quatre années et notamment pour la préparation de ma soutenance. Je voudrais aussi remercier Nacim Oijid, à la fois collègue, colocataire et ami, pour sa présence et sa bonne humeur. Merci aussi à tous mes autres collègues de recherche : Guillaume Bagan, Nicolas Bousquet, Antoine Castillon, Lucas De Meyer, Mathieu Hilaire et Sébastien Zeitoun. Un grand merci aussi à mes collègues d'enseignement pour m'avoir fait aimer la discipline : Frédéric Armetta, Sylvain Brandel, Raphaëlle Chaine, Élodie Desserée, Nathalie Guin, Hamid Ladjal, Marie Lefevre, Nicolas Louvet, Alexandre Meyer et Vincent Nivoliers. Enfin, merci à Isabelle Buisson pour son efficacité et son professionnalisme.

Mes derniers remerciements vont à ma famille. Merci à mon père, ma mère, Léa et Tom pour m'avoir apporté un contexte familial propice au travail et à la recherche. Enfin, un immense merci à ma compagne Camille pour sa présence constante, pour son soutien sans faille, pour son exigence qui m'a permis de me dépasser et pour son implication directe dans ma thèse en acceptant de me relire intégralement.

Résumé

Dans cette thèse, nous étudions les régularités dans les graphes et les représentations succinctes des graphes. Une *régularité*, ou structure, est un terme générique qui désigne un ensemble de sommets du graphe ayant certaines propriétés. Parmi les régularités les plus connues, nous pouvons citer les cliques, les sous-graphes denses, les communautés, les modules et les splits. Une *représentation succincte* d'un graphe est une façon de le décrire qui est plus efficace que de simplement lister les différentes arêtes du graphe. Chercher des régularités permet d'obtenir des représentations succinctes. Ainsi, dans un premier temps, nous avons développé un algorithme de compression de graphe qui cherche différentes régularités du graphe, en sélectionne une partie et partitionne le graphe en fonction des structures sélectionnées. Cet algorithme donne une description succincte du graphe qui est meilleure que certains algorithmes de référence.

Dans un deuxième temps, nous avons créé nos propres structures, de sorte qu'elles soient adaptées à la compression et qu'elles soient assez facile à chercher. Pour ce faire, nous sommes partis d'une structure connue, le split, et nous l'avons généralisée en créant le *r-split*, où r est un paramètre entier fixé. Nous avons alors montré que l'ensemble des r-splits d'un graphe a une cohérence globale, dans le sens où seul un nombre polynomial d'entre eux suffit à décrire l'intégralité des r-splits du graphe. Cela généralise une propriété bien connue des splits, pour lesquels seul un nombre linéaire d'entre eux suffit à retrouver tous les autres. Nous avons également montré que les r-splits peuvent se calculer en temps polynomial en utilisant des algorithmes d'optimisation de fonctions sous-modulaires.

Dans un troisième temps, nous nous sommes intéressés à la recherche de structures particulières : les motifs dans les graphes ordonnés. Un *graphe ordonné* est un graphe dans lequel les sommets sont ordonnés de 1 à n. Un *motif* est un sous-graphe ordonné partiel, dans le sens où chaque paire de sommets peut être reliée soit par une arête, soit par une non-arête, soit par ni l'une ni l'autre. Le but est de fixer un motif P et de construire un algorithme capable de détecter si P est dans n'importe quel graphe ordonné donné en paramètre. Ce problème est polynomial en la taille du graphe via une recherche exhaustive. Cependant, est-il possible de faire mieux ? Nous avons montré que oui : la plupart des motifs à trois sommets peuvent être détectés en temps linéaire là où une recherche exhaustive nécessite un temps cubique. Concernant les motifs plus grands, nous avons exhibé des classes de motifs qui peuvent être détectés en temps sous-cubique : les motifs planaires extérieurs. En ajoutant des contraintes supplémentaires, nous avons exhibé une classe de motifs qui peuvent être détectée en temps linéaire : il s'agit des motifs planaires extérieurs sans cycle et sans non-arête.

Mots-clés : Graphe, compression, splits, algorithmique, graphe ordonné, motif.

Abstract

In this thesis, we investigate regularities in graphs and succinct representations of graphs. A *regularity*, or structure, is a generic term that refers to a set of vertices in a graph with certain properties. Among the most well-known regularities, we can mention cliques, dense subgraphs, communities, modules, and splits. A *succinct representation* of a graph is a way of describing it that is more efficient than simply listing the different edges of the graph. Searching for regularities enables obtaining succinct representations. Thus, in a first step, we developed a graph compression algorithm that searches for different graph regularities, selects a portion of them, and partitions the graph based on the selected structures. This algorithm provides a succinct description of the graph that is better than some benchmark algorithms.

In a second step, we created our own structures, so they are suitable for compression and are easy enough to search for. To do this, we started from a known structure, the split, and generalized it to create the *r-split*, where r is a fixed integer parameter. We then showed that the set of r-splits of a graph has a global coherence, in the sense that only a polynomial number of them is sufficient to describe all r-splits of the graph. This generalizes a well-known property of splits, for which only a linear number of them is sufficient to recover all the others. We also demonstrated that r-splits can be computed in polynomial time using submodular function optimization algorithms.

In a third step, we focused on searching for particular regularities: patterns in ordered graphs. An *ordered graph* is a graph in which the vertices are ordered from 1 to n. A *pattern* is a partial ordered subgraph, in the sense that each pair of vertices can be connected either by an edge, a non-edge, or neither. The goal is to fix a pattern P and build an algorithm capable of detecting if P is in any ordered graph given as an input. This problem is polynomial in the size of the graph via exhaustive search. However, is it possible to do better? We were able to show that yes: most three-vertex patterns can be detected in linear time while exhaustive search requires cubic time. Regarding larger patterns, we exhibited classes of patterns that can be detected in subcubic time: outerplanar patterns. By adding additional constraints, we exhibited a class of patterns that can be detected in linear time: these are outerplanar patterns without cycles and non-edges.

Keywords: Graph, compression, splits, algorithmics, ordered graph, pattern.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Contexte et contributions

1.1.1 Réseau, graphe et compression

Les graphes sont des objets abstraits qui peuvent être utilisés pour représenter de nombreux réseaux de données tels que les réseaux sociaux ou le réseau Internet. De nos jours, les graphes sont omniprésents et leur étude constitue un des piliers de l'informatique contemporaine. Un des enjeux actuels est la compression de réseaux, afin (entre autres) de réduire l'espace dédié à leur stockage ainsi que le temps nécessaire à leur transmission. Pour résoudre ces problématiques, il convient d'étudier l'objet théorique utilisé pour modéliser ces réseaux, à savoir le graphe, et de parvenir à le compresser.

D'une manière générale, la compression se divise en deux catégories : la compression sans perte et la compression avec perte. La compression sans perte consiste à représenter les données d'une façon différente et plus optimisée afin de pouvoir stocker la même information avec moins d'espace. Souvent, cela est possible en analysant préalablement le type de données à compresser, afin de dégager des spécificités qui pourraient aider à compresser. Par exemple, dans le cas de la compression d'images, le fait de savoir que deux pixels qui se suivent sont souvent quasiment identiques permet une compression plus efficace d'images non-aléatoires. De l'autre côté, la compression avec perte s'autorise à détruire partiellement l'information si celle-ci est jugée non-essentielle. Il convient alors de juger ce qui est essentiel et ce qui ne l'est pas. Dans le cas de la compression du son par exemple, comme on sait qu'un humain moyen ne perçoit pas les fréquence en dehors de l'intervalle $[20\text{ Hz}, 20\,000\text{ Hz}]$, les fréquences en dehors de cette intervalle peuvent être ignorées dans le cas d'une compression avec perte.

Dans le cas des graphes, ces deux catégories de compression ont beaucoup été étudiées. Concernant la compression sans perte, il existe une très grande variété de techniques [BH18]. Tout d'abord, l'analyse du réseau sous-jacent au graphe joue un rôle important. Si on veut compresser le graphe qui représente le réseau du Web (graphe dans lequel chaque sommet représente une page web et chaque arête un hyperlien d'une page web à l'autre), il y a une heuristique naturelle qui aide à la compression. Il s'agit du fait que si deux pages web ont des URL très similaires (le même nom de domaine par exemple), il y a de fortes chances pour que les sommets associés dans le graphe soit aussi très similaires en termes de voisinage. C'est une des techniques qu'utilise WebGraph [BV04] par exemple. Une autre façon de compresser est de transformer le graphe en une structure plus connue, que l'on sait déjà compresser. Par exemple, un graphe peut être représenté sous forme de texte via sa liste d'adjacence, sous forme d'image binaire via sa matrice d'adjacence ou sous forme d'une suite binaire en considérant sa matrice d'adjacence de façon unidimensionnelle. Or, ces trois types de données (texte, image binaire et suite binaire) disposent d'algorithmes efficaces de compression comme la compression LZ [ZL77]. L'étape difficile de cette transformation est le choix de l'ordre des sommets du graphe. En effet, deux ordres différents aboutissent à deux textes/images/suites différentes et il y en a toujours une qui est plus compressible que l'autre. Finalement, une méthode propre aux graphes consiste à repérer des motifs du graphe facilement compressibles, comme les cliques ou les sous-graphes bipartis complets, et de les utiliser directement pour faciliter la compression globale du graphe. On peut également repérer des structures plus générales et les utiliser dans le cadre d'une compression hiérarchique. Dans ce cas, chaque structure détectée est transformée en un super-nœud indexé par la structure. Cela permet alors de simplifier le graphe global en réduisant son nombre de sommets, et la structure indexée peut indépendamment être compressée de façon directe ou récursive. Cela aboutit alors à un graphe

hiérarchique, où chaque niveau est constitué de graphes ayant des super-nœuds indexés par des graphes d'un niveau inférieur.

Concernant la compression avec perte, il faut décider et justifier ce que l'on enlève. Souvent, on ne garde que l'information nécessaire pour répondre à une ou plusieurs requêtes prédéterminées. Par exemple, la requête la plus courante est l'accessibilité : étant donné un sommet u du graphe compressé, on veut pouvoir retrouver quels sont les sommets accessibles depuis u. Pour répondre à cela, il n'y a pas besoin de stocker le graphe dans son intégralité, seul un chemin dans chaque composante connexe suffit. On peut aussi raisonner par seuillage : on associe à chaque arête un poids qui correspond à son importance dans le graphe (via des algorithmes de flot par exemple) et on ne garde que les arêtes au-delà d'un certain seuil. Dans tous les cas, la compression avec perte ne permet jamais de retrouver le graphe original, mais permet en revanche de sauvegarder certaines informations ciblées sur ce graphe.

Dans cette thèse, nous nous sommes intéressés à la compression sans perte.

1.1.2 Compresser via structures et ordonnancement

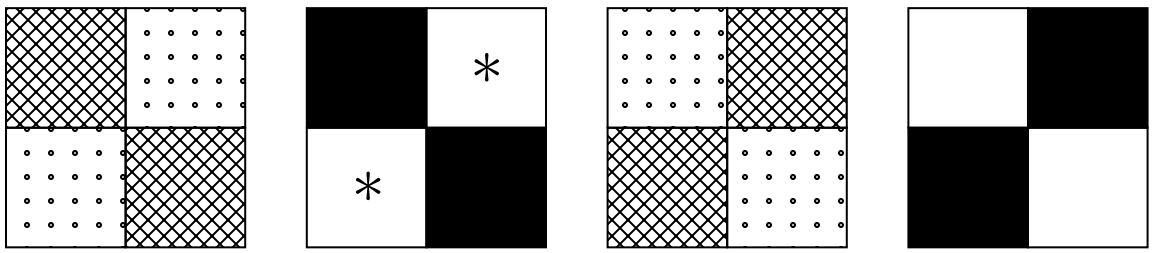
Parmi les différentes approches et les différents aspects de la compression sans perte, nous nous sommes focalisés sur les méthodes de compression qui consistent à choisir un ordre des sommets et à compresser la matrice d'adjacence du graphe suivant cet ordre. En effet, l'ordre à un grand impact sur la compressibilité de la matrice : une matrice ayant des 0 et des 1 disposés de façon quasiment aléatoire ne pourra pas être compressée, alors qu'une matrice ayant de fortes concentrations de 1 à certains endroits et de fortes concentrations de 0 à d'autres endroits pourra être plus facilement compressée en compressant indépendamment chacune de ces régions à forte concentration. Il faut donc séparer les 0 des 1 afin d'améliorer la compressibilité de la matrice Le but est alors de chercher un ordre des sommets qui maximise cette séparation des valeurs 0 et 1.

Comme il n'est pas envisageable d'essayer les $n!$ ordres possibles d'un graphe d'ordre n, il faut trouver une façon de générer un ordre des sommets directement à partir du graphe. La première idée est de repérer des structures dans le graphe et de mettre l'un après l'autre les différents sommets d'une structure ainsi repérée. Par exemple, si l'on repère une clique dans le graphe, on veut mettre ces sommets côte à côte pour obtenir un carré rempli de 1 dans la matrice d'adjacence, ce qui est une forte concentration de 1. De la même manière, un ensemble indépendant permet une forte concentration de 0. Il faut donc trouver des structures qui maximisent ce genre de segmentation. Une façon de parvenir à ce résultat est d'utiliser des algorithmes de *recherche de structures*, capables de repérer efficacement certains types de structures dans les graphes. Le fait d'utiliser plusieurs algorithmes de recherche permet de multiplier les points de vue et d'obtenir une grande variété de structures. Cependant, plusieurs problèmes surviennent. D'abord, il y a le problème du *chevauchement* : deux structures différentes peuvent avoir des sommets en commun. Comme notre ordonnancement se contente de mettre côte à côte des sommets appartenant à la même structure, il faut trouver quoi faire de ces sommets qui appartiennent à plusieurs structures. Ensuite, certaines structures ne produisent pas une assez forte séparation entre les 0 et les 1, ou chevauchent trop d'autres structures. En fait, avoir une trop grande quantité de structures n'est pas forcément positif pour la compression. Il faut donc *filtrer* les différentes structures afin de ne garder que celles qui produiront le meilleur ordonnancement des sommets. Le chapitre 3 est dédié à l'étude de cet algorithme de compression. Cette étude a conduit à la proposition d'un algorithme

de compression de graphes qui a donné lieu à une publication dans la conférence DaWak 2023 [PSH23] et à une version étendue en cours de révision pour la revue *International Journal of Data Science and Analytics*.

Comme on cherche à minimiser le coût de compression à l'aide de structures et d'un ordre des sommets, la question qui s'est posé est : pourquoi ne pas construire notre propre structure dédiée à minimiser ce coût, puis trouver un bon ordonnancement des sommets ? Pour ce faire, la première étape consiste à étudier les structures déjà connues.

1.1.3 Étudier les structures connues

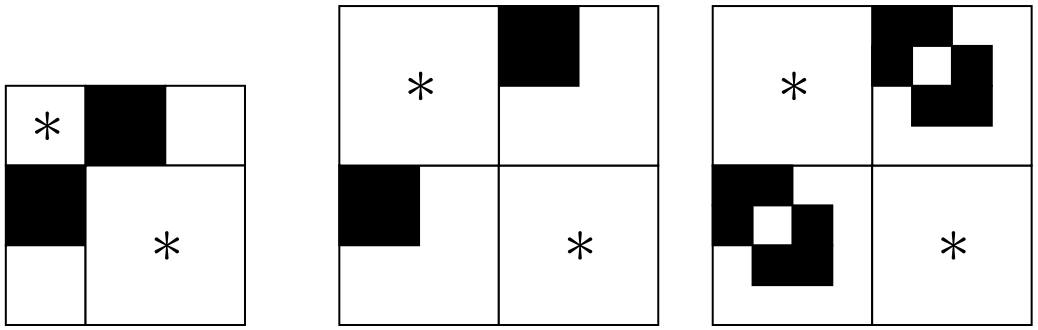


(a) Deux communautés. (b) Deux cliques. (c) Quasi-biparti. (d) Biparti complet.

FIGURE 1.1 – Schémas de matrices d'adjacence associées à différentes structures. Une région hachurée représente une forte densité de 1, une région parsemée de points représente une forte densité de 0, une région complètement noire représente une zone uniquement de 1, une région complètement blanche représente une zone uniquement de 0 et une région munie d'une étoile représente une zone avec une densité quelconque.

Il existe un certain nombre de structures qui séparent les 0 des 1 ou qui produisent une forte concentration de 1 dans la matrice d'adjacence, comme montré sur la figure 1.1. On a déjà cité les cliques, mais on peut aussi citer les communautés, qui regroupent tous les graphes denses. Les structures denses ont tendance à produire une matrice d'adjacence de la forme (1.1a), c'est-à-dire à regrouper les 1 dans des blocs qui sont sur la partie diagonale et les 0 en dehors. Grâce aux cliques, on peut obtenir une matrice de le forme (1.1b), avec une concentration maximale de 1 sur la partie diagonale de la matrice d'adjacence, mais le reste de la matrice est alors quelconque. Nous avons voulu explorer d'autres types de structures, celles qui ont tendance à produire une matrice d'adjacence de la forme (1.1c). La structure de ce type la plus connue est le sous-graphe biparti complet, qui produit une matrice (1.1d). Ce principe a déjà été un petit peu étendu via la notion de module, qui décrit un ensemble de sommets ayant le même voisinage, et la notion de split, qui décrit une bi-partition des sommets dont les arêtes reliant les deux parties forment un sous-graphe biparti complet.

Ces structures ont pour avantage d'être faciles à détecter. En effet, les modules et les splits d'un graphe possèdent une structure assez rigide qui fait qu'ils sont presque tout le temps stables par les opérations ensemblistes élémentaires comme l'intersection ou la différence. Grâce à la rigidité de l'ensemble des modules ou des splits d'un graphe, il est possible de les résumer dans une structure arborescente de taille linéaire et aussi calculable en temps linéaire. Ainsi, on peut avoir accès à l'ensemble des modules et des splits d'un graphe rapidement. De plus, ces structures sont très faciles à compresser grâce à leur rigidité interne qui offre une forte concentration de 0 et de 1. En effet, un module



(a) Un module. (b) Un split. (c) Un 2-split.

FIGURE 1.2 – Schémas de matrices d'adjacence associés à différentes structures. Une région complètement noire représente une zone uniquement de 1, une région complètement blanche représente une zone uniquement de 0 et une région munie d'une étoile représente une zone pouvant avoir des 0 et des 1.

étant un ensemble de sommets ayant le même voisinage, il suffit d'encoder une seule fois ce voisinage commun. en termes de matrice et d'ordonnancement, si on ordonne d'abord les sommets du module, puis les voisins communs à tous ces sommets, puis les non-voisins communs aux sommets du module, on a une matrice de la forme (1.2a). De même, les splits permettent d'obtenir un ordonnancement favorable, en ordonnant d'abord la première partie de la bi-partition et ensuite la seconde partie, et en mettant dans chaque partie les extrémités des arêtes traversantes proches de la frontière entre les deux parties. Cela donne une matrice de la forme (1.2b). Ainsi, en termes de contraintes, les modules et les splits imposent une certaine forme de sous-matrice pour les blocs en dehors de la diagonale et laissent libres les blocs diagonaux. Le chapitre 4 est consacré à l'étude des splits et des modules via une revue de l'état de l'art, agrémentée de preuves souvent omises de certains résultats classiques.

Si on s'intéresse à une compression plus générale que la compression par matrice, alors les modules et les splits permettent également une compression hiérarchique. Un partitionnement d'un graphe en modules ou en splits produira un ensemble de sous-graphes dont les arêtes entre sous-graphes vérifient certaines propriétés. Cela permet alors une compression hiérarchique en transformant chaque module ou split en super-nœud.

1.1.4 Créer nos propres structures

Nous avons voulu aller plus loin et introduire une généralisation des splits appelée r-split.

L'idée est d'avoir une structure qui puisse partitionner les sommets d'un graphe de sorte que les arêtes traversantes respectent certaines règles qui généralisent celles des splits, afin de regrouper au maximum les 0 et les 1 et de pouvoir être facilement compressible. Par exemple, dans le cas des splits, on partitionne les sommets du graphes de sorte à ce que la matrice d'adjacence qui représente les arêtes qui sont entre deux parties soit de rang au plus 1. Le but de notre nouvelle structure (les r-splits) est d'obtenir un tel partitionnement, mais en remplaçant la condition du rang au plus 1 par un rang au plus r. Pour $r = 2$, on obtient une matrice d'adjacence ayant la forme (1.2c). Dans cette optique, nous nous sommes penchés sur l'étude théorique des r-splits afin de savoir si un tel partitionnement était facilement trouvable ou non.

Nos résultats principaux portent sur la représentation polynomiale des r-splits. En

effet, on montre qu'un graphe peut contenir un nombre exponentiel de r-splits. Afin de pouvoir les manipuler, nous avons commencé par prouver qu'un nombre polynomial d'entre eux suffisait pour pouvoir retrouver tous les autres. De plus, nous avons montré que chacun de ces r-splits particuliers pouvait être calculé en temps polynomial, ce qui permet au final d'obtenir une représentation polynomiale en temps polynomial de l'ensemble des r-splits. Le chapitre 5 est consacré à l'étude de cette nouvelle structure que sont les r-splits. Ce travail a été présenté dans la conférence ICGT 2022 et a été accepté dans la revue DMTCS [PHST24].

1.1.5 Étudier l'ordonnancement

Vu que l'algorithme de compression fournit un ordonnancement des sommets, il pourrait être utile de déduire des informations de cet ordre. J'ai ainsi été amené à étudier les graphes ordonnés, c'est-à-dire les graphes dans lesquels les sommets sont ordonnés. Le cadre théorique consiste à prendre un graphe ordonné et à se demander si un motif apparait ou non dans ce graphe ordonné, où un motif peut être vu comme un ensemble de contraintes sur les sommets et les arêtes du graphe ordonné. Les motifs étudiés peuvent aussi être vus comme des sous-graphes ordonnés interdits dans le graphe ordonné. Le principal résultat a été de construire des algorithmes efficaces de reconnaissance pour certaines classes de motifs. On sait en effet que cela peut toujours se faire en temps polynomial, mais le but a été de réduire au maximum le degré du polynome nécessaire.

Tout d'abord, dans la continuité de [FH21b], nous nous sommes intéressés aux motifs à 3 sommets. Nous avons pu montrer que la majorité des motifs à 3 sommets pouvait être détectée en temps linéaire. De plus, si l'on essaie de détecter un ensemble de motifs à 3 sommets, c'est-à-dire savoir si le graphe ordonné en contient au moins un ou s'il les évite tous, on a pu montrer que cela pouvait toujours se faire en temps linéaire à partir du moment où l'ensemble n'est pas réduit à un singleton. Ensuite, nous nous sommes concentrés sur des classes de graphes. On a pu montrer que les motifs ordonnés planaires extérieurs positifs sans cycle peuvent se détecter en temps linéaire, que la complexité devient quadratique si l'on retire la condition positive et qu'elle devient la complexité du produit matriciel si l'on retire la condition acyclique. Finalement, cela nous a motivés à introduire notre propre paramètre pour quantifier la difficulté à détecter les motifs. Ce nouveau paramètre, que nous avons appelé *largeur de fusion* permet de donner une borne maximale sur la complexité de n'importe quel motif et s'avère être exact pour les motifs planaires extérieurs, pour lesquels on retrouve la complexité de la multiplication de matrice. Le chapitre 6 est dédié à l'étude de la détection de motifs dans les graphes ordonnés. Ce travail est en cours de soumission à SIDMA.

1.2 Organisation du manuscrit

Le chapitre 2 présente des définitions et des rappels des notions qui seront utilisées dans ce manuscrit. Tout ce qui y est présenté est globalement connu, excepté dans la section 2.4 où j'introduis une variante du produit tensoriel contracté. Les preuves et les exemples présentés dans ce chapitre sont cependant tous originaux.

Dans le chapitre 3, je présente l'algorithme que j'ai développé avec l'aide de mes encadrants Hamida Seba et Mohammed Haddad. Je détaille ce nouvel algorithme et je présente les résultats expérimentaux que nous avons obtenus. Il s'agit de l'adaptation de la publication dans la conférence DaWak 2023 [PSH23] et de la version étendue en cours de

révision pour la revue *International Journal of Data Science and Analytics*. Ces travaux ont également été présentés aux JGA 2020.

Dans le chapitre 4, je rappelle les notions de modules et de splits qui sont la base des structures que j'ai développés par la suite. Les notions ne sont pas nouvelles, mais j'ai essayé d'unifier les différentes définitions qui existaient et j'ai refait certaines preuves qui étaient difficiles à trouver dans des articles, notamment la preuve de stabilité des splits par union disjointe. Cette preuve sera publiée par la suite dans une revue.

Le chapitre 5 présente la structure que j'ai introduite, les r-splits, ainsi que les théorèmes que j'ai pu déduire. Il s'agit d'un travail nouveau fait majoritairement seul mais avec l'aide ponctuelle de mes encadrants Olivier Togni, Hamida Seba et Mohammed Haddad ainsi que d'autres chercheurs tels que Michel Habib et Mamadou M. Kanté. Ce chapitre est adapté de l'article qui a été accepté dans la revue DMTCS [PHST24]. Ce travail a également été présenté aux conférences ICGT 2022 et JGA 2022.

Enfin, le chapitre 6 s'intéresse à la recherche de motifs dans les graphes ordonnés, qui est un travail fait avec Michel Habib, Laurent Feuilloley et Guillaume Ducoffe. Il s'agit d'une adaptation assez libre de l'article qui est en cours de soumission à SIDMA. De plus, j'ai pu présenté une partie de ces travaux aux JGA 2023.

Chapitre 2

Préliminaires

Sommaire

Ce chapitre présente les notions qui seront utilisées tout au long de ce manuscrit. La section 2.1 traite des définitions, conventions et notations générales utilisées dans les différents chapitres. La section 2.2 rappelle quelques notions de compression et de représentation des données qui seront utilisées dans le chapitre 3. La section 2.3 présente certains outils mathématiques tels que les clôtures, les relations binaires et les fonctions de connectivités. Ils seront utilisés dans les chapitres 4 et 5 afin d'en introduire les notions principales. Enfin, la section 2.4 introduit certaines notions élémentaires liées aux tenseurs, à leur notation et à leur multiplication, qui seront utilisées dans le chapitre 6

2.1 Définitions générales

2.1.1 Conventions et notations mathématiques

Étant donné deux ensembles X et Y, on a les notations suivantes :

— $X \subseteq Y$ signifie que X est *inclus* dans Y,

— $X \cap Y$ désigne l'*intersection* de X et Y,

— $X \cup Y$ désigne l'*union* de X et Y,

— $X \setminus Y$ désigne la *différence* X moins Y, et

— $X \Delta Y$ désigne la *différence symétrique* de X et Y, c'est-à-dire $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$.

Les opérateurs d'intersection, d'union et de différence symétrique peuvent s'appliquer à plus de deux ensembles. Si $X_1, \ldots, X_k$ sont k ensembles, $\bigcap_{i=1}^{k} X_i$ désigne l'intersection de tous les ensembles X_i pour i allant de 1 à k. Cet ensemble est constitué des éléments qui sont présents dans tous les X_i. Comme l'intersection est associative, on a $\bigcap_{i=1}^{k} X_i = X_1 \cap \ldots \cap X_k$. De même, la notation $\bigcup_{i=1}^{k} X_i$ désigne l'union de tous les ensembles X_i pour i allant de 1 à k. Cet ensemble est constitué des éléments qui sont présents dans au moins un X_i. Comme l'union est associative, on a $\bigcup_{i=1}^{k} X_i = X_1 \cup \ldots \cup X_k$. Enfin, la différence symétrique étant elle aussi associative, on peut considérer la différence symétrique de plusieurs ensembles. L'ensemble $X_1 \Delta \ldots \Delta X_k$ est alors égal à l'ensemble dont les éléments sont ceux qui sont inclus dans un nombre impairs d'ensemble parmi les X_i.

Étant donné un ensemble fini X, le cardinal de X, noté $|X|$, est le nombre d'éléments de X. Étant donné un ensemble X, on note 2^X l'ensemble des parties de X. Étant donné un ensemble X inclus dans un ensemble E fixe et clair d'après le contexte, on note $\overline{X}$ le complémentaire de X, c'est-à-dire $\overline{X} := E \setminus X$. Étant donné deux entiers i et j, on note $\{i, \ldots, j\}$ l'ensemble des entiers entre i et j (inclus) et $[i] := \{1, \ldots, i\}$ l'ensemble des entiers entre 1 et i.

Certains ensembles ont une notation propre. L'ensemble des entiers naturels est noté $\mathbb{N}$, l'ensemble des entiers relatifs est noté $\mathbb{Z}$ et l'ensemble des nombres réels est noté $\mathbb{R}$. De plus, l'ensemble $\{0, 1\}$ peut être noté de deux façons différentes selon la définition de l'addition et de la multiplication que l'on effectue sur les nombres 0 et 1 :

— $\mathbb{F}_2$ désigne le *corps fini à deux éléments*, c'est-à-dire l'ensemble $\{0, 1\}$ muni de l'addition modulo 2 et de la multiplication modulo 2. Ainsi, dans $\mathbb{F}_2$, les additions et les multiplications s'effectuent normalement, excepté que $1 + 1 = 0$ (car 2 modulo 2 vaut 0).

— $\mathbb{B}$ désigne l'*algèbre de Boole*, c'est-à-dire l'ensemble $\{0, 1\}$ muni du OU logique et du ET logique. Le OU logique joue le rôle de l'addition et le ET logique celui

de la multiplication. Ainsi, dans $\mathbb{B}$, les additions et les multiplications s'effectuent normalement, excepté que $1 + 1 = 1$ (car 1 OU 1 vaut 1).

Le terme *famille* fait référence à un ensemble dont les éléments sont eux-même des ensembles. Par exemple, 2^X peut être appelée une famille car les éléments de 2^X sont des ensembles (ceux qui sont inclus dans X). Étant donné un ensemble X, une *partition* est une famille $\mathcal{P}$ dont les éléments sont des parties de X et telle que pour tout élément x inclus dans X, il existe un unique ensemble de la partition $\mathcal{P}$ qui contient x. De plus, une partition ne peut pas contenir l'ensemble vide. Par exemple, si $X = \{1, 2, 3, 4, 5, 6, 7\}$, une partition de X est $\mathcal{P} = \{\{1, 4\}, \{2, 3, 7\}, \{5, 6\}\}$. La *partition grossière* de X est la partition $\mathcal{P} = \{X\}$, c'est-à-dire l'unique partition de taille $|\mathcal{P}| = 1$. Une *bi-partition* est une partition $\mathcal{P}$ de taille $|\mathcal{P}| = 2$. Une partition $\mathcal{P}_1$ de X est *plus fine* qu'une partition $\mathcal{P}_2$ de X si pour tout ensemble $A_1 \in \mathcal{P}_1$, il existe un ensemble $A_2 \in \mathcal{P}_2$ tel que $A_1 \subseteq A_2$.

Dans l'intégralité de ce manuscrit, les logarithmes sont en base 2. Étant donné un entier n et deux matrices A, B de taille n par n à coefficients complexes, on note ω le nombre minimum tel que la multiplication des matrices A et B se fasse en complexité temporelle $\mathcal{O}(n^\omega)$ [CW87]. De façon triviale, on a $2 \leq \omega \leq 3$. L'article [AW21] donne la borne plus précise $\omega < 2.3728596$.

Étant donné une fonction f dont l'ensemble de départ est E et l'ensemble d'arrivé est F, ce que l'on note : $E \to F$, on définit les ensembles suivants :

— Étant donné un ensemble $A \subseteq E$, $f(A)$ est l'*ensemble image* de A et vaut $f(A) = \{y \in F \mid \exists x \in A, y = f(x)\}$, c'est-à-dire que $f(A)$ est l'ensemble des $f(x)$ pour $x \in A$.

— Étant donné un ensemble $B \subseteq F$, $f^{-1}(B)$ est l'*ensemble pré-image* de B et vaut $f^{-1}(B) = \{x \in E \mid \exists y \in B, y = f(x)\}$, c'est-à-dire que $f^{-1}(B)$ est l'ensemble des x pour lesquels $f(x) \in B$.

2.1.2 Graphes

Un graphe G est un objet mathématique constitué d'un ensemble fini de sommets noté $V(G)$ et d'un ensemble d'arêtes noté $E(G)$ qui est un sous-ensemble de $V(G) \times V(G)$. Le nombre de sommets d'un graphe est appelé sont *ordre*. Ainsi, un graphe d'ordre 6 est un graphe ayant 6 sommets. Si $e = (x, y) \in E(G)$ est une arête, alors les sommets x et y sont appelés les *extrémités* de e. Dans ce manuscrit, on ne considère que des graphes *sans boucle*, c'est-à-dire tels que $(x, x) \notin E(G)$ pour tout sommet $x \in V(G)$. De plus, on ne considère que des *graphes non orientés*, c'est-à-dire tels que pour toute arête $(x, y) \in E(G)$, on a $(y, x) \in E(G)$. L'arête (y, x) est appelée l'arête *symétrique* de (x, y). Un graphe peut se représenter graphiquement de la façon suivante : chaque sommet de $V(G)$ est représenté par un point et chaque arête (x, y) de $E(G)$ est représentée par un trait reliant le point qui représente le sommet x et celui qui représente le sommet y. Un exemple est représenté sur la figure 2.1.

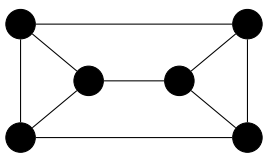


FIGURE 2.1 – Un exemple de graphe d'ordre 6.

Dans le cadre des graphes, le complémentaire d'un ensemble de sommets X fait référence à l'ensemble des sommets du graphe qui ne sont pas dans X, c'est-à-dire $\overline{X} = V(G) \setminus X$.

Étant donné un sommet $x \in V(G)$, le *voisinage* de x est l'ensemble des sommets y tels que $(x, y) \in E(G)$. On note $N(x)$ le voisinage de x. Vu que l'on ne considère que des graphes sans boucle, on a systématiquement $x \notin N(x)$. Étant donné un ensemble de sommets $X \in V(G)$ et un sommet $x \in V(G)$, le *voisinage extérieur* de x par rapport à X est l'ensemble de sommets $N(x) \setminus X$. Étant donné un ensemble de sommets $X \in V(G)$, le *bord* de X est l'ensemble des arêtes qui ont une extrémité dans X et l'autre dans $\overline{X}$. On note ∂X le bord de X. Formellement, on a $\partial X = \{(x, y) \in E(G) \mid x \in X \text{ et } y \notin X\}$.

En général, quand on travaille avec des graphes, c'est la représentation graphique qui prime pour déterminer si deux graphes sont « égaux ». Par exemple, considérons les graphes G_1 et G_2 définis par :

$$V(G_1) = \{1, 2, 3\}, \quad E(G_1) = \{(1, 2), (2, 1), (2, 3), (3, 2)\}$$
$$V(G_2) = \{4, 5, 6\}, \quad E(G_2) = \{(4, 5), (5, 4), (5, 6), (6, 5)\}.$$

Ces deux graphes peuvent être représentés de la même façon. La seule chose qui change est le nom des sommets, comme montré sur la figure 2.2.

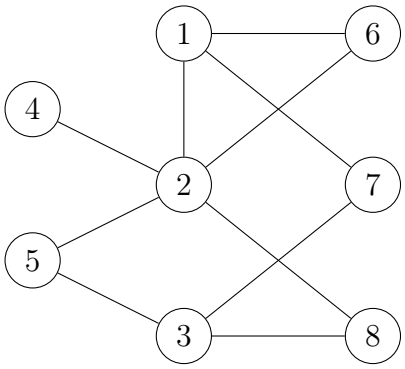
(a) Graphe G_1. (b) Graphe G_2.

FIGURE 2.2 – Deux graphes différents mais ayant la même représentation. On dit qu'ils sont isomorphes.

On veut donc pouvoir dire que ces deux graphes sont les mêmes. C'est l'idée de l'*isomorphisme de graphe*. Informellement, si on peut passer d'un graphe à l'autre en changeant uniquement le nom des sommets alors ces deux graphes sont isomorphes. Formellement, G_1 et G_2 sont isomorphes lorsqu'il existe une bijection $f : V(G_1) \to V(G_2)$ telle que $(x, y) \in E(G_1)$ si et seulement si $(f(x), f(y)) \in E(G_2)$. Ainsi, dans le cas où on travaille à isomorphisme de graphe prêt, on peut changer l'ensemble des sommets à notre guise. Notamment, on peut travailler avec l'ensemble de sommets $V(G) = [n]$, où n est le nombre de sommets dans le graphe.

En plus d'une représentation graphique, un graphe possède une représentation matricielle. Étant donné un graphe dont l'ensemble des sommets est $V(G) = [n]$, la *matrice d'adjacence* du graphe G est la matrice A de taille n par n tel que le coefficient $A_{i,j}$ vaut 1 si (i, j) est une arête de G et vaut 0 sinon. La matrice A est donc une matrice binaire. La figure 2.3 représente un graphe G avec sa matrice d'adjacence.

On peut relier les notions d'isomorphisme et de matrice d'adjacence : deux graphes G_1 et G_2 de matrices d'adjacence A_1 et A_2 sont isomorphes si et seulement s'il existe une matrice de permutation P telle que $A_1 = PA_2$. Dans le cas où l'ensemble des sommets de G n'est pas $[n]$, G ne possède pas directement de matrice d'adjacence. Plutôt, on considère un graphe G' isomorphe à G ayant $[n]$ comme ensemble de sommets et on peut alors considérer la matrice d'adjacence de G', qui est bien définie. Comme il existe $n!$ tels graphes G' isomorphes à G, G a donc $n!$ matrices d'adjacence qui le représente. On parle alors *des* matrices d'adjacence de G. La figure 2.4 montre un exemple de graphe accompagné de deux de ses matrices d'adjacence.

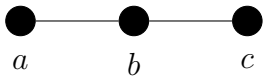


(a) Un graphe G ayant pour ensemble de sommets $\{1, 2, 3, 4, 5, 6, 7, 8\}$.

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	1	1	0
2	1	0	0	1	1	1	0	1
3	0	0	0	0	1	0	1	1
4	0	1	0	0	0	0	0	0
5	0	1	1	0	0	0	0	0
6	1	1	0	0	0	0	0	0
7	1	0	1	0	0	0	0	0
8	0	1	1	0	0	0	0	0

(b) La matrice d'adjacence de G.

FIGURE 2.3 – Un graphe et sa matrice d'adjacence.



(a) Graphe G ayant pour ensemble de sommets $\{a, b, c\}$.

	a	b	c
a	0	1	0
b	1	0	1
c	0	1	0

(b) Une matrice d'adjacence de G.

	a	c	b
a	0	0	1
c	0	0	1
b	1	1	0

(c) Une autre matrice d'adjacence de G.

FIGURE 2.4 – Un graphe et deux de ses matrices d'adjacence.

Étant donné un graphe G et un ensemble de sommets $X \subseteq V(G)$, le *graphe induit* par X est le graphe noté $G[X]$ tel que $V(G[X]) = X$ et $E(G[X]) = E(G) \cap (X \times X)$. Autrement dit, il s'agit du graphe ayant X comme ensemble de sommets et ayant comme arêtes celles de G dont les deux extrémités sont des sommets de X. Étant donné un graphe G et deux ensembles de sommets $X, Y \subseteq V(G)$, la *coupe induite* par X et Y est le graphe noté $G[X, Y]$ tel que $V(G[X, Y]) = X \cap Y$ et $E(G[X, Y]) = E(G) \cap ((X \times Y) \cup (Y \times X))$. Autrement dit, il s'agit du graphe ayant $X \cup Y$ comme ensemble de sommets et ayant comme arêtes celles de G dont une extrémité est un sommet de X et l'autre extrémité est un sommet de Y.

Dans un graphe G, un *chemin* est un ensemble de sommets $u_1, \ldots, u_k$ distincts de G tel que pour tout $1 \leq i \leq k-1$, (u_i, u_{i+1}) est une arête de G. Un *cycle* est un chemin $u_1, \ldots, u_k$ de G tel que (u_k, u_1) est une arête de G. Un graphe est *connexe* lorsque pour toute paire de sommets u, v de G il existe un chemin $u_1, \ldots, u_k$ tel que $u_1 = u$ et $u_k = v$. Une *forêt* est un graphe sans cycle et un *arbre* est une forêt connexe. Quand on considère un graphe qui est un arbre, on le note souvent T (pour l'anglais *tree*) à la place de G. De plus, les sommets du graphe T sont généralement appelés *nœuds*. Étant donné un arbre T, on peut l'*enraciner*, c'est-à-dire choisir un sommet r de T et l'appeler *racine*. Dans ce cas, on peut partitionner les voisins de chaque nœud u de T :

— d'un côté, les nœuds *fils* de u, qui sont les v voisins de u tels que l'unique chemin allant de v à la racine r contienne u ;

— de l'autre, le nœud *père* de u, qui est le voisin v de u qui est inclus dans l'unique chemin allant de u à la racine r.

La racine est le seul nœud de T à ne pas avoir de père. Un nœud sans fils est appelé une *feuille*. Les nœuds *ascendants* d'un nœud u sont définis récursivement :

— le père de u est un ascendant de u,

— le père d'un ascendant de u est un ascendant de u.

De même, les nœuds *descendants* d'un nœud u sont définis récursivement de sorte qu'un fils de u est un descendant de u et un fils d'un descendant de u est un descendant de u. La figure 2.5 montre un exemple d'arbre enraciné.

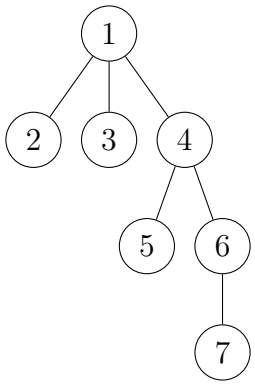


FIGURE 2.5 – Un arbre T ayant pour racine le nœud 1. Les fils de 1 sont les nœuds 2, 3 et 4. Le nœud 4 a pour père 1 et pour fils 5 et 6. Les nœuds 2,3,5 et 7 sont des feuilles. Les descendants du nœud 4 sont les nœuds 5,6 et 7.

Une *classe* de graphes est une collection (en général infinie) de graphes. Par exemple, la classe des arbres contient tous les graphes qui sont des arbres.

2.1.3 Hypergraphes

L'hypergraphe est un objet qui est globalement une famille d'ensembles. C'est aussi une généralisation de la notion de graphe. Un hypergraphe est, comme un graphe, constitué de sommets et d'arêtes. Traditionnellement, une arête d'un hypergraphe est appelée *hyper-arête*. Formellement, un hypergraphe H est composé d'un ensemble de sommets $V(H)$ et d'un ensemble d'hyper-arêtes $\mathcal{E}(H)$ qui est un sous-ensemble de $2^{V(H)}$. Là où une arête d'un graphe a deux extrémités, l'hyper-arête d'un hypergraphe a n'importe quel nombre d'extrémités. Si l'ensemble $V(H)$ est clair d'après le contexte, il arrive qu'on identifie l'hypergraphe H et la famille $\mathcal{E}(H)$. Ainsi, $\emptyset$ peut désigner l'hypergraphe sans hyper-arête dont l'ensemble des sommets est clair d'après le contexte. De même, $\{A, B\}$ peut désigner l'hypergraphe H tel que $\mathcal{E}(H) = \{A, B\}$ et tel que $V(H)$ se déduit à partir du contexte. Enfin, $A \in H$ signifie que A est une hyper-arête de H.

2.2 Compression de données

2.2.1 Suite binaire

Définition 2.1 (Notation $\texttt{pos}_s(i)$)**.** *Étant donné une suite binaire s composée de n bits, on note $\texttt{pos}_s(i)$ la position du i-ème 1 de la suite s, en commençant par la droite. L'entier i est compris entre 1 et le nombre de 1 dans la suite, et $\texttt{pos}_s(i)$ est compris entre 0 et $n-1$. On note $|s|$ la taille de la suite s, c'est-à-dire que $|s| = n$.*

On note 0^n la suite composée de n symboles 0, et on note 1^n la suite composée de n symboles 1. Étant donné deux suites s_1 et s_2, $s_1 s_2$ désigne la concaténation des deux suites. Pour désigner la concaténation de k suites $s_1, \dots s_k$, on note :

$$\prod_{i=1}^{k} s_i.$$

Exemple 2.1. Soit $s_1 = 0^4$, $s_2 = 1^3$, $s_3 = s_1 s_2$. On a $s_1 = \texttt{0000}$, $s_2 = \texttt{111}$ et $s_3 = \texttt{0000111}$.

Exemple 2.2. Soit la suite $s = \texttt{00011001}$. On a $\texttt{pos}_s(1) = 0$, $\texttt{pos}_s(2) = 3$ et $\texttt{pos}_s(3) = 4$.

On note ϵ la suite vide, c'est-à-dire l'unique suite s telle que $|s| = 0$. Étant donné une suite binaire s, on note $\overline{s}$ la suite obtenue en remplaçant chaque 0 par un 1 et chaque 1 par un 0. Enfin, étant donné deux suites s_1 et s_2, on dit que s_1 est *préfixe* de s_2 s'il existe une suite s telle que $s_2 = s_1 s$. Autrement dit, s_1 est préfixe de s_2 si $|s_1| \leq |s_2|$ et si la suite obtenue en ne gardant que les $|s_1|$ éléments les plus à gauche de s_2 donne la suite s_1.

2.2.2 Compression d'une suite binaire

N'importe quelle donnée numérique peut-être représentée par une suite de 0 et de 1. Le but de cette section est de regarder comment faire pour compresser une telle suite. Il existe de nombreuses méthodes pour cela, comme les compressions de Lempel-Ziv ou les compressions basées sur l'entropie. Je me suis intéressé à ces dernières. L'idée est la suivante : si la proportion de 0 et de 1 dans une suite est déséquilibrée (par exemple s'il y a 90 % de 0 pour 10 % de 1), on peut prendre en compte ce déséquilibre pour compresser la suite. C'est le principe de l'entropie binaire et du théorème du codage des sources de Shannon.

Définition 2.2 (Entropie binaire)**.** *Soit un nombre réel p compris entre 0 et 1. L'*entropie binaire *de p, noté $h(p)$, est la quantité $h(p) := -p \log p - (1-p)\log(1-p)$.*

Le nombre p représente soit la densité d'une suite, c'est-à-dire le rapport entre le nombre de 1 dans la suite et la taille de la suite, soit de façon équivalente la probabilité qu'un bit de la suite soit un 1. Le graphique de la fonction d'entropie binaire $h(p)$ est donnée dans la figure 2.6.

Définition 2.3 (Codage uniquement décodable)**.** *Un codage est* uniquement décodable *lorsque dès que l'on concatène n'importe quel nombre d'encodages les uns à la suite des autres, il existe une unique façon de les séparer à nouveau.*

Par exemple, l'ensemble de codes 0, 1 et 10 n'est pas uniquement décodable, car la concaténation 110 peut être découpée en 1, 1, et 0 ou en 1 et 10. En revanche, l'ensemble de codes 10, 110 et 111 est uniquement décodable.

Une façon de garantir qu'un code soit uniquement décodable consiste à imposer qu'il soit *sans préfixe*, c'est-à-dire que dans l'ensemble de codes, aucun code n'est préfixe d'un autre. Par exemple, l'ensemble de codes 0, 1 et 10 n'est pas sans préfixe car 1 est préfixe de 10. En revanche, l'ensemble de codes 10, 110 et 111 est sans préfixe. Il existe des codages uniquement décodables qui ne sont pas sans préfixe. C'est par exemple le cas de l'ensemble de codes 11 et 101.

Ces définitions permettent d'énoncer le théorème du codage des sources de Shannon appliqué aux suites binaires.

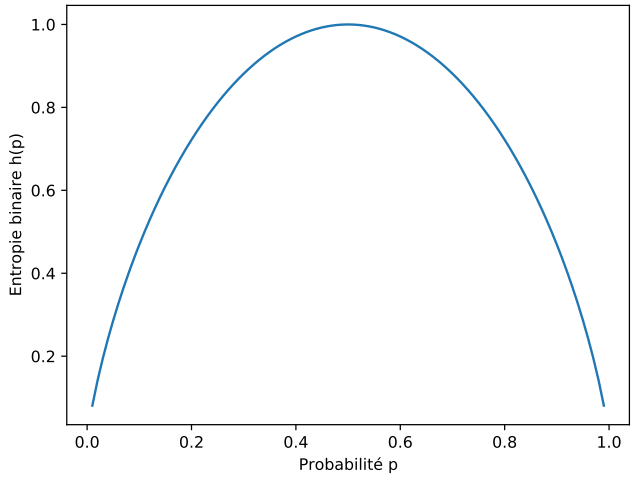


FIGURE 2.6 – Graphique de la fonction d'entropie binaire $h(p)$.

Théorème 2.1 (Codage des sources de Shannon)**.** *Soit une suite binaire aléatoire de taille n telle que chaque bit a une probabilité p d'être un 1. Alors tout encodage uniquement décodable de cette suite doit utiliser en moyenne au moins $n \cdot h(p)$ bits.*

Le résultat du codage des sources de Shannon est un résultat négatif car il donne une borne inférieure. Le but est alors d'approcher cette borne au maximum. Une version complète de ce théorème dit également que si n tend vers l'infini, la taille de l'encodage divisée par la taille de la suite tend vers $h(p)$ bits.

Exemple 2.3. Prenons une suite aléatoire s de taille $n = 3$ et une probabilité $p = 1/4$ qu'un bit vaille 1. Soit e_0 l'encodage de 000, e_1 l'encodage de 001, e_2 l'encodage de 010, et de façon générale, e_i est l'encodage de la suite qui représente i en binaire. Alors le théorème de codage des sources de Shannon nous dit que :

$$\sum_{i=0}^{7} |e_i| P(s = e_i) \geq 3 \cdot h(1/4) \geq 2,43.$$

On peut tendre vers cette borne, en choisissant par exemple le codage suivant :

s	encodage	s	encodage
000	00	100	110
001	100	101	1110
010	101	110	011
011	010	111	1111

Dans ce cas, on a :

$$\sum_{i=0}^{7} |e_i| P(s = e_i) \leq 2.65.$$

Il existe plusieurs méthodes pour atteindre cette borne. Ces méthodes s'appellent des *codages entropiques*. Bien que la plus connue soit le codage arithmétique [WNC87],

nous avons utilisé un autre codage basé sur l'entropie appelé le Système de numération combinatoire. En effet, en implémentant ces deux codages, nous nous sommes aperçus que le premier était assez lent sur de grandes instances de suites binaires. Nous avons donc opté pour le second.

Système de numération combinatoire

Afin d'appliquer le Théorème de codage des sources de Shannon, il faut se fixer une densité p et essayer d'encoder toutes suites qui ont cette densité p de 1. Cependant, comme il s'agit d'un nombre réel et que l'on travaille avec des objets discrets, une approche plus naturelle est de considérer les suites de taille n qui contiennent k symboles 1, ce qui implique bien de considérer une densité fixe de $p = k/n$. Par la suite, on note $\mathcal{S}_k^n$ l'ensemble de ces suites.

Définition 2.4 (Notation $\mathcal{S}_k^n$)**.** *On note $\mathcal{S}_k^n$ l'ensemble des suites binaires (c'est-à-dire faites de 0 et de 1) de tailles n et contenant k symboles 1.*

Pour commencer, on peut dénombrer ces suites. Par définition des coefficients, binomiaux, il y a $\binom{n}{k}$ telles suites. Ainsi, on a :

$$|\mathcal{S}_k^n| = \binom{n}{k}.$$

Une façon assez naïve d'encoder ces suites est donc d'associer à chacune d'elles un numéro entre 0 et $\binom{n}{k} - 1$. C'est exactement le principe du Système de numération combinatoire. Avant d'aller plus loin, il est intéressant de regarder quelle est la taille en binaire du nombre $\binom{n}{k} - 1$, car c'est alors la taille maximale du codage d'une suite de $\mathcal{S}_k^n$.

Lemme 2.1 ([Cov99])**.** *On a l'inégalité suivante :*

$$\log \binom{n}{k} \leq n \cdot h(n/k).$$

A première vue, cette inégalité contredit le Théorème du codage des sources de Shannon, car ce dernier affirme qu'un tel codage prend au moins $n \cdot h(n/k)$ bits. La subtilité se trouve dans le fait que ce codage n'encode que les suites de $\mathcal{S}_k^n$, alors que le théorème s'intéresse aux encodages de toutes les suites binaires dont la densité est *en moyenne* n/k.

On peut maintenant expliciter le Système de numération combinatoire. Pour ce faire, il faut trouver une formule qui associe à chaque suite de $\mathcal{S}_k^n$ un entier entre 0 et $\binom{n}{k} - 1$. Pour cela, on utilise la *représentation de Macaulay d'un entier*.

Lemme 2.2. *Étant donné deux entiers k et t, il existe une unique suite strictement croissante $c_1, \ldots, c_k$ telle que :*

$$t = \sum_{i=1}^{k} \binom{c_i}{i}.$$

Cela motive à considérer la bijection suivante entre les suites de $\mathcal{S}_k^n$ et les nombres entre 0 et $\binom{n}{k} - 1$.

Définition 2.5 (Notation $\texttt{CNS}(s)$)**.** *Soit une suite s de $\mathcal{S}_k^n$. L'encodage via le système de numération combinatoire de cette suite est le nombre :*

$$\mathit{CNS}(s) = \sum_{i=1}^{k} \binom{\mathit{pos}_s(i)}{i}.$$

On rappelle que $\binom{j}{i} = 0$ si $j < i$. De cette manière, si deux suites sont différentes, on sait que les nombres obtenus seront aussi différents grâce au lemme 2.2.

Exemple 2.4. Voici les 3 suites de $\mathcal{S}_k^n$ quand $n = 3$ et $k = 1$. L'encodage de `001` est $\binom{0}{1} = 0$. L'encodage de `010` est $\binom{1}{1} = 1$. L'encodage de `100` est $\binom{2}{1} = 2$.

Voici l'encodage de la suite `11001` : $\binom{0}{1} + \binom{3}{2} + \binom{4}{3} = 7$.

Maintenant, si on a une suite binaire s, on peut l'encoder à l'aide de trois nombres : le nombre n qui correspond à la taille de la suite, le nombre k qui correspond au nombre de 1 dans la suite et le nombre $\texttt{CNS}(s)$ qui correspond à l'encodage à proprement parlé des données de la suite. Cet encodage permet de trouver la suite binaire originale s : grâce à k et n, on sait que la suite que l'on cherche appartient à $\mathcal{S}_k^n$; et on sait que la fonction $\texttt{CNS}(\cdot)$ établit une bijection entre $\mathcal{S}_k^n$ et les entiers inférieurs à $\binom{n}{k}$.

Si $p = n/k$ n'est pas trop proche de $1/2$, on obtient une compression de s. En effet, la suite s'encode sans compression en n bits, alors qu'avec cette compression, elle s'encode en moins de $\log k + \log n + n \cdot h(n/k)$ bits. Puisqu p n'est pas trop proche de $1/2$, la quantité $h(n/k) = h(p)$ est inférieure à 1 et l'encodage peut espérer être inférieur à n.

Exemple 2.5. Soit s la suite `00000011 00000001 00001101 00011001`. On a :

$$\begin{aligned}\texttt{CNS}(s) &= \binom{0}{1} + \binom{3}{2} + \binom{4}{3} + \binom{8}{4} + \binom{10}{5} + \binom{11}{5} + \binom{16}{6} + \binom{24}{7} + \binom{25}{8} \\ &= 0 + 3 + 4 + 70 + 252 + 462 + 11440 + 735471 + 2042975 = 2790677.\end{aligned}$$

Ainsi, la suite s est entièrement représentable par les nombres $n = 32$, $k = 9$ et $\texttt{CNS}(s) = 2790677$.

Remarque 2.1. La formule de $\texttt{CNS}(s)$ ne dépend pas de la longueur n de la suite. Ainsi, si l'on considère une suite s et la suite $s' = 0^\ell s$ obtenue en ajoutant ℓ 0 devant s, pour un certain entier ℓ, alors on a $\texttt{CNS}(s) = \texttt{CNS}(s')$. La donnée du nombre n est donc à la fois indispensable pour connaitre la longueur de la suite, mais ne donne au final que l'information du nombre de zéros initiaux. On peut alors se placer dans l'ensemble $\mathcal{S}_k^* := \bigcup_n \mathcal{S}_k^n$ des suites binaires ayant k 1 et n'importe quel nombre de zéros. Dans cet ensemble $\mathcal{S}_k^*$, deux suites s et s' peuvent alors vérifier $\texttt{CNS}(s) = \texttt{CNS}(s')$, mais seulement si l'une est obtenue en ajoutant des zéros initiaux à l'autre. Si l'on considère maintenant uniquement les suites de $\mathcal{S}_k^*$ sans zéro initial, on a une bijection entre cet ensemble et l'ensemble $\mathbb{N}$ des entiers via la fonction $\texttt{CNS}(\cdot)$.

2.2.3 Représentation de plusieurs suites binaires

On considère maintenant que l'on veut encoder une suite de suites binaires. Grâce à la partie précédente, on sait qu'une seule suite binaire peut être représentée par trois entiers. Ainsi, encoder une suite de suites binaires revient à vouloir encoder une suite d'entiers. En général, pour encoder un seul entier, on utilise le binaire. Cependant, si on veut encoder plusieurs entiers, il faut une façon de délimiter où commence et où se termine chaque entier.

Exemple 2.6. On veut encoder les entiers 3 et 12. En binaire, 3 s'écrit `11` et 12 s'écrit `1100`. Si on se contente de juxtaposer ces deux encodages, on obtient `111100`. Cependant, comme on ne sait pas où commence et où finit chaque encodage, on ne sait pas si cette suite représente 3 et 12, ou par exemple 1 et 28 (en la découpant en `1 11100`), ou encore 7 et 4 (en la découpant en `111 100`), ou même 3, 1 et 4 (en la découpant en `11 1 100`).

Il existe plusieurs façon de remédier à ce problème :

— Utiliser un délimiteur entre chaque entier. Cela nécessite d'interdire une suite de bits afin de l'utiliser comme délimiteur. Par exemple, une façon de faire est de travailler en base 15 à la place de l'hexadécimal. Dans ce cas, on peut utiliser la suite `1111` comme délimiteur. Le problème principal de cette approche est qu'il faut ré-encoder tous les entiers, vu que les ordinateurs utilisent le système hexadécimal.

— Utiliser un *code à longueur fixe*. Cela consiste à encoder chaque entier en utilisant le même nombre de bits pour chaque entier. Pour que cette approche soit efficace, il faut que les différents entiers aient le même ordre de grandeur.

— Utiliser un *code préfixe*. Il s'agit d'un code où aucun encodage n'est préfixe d'un autre. Cela garantit qu'il n'y a qu'une seule façon de décoder une suite donnée et cela enlève l'ambigüité présentée dans l'exemple 2.6.

J'ai décidé d'opter pour la dernière option car elle offre plus de flexibilité. Il existe de nombreux codes préfixes. Je me suis intéressé aux représentations d'Élias [Eli75]. Plus particulièrement, je me suis intéressé à la représentation γ composée d'Élias et la représentation δ doublement composée d'Élias. Afin de les décrire, on commence par les représentations unaires α et binaires β.

Définition 2.6 (Représentation unaire)**.** *Soit n un entier. La représentation unaire de n est $\alpha(n) := 0^n1$.*

Définition 2.7 (Représentation binaire)**.** *Soit n un entier. La représentation binaire de n est son écriture en base 2. Formellement, cela s'écrit :*

$$\beta(n) = \prod_{i=1}^{\lceil \log(n+1) \rceil} \left\lfloor \frac{n \bmod 2^i}{2^{i-1}} \right\rfloor$$

Définition 2.8 (Représentation gamma)**.** *Soit n un entier. La représentation γ composée d'Élias est $\gamma(n) := \alpha(|\beta(n)|)\ \beta(n)$.*

Définition 2.9 (Représentation delta)**.** *Soit n un entier. La représentation δ doublement composée d'Élias est $\delta(n) := \gamma(|\beta(n)|)\ \beta(n) = \alpha(|\beta(|\beta(n)|)|)\ \beta(|\beta(n)|)\ \beta(n)$.*

Exemple 2.7. Soit n valant 34 millions. On a :

$\alpha(n) = 0^n1$, $\beta(n) = 10000001101100110010000000$, $|\beta(n)| = 26$, $\alpha(|\beta(n)|) = 0^{26}1$, $\gamma(n) = 0^{26}11000000110110011001000000$, $\beta(|\beta(n)|) = \beta(26) = 11010$, $\alpha(|\beta(|\beta(n)|)|) = \alpha(|11010|) = \alpha(5) = 000001$ et $\delta(n) = 00000111010100000011011001100100000000$.

La taille des deux représentations d'Élias est ici $|\gamma(n)| = 53$ et $|\delta(n)| = 37$.

De façon générale, on a $|\gamma(n)| \approx 2\log n$ et $|\delta(n)| \approx \log n + 2\log\log n$.

Interprétation

Le but des représentations d'Élias est d'avoir un code préfixe (de sorte à ce qu'on puisse le décoder sans ambigüité) qui soit le plus concis possible. En effet, il existe un code préfixe très simple, qui est le code unaire. Puisque l'ensemble des $\alpha(n)$ pour n allant de 0 à l'infini est l'ensemble des suites 0^n1, on voit qu'aucune suite n'est préfixe d'une autre. Cependant, cet encodage n'est pas du tout efficace. L'idée derrière la représentation γ composée d'Élias est d'utiliser le fait que la représentation unaire soit préfixe pour pouvoir encoder un premier nombre et ensuite utiliser ce nombre pour définir la taille de la donnée que l'on veut encoder.

Cette idée peut être généralisée : pour créer un nouveau codage préfixe capable d'encoder tout entier n, on concatène d'une part le codage de la valeur de $|\beta(n)|$ en utilisant un ancien code préfixe et d'autre part le codage binaire de n, c'est-à-dire $\beta(n)$. En faisant cela avec le code préfixe γ, ou obtient le code δ. On pourrait continuer davantage.

Généralisation

[Eli75] propose une version finale de cette idée en utilisant la récursivité. Il définit la représentation ω pénultième par : $\omega(n) = \omega(|\beta(n) - 1|)\ \beta(n)$ et $\omega(1) = \epsilon$. Le fait de soustraire 1 permet de faire en sorte que 1 soit point fixe de $n \to |\beta(n) - 1|$. Sinon, on a à la fois 1 et 2 qui sont points fixes de $n \to |\beta(n)|$. Un problème qui apparait ici est de savoir combien de fois ω a été appelé récursivement, afin de savoir quand s'arrêter de décoder. Pour pallier ce problème, on remarque que $\beta(n)$ commence toujours par un 1 dès que $n \geq 1$. Il suffit alors de mettre un 0 à la fin de l'encodage pour marquer la fin.

Variantes et optimisations

Une première optimisation vient de la remarque précédente : tout nombre strictement positif possède un 1 comme symbole le plus à gauche dans sa notation binaire. Puisque ce 1 est toujours présent, il n'est pas nécessaire de l'encoder. Une variante possible consiste à entremêler les bits de la représentation unaire et de la représentation binaire de la représentation gamma (et respectivement, d'entremêler les bits de la représentation gamma et de la représentation binaire dans la représentation delta). Si on prend en compte cette optimisation et cet entremêlement, on obtient les représentations gamma et delta d'Élias telles que décrites dans [Eli75]. Cependant, j'ai décidé d'ignorer ce détail à des fins de simplicité, étant que ça ne change aucune borne pratique ou théorique.

Finalement, j'ai décidé de créer une autre variante du codage gamma. Ce qui est problématique avec ce codage, c'est que la taille de la suite qui encode n est environ $2 \log n$, soit deux fois plus que la représentation binaire habituelle. J'ai décidé de le modifier. On rappelle que le codage gamma produit une suite de la forme $s_1 s_2$, où s_1 encode la taille de s_2 et s_2 le nombre n. On peut modifier cela en disant que s_1 encode un nombre capable de retrouver la taille de s_2. Mon approche a été de dire que s_1 encode le quart de la taille de s_2, arrondi au supérieur. Il suffit alors de multiplier ce nombre par 4 pour trouver la taille de s_2. Un défaut qui arrive immédiatement est qu'on ne peut encoder que des tailles de s_2 multiples de 4 avec ce système. Mon idée est alors d'ajouter des zéro en tête de la suite de s_2 jusqu'à ce qu'elle soit multiple de 4. Comme s_2 est encodé en binaire, ajouter des 0 au début ne modifie pas la valeur de ce qui est encodé.

Je note mes variantes γ^* et δ^*.

Définition 2.10 (Représentation gamma modifiée)**.** *Soit n un entier. La représentation γ modifiée est :*

$$\gamma^*(n) := \alpha\left(\left\lceil \frac{|\beta(n)|}{4} \right\rceil\right) 0^{(-|\beta(n)|) \bmod 4} \beta(n).$$

Exemple 2.8. On a $\gamma^*(9) = 01\ 1001$, $\gamma^*(15) = 01\ 1111$ et $\gamma^*(16) = 001\ 00010000$. Les espaces ont été ajoutés pour la lisibilité.

Ainsi, on peut définir la représentation δ^* à partir de γ^* de la même façon que la représentation δ est définie à partir de γ :

Définition 2.11 (Représentation delta modifiée)**.** *Soit n un entier. La représentation δ modifiée est $\delta^*(n) := \gamma^*(|\beta(n)|)\ \beta(n)$.*

On peut maintenant définir une nouvelle représentation préfixe, qui est celle que j'ai utilisée en pratique par la suite. Le but est de combiner les représentations γ^* et δ^* en une seule représentation, en fonction de laquelle des deux est plus courte. En effet, quand n est petit, on a généralement $|\gamma^*(n)| \leq |\delta^*(n)|$, et quand n est grand, on a généralement l'inégalité inverse. De façon précise, on a le graphique de la figure 2.7.

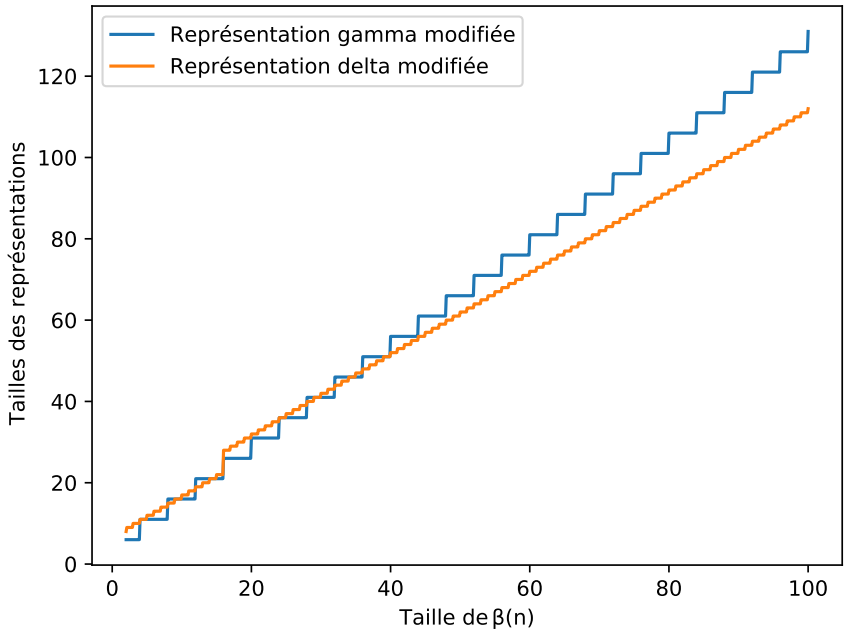


FIGURE 2.7 – Comparaison de $|\gamma^*(n)|$ et $|\delta^*(n)|$ pour les différentes valeurs possibles de $|\beta(n)| = \lceil \log(n) \rceil$. On remarque que la valeur de $|\gamma^*(n)|$ est plus petite quand n est petit mais que celle de $|\delta^*(n)|$ est plus petite quand n est grand. Le point de bascule se fait entre $n = 2^{25}$ et $n = 2^{40}$.

Une façon naïve de combiner ces deux représentations en une seule est d'ajouter tout à gauche un bit qui indique quelle est la représentation utilisée. Une façon plus optimisée consiste à remarquer que le choix de la représentation unaire utilisée est arbitraire. On aurait tout aussi bien pu définir $\alpha(n)$ comme étant 1^n0. On définit alors les représentations γ^* et δ^* complémentaires en utilisant cette version alternative de la représentation unaire.

Définition 2.12 (Représentations gamma et delta modifiées complémentaires)**.** *Soit n un entier. On note $\overline{\alpha}(n) = 1^n0$. On définit alors :*

$$\overline{\gamma}^*(n) := \overline{\alpha}\left(\left\lceil \frac{|\beta(n)|}{4} \right\rceil\right) 0^{(-|\beta(n)|) \bmod 4}\beta(n) \quad \textit{et} \quad \overline{\delta}^*(n) := \overline{\gamma}^*(|\beta(n)|)\beta(n).$$

On remarque que $\gamma^*(n)$ et $\overline{\gamma}^*(n)$ sont identiques à l'exception de l'encodage unaire initial. De même pour $\delta^*(n)$ et $\overline{\delta}^*(n)$. On peut maintenant définir la *représentation mixte* que j'utilise en pratique par la suite. Je la note $\texttt{mixte}(n)$.

Définition 2.13 (Représentation mixte)**.** *Soit n un entier. Si $n \leq 2^{25}$, on pose $\texttt{mixte}(n) = \gamma^*(n)$. Sinon, on pose $\texttt{mixte}(n) = \overline{\delta}^*(n)$.*

Le choix de 2^{25} s'est fait en accord avec la figure 2.7. D'après cette figure, n'importe quelle valeur entre 2^{25} et 2^{40} semble possible.

Exemple 2.9. On a $\beta(26) = 11010$, $\texttt{mixte}(26) = 001\ 00011010$ (les espaces sont ajoutées pour la lisibilité). Avec n valant 34 millions, on a $\beta(n) = 10000001101100110010000000$, $|\beta(n)| = 26$ et $\texttt{mixte}(n) = 110\ 00011010\ 10000001101100110010000000$.

Tailles et triple composition

De façon approximative, on a $|\gamma^*(n)| \approx 5/4 \log n$ et $|\delta^*(n)| \approx \log n + 5/4 \log\log n$. Puisque la représentation γ (et γ^*) correspond à une simple composition et que la représentation δ (et $\overline{\delta}^*$) correspond à une double composition, on pourrait se demander s'il est envisageable d'utiliser une triple composition pour encoder des entiers. Pour cela, il faut se demander à partir de quelle taille d'entier cela devient plus efficace que la double composition. On a pu constater que le saut entre gamma et delta se fait autour de $n = 2^{25}$. C'est-à-dire que pour $n \geq 2^{25}$, on a $|\delta^*(n)| - |\gamma^*(n)| \geq 0$. Ainsi, en notant η la triple composition, on étudie la différence $|\eta^*(n)| - |\delta^*(n)| \approx |\delta^*(\log n)| + \log n - (|\gamma^*(\log n)| + \log n) = |\delta^*(\log n)| - |\gamma^*(\log n)|$. Or, on sait que cette quantité devient positive quand $\log n \geq 2^{25}$, c'est-à-dire quand n possède plus de 2^{25} chiffres binaires, ce qui équivaut à un nombre dont l'écriture binaire nécessite environ un gigaoctet. De plus, à ce stade là, on n'économise encore aucun bit. Cela justifie le fait que la triple composition n'a pas été envisagée.

2.2.4 Résumé et exemple

Finalement, d'après les sections précédentes, il est possible d'encoder n'importe quel tableau d'objets. Pour encoder un tableau d'objets $[o_1, \ldots, o_\ell]$, on commence par transformer chaque objet en une suite de 0 et de 1 qui représente cet objet. En général, cela est déjà fait quand on manipule des objets informatiques. On obtient alors un tableau de suites binaires $[s_1, \ldots, s_\ell]$. Ensuite, chaque suite de 0 et de 1 est transformée en trois entiers en utilisant ce qui a été vu à la section 2.2.2. On a alors transformé notre tableau de ℓ objets quelconques en un tableau de 3ℓ entiers $[k_1, n_1, \texttt{CNS}(s_1), \ldots, k_\ell, n_\ell, \texttt{CNS}(s_\ell)]$. On utilise alors ce que l'on a vu dans la section 2.2.3 pour représenter ce tableau d'entiers, c'est-à-dire que l'on va encoder la suite suivante :

$$\texttt{mixte}(k_1)\ \texttt{mixte}(n_1)\ \texttt{mixte}(\texttt{CNS}(s_1))\ \ldots\ \texttt{mixte}(k_\ell)\ \texttt{mixte}(n_\ell)\ \texttt{mixte}(\texttt{CNS}(s_\ell)).$$

Cependant, si cette représentation est utilisée au milieu d'autres données, le problème de fin de données se pose à nouveau : on ne sait pas quand cette représentation se termine ; ou autrement dit, on ne connait pas la taille 3ℓ du tableau. Ainsi, la représentation

complète (et qui est bien un code préfixe) du tableau est :

$$\texttt{mixte}(3\ell)\ \texttt{mixte}(k_1)\ \texttt{mixte}(n_1)\ \texttt{mixte}(\texttt{CNS}(s_1))\ \ldots$$

Exemple 2.10. On a un tableau de trois objets dont les représentations binaires sont :

$$\begin{aligned}
s_1 &= 1001001\\
s_2 &= 11011001101111111101101100101\\
s_3 &= 10000111100001110110111010011101100110110011001100101
\end{aligned}$$

En utilisant le système de numération combinatoire :

$$\begin{array}{lll}
n_1 = 7, & k_1 = 3, & \texttt{CNS}(s_1) = 23\\
n_2 = 28, & k_2 = 19, & \texttt{CNS}(s_2) = 4409015\\
n_3 = 49, & k_3 = 27, & \texttt{CNS}(s_3) = 23233639441304
\end{array}$$

Alors on a :

$\texttt{mixte}(9) = 01\ 1001$
$\texttt{mixte}(n_1) = 01\ 0111$
$\texttt{mixte}(k_1) = 01\ 0011$
$\texttt{mixte}(\texttt{CNS}(s_1)) = 001\ 00010111$
$\texttt{mixte}(n_2) = 001\ 00011100$
$\texttt{mixte}(k_2) = 001\ 00010011$
$\texttt{mixte}(\texttt{CNS}(s_2)) = 0000001\ 010000110100011010110111$
$\texttt{mixte}(n_3) = 001\ 00110001$
$\texttt{mixte}(k_3) = 001\ 00011011$
$\texttt{mixte}(\texttt{CNS}(s_3)) = 110\ 00101101\ 101010010000110000000110100110110001110011000$

Finalement, le tableau contenant ces trois objets s'encode :

$S = 01\ 1001\quad 01\ 0111\quad 01\ 0011\quad 001\ 00010111\quad 001\ 00011100\quad 001\ 00010011$
$0000001\ 010000110100011010110111\quad 001\ 00110001\quad 001\ 00011011$
$110\ 00101101\ 101010010000110000000110100110110001110011000$

Dans cet exemple, on a représenté trois objets avec une taille totale de $|s_1|+|s_2|+|s_3| = 84$ via un suite binaire S de taille 160. Au premier abord, on a l'impression que cette compression produit une suite plus grande que la suite originale. Cependant, en plus de compresser, S a aussi pour rôle d'introduire une délimitation entre les données. On rappelle que la façon naïve de faire cela consiste à utiliser une représentation unaire de chaque taille $|s_1|$, $|s_2|$ et $|s_3|$, ce qui double la taille de la représentation. On aurait donc naïvement une représentation du tableau $[s_1, s_2, s_3]$ de $84 \times 2 = 168$ bits. S est donc plus petit que la représentation naïve. De plus, les objets o_1, o_2 et o_3 ont une densité $p_i = k_i/n_i$ proche de $1/2$. Dans les cas réels, on cherchera à avoir des objets de densité loin de $1/2$ qui permettront une meilleure compression. L'exemple 2.11 montre un cas où la densité de chaque objet est environ $p \simeq 1/7$. L'écart avec la représentation naïve est alors bien plus net, la représentation naïve a une taille de $84 \times 2 = 168$ bits, alors que la représentation compressée a une taille de 124 bits.

Exemple 2.11. On a un tableau de trois objets dont les représentations binaires sont :

$$s_1 = 1000001$$
$$s_2 = 10000001000000100000001000001$$
$$s_3 = 100000010000001000000100000010000001000000100000001$$

En utilisant le système de numération combinatoire :

$$n_1 = 7, \quad k_1 = 2, \quad \texttt{CNS}(s_1) = 15$$
$$n_2 = 28, \quad k_2 = 5, \quad \texttt{CNS}(s_2) = 85876$$
$$n_3 = 49, \quad k_3 = 8, \quad \texttt{CNS}(s_3) = 401261714$$

Alors on a :

$$\texttt{mixte}(9) = 01\ 1001$$
$$\texttt{mixte}(n_1) = 01\ 0111$$
$$\texttt{mixte}(k_1) = 01\ 0010$$
$$\texttt{mixte}(\texttt{CNS}(s_1)) = 01\ 1111$$
$$\texttt{mixte}(n_2) = 001\ 00011100$$
$$\texttt{mixte}(k_2) = 01\ 0101$$
$$\texttt{mixte}(\texttt{CNS}(s_2)) = 000001\ 00010100111101110100$$
$$\texttt{mixte}(n_3) = 001\ 00110001$$
$$\texttt{mixte}(k_3) = 01\ 1000$$
$$\texttt{mixte}(\texttt{CNS}(s_3)) = 110\ 00011101\ 10111111010101100010010010010$$

2.3 Quelques outils mathématiques

On présente dans cette section des outils qui sont utiles pour comprendre et définir les notions centrales de la décomposition de graphes. Ces outils ont déjà été étudiés par le passé, mais je les présente et je les redéfinis par souci d'exhaustivité.

2.3.1 Les clôtures et leurs représentations

Opérateurs de clôture

Informellement, une clôture est une façon naturelle de compléter un ensemble d'objets. Un *opérateur de clôture* sur un ensemble F est une fonction qui prend en argument une partie A de F et renvoie une partie de F, notée $\mathrm{cl}(A)$. Un opérateur de clôture doit être *extensif*, ce qui signifie que $\mathrm{cl}(A)$ contient A. En d'autres termes, cette condition garantit que l'on peut toujours obtenir $\mathrm{cl}(A)$ en partant de A et en ajoutant des éléments. Un opérateur de clôture doit aussi être *monotone*, ce qui signifie que si on part d'un ensemble A et de sa clôture $\mathrm{cl}(A)$, et que l'on décide d'ajouter des éléments à A, alors cela va aussi ajouter des éléments à $\mathrm{cl}(A)$. Enfin, un opérateur de clôture doit être *idempotent*, ce qui signifie que l'appliquer plusieurs fois revient au même que de l'appliquer une seule fois. Formellement, cela donne la définition suivante :

Définition 2.14 (Opérateur de clôture)**.** *Soit F un ensemble. Une fonction $cl(\cdot) : 2^F \to 2^F$ est un* opérateur de clôture *si elle vérifie les trois conditions suivantes pour tous les ensembles $A, B \subseteq F$:*

— *(Extensif)* $A \subseteq cl(A)$,
— *(Monotone)* $A \subseteq B \implies cl(A) \subseteq cl(B)$, *et*
— *(Idempotent)* $cl(cl(A)) = cl(A)$.

Exemple 2.12. Prenons comme ensemble d'objets les entiers, c'est-à-dire que l'on prend $F = \mathbb{N}$, et prenons comme fonction f la fonction qui a un ensemble d'entiers associe le plus petit intervalle qui contient ces entiers. On a $f(\{1,4\}) = \{1,2,3,4\}$ et $f(\{1,4,7\}) = \{1,2,3,4,5,6,7\}$. Cette fonction est un opérateur de clôture. En effet, cette fonction est extensive (on obtient $f(A)$ en ajoutant des éléments à A pour « boucher les trous »), elle est monotone (ajouter des éléments à A ne peut pas enlever des éléments à $\mathrm{cl}\,(A)$) et idempotente (une fois que l'on a bouché les trous de A, appliquer f à nouveau ne va ajouter aucun élément).

Exemple 2.13. Un autre exemple vient de l'algèbre linéaire. On prend comme ensemble d'objets l'ensemble des vecteurs d'un espace vectoriel. On considère la fonction qui a un ensemble de vecteurs associe le plus petit sous-espace vectoriel qui contient ces vecteurs. Traditionnellement, cette fonction se note $<\cdot>$. Par exemple, avec $F = \mathbb{R}^3$, et on notant les vecteurs horizontalement, on a $<\{(0 \quad 0 \quad 1)\}> = \{(0 \quad 0 \quad \lambda) \mid \lambda \in \mathbb{R}\}$ et $<\{(0 \quad 1 \quad 0), (0 \quad 0 \quad 2)\}> = \{(0 \quad \lambda \quad \mu) \mid \lambda, \mu \in \mathbb{R}\}$.

Créer un opérateur de clôture

Lemme 2.3. *Soit $\mathcal{P}$ une famille de parties d'un ensemble F. Alors la fonction f qui a un ensemble $A \subseteq F$ associe $\bigcap_{A \subseteq X \in \mathcal{P}} X$ est un opérateur de clôture (en ayant pour convention qu'une intersection vide vaut F).*

Démonstration. Il faut montrer que f est extensive, monotone et idempotente. Pour simplifier les notations, on note $N(A) := \{X \in \mathcal{P} \mid A \subseteq X\}$. Ainsi, $f(A) = \bigcap_{X \in N(A)} X$.

— (Extensive) On montre que $A \subseteq f(A)$. Si l'ensemble $N(A)$ est vide, $f(A) = F$ par convention. Sinon, chaque $X \in N(A)$ contient A, donc leur intersection aussi, c'est-à-dire $f(A)$.
— (Monotone) On montre que si $A \subseteq B$, alors $f(A) \subseteq f(B)$. Pour cela, on remarque que si $A \subseteq B$, alors $N(B) \subseteq N(A)$. On conclut par décroissance de l'intersection.
— (Idempotente) Il suffit de montrer que $f(f(A)) \subseteq f(A)$, puisque l'autre inclusion se déduit par extensivité et monotonie. Pour cela, on montre que $N(A) \subseteq N(f(A))$. Soit un ensemble $Y \in N(A)$. On a $f(A) = \bigcap_{X \in N(A)} X \subseteq Y$, ce qui signifie que $Y \in N(f(A))$. Ainsi, par décroissance de l'intersection, $f(f(A)) \subseteq f(A)$. □

Familles de Moore

On a défini une clôture via un opérateur de clôture. Cependant, il existe plusieurs façons de représenter une clôture. On rappelle qu'une clôture est le concept selon lequel on complète de façon naturelle un ensemble d'objets. L'opérateur de clôture est une façon formelle de définir cette notion, mais il en existe d'autres [Vil21].

Définition 2.15 (Ensemble clos)**.** *Soit $cl(\cdot)$ un opérateur de clôture sur un ensemble F. Un ensemble $A \subseteq F$ est un ensemble* clos *lorsque $cl(A) = A$.*

Une façon de représenter une clôture est de lister tous les ensembles clos. En effet, on peut montrer que si on a la liste des ensembles clos, alors on peut retrouver l'opérateur de clôture $\mathrm{cl}\,(\cdot)$.

Lemme 2.4. *Soit $cl(\cdot)$ un opérateur de clôture sur l'ensemble F. Soit $\mathcal{C}$ l'ensemble des ensembles clos de $cl(\cdot)$, c'est-à-dire que $\mathcal{C} = \{A \subseteq F \mid cl(A) = A\}$. Alors pour tout ensemble $A \subseteq F$, on a :*

$$cl(A) = \bigcap_{A \subseteq X \in \mathcal{C}} X.$$

Démonstration. On montre cette égalité par double inclusion. Pour simplifier, on note $N := \bigcap_{A \subseteq X \in \mathcal{C}} X$.

- $\mathrm{cl}(A) \subseteq N$: Il suffit de montrer que pour tout ensemble X vérifiant $A \subseteq X \in \mathcal{C}$, on a $\mathrm{cl}(A) \subset X$. Cela est vrai par monotonie de l'opérateur de clôture : comme $A \subseteq X$, on a $\mathrm{cl}(A) \subseteq \mathrm{cl}(X)$. Ensuite, comme X est clos ($X \in \mathcal{C}$), cela signifie que $\mathrm{cl}(X) = X$, d'où l'inclusion voulue.
- $N \subseteq \mathrm{cl}(A)$: Il suffit de trouver un ensemble X vérifiant $A \subseteq X \in \mathcal{C}$ et tel que $X \subseteq \mathrm{cl}(A)$. On choisit $X = \mathrm{cl}(A)$. Il est évident que ce X satisfait $X \subseteq \mathrm{cl}(A)$. Il reste à montrer que X vérifie les conditions $A \subseteq X \in \mathcal{C}$. Par idempotence, on a $X = \mathrm{cl}(A) = \mathrm{cl}(\mathrm{cl}(A)) = \mathrm{cl}(X)$, ce qui prouve que X est clos, et que donc $X \in \mathcal{C}$. De plus, par extensivité, on a $A \subseteq \mathrm{cl}(A) = X$, ce qui montre que X vérifie bien les conditions qu'il doit vérifier.

Par double inclusion, on a l'égalité voulue. □

Autrement dit, on peut retrouver l'opérateur de clôture à partir de la liste des éléments clos par cet opérateur. Cette remarque amène à définir les *familles de Moore*, qui sont les familles pour lesquelles on peut montrer qu'elles sont constituées des éléments clos d'un certain opérateur de clôture.

Définition 2.16 (Famille de Moore)**.** *Une* famille de Moore $\mathcal{M}$ *sur un ensemble F est un ensemble de parties de F telle que :*

- *(Majoré) $F \in \mathcal{M}$, et*
- *(Stable par intersection) pour tout ensemble $A, B \in \mathcal{M}$, on a $A \cap B \in \mathcal{M}$.*

Exemple 2.14. L'ensemble des intervalles de $\mathbb{N}$ est une famille de Moore parce que $\mathbb{N}$ est un intervalle et que l'intersection de deux intervalles est un intervalle.

Exemple 2.15. L'ensemble des sous-espaces vectoriels de $\mathbb{R}^3$ est une famille de Moore parce que $\mathbb{R}^3$ est un sous-espace vectoriel de $\mathbb{R}^3$ et que l'intersection de deux sous-espaces vectoriels est un sous-espace vectoriel.

On montre maintenant que les familles de Moore et les opérateurs de clôture sont en fait deux représentations d'un même objet - les clôtures -, à condition que l'ensemble F sous-jacent soit finie. Cela signifie qu'à chaque famille de Moore, on peut associer un opérateur de clôture, et qu'à chaque opérateur de clôture on peut associer une famille de Moore. Cette correspondance a été présagée par le lemme 2.4.

Lemme 2.5. *Soit F un ensemble fini. On a les deux résultats suivants :*

1. *Soit $cl(\cdot)$ un opérateur de clôture sur un ensemble F. Alors l'ensemble $\mathcal{C}$ des éléments clos pour $cl(\cdot)$ est une famille de Moore.*
2. *Soit $\mathcal{M}$ une famille de Moore sur un ensemble F. Alors la fonction f qui a un ensemble $A \subseteq F$ associe $\bigcap_{A \subseteq X \in \mathcal{M}} X$ est un opérateur de clôture, et l'ensemble des éléments clos de f est $\mathcal{M}$.*

Démonstration. On montre chacun des deux points du lemme :

1. Il suffit de montrer que F est clos, et que l'intersection de deux éléments clos est clos. Comme $\mathrm{cl}(\cdot)$ est une fonction de 2^F dans 2^F, on sait que $\mathrm{cl}(F) \in 2^F$, c'est-à-dire que $\mathrm{cl}(F) \subseteq F$. De plus, par extensivité de $\mathrm{cl}(\cdot)$, on sait que $F \subseteq \mathrm{cl}(F)$. Ainsi, par double inclusion, on a $\mathrm{cl}(F) = F$. Prenons deux éléments clos A et B et montrons que $A \cap B$ est clos. Il faut donc montrer que $\mathrm{cl}(A \cap B) = A \cap B$. Par extensivité, on a l'inclusion $A \cap B \subseteq \mathrm{cl}(A \cap B)$. Il reste à montrer que $\mathrm{cl}(A \cap B) \subseteq A \cap B$. Par monotonie, on a $\mathrm{cl}(A \cap B) \subseteq \mathrm{cl}(A)$ et $\mathrm{cl}(A \cap B) \subseteq \mathrm{cl}(B)$. Ainsi, $\mathrm{cl}(A \cap B) \subseteq \mathrm{cl}(A) \cap \mathrm{cl}(B)$. Comme A et B sont clos, $\mathrm{cl}(A) \cap \mathrm{cl}(B) = A \cap B$, d'où le résultat.
2. On sait que f est un opérateur de clôture via le lemme 2.3. Il reste à montrer que l'ensemble des éléments clos de f est $\mathcal{M}$. Soit A un élément clos, c'est-à-dire que $A = f(A)$. Comme F est finie, $\mathcal{M}$ aussi, et donc $f(A)$ est une intersection finie d'éléments de $\mathcal{M}$. Comme $\mathcal{M}$ est stable par intersection de deux éléments, elle est aussi stable par intersection d'un nombre fini d'éléments. Ainsi, $f(A) \in \mathcal{M}$. Dans l'autre sens, soit $A \in \mathcal{M}$. On sait que $A \subseteq f(A)$ par extensivité de f. De plus, $f(A)$ est l'intersection des X vérifiant $A \subseteq X \in \mathcal{M}$. L'ensemble A vérifie cette condition, ce qui signifie que $f(A)$ est l'intersection de A et d'éventuellement d'autres éléments, ce qui montre que $f(A) \subseteq A$. □

Le caractère fini de F est nécessaire pour que l'ensemble des éléments clos soit le même que la famille de Moore considérée.

Exemple 2.16. Soit $F = \mathbb{R}$, et soit $\mathcal{M} = \{\mathbb{R}\} \cup \{\,]0, 1+1/n[\mid n \in \mathbb{N}\}$. La famille $\mathcal{M}$ est bien une famille de Moore car deux ensembles de $\mathcal{M}$ sont toujours comparables, ce qui signifie que leur intersection est le plus petit des deux ensembles. Or, on a $f(\,]0, 1/2[\,) =]0, 1] \notin \mathcal{M}$.

2.3.2 Relations et coupes

Définition

Soit F un ensemble d'objets. On peut définir une *relation* $\mathfrak{R}$ sur les éléments de F. Cela signifie que pour tout $x, y \in F$, $x \mathfrak{R} y$ est soit vrai, soit faux. On définit aussi l'inverse de $\mathfrak{R}$, noté $\mathfrak{R}^{-1}$, par $x \mathfrak{R}^{-1} y$ si et seulement si $y \mathfrak{R} x$. Quand deux relations $\mathfrak{R}_1$ et $\mathfrak{R}_2$ sont les mêmes, on note $\mathfrak{R}_1 \equiv \mathfrak{R}_2$. Cela signifie que pour tout $x, y \in F$, $x \mathfrak{R}_1 y$ si et seulement si $x \mathfrak{R}_2 y$

Exemple 2.17. Soit $F = \mathbb{Z}$ l'ensemble des entiers relatifs. Des relations possibles sur F sont l'égalité ($\mathfrak{R} \equiv =$, c'est-à-dire que $x \mathfrak{R} y$ si et seulement si $x = y$), la supériorité ($\mathfrak{R} \equiv <$, c'est-à-dire que $x \mathfrak{R} y$ si et seulement si $x < y$), la succession ($x \mathfrak{R} y$ si et seulement si $y = x + 1$) ou la divisibilité ($x \mathfrak{R} y$ si et seulement si x est un diviseur de y). Dans le cas de l'égalité, $\mathfrak{R}$ est son propre inverse ($=^{-1} \equiv =$, c'est-à-dire que $\mathfrak{R}^{-1}$ représente aussi l'égalité). Pour les autres exemples, $\mathfrak{R}^{-1}$ est une autre relation, à savoir l'infériorité, la précession et la multiplicité.

Il est possible de définir un opérateur de clôture à partir d'une relation $\mathfrak{R}$ [Rig48]. Pour ce faire, définissons d'abord quelques notions intermédiaires. Étant donné une relation $\mathfrak{R}$ sur un ensemble d'objets F et un objet $x \in F$, on s'intéresse naturellement à l'ensemble des éléments de F qui sont en relation avec x.

Définition 2.17 (Coupe d'une relation)**.** *On appelle* coupe *de* $\mathfrak{R}$ *suivant* x*, et on le note* $\mathfrak{R}(x)$*, l'ensemble* $\mathfrak{R}(x) := \{y \in F \mid x \mathfrak{R} y\}$.

On pourrait aussi avoir envie de décrire l'ensemble $\{x \in F \mid x \mathfrak{R} y\}$. Pour se faire, il suffit de considérer la relation inverse $\mathfrak{R}^{-1}$. Ainsi, $\{x \in F \mid x \mathfrak{R} y\} = \{x \in F \mid y \mathfrak{R}^{-1} x\} = \mathfrak{R}^{-1}(y)$.

Exemple 2.18. Soit $F = \mathbb{Z}$ l'ensemble des entiers relatifs.

1. La coupe de la relation d'infériorité $<$ suivant un entier x est l'ensemble des entiers $\{y \in \mathbb{Z} \mid x < y\}$, c'est-à-dire l'ensemble des entiers plus grands que x.
2. La coupe de la relation de divisibilité suivant un entier x est l'ensemble des entiers qui sont multiples de x, c'est-à-dire $x\mathbb{Z} = \{x \cdot n, n \in \mathbb{Z}\}$.
3. La coupe de la relation d'égalité suivant un entier x est l'ensemble $\{x\}$, car le nombre x est le seul à être égal à lui-même.

Afin de se familiariser avec ces notions, voici un lemme simple qui servira également ultérieurement :

Lemme 2.6. *Soit $\mathfrak{R}$ une relation anti-symétrique (c'est-à-dire que si $x \mathfrak{R} y$ est vrai alors $y \mathfrak{R} x$ est faux) et anti-réflexive (c'est-à-dire que $x \mathfrak{R} x$ est faux pour tout x) sur $F = [n]$. Alors $\mathfrak{R} \equiv <$ si et seulement si $\forall i, |\mathfrak{R}^{-1}(i)| = i - 1$.*

Démonstration. D'un côté, si $\mathfrak{R} \equiv <$, alors pour tout i, $\mathfrak{R}^{-1}(i) = \{j \in F \mid j < i\} = [i-1]$ et $|\mathfrak{R}^{-1}(i)| = i-1$. Dans l'autre sens, on le prouve par récurrence sur n. Si $n = 2$, les deux seules relations anti-symétriques et anti-réflexives sont $<$ et $>$. Comme $|\mathfrak{R}^{-1}(2)| = 1$, on a $1 \mathfrak{R} 2$ et donc $\mathfrak{R} \equiv <$. On suppose que l'implication est vraie pour n et on la prouve pour $n+1$. Puisque $|\mathfrak{R}^{-1}(n+1)| = n$ et $\mathfrak{R}$ est anti-réflexive, pour tout i tel que $1 \leq i \leq n$, on a $i \mathfrak{R}^{-1} n+1$. Puisque $\mathfrak{R}$ est anti-symétrique, pour tout i tel que $1 \leq i \leq n$, on a $n+1 \mathfrak{R}^{-1} i$ qui est faux. Cela signifie que pour tout i tel que $1 \leq i \leq n$, $\mathfrak{R}(i) \subseteq [n]$. On applique l'hypothèse d'induction à $\mathfrak{R}$ restreint à $[n]$, ce qui signifie que cette relation est $<$. Vu que pour tout i tel que $1 \leq i \leq n$ on a $i \mathfrak{R}^{-1} n+1$, la relation $\mathfrak{R}$ sur $[n+1]$ est bien $<$. □

Il est possible d'aller au-delà d'une coupe suivant un objet en considérant une coupe suivant un ensemble d'objets.

Définition 2.18 (Coupe de seconde espèce)**.** *Soit F un ensemble d'objets, $\mathfrak{R}$ une relation sur F, et X un sous-ensemble de F. On appelle* coupe de seconde espèce *de $\mathfrak{R}$ suivant X, et on le note $\mathfrak{R}[X]$, l'ensemble :*

$$\mathfrak{R}[X] := \bigcap_{x \in X} \mathfrak{R}(x).$$

En français, la coupe de seconde espèce de $\mathfrak{R}$ suivant X est l'ensemble des objets qui sont en relation via $\mathfrak{R}$ avec tous les éléments de X. Le fait de prendre l'intersection de toutes les coupes peut sembler arbitraire. De fait, remplacer cette intersection par une union donne lieu à la *coupe de première espèce*, mais nous n'en parlerons pas ici, car les résultats suivants ne sont valables qu'avec une coupe de seconde espèce.

Exemple 2.19. Soit $F = \mathbb{Z}$ l'ensemble des entiers relatifs.

1. La coupe de seconde espèce de la relation d'infériorité $<$ suivant un ensemble d'entiers X est l'ensemble des entiers qui sont tous plus grands que chaque élément de X, c'est-à-dire l'intervalle $[x, +\infty[$ où $x = \max X$. Si $\max X = +\infty$, cet ensemble est vide.

2. La coupe de seconde espèce de la relation de divisibilité suivant un ensemble d'entiers x est l'ensemble des entiers qui sont multiples de chacun des éléments de X, c'est-à-dire $x\mathbb{Z} = \{x \cdot n, n \in \mathbb{Z}\}$, où x est le PGCD des éléments de X si cet ensemble est fini. Sinon, il d'agit de l'ensemble vide.
3. La coupe seconde espèce de la relation d'égalité suivant un ensemble d'entiers X est vide dès que $|X| \geq 2$.

Propriétés

On démontre ici quelques propriétés élémentaires de la coupe de seconde espèce. Ces démonstrations ont été adaptées de [Rig48]. Tout d'abord, on montre que la coupe de seconde espèce est décroissante :

Lemme 2.7. *Si $X \subseteq Y$ alors $\mathfrak{R}[Y] \subseteq \mathfrak{R}[X]$.*

Démonstration. Soit $x \in \mathfrak{R}[Y]$. D'après la définition de la coupe de seconde espèce, $x \in \mathfrak{R}[Y]$ si et seulement si $\forall y \in Y, y \mathfrak{R} x$. Comme $X \subseteq Y$, cela signifie que $\forall y \in X, y \mathfrak{R} x$, ce qui équivaut à $x \in \mathfrak{R}[X]$. □

On définit ensuite la notion de *rectangle* d'une relation $\mathfrak{R}$. Il s'agit d'une paire d'ensembles (X, Y) telle que pour tout objet $x \in X$ et pour tout objet $y \in Y$, on a $x \mathfrak{R} y$. La notion de rectangle peut être caractérisée via une coupe de seconde espèce, comme le montre le lemme suivant.

Lemme 2.8. *(X, Y) est un rectangle de $\mathfrak{R}$ si et seulement si $Y \subseteq \mathfrak{R}[X]$.*

Démonstration. On a la suite d'équivalences suivante :

$$Y \subseteq \mathfrak{R}[X] \iff Y \cap \overline{\mathfrak{R}[X]} = \emptyset \iff Y \cap \overline{\bigcap_{x \in X} \mathfrak{R}(x)} = \emptyset \iff Y \cap \bigcup_{x \in X} \overline{\mathfrak{R}(x)} = \emptyset$$
$$\iff \forall y \in Y, y \notin \bigcup_{x \in X} \overline{\mathfrak{R}(x)} \iff \forall y \in Y, \forall x \in X, y \notin \overline{\mathfrak{R}(x)}$$
$$\iff \forall y \in Y, \forall x \in X, x \mathfrak{R} y. \iff (X, Y) \text{ est un rectangle de } \mathfrak{R}. \quad \square$$

En substituant X par $\mathfrak{R}^{-1}[Y]$ dans cette équivalence, on obtient que $(\mathfrak{R}^{-1}[Y], Y)$ est un rectangle si et seulement si $Y \subseteq \mathfrak{R}[\mathfrak{R}^{-1}[Y]]$. Comme le première partie de l'équivalence est toujours vraie par définition de $\mathfrak{R}^{-1}[Y]$, on aboutit au lemme suivant, qui correspond à la propriété 22 de [Rig48].

Lemme 2.9. *Pour toute relation $\mathfrak{R}$ sur un ensemble d'objets F et tout $Y \subseteq F$, on a l'inclusion $Y \subseteq \mathfrak{R}[\mathfrak{R}^{-1}[Y]]$.*

Comme ce lemme est vrai pour toute relation $\mathfrak{R}$, on a également pour tout $X \subseteq F$ l'inclusion $X \subseteq \mathfrak{R}^{-1}[\mathfrak{R}[X]]$ en l'appliquant à la relation $\mathfrak{R}^{-1}$. Grâce à ce lemme, on démontre finalement une égalité qui fait intervenir une triple composition de l'opération de coupe de seconde espèce :

Lemme 2.10. *Pour toute relation $\mathfrak{R}$ sur un ensemble d'objets F et tout $X \subseteq F$, on a l'égalité $\mathfrak{R}[X] = \mathfrak{R}[\mathfrak{R}^{-1}[\mathfrak{R}[X]]]$.*

Démonstration. On démontre cette égalité par double inclusion :

($\subseteq$) On remplace Y par $\mathfrak{R}[X]$ dans l'inclusion $Y \subseteq \mathfrak{R}[\mathfrak{R}^{-1}[Y]]$.

($\supseteq$) On applique la décroissance de la coupe de seconde espèce (lemme 2.7) à l'inclusion $X \subseteq \mathfrak{R}^{-1}[\mathfrak{R}[X]]$. □

Grâce à ces lemmes, on peut définir un opérateur de clôture à partir d'une relation $\mathfrak{R}$.

Clôture

Nous avons maintenant tous les outils pour définir un opérateur de clôture à partir d'une relation $\mathfrak{R}$.

Lemme 2.11. *Soit F un ensemble et soit $\mathfrak{R}$ une relation sur F. La fonction $f : 2^F \to 2^F$ définie par $f(X) = \mathfrak{R}^{-1}[\mathfrak{R}[X]]$ est un opérateur de clôture.*

Démonstration. La fonction f est extensive (via le lemme 2.9), monotone (en appliquant deux fois le lemma 2.7) et idempotente (en considérant la coupe de seconde espèce des deux membres de l'égalité du lemme 2.10). □

On peut donc obtenir un opérateur de clôture à partir d'une relation. Un intérêt d'exprimer une clôture de cette façon est la complexité algorithmique du calcul de $\mathrm{cl}(X)$. Si le calcul de $\mathfrak{R}[X]$ est rapide, il devient facile de calculer $\mathrm{cl}(X)$ en l'exprimant sous la forme $\mathfrak{R}^{-1}[\mathfrak{R}[X]]$. Cependant, contrairement aux familles de Moore vues dans la section 2.3.1, il n'y a pas de correspondance totale entre les deux. Cela signifie qu'il existe des opérateurs de clôture qui ne peuvent pas s'exprimer sous la forme $\mathfrak{R}^{-1}[\mathfrak{R}[X]]$.

Lemme 2.12. *Soit F un ensemble d'au moins 12 objets. Il existe un opérateur de clôture $cl(\cdot) : 2^F \to 2^F$ tel que pour toute relation $\mathfrak{R}$ sur F, la fonction $f : 2^F \to 2^F$ définie par $f(X) = \mathfrak{R}^{-1}[\mathfrak{R}[X]]$ soit différente de $cl(\cdot)$.*

Démonstration. On considère d'abord le cas où F est finie, et on procède par dénombrement. On note n le cardinal de F.

— On calcule dans un premier temps une borne supérieure sur le nombre de fonctions f. Ce nombre est borné par le nombre de relations $\mathfrak{R}$ différentes sur F. Pour définir une relation $\mathfrak{R}$ sur E, il suffit de décider pour chaque pair d'éléments $(x, y) \in E \times E$ si $x \mathfrak{R} y$ est vrai ou faux. Il y a donc $2^{(n^2)}$ relations différentes et au plus ce nombre de fonctions f différentes.

— On calcule dans un second temps une borne inférieure sur le nombre d'opérateurs de clôture. Pour cela, on dénombre seulement ceux qui ont une forme particulière. Soit k un entier, on construit un opérateur de clôture en choisissant un ensemble de parties $\mathcal{P}$ de F où chaque partie est de taille k et on construit un opérateur de clôture en appliquant le lemme 2.3. Si l'on prend deux tels ensembles de parties $\mathcal{P}_1$ et $\mathcal{P}_2$, on obtient deux opérateurs de clôtures différents. En effet, soit $\mathrm{cl}_1(\cdot)$ et $\mathrm{cl}_2(\cdot)$ les deux opérateurs de clôture associés. Vu que $\mathcal{P}_1 \neq \mathcal{P}_2$, il existe une partie A (de taille k) qui est dans $\mathcal{P}_1$ mais pas dans $\mathcal{P}_2$. Ainsi, $\mathrm{cl}_1(A) = \bigcap_{A \subseteq X \in \mathcal{P}_1} X = A$ mais $\mathrm{cl}_2(A) = \bigcap_{A \subseteq X \in \mathcal{P}_2} X = F$ car il n'y a aucun ensemble X vérifiant $A \subseteq X \in \mathcal{P}_2$ (vu que tous les ensembles de $\mathcal{P}_2$ sont de taille k et que A n'en fait pas partie) et car une intersection vide d'ensembles de F vaut F par convention. Le nombre de choix de $\mathcal{P}$ est égal à deux élevé à une puissance égale au nombre d'ensembles de taille k parmi un ensemble de taille n, c'est-à-dire $n^{\binom{n}{k}}$. En prenant k suffisamment grand (par exemple de l'ordre de $n/2$), on obtient une borne inférieure de l'ordre de $2^{(2^{n-2}n^{-1/2})}$.

Finalement, dès que $n \geq 12$, le nombre de fonctions f est strictement inférieur au nombre d'opérateurs de clôture, ce qui montre que certains opérateurs de clôture ne peuvent pas s'exprimer à partir d'une relation $\mathfrak{R}$.

Dans le cas où F est infinie, on peut utiliser le même genre de raisonnement. On obtient qu'il a une injection entre le nombre de fonction f et l'ensemble 2^F ainsi qu'une

surjection entre 2^{2^F} et le nombre d'opérateurs de clôture. Le fait qu'il n'existe pas de bijection entre E et 2^E pour tout ensemble E permet de conclure. □

2.3.3 Fonctions de connectivité

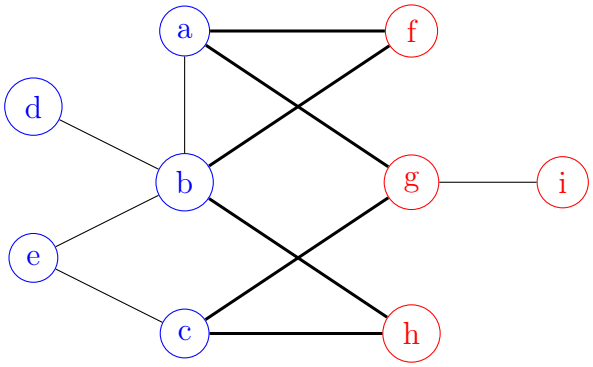
Soit E un ensemble, et soit $f : 2^E \to \mathbb{N}$ une fonction qui associe à chaque sous-ensemble de E une valeur entière. On dit que f est une fonction de *connectivité* quand elle vérifie les conditions suivantes [Cun83] :

- (normalisée) $f(\emptyset) = 0$,
- (symétrique) pour tout ensemble $X \subseteq E$, on a $f(E \setminus X) = f(X)$,
- (sous-modulaire) pour tous les ensembles $X, Y \subseteq E$, on a $f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y)$.

Rang de coupe

Définition 2.19 (Rang de coupe)**.** *Soit G un graphe. Le* rang de coupe *d'un ensemble de sommets X de G, notée ρ, est égal au rang de la matrice d'adjacence de $G[X, \overline{X}]$.*

Il existe plusieurs façons de calculer le rang d'une matrice booléenne [HP23]. Classiquement, le rang d'un matrice M de taille m par n peut être défini comme le plus petit entier k tel qu'il existe une matrice C de taille m par k et une matrice R de taille k par n de sorte que $M = CR$. On considère ici que le rang est calculé sur le corps fini à deux éléments $\mathbb{F}_2$.



	a	b	c	d	e	f	g	h	i
a	0	1	0	0	0	**1**	**1**	**0**	**0**
b	1	0	0	1	1	**1**	**0**	**1**	**0**
c	0	0	0	0	1	**0**	**1**	**1**	**0**
d	0	1	0	0	0	**0**	**0**	**0**	**0**
e	0	1	1	0	0	**0**	**0**	**0**	**0**
f	**1**	**1**	**0**	**0**	**0**	0	0	0	0
g	**1**	**0**	**1**	**0**	**0**	0	0	0	1
h	**0**	**1**	**1**	**0**	**0**	0	0	0	0
i	**0**	**0**	**0**	**0**	**0**	0	1	0	0

FIGURE 2.8 – Un graphe et l'une de ses coupes. Les sommets bleus représentent un côté de la coupe et les sommets rouges l'autre côté. Les arêtes en gras sont les arêtes qui traversent la coupe. La matrice d'adjacence du graphe est représentée en utilisant le même code couleur.

Exemple 2.20. Dans le graphe de la figure 2.8, la coupe $(\{a, b, c, d, e\}, \{f, g, h, i\})$ a un rang de 2 (cf. figure 2.9). Comme le rang est calculé dans le corps $\mathbb{F}_2$, les additions et les multiplications sont effectuées modulo 2. Cela signifie par exemple, que l'addition des trois premières lignes de la matrice ([1 1 0 0], [1 0 1 0] et [0 1 1 0]) fait [0 0 0 0], vu que $1 + 1 = 0 \mod 2$.

On a la propriété suivante.

Proposition 2.13 ([OS06])**.** *Le rang de coupe est une fonction de connectivité.*

$$A(G)[\{a,b,c,d,e\},\{f,g,h,i\})] = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccc} f & g & h & i \end{array} \\ \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \end{array} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

FIGURE 2.9 – La coupe $(\{a,b,c,d,e\},\{f,g,h,i\})$ a un rang de 2.

Il est clair que le rang de coupe est une fonction normalisée et symétrique. Pour la propriété de sous-modularité, diverses preuves existent, notamment des preuves utilisant des manipulations matricielles [Tru92, Mur99]. Je propose une preuve purement algébrique.

Définition 2.20 (Notation I_X)**.** *Soit un ensemble $X \subseteq [n]$, on note I_X la matrice $diag(\mathbb{1}_X)$, c'est-à-dire la matrice diagonale telle que le i-ème élément diagonal soit 1 si $i \in X$ et 0 sinon.*

Observation 2.1. *Par définition, $I_{X\cap Y} = I_X I_Y$ et $I_{\overline{X}} = I - I_X$. Puisque $X \cup Y = \overline{\overline{X} \cap \overline{Y}}$ par les lois de De Morgan, on a $I_{X\cup Y} = I - (I - I_X)(I - I_X)$.*

Avec ces notations, le rang de coupe s'écrit $\rho(X) = \text{rg } I_X A(I - I_X)$, où A est la matrice d'adjacence du graphe G. Le rang est calculé dans le corps à deux éléments $\mathbb{F}_2$.

On rappelle l'inégalité de Frobenius :

Lemme 2.14 (Inégalité de Frobenius)**.** *Soit E, F, G, H quatre espaces vectoriels de dimension finie et soit $g_1 : G \to H, g_2 : F \to G, g_3 : E \to F$ trois morphismes. Alors on a :*

$$\text{rg } g_1 \circ g_2 + \text{rg } g_2 \circ g_3 \leq \text{rg } g_1 \circ g_2 \circ g_3 + \text{rg } g_2.$$

On se place dans le contexte suivant : E, F sont deux espaces vectoriels de dimension finie, $f : E \to F$ est un morphisme, $p_1, p_2, p_3 : E \to E$ sont trois projections telles que $p_i \circ p_j = 0$ pour tout $i \neq j$, et enfin $q_1, q_2, q_3 : F \to F$ sont trois projections telles que $q_i \circ q_j = 0$ pour tout $i \neq j$.

Corollaire 2.15. *On a :*

$$\begin{aligned} &\text{rg } q_2 \circ f \circ (p_1 + p_2) + \text{rg } q_2 \circ f \circ (p_2 + p_3) - \text{rg } q_2 \circ f \circ p_2 \\ &+\text{rg } (q_1 + q_2) \circ f \circ p_2 + \text{rg } (q_2 + q_3) \circ f \circ p_2 - \text{rg } q_2 \circ f \circ p_2 \\ &\leq \text{rg } (q_1 + q_2) \circ f \circ (p_1 + p_2) + \text{rg } (q_2 + q_3) \circ f \circ (p_2 + p_3). \end{aligned}$$

Démonstration. On applique l'inégalité de Frobenius à $g_1 = q_2, g_2 = (q_1 + q_2) \circ f \circ (p_1 + p_2), g_3 = p_2$:

$$\begin{aligned} &\text{rg } q_2 \circ (q_1 + q_2) \circ f \circ (p_1 + p_2) + \text{rg } (q_1 + q_2) \circ f \circ (p_1 + p_2) \circ p_2 \\ &\leq \text{rg } q_2 \circ (q_1 + q_2) \circ f \circ (p_1 + p_2) \circ p_2 + \text{rg } (q_1 + q_2) \circ f \circ (p_1 + p_2). \end{aligned}$$

Puisque q_2 est une projection, $q_2 \circ q_2 = q_2$. Ainsi, puisque $q_2 \circ q_1 = 0$, on a $q_2 \circ (q_1 + q_2) = q_2$. De même, on a $(p_1 + p_2) \circ p_2 = p_2$. Finalement, cette inégalité peut être réécrite en :

$$\text{rg } q_2 \circ f \circ (p_1 + p_2) + \text{rg } (q_1 + q_2) \circ f \circ p_2 \leq \text{rg } q_2 \circ f \circ p_2 + \text{rg } (q_1 + q_2) \circ f \circ (p_1 + p_2).$$

En faisant la même chose avec $g_1 = q_2, g_2 = (q_2 + q_3) \circ f \circ (p_2 + p_3), g_3 = p_2$, on obtient :

$$\text{rg } q_2 \circ f \circ (p_2 + p_3) + \text{rg } (q_2 + q_3) \circ f \circ p_2 \leq \text{rg } q_2 \circ f \circ p_2 + \text{rg } (q_2 + q_3) \circ f \circ (p_2 + p_3).$$

On obtient le résultat en additionnant les deux inégalités. □

Lemme 2.16 (Sous-modularité du rang)**.** *On a les deux inégalités suivantes :*

$$\text{rg } g \circ (p_1 + p_2 + p_3) + \text{rg } g \circ p_2 \leq \text{rg } g \circ (p_1 + p_2) + \text{rg } g \circ (p_2 + p_3),$$
$$\text{rg } (q_1 + q_2 + q_3) \circ g + \text{rg } q_2 \circ g \leq \text{rg } (q_1 + q_2) \circ g + \text{rg } (q_2 + q_3) \circ g.$$

Démonstration. On rappelle qu'étant donné deux ensembles de vecteurs A, B, on a rg $(A \cup B)$+rg $(A \cap B) \leq$ rg A+rg B. Cela peut se démontrer en utilisant la théorie des matroïdes : on applique la sous-modularité du rang [Oxl06] au matroïde dont l'ensemble de base est l'ensemble des vecteurs et dont l'ensemble des indépendants est l'ensemble des sous-ensembles linéairement indépendants. Puisque $p_i \circ p_j = 0$, tous les p_i's commutent, et ils sont donc simultanément diagonalisables. Soit P une telle base de E. Soit A et B l'ensemble des colonnes de $g \circ (p_1 + p_2)$ et $g \circ (p_2 + p_3)$ dans la base P, vu comme des vecteurs. Ainsi, $A \cup B$ et $A \cap B$ sont respectivement les ensembles des colonnes de $g \circ (p_1 + p_2 + p_3)$ et $g \circ p_2$ dans la base P, vu comme des vecteurs. Cela prouve la première inégalité. Pour la seconde inégalité, on effectue le même raisonnement sur les lignes au lieu des colonnes. □

Corollaire 2.17. *On a :*

$$\begin{aligned} &\text{rg } q_2 \circ f \circ (p_1 + p_2 + p_3) + \text{rg } (q_1 + q_2 + q_3) \circ f \circ p_2 \\ \leq\ &\text{rg } q_2 \circ f \circ (p_1 + p_2) + \text{rg } q_2 \circ f \circ (p_2 + p_3) - \text{rg } q_2 \circ f \circ p_2 \\ &+\text{rg } (q_1 + q_2) \circ f \circ p_2 + \text{rg } (q_2 + q_3) \circ f \circ p_2 - \text{rg } q_2 \circ f \circ p_2. \end{aligned}$$

Démonstration. On applique la première inégalité du lemme 2.16 avec $g = q_2 \circ f$ ainsi que la seconde inégalité avec $g = f \circ p_2$. On obtient le résultat en additionnant les deux inégalités. □

Proposition 2.18. *On a :*

$$\begin{aligned} &\text{rg } q_2 \circ f \circ (p_1 + p_2 + p_3) + \text{rg } (q_1 + q_2 + q_3) \circ f \circ p_2 \\ \leq\ &\text{rg } (q_1 + q_2) \circ f \circ (p_1 + p_2) + \text{rg } (q_2 + q_3) \circ f \circ (p_2 + p_3). \end{aligned}$$

Démonstration. On obtient le résultat en mettant bout à bout le corollaire 2.17 et le corollaire 2.15. □

On peut maintenant montrer que le rang de coupe est une fonction sous-modulaire.

Corollaire 2.19. *Soit A une matrice carrée $n \times n$ sur le corps $\mathbb{K}$. La fonction $\rho(X) =$ rg $I_X A(I - I_X)$ est sous-modulaire.*

Démonstration. Soit $X, Y \subseteq [n]$. Soit $E = \mathbb{R}^n$. Dans la base canonique, soit $f : E \to E$ le morphisme associé à A, soit $q_1 = p_3 : E \to E$ les morphismes associés à $I_X(I - I_Y)$, soit $q_2 : E \to E$ le morphisme associé à $I_X I_Y$, soit $q_3 = p_1 : E \to E$ les morphismes associés $(I - I_X)I_Y$, soit $p_2 : E \to E$ le morphisme associé à $(I - I_X)(I - I_Y)$. Tous les morphismes p_i et q_i sont des projections car ils sont diagonaux dans la base canonique, avec pour valeurs diagonales uniquement des 0 et des 1. On vérifie facilement que $p_i \circ p_j = 0$ et $q_i \circ q_j = 0$

en utilisant $I_X^2 = I_X$, $I_Y^2 = I_Y$ et $I_X I_Y = I_Y I_X$. Ainsi, on peut appliquer la proposition 2.18 puis remplacer tous les morphismes par leur matrice dans la base canonique. Dans cette base, $p_1 + p_2$ a pour matrice $I - I_X$, $p_2 + p_3$ a pour matrice $I - I_Y$, $p_1 + p_2 + p_3$ a pour matrice $I - I_X I_Y$, $q_1 + q_2$ a pour matrice I_X, $q_2 + q_3$ a pour matrice I_Y et $q_1 + q_2 + q_3$ a pour matrice $I - (I - I_X)(I - I_Y)$. Ainsi, on obtient :

$$\begin{aligned}\operatorname{rg} I_X I_Y A(I - I_X I_Y) + \operatorname{rg}\,[I - (I - I_X)(I - I_Y)]A[(I - I_X)(I - I_Y)] \\ \leq \operatorname{rg} I_X A(I - I_X) + \operatorname{rg} I_Y A(I - I_Y),\end{aligned}$$

c'est-à-dire $\rho(X \cap Y) + \rho(X \cup Y) \leq \rho(X) + \rho(Y)$. □

2.4 Tenseurs

2.4.1 Étiquetage des dimensions

Dans cette partie, je manipulerai des tableaux à d dimensions, aussi appelés *tenseurs*.

Un tenseur $\mathfrak{T}$ d'ordre d est un tableau à d dimensions qui généralise la notion de nombre (tenseur d'ordre 0), de vecteur (tenseur d'ordre 1) et de matrice (tenseur d'ordre 2). Je manipulerai surtout des tenseurs *étiquetés*, ce qui signifie que chaque dimension possède une étiquette.

Exemple 2.21. Soit $\mathfrak{T}$ un tenseur d'ordre 3, étiqueté par x, y et z, où chaque dimension est de taille 6, tel que $\mathfrak{T}[x : i, y : j, z : k] = (i - j) \times k$. Ainsi, $\mathfrak{T}[x : 0, y : 2, z : 5] = -10$ (on peut également noter cet élément $\mathfrak{T}[0, 2, 5]$ quand il n'y a pas d'ambigüité sur les étiquettes).

Le but est de définir les opérations dont nous aurons besoin par la suite. Pour ce faire, on commence par des rappels sur les vecteurs, qui sont des tenseurs d'ordre 1.

Définition 2.21 (Produit scalaire $u \cdot v$ et produit terme à terme $u * v$)**.** *Soit u et v deux vecteurs de même taille. Le* produit scalaire *de u et v est le nombre $u \cdot v = \sum_i u_i v_i$. Le* produit terme à terme *de u et v est le vecteur $u * v$ tel que $(u * v)_i = u_i v_i$.*

En terme de tenseur, le produit scalaire prend deux tenseurs d'ordre 1 et renvoie un tenseur d'ordre 0, et le produit terme à terme prend deux tenseurs d'ordre 1 et renvoie un tenseur d'ordre 1.

Il existe un produit entre tenseur d'ordre quelconques, appelé *produit tensoriel* [KB09, McC14].

Définition 2.22 (Produit tensoriel $\mathfrak{T}_1 \otimes \mathfrak{T}_2$)**.** *Soit $\mathfrak{T}_1$ un tenseur d'ordre d_1 et $\mathfrak{T}_2$ un tenseur d'ordre d_2. Le* produit tensoriel *de $\mathfrak{T}_1$ et $\mathfrak{T}_2$ est le tenseur à $d_1 + d_2$ dimensions $\mathfrak{T}_1 \otimes \mathfrak{T}_2$ tel que $(\mathfrak{T}_1 \otimes \mathfrak{T}_2)[i_1, \ldots i_{d_1}, j_1, \ldots, j_{d_2}] = \mathfrak{T}_1[i_1, \ldots i_{d_1}] \times \mathfrak{T}_2[j_1, \ldots, j_{d_2}]$.*

Un des problèmes des tenseurs est que les notations deviennent rapidement compliquées. Pour pallier ce problème, j'utilise les étiquettes des tenseurs.

Définition 2.23 (Dictionnaire et étiquette)**.** *Un* dictionnaire $\mathcal{D}$ *est un objet qui associe à chaque étiquette une valeur. On note* $\{$*étiquette-1* : *valeur-1*, *étiquette-2* : *valeur-2*, $\ldots\}$*. L'ordre des paires* *étiquette* : *valeur* *d'un dictionnaire n'a pas d'importance. On peut utiliser un dictionnaire pour noter les indices d'un tenseur, de sorte que* $\mathfrak{T}[\mathcal{D}] = \mathfrak{T}[$*étiquette-1* : *valeur-1*, *étiquette-2* : *valeur-2*, $\ldots]$.

Définition 2.24 (Notations $\alpha : i$ et $\Sigma : I$)**.** *Étant donné une étiquette α et un entier i, on note $\alpha : i$ le dictionnaire $\{\alpha : i\}$. Étant donné un ensemble d'étiquettes $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ et une liste d'entiers $I = \{i_1, \ldots i_n\}$, on note $\Sigma : I$ le dictionnaire $\{\alpha_1 : i_1, \ldots, \alpha_n : i_n\}$.*

Exemple 2.22. Soit $\mathfrak{T}$ un tenseur à 3 dimensions, étiqueté par x, y et z, où chaque dimension est de taille 6, tel que $\mathfrak{T}[x : i, y : j, z : k] = (i - j) \times k$. Soit $\mathcal{D}$ le dictionnaire $\{x : 0, y : 2, z : 5\}$. Alors $\mathfrak{T}[\mathcal{D}] = \mathfrak{T}[x : 0, y : 2, z : 5] = \mathfrak{T}[0, 2, 5] = -10$.

Définition 2.25 (Notation $\mathcal{D}_1, \mathcal{D}_2$)**.** *Soit $\mathcal{D}_1$ et $\mathcal{D}_2$ deux dictionnaires avec des étiquettes différentes. Alors $\mathcal{D}_1, \mathcal{D}_2$ est le dictionnaire obtenu en combinant les paires* `étiquette` : `valeur` *de chaque dictionnaire.*

Exemple 2.23. Si $\mathcal{D}_1 = \{x : 0, z : 5\}$ et $\mathcal{D}_2 = \{y : 2\}$, alors $\mathcal{D}_1, \mathcal{D}_2 = \{x : 0, z : 5, y : 2\} = \{x : 0, y : 2, z : 5\}$.

Maintenant, si on a deux tenseurs avec des étiquettes toutes différentes, on peut réécrire simplement le produit tensoriel :

Remarque 2.2. Soit $\mathfrak{T}_1$ un tenseur d'ordre d_1 et $\mathfrak{T}_2$ un tenseur d'ordre d_2. Soit $\mathcal{D}_1$ un dictionnaire qui correspond aux étiquettes de $\mathfrak{T}_1$ et $\mathcal{D}_2$ un dictionnaire qui correspond aux étiquettes de $\mathfrak{T}_1$. On a alors $(\mathfrak{T}_1 \otimes \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2] = \mathfrak{T}_1[\mathcal{D}_1] \times \mathfrak{T}_2[\mathcal{D}_2]$.

2.4.2 Opérations

Un problème se pose quand les tenseurs ont une étiquette en commun. Il faut alors soit combiner cette étiquette en une nouvelle étiquette (c'est-à-dire combiner deux tenseurs d'ordre 1 en un nouveau tenseur d'ordre 1), ou alors supprimer cette étiquette (c'est-à-dire combiner deux tenseurs d'ordre 1 en un nouveau tenseur d'ordre 0). Pour ce faire, une façon naturelle est d'utiliser les produits de la définition 2.21. Cela permet d'introduire les produits *contractés* [McC14] et *semi-contractés* de tenseurs :

Définition 2.26 (Produit contracté $\mathfrak{T}_1 \odot_\alpha \mathfrak{T}_2$ et produit semi-contracté $\mathfrak{T}_1 \circledast_\alpha \mathfrak{T}_2$)**.** *Soit $\mathfrak{T}_1$ un tenseur d'ordre d_1 et $\mathfrak{T}_2$ un tenseur d'ordre d_2 ayant une étiquette α en commun tels que la taille de la dimension correspondant à cette étiquette soit la même chez $\mathfrak{T}_1$ et chez $\mathfrak{T}_2$. Soit $\mathcal{D}_1$ un dictionnaire qui correspond aux étiquettes de $\mathfrak{T}_1$ moins l'étiquette α et $\mathcal{D}_2$ un dictionnaire qui correspond aux étiquettes de $\mathfrak{T}_2$ moins l'étiquette α. On définit alors le* produit contracté *sur α, qui est un tenseur d'ordre $d_1 + d_2 - 2$ et qui est noté $\mathfrak{T}_1 \odot_\alpha \mathfrak{T}_2$, et le* produit semi-contracté *sur α, qui est un tenseur d'ordre $d_1 + d_2 - 1$ et qui est noté $\mathfrak{T}_1 \circledast_\alpha \mathfrak{T}_2$, tels que :*

$$(\mathfrak{T}_1 \odot_\alpha \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2] = \sum_{\alpha:i} \mathfrak{T}_1[\mathcal{D}_1, \alpha : i] \times \mathfrak{T}_2[\mathcal{D}_2, \alpha : i]$$
$$(\mathfrak{T}_1 \circledast_\alpha \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2, \alpha : i] = \mathfrak{T}_1[\mathcal{D}_1, \alpha : i] \times \mathfrak{T}_2[\mathcal{D}_2, \alpha : i].$$

On note que la somme s'effectue sur toutes les valeurs de i entre 0 et la taille de la dimension étiquetée α moins un.

Le produit contracté sur une étiquette de $\mathfrak{T}_1$ et $\mathfrak{T}_2$ est un tenseur qui a deux dimensions de moins que $\mathfrak{T}_1 \otimes \mathfrak{T}_2$ (si on imagine que l'on change les étiquettes pour qu'elles soient toutes différentes), là où le produit semi-contracté a une dimension de moins. De plus, ces produits généralisent les produits vus dans la définition 2.21, ainsi que le produit matriciel.

Remarque 2.3. Soit $\mathcal{T}_1$ et $\mathcal{T}_2$ deux tenseurs d'ordre 1, ayant la même étiquette α et de même taille. Alors $\mathcal{T}_1 \odot_\alpha \mathcal{T}_2 = \mathcal{T}_1 \cdot \mathcal{T}_2$ et $\mathcal{T}_1 \circledast_\alpha \mathcal{T}_2 = \mathcal{T}_1 * \mathcal{T}_2$.

Remarque 2.4. Soit $\mathcal{T}_1$ et $\mathcal{T}_2$ deux tenseurs d'ordre 2, avec $\mathcal{T}_1$ étiqueté x et y et $\mathcal{T}_2$ étiqueté y et z, tels que la taille de la dimension correspondant à l'étiquette y soit la même chez $\mathcal{T}_1$ et chez $\mathcal{T}_2$. Alors $\mathcal{T}_1 \odot_y \mathcal{T}_2 = \mathcal{T}_1\,\mathcal{T}_2$, c'est-à-dire le produit matriciel entre $\mathcal{T}_1$ et $\mathcal{T}_2$.

Exemple 2.24. Soit $\mathcal{T}_1$ et $\mathcal{T}_2$ deux tenseurs d'ordre 3 et 2, tels que chaque dimension soit de taille n, et définis par :

$$\begin{aligned} &\mathcal{T}_1[x:i, y:j, t:k] = i+j+k && \text{pour } 0 \le i \le n-1, 0 \le j \le n-1, 0 \le k \le n-1 \\ &\mathcal{T}_2[t:k, z:l] = k-l && \text{pour } 0 \le k \le n-1, 0 \le l \le n-1. \end{aligned}$$

Alors les tenseurs $\mathcal{T}_1 \odot_t \mathcal{T}_2$ et $\mathcal{T}_1 \circledast_t \mathcal{T}_2$ vérifient :

$$\begin{aligned} (\mathcal{T}_1 \odot_t \mathcal{T}_2)[x:i, y:j, z:l] &= \sum_{k=0}^{n-1} \mathcal{T}_1[x:i, y:i, t:k] \times \mathcal{T}_2[t:k, z:l] \\ &= \sum_{k=0}^{n-1} (i+j+k) \times (k-l) \\ &= \frac{n(n-1)(2n-1)}{6} + (i+j-l)\frac{(n-1)(n-2)}{2} \\ &\qquad - (i+j)(n-1)l \end{aligned}$$

$$\begin{aligned} (\mathcal{T}_1 \circledast_t \mathcal{T}_2)[x:i, y:j, z:l, t:k] &= \mathcal{T}_1[x:i, y:i, t:k] \times \mathcal{T}_2[t:k, z:l] \\ &= (i+j+k)(k-l). \end{aligned}$$

Il est possible de contracter sur plusieurs étiquettes à la fois. Dans ce cas, on peut vouloir contracter certaines étiquettes comme avec le produit contracté et d'autres étiquettes comme avec le produit semi-contracté. Cela amène à définir le produit contracté mixte. Avant de le faire, il nous faut définir quelques notions et quelques notations qui seront nécessaires dès lors que l'on manipule deux tenseurs dont on veut faire le produit.

Définition 2.27 (Tenseurs compatibles pour le produit)**.** *Soit $\mathcal{T}_1$ et $\mathcal{T}_2$ deux tenseurs ayant une ou plusieurs étiquettes en commun. On dit qu'ils sont* compatibles pour le produit *lorsque la taille de la dimension correspondant à une étiquette présente chez les deux tenseurs est la même chez $\mathcal{T}_1$ et chez $\mathcal{T}_2$.*

Exemple 2.25. Soit $\mathcal{T}_1$ un tenseur d'ordre 3 ayant pour étiquettes x, y et z, et $\mathcal{T}_2$ un tenseur d'ordre 3 ayant pour étiquettes x, y et t. Afin que $\mathcal{T}_1$ et $\mathcal{T}_2$ soient compatibles pour le produit, il faut que la taille de la dimension correspondant à l'étiquette x soit la même chez $\mathcal{T}_1$ et chez $\mathcal{T}_2$, et de même pour l'étiquette y.

Quand on a deux tenseurs $\mathcal{T}_1$ et $\mathcal{T}_2$ compatibles pour le produit, on est amené à considérer certains dictionnaires. On note $\mathcal{D}_1$ un dictionnaire qui correspond aux étiquettes de $\mathcal{T}_1$ sans les étiquettes en commun à $\mathcal{T}_1$ et $\mathcal{T}_2$. De même, on note $\mathcal{D}_2$ un dictionnaire qui correspond aux étiquettes de $\mathcal{T}_1$ sans les étiquettes en commun à $\mathcal{T}_1$ et $\mathcal{T}_2$. Si Π est un ensemble d'étiquettes en commun à $\mathcal{T}_1$ et $\mathcal{T}_2$, on note $\mathcal{D}_\pi$ un dictionnaire qui correspond aux étiquettes de Π. On définit maintenant le produit contracté mixte de la façon suivante :

Définition 2.28 (Produit mixte $\mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2$)**.** *Soit $\mathfrak{T}_1$ un tenseur d'ordre d_1 et $\mathfrak{T}_2$ un tenseur d'ordre d_2, compatibles pour le produit, et soit (Π, Σ) une partition des étiquettes en commun. On définit le produit contracté mixte sur Π et Σ, qui est un tenseur d'ordre $d_1 + d_2 - 2|\Sigma| - |\Pi|$ et qui est noté $\mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2$, par :*

$$(\mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] = \sum_{\Sigma:I} \mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, \Sigma : I] \times \mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, \Sigma : I].$$

On note que la somme s'effectue sur tous les dictionnaires $\Sigma : I$.

Exemple 2.26. Soit $\mathfrak{T}_1$ et $\mathfrak{T}_2$ deux tenseurs d'ordre 4, tels que chaque dimension soit de taille n, et définis par :

$$\mathfrak{T}_1[x : i, y : i, t : k, u : v] = i + j + k + v$$
$$\mathfrak{T}_2[y : i, z : l, t : k, u : v] = jk - lv.$$

Alors le tenseur $\mathfrak{T}_1 \odot_{t,u} \circledast_y \mathfrak{T}_2$ vérifie :

$$\begin{aligned}
&(\mathfrak{T}_1 \odot_{t,u} \circledast_y \mathfrak{T}_2)[x : i, y : j, z : l] \\
&\quad = \sum_{k=0}^{n-1} \sum_{v=0}^{n-1} \mathfrak{T}_1[x : i, y : i, t : k, u : v] \times \mathfrak{T}_2[y : i, z : l, t : k, u : v] \\
&\quad = \sum_{k=0}^{n-1} \sum_{v=0}^{n-1} (i + j + k + v) \times (jk - lv) \\
&\quad = \ldots = f(i, j, l, n).
\end{aligned}$$

Dans notre cas, nous manipulerons des tenseurs binaires, donc uniquement constitués de 0 et de 1. Un problème se pose alors dans le cas de l'addition, car 1 et 1 font 2. La solution est alors de poser 1+1=1, ce qui revient à remplacer dans toutes les définitions précédentes la notion de somme par celle de maximum. Le produit scalaire de u et v devient alors $u \cdot v = \max_i u_i v_i$ et il en va de même pour toutes les autres sommations. Dans ce contexte, le maximum est équivalent au « ou » logique et le produit au « et » logique. Autrement dit, on effectue nos calculs dans l'*algèbre de Boole* $\mathbb{B}$.

2.4.3 Complexité

Le but de cette section est de montrer la proposition suivante :

Proposition 2.20. *Soit $\mathfrak{T}_1$ un tenseur d'ordre d_1 et $\mathfrak{T}_2$ un tenseur d'ordre d_2 ayant plusieurs étiquettes en commun, tels que la taille de chaque dimension de $\mathfrak{T}_1$ et de $\mathfrak{T}_2$ soit n. Soit (Π, Σ) une partition des étiquettes en commun. Soit $\mathfrak{T} = \mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2$, qui est donc un tenseur d'ordre $d = d_1 + d_2 - 2|\Sigma| - |\Pi|$. Soit $d_{\max} = \max(d, d_1, d_2)$. Alors il est possible de calculer $\mathfrak{T}$ à partir de $\mathfrak{T}_1$ et $\mathfrak{T}_2$ en temps $\mathcal{O}(\sqrt{n^{\omega d_{\max}}})$.*

Ce résultat est à comparer avec l'algorithme naïf qui consisterait à effectuer le calcul pour chaque entrée de $\mathfrak{T}$. Dans ce cas-là, pour chacune des d entrées du tableau, il faudrait calculer une somme impliquant $n^{|\Sigma|}$ termes, ce qui ferait une complexité en $\mathcal{O}(n^x)$ avec $x = d + |\Sigma| = d_1 + d_2 - |\Sigma| - |\Pi|$. Selon les valeurs de d_1, d_2, $|\Sigma|$ et $|\Pi|$, l'une ou l'autre des complexités est meilleure. Le point important est qu'il existe des cas pour lesquels la proposition 2.20 donne de meilleurs résultats que l'algorithme naïf. Notamment, dans le

cas où $d_1 = 2$, $d_2 = 2$, $|\Sigma| = 1$ et $|\Pi| = 0$, on se retrouve dans le cas d'une multiplication classique de matrices. On a alors les résultats habituels, à savoir que l'algorithme naïf a une complexité en $\mathcal{O}(n^3)$ et l'algorithmique issu de la proposition 2.20 a une complexité en $\mathcal{O}(n^\omega)$.

Préambule : multiplication de matrices rectangulaires

On rappelle que le coût optimal pour multiplier deux matrices n par n est $\mathcal{O}(n^\omega)$ De même, pour chaque entiers a, b, c, il a été définit une constante qui sert à représenter le coût optimal de la multiplication entre une matrice n^a par n^b et une matrice n^b par n^c. Ce coût est $\mathcal{O}(n^{\omega(a,b,c)})$. En découpant ces matrices à priori rectangulaire en blocs et en faisant de la multiplication de matrice bloc par bloc, on peut montrer que $\omega(a, b, c) \leq \omega \times \max(a + b, b + c, a + c)/2$.

Tout d'abord, [HP98] montre les trois faits suivants :

Lemme 2.21. *1.* $\omega(a, 0, c) = a + c$,

2. $\omega(a, b, c) \leq \omega(a - m, b - m, c - m) + m\omega(1, 1, 1)$, *où* $m = \min(a, b, c)$,

3. $\omega(a, b, c) = \omega(a, c, b) = \omega(b, a, c) = \omega(b, c, a) = \omega(c, a, b) = \omega(c, b, a)$.

On rappelle que $\omega(1, 1, 1)$ désigne le nombre ω correspondant à la multiplication de matrices carrées.

Pour le premier point, on fait la multiplication entre deux vecteurs. On obtient une matrice dont chaque entrée contient le produit d'un élément du premier vecteur par un élément du second vecteur. Le résultat étant une matrice de taille n^a par n^c, écrire l'entrée demande un temps n^{a+c}, d'où $\omega(a, 0, c) \geq a + c$. De plus, on peut atteindre cette borne trivialement en calculant les n^{a+c} entrées une par une.

Pour le second point, l'idée est de faire du calcul par bloc. Si $m = \min(a, b, c)$, on peut découper chaque matrice en blocs de taille n^m par n^m. Ainsi, grâce à la multiplication de matrices par blocs, on peut voir le produit entre une matrice n^a par n^b et une matrice n^b par n^c comme le produit entre une matrice n^a/n^m par n^b/n^m et une matrice n^b/n^m par n^c/n^m. Dans ce cas, le produit de deux éléments de ces matrices est un produit de deux matrices carrées de taille n^m par n^m. Le coût du produit de deux matrices carrées de taille n^m par n^m est $\mathcal{O}((n^m)^{\omega(1,1,1)}) = \mathcal{O}(n^{m\omega(1,1,1)})$ et le nombre de telles produits est $\mathcal{O}(n^{\omega(a-m,b-m,c-m)})$. D'où une complexité de $\mathcal{O}(n^{\omega(a-m,b-m,c-m)} n^{m\omega(1,1,1)}) = \mathcal{O}(n^{\omega(a-m,b-m,c-m)+m\omega(1,1,1)})$.

Pour le troisième point, cela vient de la représentation du nombre minimum d'opérations nécessaires pour effectuer un produit matriciel sous la forme du rang d'un tenseur à 3 dimensions (ou forme trilinéaire) [Pro73, LR83, GU18]. L'idée est que $\omega(a, b, c)$ est égal au rang d'une certaine forme trilinéaire. Cette forme représente la multiplication entre une matrice M_1 de taille n^a par n^b et une matrice M_2 de taille n^b par n^c, le tout donnant une matrice M_3 de taille n^a par n^c. Or, cette forme est symétrique en les variables formelles correspondant aux termes de M_1, M_2 et M_3 et permet ainsi de déduire un algorithe pour les autres permutations des nombres a, b et c qui soit de la même complexité.

Grâce à ces trois points, on montre le lemme suivant :

Lemme 2.22. *On a l'inégalité* $\omega(a, b, c) \leq \omega \times \max(a + b, b + c, a + c)/2$.

Démonstration. Soit $m = \min(a, b, c)$ et $M = \max(a + b, b + c, a + c)$. On note que $a + b + c = m + M$, puisque pour maximiser la somme de deux valeurs parmi trois, il

faut écarter la plus petite. On note également que $m \leq M/2$, puisque M est la somme de deux nombres chacun plus grand que m.

D'après le second point du lemme 2.21, on sait que $\omega(a,b,c) \leq \omega(a-m, b-m, c-m) + m\omega$. Parmi les quantités $a-m$, $b-m$ et $c-m$, l'une vaut zéro puisque m est le minimum de a, b et c. On utilise le troisième point du lemme 2.21 pour mettre la quantité qui vaut 0 au milieu et on utilise ensuite le premier point du lemme 2.21 afin d'obtenir que $\omega(a-m, b-m, c-m) = (a-m) + (b-m) + (c-m) = M - 2m$. On a donc $\omega(a,b,c) \leq \omega(a-m, b-m, c-m) + m\omega = M - 2m + m\omega \leq M - M + M\omega/2$, d'où le résultat. □

Preuve de la proposition

Comme suggéré par la présence de ω dans la proposition 2.20 et par le paragraphe précédent, le but est d'utiliser une multiplication de matrices. Pour ce faire, il faut donc convertir les différents tenseurs en matrices. Il s'agit d'une opération habituelle sur les tenseurs [Kol06, KB09] appelée *matricisation* [TBR18]. Cependant, pour faciliter cette transformation, on va considérer des matrices dont les lignes et les colonnes sont indexées par des dictionnaires plutôt que par des entiers.

On commence par introduire quelques notations : Si $\mathcal{T}$ est un tenseur et α une étiquette de $\mathcal{T}$, on note n_α la taille de la dimension associée à α. Si (Σ_1, Σ_2) est une partition des étiquettes de $\mathcal{T}$, $\mathcal{D}_1$ représente un dictionnaire correspondant aux étiquettes Σ_1 et $\mathcal{D}_2$ représente un dictionnaire correspondant aux étiquettes Σ_2.

Définition 2.29 (Matricisation)**.** *Soit $\mathcal{T}$ un tenseur. Soit (Σ_1, Σ_2) une partition des étiquettes de $\mathcal{T}$. On définit la* matricisation de $\mathcal{T}$ par rapport à (Σ_1, Σ_2) *comme étant la matrice M ayant un nombre de lignes égal à $\sum_{\alpha \in \Sigma_1} n_\alpha$ et un nombre de colonnes égal à $\sum_{\alpha \in \Sigma_2} n_\alpha$ et dont l'élément indexé par la ligne $\mathcal{D}_1$ et par la colonne $\mathcal{D}_2$ est $\mathcal{T}[\mathcal{D}_1, \mathcal{D}_2]$. On note cet élément $M[(\mathcal{D}_1), (\mathcal{D}_2)]$.*

Les parenthèses autour de chaque dictionnaire de $M[(\mathcal{D}_1), (\mathcal{D}_2)]$ permettent de bien distinguer que M ne prend que 2 arguments et non d comme $\mathcal{T}$.

Nous avons vu au cours de la remarque 2.4 que le produit contracté de deux tenseurs d'ordre 2 correspondait au produit matricielle. Cela se généralise aux tenseurs de n'importe quels ordres en considérant la matricisation des deux tenseurs. Ainsi, si $\mathcal{T}_1$ et $\mathcal{T}_2$ sont deux tenseurs compatibles pour le produit et $\mathcal{T} = \mathcal{T}_1 \odot_\Sigma \mathcal{T}_2$, en notant Σ les étiquettes en commun à $\mathcal{T}_1$ et $\mathcal{T}_2$, Σ_1 les étiquettes appartenant à $\mathcal{T}_1$ mais pas à $\mathcal{T}_2$ et Σ_2 les étiquettes appartenant à $\mathcal{T}_2$ mais pas à $\mathcal{T}_1$, et en notant M_1 la matricisation de $\mathcal{T}_1$ par rapport à (Σ_1, Σ), M_2 la matricisation de $\mathcal{T}_2$ par rapport à (Σ_2, Σ) et M la matricisation de $\mathcal{T}$ par rapport à (Σ_1, Σ_2), alors $M = M_1 M_2$. Cela signifie que le produit contracté $\mathcal{T} = \mathcal{T}_1 \odot_\Sigma \mathcal{T}_2$ de deux tenseurs peut être calculé au moyen du produit matriciel $M = M_1 M_2$. Cette propriété se généralise au produit mixte $\mathcal{T}_1 \odot_\Sigma \circledast_\Pi \mathcal{T}_2$, en enlevant les dimensions correspondant aux étiquettes de Π :

Lemme 2.23. *Soit $\mathcal{T}_1$ et $\mathcal{T}_2$ deux tenseurs compatibles pour le produit et soit (Π, Σ) une partition des étiquettes en commun. Soit $\mathcal{T} = \mathcal{T}_1 \odot_\Sigma \circledast_\Pi \mathcal{T}_2$. Pour chaque dictionnaire $\mathcal{D}_\pi$ qui correspond aux étiquettes de Π, on définit les matrices suivantes :*

$$M^{(\mathcal{D}_\pi)}[(\mathcal{D}_1), (\mathcal{D}_2)] = \mathcal{T}[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi],$$
$$M_1^{(\mathcal{D}_\pi)}[(\mathcal{D}_1), (\Sigma : I)] = \mathcal{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, \Sigma : I],$$
$$M_2^{(\mathcal{D}_\pi)}[(\Sigma : I), (\mathcal{D}_2)] = \mathcal{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, \Sigma : I].$$

On a $M^{(\mathcal{D}_\pi)} = M_1^{(\mathcal{D}_\pi)} M_2^{(\mathcal{D}_\pi)}$, c'est-à-dire que $M^{(\mathcal{D}_\pi)}$ est le produit matriciel de $M_1^{(\mathcal{D}_\pi)}$ par $M_2^{(\mathcal{D}_\pi)}$.

Démonstration. On a :

$$\begin{aligned}
M^{(\mathcal{D}_\pi)}[(\mathcal{D}_1),(\mathcal{D}_2)] &= \mathfrak{T}[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] \\
&= (\mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] \\
&= \sum_{\Sigma:I} \mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, \Sigma : I] \times \mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, \Sigma : I] \\
&= \sum_{\Sigma:I} M_1^{(\mathcal{D}_\pi)}[(\mathcal{D}_1), (\Sigma : I)] \times M_2^{(\mathcal{D}_\pi)}[(\Sigma : I), (\mathcal{D}_2)] \\
&= M_1^{(\mathcal{D}_\pi)} M_2^{(\mathcal{D}_\pi)}.
\end{aligned}$$

□

Grâce au lemme 2.20, on peut calculer l'intégralité des coefficients de $T = \mathfrak{T}_1 \odot_\Sigma \circledast_\Pi \mathfrak{T}_2$ en faisant un produit matriciel pour chaque dictionnaire qui associe un indice à chaque étiquette de Π. Dans notre cas, chaque dimension est de taille n. Cela signifie qu'il y a donc $n^{|\Sigma|}$ tels dictionnaires. Il reste à calculer le coût pour effectuer un de ces produits matriciels. Pour ce faire, on a besoin de la taille de chacune des matrices. La matrice $M_1^{(\mathcal{D}_\pi)}$ possède un nombre de lignes égal à n mis à un exposant égal au nombre d'étiquettes dans $\mathcal{D}_1$, c'est-à-dire le nombre d'étiquettes dans $\mathfrak{T}_1$ mais pas dans $\mathfrak{T}_2$. Vu que (Σ, Π) partitionne les étiquettes en commun entre $\mathfrak{T}_1$ et $\mathfrak{T}_2$, le nombre d'étiquettes dans $\mathcal{D}_1$ vaut $d_1 - |\Sigma| - |\Pi|$. Le nombre de colonnes de $M_1^{(\mathcal{D}_\pi)}$ vaut $n^{|\Sigma|}$. De même, le nombre de lignes de $M_1^{(\mathcal{D}_\pi)}$ est $n^{|\Sigma|}$ et son nombre de colonnes est $n^{d_2 - |\Sigma| - |\Pi|}$.

D'après le lemme 2.22, le coût optimal pour multiplier une matrice n^a par n^b par une matrice n^b par n^c est $\mathcal{O}(n^{\omega(a,b,c)})$, et on a l'inégalité $\omega(a,b,c) \leq \omega \times \max(a+b, b+c, a+c)/2$. Dans notre cas, on a $a = d_1 - |\Sigma| - |\Pi|$, $b = |\Sigma|$ et $c = d_2 - |\Sigma| - |\Pi|$. Ainsi, $a + b = d_1 - |\Pi|$, $b + c = d_2 - |\Pi|$ et $a + c = d_1 + d_2 - 2|\Sigma| - 2|\Pi| = d - |\Pi|$. On a donc que $\max(a+b, b+c, a+c) = d_{\max} - |\Pi|$.

Le coût total est alors :

$$n^{|\Pi|} n^{\omega(d_{\max} - |\Pi|)/2} \leq n^{\omega|\Pi|/2} n^{\omega(d_{\max} - |\Pi|)/2} = n^{\omega \cdot d_{\max}/2} = \sqrt{n^{\omega d_{\max}}}.$$

On a utilisé le fait que $\omega \geq 2$ est la borne inférieure triviale pour calculer le produit de deux matrices.

Chapitre 3

Une nouvelle méthode de compression de graphes respectant les chevauchements

Sommaire

Dans ce chapitre, je présente un travail qui a été publié à DaWaK 2023 [PSH23]. Étant donné un graphe G, on cherche à le compresser en une suite binaire la plus petite possible. Il existe de nombreuses façons de compresser un graphe, comme cela est présenté dans la section 3.1. Une de ces façons consiste à repérer des structures dans le graphe G, c'est-à-dire des sous-graphes de G avec des propriétés assez fortes permettant une compression efficace de ce sous-graphe. En partitionnant le graphe G en structures, on peut les compresser chacune indépendamment et ainsi compresser G. Pour compresser une structure en une suite binaire, nous utiliserons certains des éléments présentés dans la section 2.2. Cependant, en cas de chevauchement des structures, une partition de G n'est plus possible. La section 3.2 présente notre méthode de compression qui répond à cette problématique. Ensuite, la section 3.3 décrit en détail notre algorithme, dont le code source est disponible à l'adresse `https://gitlab.liris.cnrs.fr/fpitois/fgsp.git`. Finalement, les résultats obtenu avec notre algorithme sont présentés dans la section 3.4.

3.1 État de l'art

Il existe déjà un grand nombre d'algorithmes permettant de compresser un graphe sans perte de données. Nous citons une sélection de méthodes référencées dans [BH18] afin de montrer cette grande variété. Il existe deux grandes familles d'algorithmes de compression : ceux qui compressent le graphe sans avoir besoin d'un ordre particulier et ceux dont la qualité de la compression dépend directement de l'ordre choisi pour les sommets du graphe. Dans le premier cas, on peut citer les techniques de compression suivantes :

- **Les super-sommets.** Cette méthode consiste à regrouper plusieurs sommets ayant un voisinage similaire en un seul nouveau super-sommet [KKVF15].
- **Les nœuds virtuels.** Très similaires aux super-sommets, cette méthode consiste à regrouper ensemble plusieurs arêtes qui ont une extrémités en commun. Cela est utilisé pour représenter succinctement des sous-graphes bipartis complets [BC08] ou des cliques du graphes [RZ18]. Dans les deux cas, on introduit un nouveau sommet appelé *nœud virtuel* que l'on relie via des arêtes orientées aux différents sommets. Il y a une arrête entre deux sommets du graphe original si et seulement si dans le graphe compressé, il y a une arête allant du premier sommet vers le nœud virtuel et une arête allant du nœud virtuel vers le second sommet.
- **La représentation implicite de graphe**. Cela consiste à donner un identifiant particulier à chaque sommet de sorte que l'on puisse déduire s'il y a une arête entre deux sommets uniquement en regardant l'identifiant de ces deux sommets. Il n'y a donc plus besoin de stocker la liste des arêtes du graphes mais seulement la liste des identifiants [KNR88].

Dans le second cas, il faut trouver un ordre pertinent des sommets. Pour les graphes du web, un ordre est généralement privilégié ; il s'agit de l'ordre lexicographique des URL. Pour les autres graphes, un algorithme d'ordonnancement des sommets doit être utilisé. Un de ces algorithmes s'appelle SlashBurn [KF11]. Il ordonne les sommets d'un graphe de sorte à faire apparaitre beaucoup de régions vides dans la matrice d'adjacence, en utilisant une notion de centralité des sommets. De fait, il existe tout un pan de la recherche dédié à trouver un ordre des sommets qui minimise une certaine quantité [DPS02]. Cela se fait généralement en utilisant de la programmation linéaire entière [PS98]. Une fois les sommets ordonnés, on trouve plusieurs techniques de compression :

— **Les grammaires.** L'idée est de donner une grammaire capable de reconstruire le graphe. Par exemple, l'algorithme Re-Pair [LM00] remplace des paires d'éléments consécutifs dans les listes d'adjacence par un seul symbole stocké dans un dictionnaire [CN10]. Un autre algorithme repère des séquences de sommets consécutifs et les remplace par un seul nouveau symbole [GB11]. Ces deux algorithmes sont utilisés pour compresser les graphes du web. On peut aussi citer [MP16] qui généralise Re-Pair en cherchant des petits motifs fréquents dans le graphe.

— **Les arbres quaternaires**. On divise la matrice d'adjacence en quatre morceaux de taille égale et on recommence récursivement sur les morceaux qui ne contiennent pas un petit nombre de 1. Ce seuil est souvent fixé à 1 (c'est-à-dire qu'on continue récursivement dès qu'il y a deux 1 ou plus), mais par exemple [CLL+16] fixe un seuil à 3.

— **Les k^2-arbres.** Il s'agit d'une généralisation des arbres quaternaires. On divise la matrice d'adjacence en k^2 morceaux de taille égale et on recommence récursivement sur les morceaux qui ne sont pas vides. Cet algorithme est utilisé dans les graphes du web avec l'ordre lexicographique des adresses des pages web [BLN09] ou avec un parcours en largeur préalable [SXB+12]. L'approche est donc d'ordonner la matrice d'adjacence via des notions liées aux graphes, puis d'encoder la matrice d'adjacence en oubliant tout lien avec les graphes.

L'ordonnancement des sommets joue un rôle crucial pour chacune de ces techniques de compression. Celui-ci ne doit donc pas être arbitraire et doit correspondre aux mieux à la structure du graphe. Par exemple, si le graphe contient une clique, il est important que les différents sommets composant cette clique soient consécutifs dans l'ordre choisi. Le but est donc de suivre les différentes structures du graphes. Or, celles-ci peuvent se chevaucher, rendant compliqué l'ordonnancement linéaire des sommets. Le but de notre algorithme est alors de se focaliser sur ces chevauchements.

3.2 Notre approche

Notre idée consiste à chercher une partition $\mathcal{P}$ des sommets du graphe G qui soit compatible avec les différentes structures de G, c'est-à-dire telle que chaque structure soit une union de certaines parties de $\mathcal{P}$. Pour ce faire, on effectue du raffinement de partition à partir des structures de G trouvées via des algorithmes de détection de structures. Cela permet d'éliminer les chevauchements de structures, puisqu'au lieu d'avoir une liste de structures qui peuvent se chevaucher, on a une partition des sommets (donc sans chevauchement par définition d'une partition) dont les parties sont des sous-ensembles des structures trouvées. Une fois $\mathcal{P}$ construit, on ordonne les sommets de sorte à ce que les sommets d'une partie soient contigus. Cet ordre et cette partition permettent enfin de compresser le graphe en utilisant des méthodes classiques de compression de données (cf. section 2.2) sur les différentes parties du graphe.

3.2.1 Principe

Soit $\mathcal{P}$ une partition de l'ensemble des sommets de G. Soit p_i, p_j deux parties de $\mathcal{P}$. On considère le sous-graphe $G[p_i, p_j]$. Si $p_i = p_j$, alors il s'agit du sous-graphe induit par les sommets de $p_i = p_j$. Sinon, on sait que p_i et p_j sont disjoints et $G[p_i, p_j]$ est un graphe biparti dont les deux parties sont p_i et p_j. On appelle $G[p_i, p_j]$ un *bloc* de G. Un exemple

de décomposition d'un graphe en blocs est représenté sur la figure 3.1. Notre méthode de compression cherche à construire une partition des sommets de G qui corresponde aux structures que l'on peut trouver naturellement dans G. Le problème principal concerne les structures qui se chevauchent, c'est-à-dire qui ont une intersection non-vide mais où aucune n'est incluse dans l'autre. Ainsi, étant donné plusieurs structures, le but est de découper chacune de ces structures de sorte à ce qu'il n'y ait plus de chevauchement.

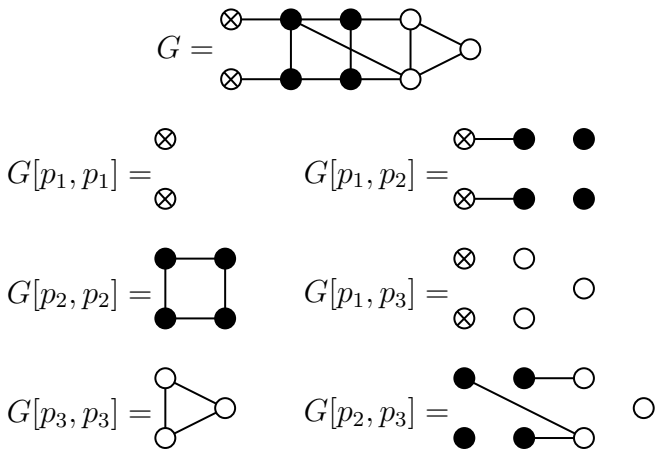


FIGURE 3.1 – Exemple d'une partition d'un graphe et des blocs associés. Les sommets du graphe G sont partitionnés en trois parties : p_1 (les cercles avec une croix), p_2 (les cercles noirs) et p_3 (les cercles blancs).

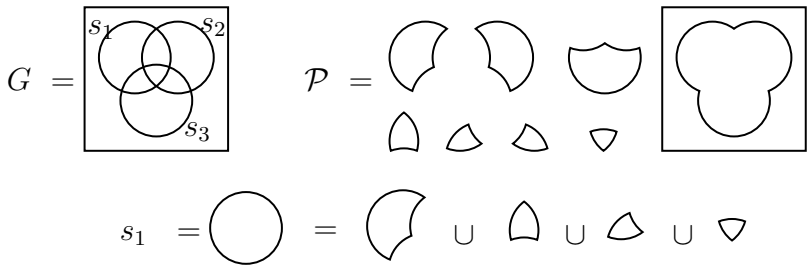
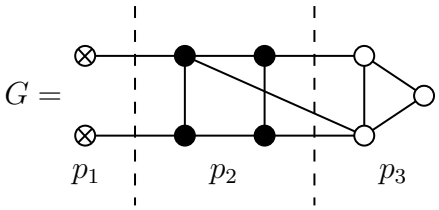


FIGURE 3.2 – Exemple d'une partition induite par trois structures. Le graphe G est représenté par un carré. Il y a trois structures s_1, s_2 et s_3 qui se chevauchent, représentées chacunes par un cercle. Cela induit une partition $\mathcal{P}$ faite de huit parties. Chaque structure peut être écrite comme une union disjointe de plusieurs parties ; seul le cas de la partie s_1 est représenté.

Étant donné un ensemble $\{s_1, \ldots, s_k\}$ de structures de G, on peut déduire une partition $\mathcal{P}$ des sommets de G de la façon suivante : deux sommets u et v de $V(G)$ sont dans la même partie si, et seulement si, ils appartiennent exactement aux mêmes structures. En d'autres termes, $\mathcal{P}$ est la partition la plus grossière telle que chaque structure est l'union d'une ou plusieurs parties de $\mathcal{P}$. Un aperçu de cette idée est représenté dans la figure 3.2. On note cette partition $\mathcal{P}(s_1, \ldots, s_k)$. On peut la définir formellement de la façon suivante :

Définition 3.1 (Notation $\mathcal{P}(s_1, \ldots, s_k)$)**.** *Soit $s_1, \ldots, s_k$ des ensembles de sommets d'un même graphe G. On note $\mathcal{P}(s_1, \ldots, s_k)$ la partition la plus grossière qui soit plus fine que chaque bi-partition $\{s_i, V(G) \setminus s_i\}$ pour chaque i. De façon équivalente, il s'agit de la partition définie par la relation d'équivalence suivante entre les éléments de $V(G)$: u est*



$$M = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

FIGURE 3.3 – Exemple d'un graphe G partitionné en trois parties p_1, p_2 et p_3 et de la méta-matrice M associée. Un 0 entre p_i et p_j dans M signifie que le bloc $G[p_i, p_j]$ n'a aucune arête. Un 1 signifie qu'il y a au moins une arête et que ce bloc est donc non-vide.

équivalent à v quand l'ensemble des s_i qui contiennent u est le même que l'ensemble des s_j qui contiennent v, c'est-à-dire $\{s_i \mid u \in s_i\} = \{s_j \mid v \in s_j\}$.

Pour encoder un graphe en utilisant une partition $\mathcal{P}$, il suffit d'encoder chaque bloc induit par chaque paire de parties de la partition, c'est-à-dire encoder $G[p_i, p_j]$ pour chaque $p_i, p_j \in \mathcal{P}$. Cela encode bien chaque arête du graphe G.

On peut réaliser une optimisation de cet encodage : quand on considère des graphes réels et des partitions bien choisies, on observe que la majorité des blocs ne contiennent aucune arête. Afin d'éviter d'utiliser de l'espace mémoire pour n'encoder aucune information, on peut n'encoder que les blocs non-vides. Il faut alors encoder d'une façon ou d'une autre l'emplacement des blocs non-vides. On fait cela au moyen d'un objet que l'on a appelé la *méta-matrice*. Un exemple de méta-matrice est représenté dans la figure 3.3.

3.2.2 Estimation de l'encodage

Pour compresser un graphe G en utilisant notre méthode, on doit trouver une partition $\mathcal{P}$ qui minimise le coût total d'encodage de G en utilisant cette partition. Pour encoder toutes les informations nécessaires, on doit encoder la partition en question, la méta-matrice ainsi que chaque bloc non-vide. Pour encoder la partition $\mathcal{P}$, l'ordre des sommets n'ayant pas d'importance, on réordonne les sommets du graphe de sorte à ce que chaque partie de $\mathcal{P}$ forme un intervalle. On rappelle que l'ensemble des sommets $V(G)$ est choisi comme étant l'ensemble des entiers de 1 à n. Ainsi, pour encoder la partition, on a simplement besoin d'encoder la liste des tailles des différentes parties, et non le contenu de chacune d'elles.

Ainsi, pour encoder complètement la partition $\mathcal{P}$, on encode le nombre de sommets de G, la taille de $\mathcal{P}$ (c'est-à-dire le nombre de parties), et la taille de chacune des parties de $\mathcal{P}$.

Il reste à encoder la donnée principale, à savoir les différentes arêtes du graphe G. Pour ce faire, on encode la méta-matrice ainsi que chaque bloc non-vide. Pour encoder ces données et utiliser ce qui a été vu dans la section 2.2.4 et les précédentes, il faut pouvoir transformer la méta-matrice et chaque bloc en une suite binaire. Pour ce faire, on commence par représenter chacune des ces données par une matrice binaire, qui est la représentation la plus naturelle :

— La méta-matrice est représentée par une matrice binaire carrée symétrique de taille $|\mathcal{P}|$. Un 1 en position (i, j) signifie que le bloc $G[p_i, p_j]$ est non-vide.

— Un bloc non-diagonal $G[p_i, p_j]$ avec $i \neq j$ est représenté par une matrice binaire rectangulaire de taille $|p_i|$ par $|p_j|$. Un 1 en position (i', j') signifie qu'il y a une arête entre le i'-ième sommet de p_i et le j'-ième sommet de p_j.

— Un bloc diagonal $G[p_i, p_i]$ est représenté par une matrice binaire carrée de taille $|p_i|$ par $|p_i|$. Un 1 en position (i', j') signifie qu'il y a une arête entre le i'-ième sommet et le j'-ième sommet de p_i. Comme on ne considère que des graphes sans boucle, la diagonale est remplie de 0.

Un exemple est représenté dans la figure 3.4.

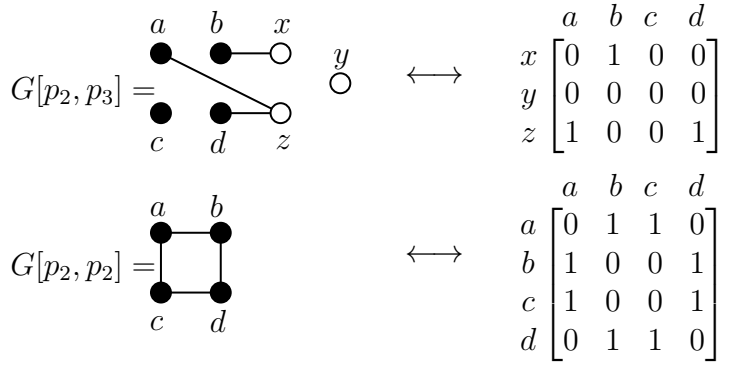


FIGURE 3.4 – Exemple de blocs avec leur représentation matricielle.

Ensuite, on transforme une matrice binaire en une suite binaire en concaténant ses différentes lignes. En faisant cela, on élague également les informations dupliquées ou inutiles. Cela signifie que pour une matrice symétrique, on n'écrit pas les éléments qui sont sous la diagonale, et pour une matrice dont on sait la diagonale nulle, on n'écrit pas cette diagonale de zéros.

Exemple 3.1. On considère la méta-matrice M ci-dessous, la matrice N correspondant à un bloc non-diagonal et la matrice D correspondant à un bloc diagonal. On sait qu'une méta-matrice est symétrique, et on sait qu'une matrice correspondant à un bloc diagonal est symétrique et a une diagonale de zéros.

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, N = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

La matrice M se représente via la suite binaire 010 11 1 (on a concaténé les trois lignes de M, mais en tronquant le début de chaque ligne pour ne pas répéter les éléments identiques dus à la symétrie de la matrice). La matrice N se représente via la suite binaire 0100 0000 1001 (on a juste concaténé les trois lignes de N). La matrice D se représente via la suite binaire 110 01 1 (on a concaténé les quatre lignes de D, mais en tronquant le début de chaque ligne pour ne pas répéter les éléments identiques dus à la symétrie de la matrice et on omettant le zéro de la diagonale ; cela fait que la dernière ligne de la matrice n'est pas du tout écrite).

Pour estimer cet encodage, on utilise le Théorème de code des sources de Shannon pour dire qu'une suite de a zéros et b uns s'encode environ avec $(a+b)h(b/(a+b))$ bits. Cela n'est qu'une borne inférieure, mais cela donne une estimation suffisamment bonne et bien plus rapide que d'encoder réellement cette matrice via le procédé décrit dans la section 2.2.4. On estime le coût pour encoder un entier i par $2\log(i)$, ce qui est une borne supérieure de la taille de `mixte`(i).

Définition 3.2 (Notations `inline`(M), `shannon-estimate`(M) et `count`(M))**.** *Étant donné une matrice M qui est soit une méta-matrice soit une matrice correspondant à un bloc, on note* $\texttt{inline}(M)$ *la représentation de M sous forme d'une suite binaire comme décrit dans l'exemple 3.1. Si a est le nombre de zéros de* $\texttt{inline}(M)$ *et b son nombre de uns, on note* $\texttt{shannon-estimate}(M)$ *la quantité $(a+b)h(b/(a+b))$ qui correspond à l'estimation de nombre de bits de* $\texttt{inline}(M)$ *via le Théorème de code des sources de Shannon. On note également* $\texttt{count}(M)$ *le nombre de 1 dans* $\texttt{inline}(M)$*.*

Étant donné une partition $\mathcal{P}$ de sommets de G, on note `estimate`$(G, \mathcal{P})$ la somme de toutes ces quantités. Précisément, `estimate`$(G, \mathcal{P})$ est défini de la façon suivante :

Définition 3.3 (Notation `estimate`$(G, \mathcal{P})$)**.** *Soit $\mathcal{P}$ une partition de G et soit M la méta-matrice associée. On définit :*

$$\begin{aligned}
\texttt{estimate}(G, \mathcal{P}) = {} & 2\log|V(G)| \\
& + 2\log|\mathcal{P}| \\
& + \sum_i 2\log|p_i| \\
& + 2\log(\texttt{count}(M)) \\
& + \texttt{shannon-estimate}(M) \\
& + \sum_{i \leq j} 2\log(\texttt{count}(G[p_i, p_j])) \\
& + \sum_{i \leq j} \texttt{shannon-estimate}(G[p_i, p_j])
\end{aligned}$$

3.3 Algorithme

Notre algorithme est divisé en trois étapes. Ces étapes ont été implémentées indépendamment, ce qui permet de modifier une étape sans impacter les deux autres.

— Étape 1 : on exécute un certain nombre d'algorithmes capables de détecter des structures dans un graphe. On a choisi d'utiliser SlashBurn [KF11], BigClam [YL13], Louvain [BGLL08] et Metis [Kar97]. Ces algorithmes sont expliqués dans la section 3.3.1. Il s'agit principalement d'algorithmes de partitionnement de graphe, qui peuvent trouver des communautés et plus généralement des sous-graphes denses. À la fin de cette étape, on a une liste de structures $s_1, \ldots, s_k$.

— Étape 2 : on utilise une heuristique gloutonne pour trouver une partition $\mathcal{P}$ de $V(G)$. On commence avec un ensemble de structures $S = \emptyset$ et la partition la plus grossière $\mathcal{P} = \{V(G)\}$. À chaque itération, on choisit de façon gloutonne une structure s_i et on l'ajoute à S. Cela signifie que l'on choisit s_i de sorte à ce que `estimate`$(G, \mathcal{P}(S))$ soit minimale. On s'arrête dès qu'on ne peut plus trouver de structure s_i qui fasse diminuer `estimate`$(G, \mathcal{P}(S))$. Cet algorithme est expliqué en détail dans la section 3.3.2.

— Étape 3 : on encode le graphe G en utilisant la partition $\mathcal{P}$. Cette étape est décrite dans la section 3.3.3.

Le code source de notre algorithme est disponible à l'adresse `https://gitlab.liris.cnrs.fr/fpitois/fgsp.git`. Les trois étapes décrites ci-dessus sont bien séparées dans le

code fourni. L'étape 1 est programmée en Python afin d'utiliser des bibliothèques qui implémentent la plupart des algorithmes utilisés dans cette étape. L'étape 2 est programmée en Java pour une question de lisibilité du code. L'étape 3 est programmée en Python.

3.3.1 Étape 1 : recherche de structures

On a utilisé les quatre algorithmes suivants pour trouver des structures denses dans un graphe :

— *SlashBurn [KF11] :* SlashBurn est un algorithme d'ordonnancement des sommets qui peut être utilisé pour détecter des communautés ayant une forme d'étoile. Cet algorithme choisit k sommets ayant la meilleure *centralité*, les supprime du graphe, conserve la plus grande composante connexe et on supprime les autres composantes connexes. L'ensemble des k sommets ayant la plus forte centralité est appelé un *centre*. Chaque composante qui est supprimée est appelé une *branche*. Notre version de SlashBurn effectue ces opérations et conserve chaque centre et chaque branche trouvée. Il s'agit des structures trouvées par notre version de SlashBurn.

— *BigClam [YL13] :* BigClam est un algorithme de détection de communautés utilisant la factorisation de matrices positives. L'idée est de factoriser la matrice d'adjacence du graphe de taille n par n en une matrice F de taille n par k et une matrice k par n, avec $k < n$. La matrice F peut alors s'interpréter comme « le sommet $i \in [n]$ appartient à la communauté $j \in [k]$ avec une force de $F_{i,j}$ ». Cela permet de détecter des structures denses se chevauchant.

— *Louvain [BGLL08] :* Louvain permet de détecter des communautés qui ne se chevauchent pas en partitionnant le graphe. Pour ce faire, Louvain commence par la partition la plus fine (celle où chaque partie est constituée d'un unique sommet) et alterne ensuite entre des changements locaux dans les parties (comme changer un sommet de partie) et des fusions de parties (afin d'obtenir une partition moins fine). Ces changements sont décidés en évaluant la *modularité* du graphe et en essayant de l'augmenter au maximum. Dans ce contexte, la modularité est un nombre réel entre -1/2 et 1 qui indique s'il y a plus d'arêtes à l'intérieur des parties qu'entre celles-ci [BDG+07].

— *Metis [Kar97] :* Metis permet de détecter des communautés qui ne se chevauchent pas en réduisant la taille du graphe, en partitionnant le graphe réduit et en projetant cette partition sur le graphe original. Le graphe est réduit en fusionnant deux sommets en un seul et en répétant cette opération jusqu'à obtenir une taille satisfaisante. Les sommets fusionnés ainsi que leur ordre de fusion sont retenus de sorte à pouvoir les séparer à nouveau plus tard dans l'ordre inverse. Ensuite, un algorithme de partitionnement est utilisé sur le graphe réduit. Cet algorithme divise le graphe en k parties, où l'entier k est un paramètre de l'algorithme. Puisque ce graphe est petit, l'exécution est relativement rapide. Enfin, pour étendre cette partition au graphe original, les sommets fusionnés sont séparés un à un. La partition est alors légèrement adaptée pour correspondre au mieux au graphe agrandi. Cette opération est recommencée jusqu'à ce que tous les sommets soient séparés.

3.3.2 Étape 2 : sélection des structures

L'algorithme 1 décrit la partie dédiée à la sélection de structures. Cela est fait via un algorithme glouton. L'algorithme prend en entrée un graphe G et une liste de structures L

et il produit en sortie un ensemble de structures S qui est un sous-ensemble des structures de la liste L. Le but est que l'ensemble S permette de compresser efficacement G en utilisant la partition $\mathcal{P}(S)$. L'algorithme glouton commence en initialisant S à l'ensemble vide. À chaque étape, il cherche la structure $s \in L \setminus S$ (c'est-à-dire qu'il cherche s dans la liste des structures potentielles L qui n'ont pas déjà été ajoutées à S) qui minimise la compression estimée si on ajoute s à S. Pour ce faire, l'algorithme essaie chaque s dans $L \setminus S$ et garde celui qui produit le meilleur résultat, si tant est qu'il permet de diminuer la compression estimée. En effet, il est possible qu'ajouter une structure qui augmente la compression estimée, notamment parce que cela fragmente trop $\mathcal{P}(S)$. Dans le cas où aucune structure ne permet de diminuer la compression estimée, l'algorithme s'arrête et renvoie S. Sinon, la structure s qui a été la meilleure est définitivement ajoutée à S et ce processus de sélection est recommencé. Pour que l'algorithme soit efficace, la partition $\mathcal{P}(S)$ est mise à jour dynamiquement en utilisant du raffinement de partition. Cela signifie que l'instruction $S \leftarrow S \cup \{s\}$ met aussi à jour la valeur de $\mathcal{P}(S \cup \{s\})$, en utilisant la valeur de $\mathcal{P}(S)$. Les différents blocs de G induits par $\mathcal{P}(S)$ sont aussi mis à jour dynamiquement d'une manière similaire. A contrario, l'instruction $S \leftarrow S \setminus \{s\}$ met à jour $\mathcal{P}(S)$ en mémorisant la valeur de $\mathcal{P}(S \setminus \{s\})$ pour ne pas avoir à la recalculer.

Algorithme 1 : Algorithme glouton de sélection de structures.

Entrée : Un graphe G, une liste de structures L
Sortie : Un sous-ensemble de structures $S \subseteq L$ tel que G peut être compressé efficacement en utilisant la partition $\mathcal{P}(S)$

```
1  S ← ∅
2  meilleureCompression ← estimate(G, P(S))
3  Boucle
       // Chaque exécution de la boucle ajoute une structure à S
4      meilleureStructure ← NONE        // On l'ajoutera à la fin de la boucle
5      Pour s ∈ L \ S faire
           // On parcourt l'ensemble des structures potentielles
6          S ← S ∪ {s}                                  // On ajoute s provisoirement
7          compressionActuelle ← estimate(G, P(S))
8          Si compressionActuelle < meilleureCompression alors
               // Si ajouter s à S donne une meilleure compression, on le stocke
                  dans meilleureStructure
9              meilleureCompression ← compressionActuelle
10             meilleureStructure ← s
11         S ← S \ {s}                          // On a fini d'estimer s, on l'enlèvre
12     Si meilleureStructure ≠ NONE alors
           /* On ajoute définitivement à S la structure qui donne une meilleure
              compression, si elle existe                                        */
13         S ← S ∪ {meilleureStructure}
14     Sinon
15         Retourner S      // Sinon, on s'arrête et on retourne l'ensemble trouvé
```

3.3.3 Étape 3 : encodage

La dernière partie de notre algorithme consiste à encoder le graphe en utilisant la partition trouvée avec l'algorithme 1. Pour ce faire, on encode les différentes informations nécessaires à la décompression du graphe. Il s'agit des mêmes informations que celles qui sont utilisées pour estimer la taille du graphe compressé (cf. définition 3.3), à savoir :

1. le nombre de sommets n du graphe,
2. le nombre de parties de la partition $\mathcal{P}(S)$,
3. le nombre de sommets de chaque partie de $\mathcal{P}(S)$,
4. le nombre de blocs non-vides de la méta-matrice M,
5. la méta-matrice M,
6. pour chaque bloc de M, le nombre de 1 dans la suite binaire qui représente ce bloc,
7. pour chaque bloc de M, la suite binaire qui le représente.

La compression des données se fait comme rappelé dans la section 2.2.4. L'idée est qu'à chaque fois que l'on utilise la compression entropique pour encoder une suite de bits, il faut aussi stocker la taille de cette suite non-compressée et le nombre de 1 de cette suite. On peut vérifier que cela est bien le cas dans notre compression. Notre algorithme compresse de façon entropique la méta-matrice M ainsi que les différents blocs de M. On vérifie que l'on a bien chacune des ces informations :

— On peut déduire la taille de la suite binaire correspondant à la méta-matrice M non-compressée, car elle correspond à $k(k+1)/2$, où k est la taille de la méta-matrice, c'est-à-dire le nombre de parties dans $\mathcal{P}(S)$, qui est encodé en (2). On rappelle en effet que la méta-matrice est une matrice carrée symétrique. On n'encode alors qu'une seule moitié de la matrice et on retrouve l'autre par symétrie.

— Le nombre de 1 dans la méta-matrice est le nombre de blocs non-vides, qui est encodé en (4).

— Pour chaque bloc de M, on peut déduire la taille de la suite binaire correspondant au bloc non-compressé, car elle correspond soit à $k_i k_j$ (si le bloc n'est pas sur la diagonale de la méta-matrice), soit à $k_i(k_j-1)/2$ (s'il est sur la diagonale), où k_i est la hauteur du bloc, k_j sa largeur. Le nombre k_i est le nombre de sommets de la i-ème partie de $\mathcal{P}(S)$ et k_j le nombre de sommets de la j-ème partie de $\mathcal{P}(S)$. Cela est encodé en (3). On rappelle qu'un bloc diagonal est une matrice carrée symétrique avec une diagonale vide. On n'encode alors qu'une seule moitié de la matrice et on retrouve l'autre par symétrie.

— Pour chaque bloc de M, le nombre de 1 dans ce bloc est encodé en (6).

Afin de s'assurer que rien ne manquait, nous avons aussi construit un algorithme de décompression et nous avons vérifié que les graphes avant et après avoir effectué une compression et une décompression étaient isomorphes.

3.4 Résultats expérimentaux

Nous avons testé notre algorithme appelé FGSP (Fine-Grained Structural Partitioning) sur divers graphes réels disponibles sur différentes bases de données : le dépôt SNAP (`https://snap.stanford.edu/data/`), Network Repository [RA15] et le projet MUSAE [RAS19]. La table 3.1 résume les caractéristiques des graphes utilisés. La table 3.2 montre

TABLE 3.1 – Jeu de données.

Graphe	Nœuds	Arêtes
hamming8-2	256	31,616
ia-infect-dublin	410	2,765
web-google	1,299	2,773
ia-email-univ	1,133	5,451
choc	2,899	5,467
web-BerkStan	12,305	19,500
rt_lolgop	9,765	10,324
as-oregon	13,579	37,448
as-caida	65,536	53,381
facebook_combined	4,039	88,234
musae-twitch	6,549	112,666
ca-HepPh	11,204	117,619
ca-AstroPh	17,903	196,972
web-sk-2005	121,422	334,419
enron	80,163	288,364
email-eu-all	265,214	365,570
ca-MathSciNet	332,689	820,644
roadNet-PA	1,090,920	1,541,898

TABLE 3.2 – Résultats expérimentaux. **Struct** est le nombre de structures **Gardé** est le pourcentage de structures gardées à l'étape 2. **Part** est le nombre de parties dans la partition $\mathcal{P}(S)$. **Estimée** est la taille estimée (en octets) du graphe compressé en utilisant $\mathcal{P}(S)$. **Taille** est la taille réelle (en octets) du graphe compressé à l'étape 3. **BPE** est le nombre de bits par arête (Bits Per Edge) du graphe compressé. **T** est le temps total d'exécution des 3 parties (« s » signifie « seconde », « m » signifie « minute », « h » signifie « heure » et « j » signifie « jour »).

Graphe	Struct	Gardé	Part	Estimée	Taille	BPE	T
hamming8-2	40	5.0 %	3	835	713	0.18	4 s
ia-infect-dublin	59	33.9 %	32	1,302	1,420	4.11	4 s
web-google	79	55.7 %	79	1,429	1,597	4.61	7 s
ia-email-univ	117	13.7 %	46	4,447	4,949	7.26	7 s
choc	205	28.3 %	245	4,440	5,635	8.25	16 s
web-BerkStan	306	63.1 %	721	14,153	15,694	6.44	2 m
rt_lolgop	7,322	1.7 %	145	1,274	1,688	1.31	5 m
as-oregon	256	39.1 %	568	26,934	32,399	6.92	3 m
as-caida	285	44.9 %	802	41,140	47,628	7.14	7 m
facebook_combined	292	39.4 %	299	31,487	32,897	2.98	49 s
musae-twitch	551	8.5 %	241	95,833	102,247	7.26	4 m
ca-HepPh	602	46.8 %	2,013	61,010	73,958	5.03	10 m
ca-AstroPh	989	25.1 %	1,842	170,502	198,996	8.08	24 m
web-sk-2005	751	67.1 %	1,610	253,072	256,992	6.15	1 h
enron	1,909	12.3 %	2,755	360,412	268,805	7.46	2 h
email-eu-all	16,577	2.9 %	3,239	209,606	268,717	5.88	2 j
ca-MathSciNet	1,546	40.9 %	16,256	1,090,035	1,254,030	12.22	2 j
roadNet-PA	1,729	63.5 %	19,780	1,835,527	1,894,070	9.83	8 j

les résultats de FGSP sur ces graphes. Les tests ont été effectués sur un ordinateur portable Intel Core avec un CPU i5, en utilisant un seul cœur à 1.70 GHz.

3.4.1 Choix de la recherche de structures

TABLE 3.3 – Sélection des structures, représenté par le nombre de structures gardées sur le nombre total de structures trouvées par l'algorithme indiqué en tête de colonne.

Graphe	SlashBurn	BigClam	Louvain	Metis
`hamming8-2`	0/26	1/1	1/2	0/11
`ia-infect-dublin`	6/34	0/3	6/8	8/14
`web-google`	4/6	0/2	26/46	14/25
`ia-email-univ`	2/76	1/2	12/16	1/23
`choc`	17/35	1/2	28/131	12/37
`web-BerkStan`	44/68	0/1	83/159	66/78
`rt_lolgop`	3/4	0/1	109/7248	10/69
`as-oregon`	43/110	0/1	39/63	18/82
`as-caida`	54/112	1/1	50/57	7/115
`facebook_combined`	70/228	0/2	13/18	32/44
`musae-twitch`	34/468	1/2	9/24	3/57
`ca-HepPh`	186/462	0/2	30/64	66/74
`ca-AstroPh`	128/850	0/1	32/44	88/94
`web-sk-2005`	65/76	0/2	234/427	205/246
`enron`	78/433	1/1	123/1276	81/199
`ca-MathSciNet`	22/536	0/1	212/602	398/407

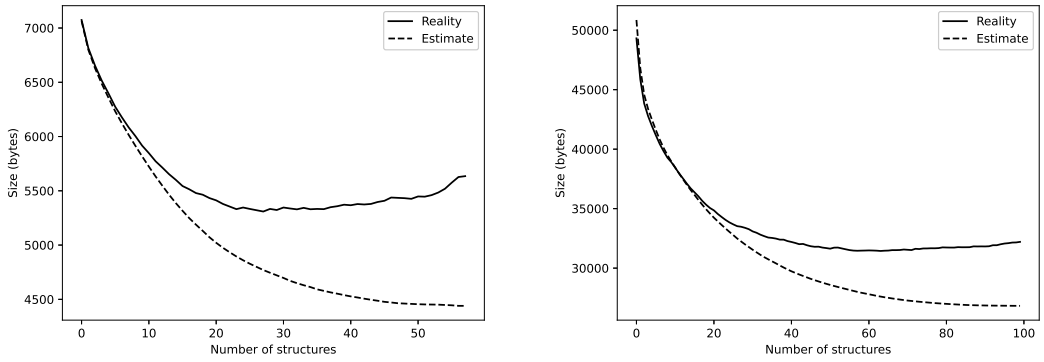
Le but de l'étape 1 est d'utiliser des algorithmes qui détectent chacun des structures différentes, afin de rassembler le maximum de points de vue sur le graphe. La table 3.3 montre pour chaque graphe et chaque algorithme de détection de structures le nombre de structures détectées par cet algorithme, ainsi que le nombre et le pourcentage de structures gardées à l'étape 2. Nous remarquons que SlashBurn, Louvain et Metis produisent de nombreuses structures qui ont tendance à être gardées, alors que l'impact de BigClam est plus subtil. Cependant, bien que le nombre de structures gardées par BigClam soit toujours 0 ou 1, nous avons remarqué que l'enlever pouvait conduire à une compression jusqu'à 4 % moins bonne. En effet, quand une structure de BigClam est gardée par l'algorithme glouton de la partie 2, c'est souvent dans les premiers tours de boucle : pour `ia-email-univ`, il s'agit de la toute première structure sélectionnée ; pour `as-caida` et `musae-twitch`, il s'agit de la deuxième ; pour `choc`, c'est la 39^{e} structure ; et pour `enron`, c'est la 158^{e} sur un total de 1909 structures sélectionnées. En enlevant BigClam, on obtient une compression 1 % moins bonne pour le graphe `as-caida` et 4 % moins bonne pour les graphes `ia-email-univ` et `choc`. En revanche, on a obtenu une meilleure compression pour le graphe `musae-twitch` (mais de seulement 1 %). Cela montre aussi l'influence négative que peut avoir un trop grand ensemble de structures potentielles. Ainsi, un paramètre important pour l'efficacité de notre algorithme est l'ensemble des algorithmes de détection de structures choisis.

Un autre facteur sur lequel il est possible de jouer est celui des paramètres internes des différents algorithmes. Par exemple, SlashBurn a besoin d'un paramètre k qui influe fortement sur les structures qu'il peut détecter. D'après [KF11], il est conseillé de choisir

une valeur de k qui dépend linéairement du nombre de sommets n du graphe. Nous avons opté pour $k = 0.001n$, car cela donnait des résultats satisfaisants. Nous avons également légèrement modifié SlashBurn pour que la valeur de k change dynamiquement avec la taille du graphe. En effet, SlashBurn réduit le graphe au fur et à mesure, il semblait cohérent d'adapter k en conséquence afin qu'il reste dépendant linéairement du nombre de sommets considérés. Cependant, afin de ne pas obtenir des structures trop petites (ce qui est globalement mauvais pour notre algorithme de compression), nous avons choisi une valeur minimale pour k valant 10. Concernant Metis, nous avons suivi l'avis des auteurs et pris $\sqrt{n/2}$ comme nombre de communautés.

3.4.2 Précision de l'estimation de l'encodage

À l'étape 2, nous utilisons une fonction `estimate` pour estimer la taille du graphe compressé. Cependant, comme ce n'est qu'une estimation, il peut arriver que l'algorithme glouton (qui utilise cette estimation pour savoir s'il faut continuer d'ajouter des structures) pense qu'il faut continuer alors que les structures ajoutées font en fait augmenter la taille de la compression réelle. Cela amène à un *point de demi-tour* dans la recherche gloutonne, c'est-à-dire un moment où la compression devient de moins en moins bonne alors que l'estimation nous dit le contraire. Nous avons observé ce phénomène avec tous les graphes. La figure 3.5 montre ce phénomène pour les graphes `choc` et `as-oregon`.



(a) Choc (b) As-oregon

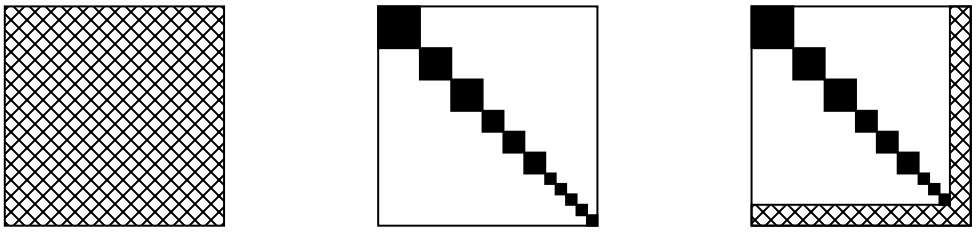
FIGURE 3.5 – Estimation versus réalité.

Une façon d'y remédier serait de réellement compresser le graphe après chaque ajout de structure, pour vérifier que cet ajout glouton fait réellement diminuer la taille du graphe compressé. Il ne serait pas possible de remplacer chaque appel à `estimate` par un appel à une compression réelle du graphe, car la compression réelle est bien plus lente que la simple estimation. Pour résumer, notre algorithme actuel appelle la fonction `estimate` un nombre quadratique de fois (dans chaque tour de boucle, elle est appelée pour chaque structure potentielle) et une seule fois la fonction de compression réelle ; il serait envisageable d'appeler un nombre linéaire de fois la fonction de compression réelle (une fois par tour de boucle) ; mais il ne serait pas envisageable de remplacer le nombre quadratique d'appels à `estimate` par un appel à la compression réelle du graphe.

Pour gagner encore un peu de temps, on ne pourrait appeler la fonction de compression réelle que si le gain calculé par `estimate` passe sous un certain seuil. En faisant cela, on pourrait globalement accélérer et améliorer notre algorithme global, car cela permettrait

d'arrêter la recherche gloutonne plus tôt (ce qui fait gagner du temps) et avec un meilleur résultat (car on s'est arrêté avant que la compression n'augmente).

3.4.3 Évolution de la matrice d'adjacence



(a) Au début. (b) Vers la fin. (c) À la fin.

FIGURE 3.6 – Schéma de la forme globale de la matrice d'adjacence pendant la recherche gloutonne. Une région noire représente une région dense, une région blanche représente une région presque vide et une région hachurée représente une région au contenu plutôt aléatoire.

La figure 3.7 montre l'exécution de la recherche gloutonne de l'étape 2 sur le graphe `as-oregon`. Elle représente la matrice d'adjacence du graphe après l'exécution de plusieurs tours de la boucle principale de l'algorithme 1. On rappelle que chaque tour de boucle permet d'ajouter une structure supplémentaire dans l'ensemble S des structures sélectionnées. L'ordre des sommets est celui induit par la partition $\mathcal{P}(S)$, en triant les différentes parties par ordre décroissant. On remarque que plus on sélectionne de structures, plus la matrice tend vers une matrice par blocs où chaque bloc diagonal est dense et où les autres blocs sont vides, excepté pour une petite *bande* en bas et à droite, qui se fait de plus en plus grande avec le temps. De façon générale, le même phénomène est observé avec les autres graphes. Une version schématique de ce phénomène est représentée sur la figure 3.6. On a globalement trois phases :

(a) au début, le contenu est aléatoire,

(b) vers la fin, la matrice est bien organisée avec une diagonale dense,

(c) à la fin, la bande apparait.

La forme qui parait la plus facile à compresser est la forme 3.6b. Une idée serait donc d'arrêter la recherche gloutonne à ce moment-là. Cela nous fait penser au *point de demi-tour* de la section 3.4.2 et nous conforte dans l'idée qu'il faut arrêter la recherche gloutonne un peu plus tôt que ce que nous avons fait, typiquement à ce point de demi-tour qui est facile à atteindre. À ce moment-là, bien que la *bande* soit assez fine, elle semble être toujours présente. Une autre optimisation pourrait être de l'encoder sous la forme d'un seul bloc, car sa fragmentation n'apporte pas grand chose, voire peut être négatif.

La figure 3.8 montre la matrice d'adjacence de certains graphes de la table 3.1, une fois l'algorithme glouton complètement terminé. On remarque à nouveau sur ces graphes le schéma global d'une matrice par blocs à la diagonale dense. Cependant, certaines matrices sont légèrement différentes. Une perspective serait de comprendre pourquoi on obtient ces différences et s'il serait possible de les anticiper et d'adapter notre méthode de compression à ces cas particuliers.

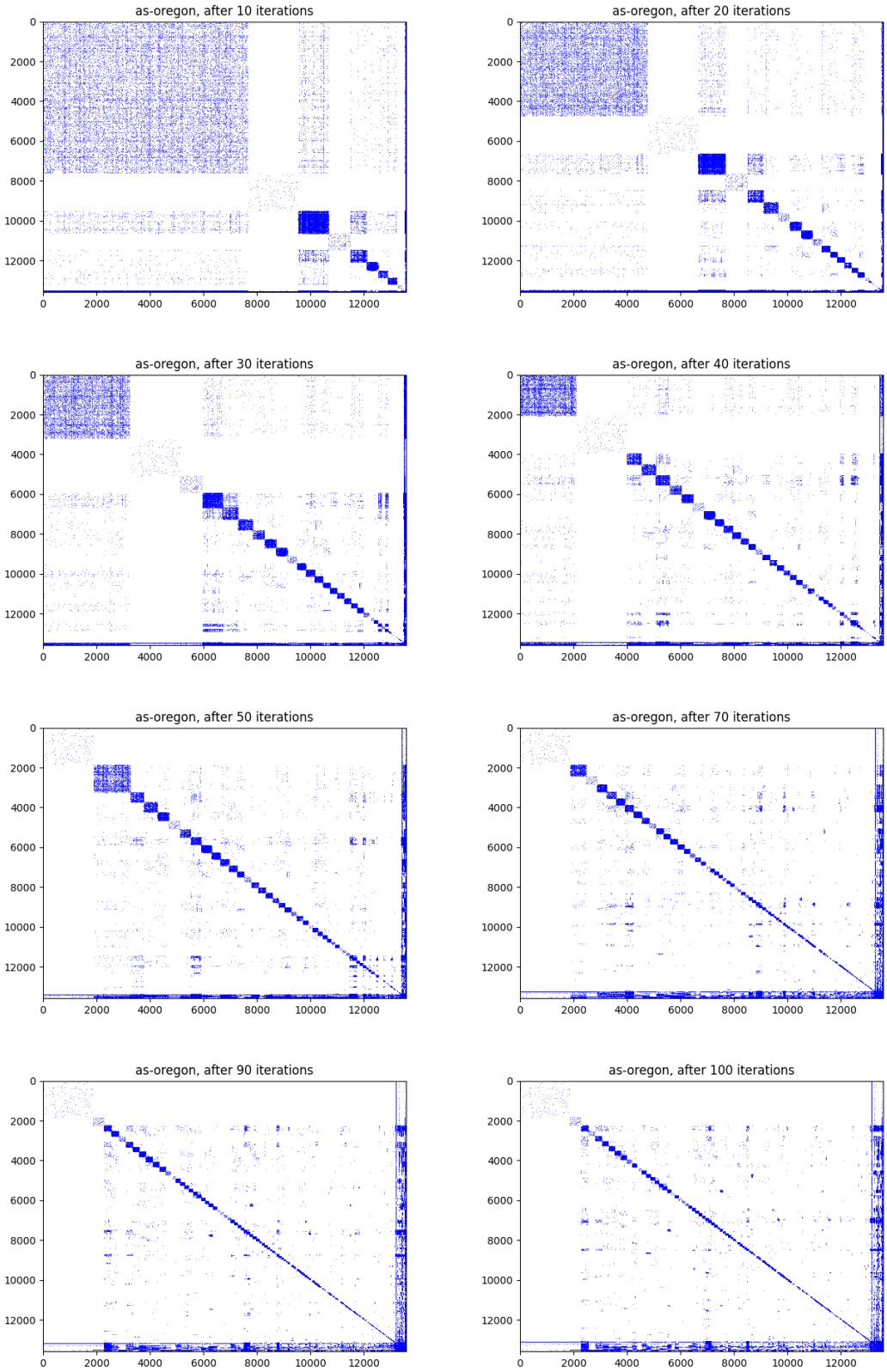


FIGURE 3.7 – Évolution de la matrice d'adjacence de `as-oregon` pendant la recherche gloutonne.

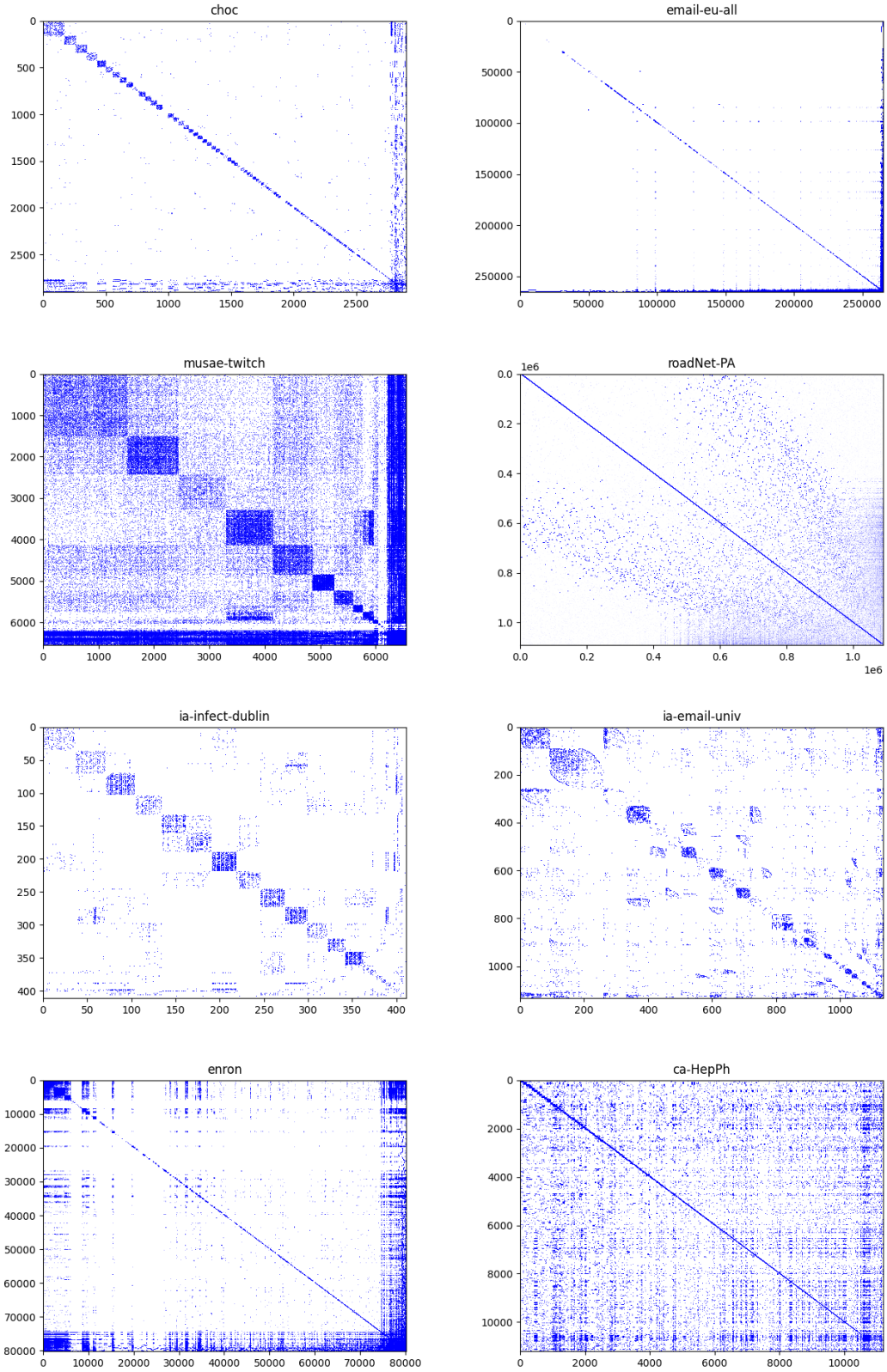


FIGURE 3.8 – La matrice d'adjacence de certains graphes ordonnés par notre algorithme.

3.4.4 Comparaison avec d'autres méthodes

Nous avons comparé notre algorithme avec GraphZIP [RZ18]. Nos résultats sont résumés dans la table 3.4. Les résultats de compression de GraphZIP ont été obtenus à partir de leur article [RZ18]. Comme le code de GraphZIP n'est pas disponible, nous n'avons pas pu le comparer sur l'intégralité des graphes que nous avons utilisés pour notre algorithme. De façon générale, notre approche permet d'obtenir des résultats qui sont entre 5 et 10 fois meilleurs. Le seul point négatif étant le temps d'exécution de notre algorithme, qui peut prendre plusieurs heures voire plusieurs jours, là où GraphZIP finit systématiquement en moins d'une minute. Ce n'est pas tant un problème, car notre algorithme se concentre sur le résultat final plutôt que le temps d'exécution. Une perspective reste cependant d'optimiser ce temps d'exécution, notamment en arrêtant l'étape de recherche gloutonne plus tôt.

3.5 Perspectives

Il semble possible d'améliorer davantage notre algorithme. Tout d'abord, l'étape 1 de notre algorithme, qui consiste à utiliser d'autres algorithmes de recherche de structures, peut être améliorée en choisissant d'autres algorithmes supplémentaires. Le chapitre 4 est dédié à l'étude de certains algorithmes de recherche de structures, à savoir les modules et les splits. Nous pourrions aussi effectuer une analyse préalable du graphe afin de ne lancer qu'un nombre restreint d'algorithmes pertinents. Par exemple, si le graphe est plutôt creux, il ne semble pas utile de lancer des algorithmes de recherche de clique. Nous pourrions enfin essayer de construire notre propre algorithme de recherche de structures ; le chapitre 5 est dédié à cela. Concernant l'étape 2, l'heuristique gloutonne pourrait être modifiée ou remplacée par une autre. Par exemple, on pourrait donner un score à chaque structure en fonction de sa taille et de sa densité et ajouter les structures une à une suivant ce score. Ainsi, chaque structure n'est évaluée qu'une seule fois et non une fois par tour de boucle. De plus, puisque notre algorithme se concentre sur les chevauchements entre structures, on pourrait construire un graphe de chevauchement que l'on analyserait pour en déduire un ensemble de structures qui ne se chevauchent pas trop. Combinées ensemble, on obtiendrait un graphe de chevauchement pondéré, dont le but serait de trouver un ensemble indépendant maximal pour éviter tout chevauchement (trouver un tel ensemble est NP-complet, mais [LSS$^+$19] propose un algorithme qui trouve de bonnes approximations en temps raisonnable pour des cas réels), ou au moins un sous-graphe assez creux. Enfin, il serait possible d'utiliser l'ordre ainsi trouvé pour chercher efficacement de nouvelles structures. Le chapitre 6 est dédié à la recherche efficace de structures dans les graphes ordonnés.

TABLE 3.4 – Comparaison entre **GraphZIP** et notre algorithme **FGSP**. **Brut** correspond au graphe donné sous forme d'une liste d'arêtes.

(a) Comparaison exprimée en octets.

Graphe	**Brut**	**GraphZIP**	**FGSP**
`hamming8-2`	226,252	110,838	713
`ia-infect-dublin`	20,597	13,287	1,420
`web-google`	22,977	12,216	1,597
`ia-email-univ`	41,802	32,405	4,949
`web-BerkStan`	199,261	135,813	15,694
`ca-HepPh`	1,178,601	498,661	73,958
`ca-AstroPh`	2,121,144	1,423,607	198,996
`web-sk-2005`	4,083,711	2,040,783	256,992
`ca-MathSciNet`	10,482,316	8,370,757	1,254,030

(b) Comparaison exprimée en BPE (nombre de bits par arête).

Graphe	**Brut**	**GraphZIP**	**FGSP**
`hamming8-2`	57.25	28.05	0.18
`ia-infect-dublin`	59.59	38.44	4.11
`web-google`	66.29	35.24	4.61
`ia-email-univ`	61.35	47.56	7.26
`web-BerkStan`	81.75	55.72	6.44
`ca-HepPh`	80.16	33.92	5.03
`ca-AstroPh`	86.15	57.82	8.08
`web-sk-2005`	97.69	48.82	6.15
`ca-MathSciNet`	102.19	81.60	12.22

(c) Comparaison des temps d'exécution.

Graphe	**GraphZIP**	**FGSP**
`hamming8-2`	1 s	4 s
`ia-infect-dublin`	7 ms	4 s
`web-google`	4 ms	7 s
`ia-email-univ`	17 ms	7 s
`web-BerkStan`	170 ms	2 m
`ca-HepPh`	5.6 s	10 m
`ca-AstroPh`	1.3 s	24 m
`web-sk-2005`	9 s	1 h
`ca-MathSciNet`	25 s	2 j

Chapitre 4

Décomposition modulaire et en splits : une approche unifiée

Sommaire

Dans ce chapitre, je reviens sur les décompositions en modules et en splits. Il s'agit majoritairement d'un état de l'art, même si je me suis attelé à présenter des preuves nouvelles. Je commence par rappeler la définition d'un module et je montre les différentes propriétés de stabilité des modules (ce sont des propriétés de la forme « si X et Y sont des modules vérifiant certaines conditions, alors un autre ensemble construit à partir de X et Y est aussi un module »). Il s'agit de propriétés connues des modules que je démontre succinctement par soucis d'exhaustivité. La majeure partie de ce chapitre concerne les splits, qui sont à la fois une généralisation des modules mais aussi l'objet que je vais à mon tour généraliser dans la chapitre 5. Je commence par présenter six définitions différentes des splits que j'ai pu rencontrer depuis leur introduction par Cunningham au début des années 80 jusqu'à aujourd'hui et je montre leur équivalence. Tout comme les modules, les splits ont des propriétés de stabilité, qui sont elles aussi bien connues. Cependant, il peut être difficile d'en trouver la preuve et l'une des propriétés de stabilité en particulier n'est pas triviale. Je propose donc une nouvelle preuve de la stabilité des splits par union disjointe qui n'utilise que des outils liés au graphe (par opposition à la preuve des autres stabilités qui utilisent des outils empruntés à l'algèbre linéaire). Je finis enfin par présenter les familles symétriques traversantes et les familles sans croisement qui permettent d'abstraire la notion de split. Ces familles seront par la suite généralisées sous la forme d'hyper-graphe dans le chapitre 5, d'où la nécessité d'en présenter la version originale dans ce chapitre.

4.1 Décomposition modulaire

4.1.1 Module

On s'intéresse dans cette partie aux *modules* d'un graphe. Ils ont été introduits par Gallai dans [Gal67]. Ils ont eu plusieurs noms et définitions. Dans ce manuscrit, je me suis basé sur les travaux de Habib et al. [HP10].

Définition 4.1 (Sommet distinguant et sommet homogène [HP10])**.** *Soit $M \subseteq V(G)$ un ensemble de sommets d'un graphe G et soit x un sommet de $V(G) \setminus M$. On dit que x* distingue *M (ou que x est un* sommet distinguant *de M) s'il existe $y \in M$ et $z \in M$ tels que (x, y) soit une arête de G mais (x, z) ne soit pas une arête de G. Dans le cas contraire, on dit que x est* homogène *par rapport à M.*

Dans le cas où x est homogène par rapport à M, on a soit que tous les sommets de M sont adjacents à x (c'est-à-dire que $M \subseteq N(x)$), soit qu'aucun d'eux ne l'est (c'est-à-dire que $N(x) \cap M = \emptyset$).

Exemple 4.1. On considère le graphe G à 6 sommets représenté en figure 4.1. Si l'on considère l'ensemble de sommets $M = \{4, 5, 6\}$, on a :

- le sommet 1 est homogène par rapport à M car $N(1) = \{2, 4, 5, 6\} \supseteq \{4, 5, 6\} = M$;
- le sommet 2 est homogène par rapport à M car les ensembles $N(2) = \{1, 3\}$ et $M = \{4, 5, 6\}$ sont disjoints ;
- le sommet 3 distingue M car $(3, 6)$ est une arête mais $(3, 5)$ ne l'est pas.

Définition 4.2 (Ensembles qui se chevauchent $X \between Y$ et ensembles orthogonaux $X \perp Y$)**.** *On dit de deux ensembles X et Y qu'ils se* chevauchent *quand $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ et $Y \setminus X \neq \emptyset$. On note $X \between Y$. Quand deux ensembles X et Y ne se chevauchent pas, on dit qu'ils sont* orthogonaux*. On note $X \perp Y$.*

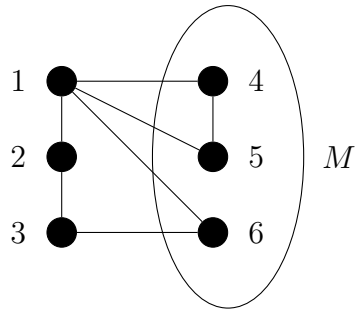


FIGURE 4.1 – Graphe ayant 6 sommets numérotés de 1 à 6 et ayant pour arêtes $(1,2),(1,4),(1,5),(1,6),(2,3),(3,6)$ et $(4,5)$.

Observation 4.1. *La notion de chevauchement permet de catégoriser les paires d'ensembles. Étant donné deux ensembles X et Y différents, exactement un des cas de figure suivant est vrai :*

- $X \cap Y = \emptyset$
- $X \subset Y$
- $X \supset Y$
- $X \bigcirc\!\!\!\!\bigcirc Y$

Définition 4.3 (Module [Gal67])**.** *Un* module M *d'un graphe G est un ensemble de sommets de $V(G)$ qui n'admet pas de sommet distinguant, c'est-à-dire que tout sommet $x \in V(G) \setminus M$ est homogène par rapport à M.*

Certains auteurs (comme [BLS99]) demandent en plus qu'un module M soit non-vide. Dans ma thèse, je considère qu'un module puisse être vide. Je discuterai de la volonté de ce choix dans les sections suivantes. Pour mieux comprendre la définition d'un module, voici d'autres caractérisations équivalentes.

Lemme 4.1. *Soit G un graphe et M un ensemble de sommets. Les propositions suivantes sont équivalentes :*

1. *M est un module.*
2. *Le complémentaire de M peut être partitionné entre les voisins de M et les non-voisins de M, c'est-à-dire qu'il existe deux ensembles de sommets A, B tels que $A \cup B = \overline{M}$, $A \cap B = \emptyset$ et pour tout sommet $x \in M$ on a $N(x) \setminus M = A$.*
3. *Tous les sommets de M ont le même voisinage extérieur, c'est-à-dire que pour tout $x, y \in M$ on a $N(x) \setminus M = N(y) \setminus M$.*

Démonstration. On procède par implication circulaire :

- (1 $\implies$ 2) Comme M est un module, chaque sommet x de $\overline{M}$ est homogène, c'est-à-dire que soit tous les sommets de M sont adjacents à x, soit aucun ne l'est. On définit A comme étant l'ensemble de sommets vérifiant le premier cas et B comme étant l'ensemble des sommets vérifiant le second cas.
- (2 $\implies$ 3) Tous les sommets x de M ont pour voisinage extérieur $N(x) \setminus M = A$.
- (3 $\implies$ 1) On note A ce voisinage extérieur commun et on vérifie que chaque sommet de $x \in A$ vérifie $M \subseteq N(x)$ et que chaque sommet $x \in \overline{M \cup A}$ vérifie $N(x) \cap M = \emptyset$. □

Ainsi, un module induit une partition du graphe en trois ensembles M, A, B (avec A et/ou B éventuellement vides), avec les contraintes suivantes :

— Il y a toutes les arêtes possibles entre M et A.

— Il n'y a aucune arête entre M et B.

Ces caractérisations sont utiles pour se représenter ce à quoi ressemble un module, comme montré en figure 4.2. Cependant, c'est souvent la définition première d'un module qui permet de prouver des lemmes sur les modules, à savoir qu'un module est un ensemble sans sommet distinguant.

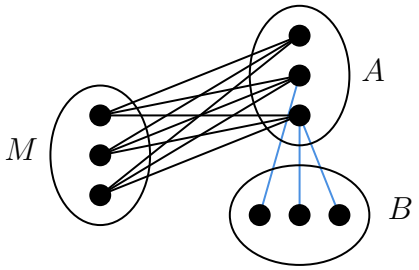


FIGURE 4.2 – Exemple du module. Il y a toutes les arêtes possibles entre M et A mais il n'y a aucune arête entre M et B. Il est possible qu'il y ait des arêtes entre A et B, elles sont représentées en bleu sur la figure.

Dans le cas où A est vide, on observe que M est une union de composantes connexes du graphe. Si B est vide, A est aussi un module. Ce sont les deux seuls cas où le complémentaire d'un module est aussi un module. Le cas où A et B sont tous les deux vides correspond au module $M = V(G)$.

Propriétés

Les modules vérifient des propriétés de stabilité.

Lemme 4.2 ([CHM81])**.** *Soit X et Y deux modules de G tels que $X \bigcirc\!\!\!\!\bigcirc Y$. Alors les ensembles de sommets $X \cap Y$, $X \cup Y$, $X \setminus Y$, $Y \setminus X$ et $X \Delta Y$ sont aussi des modules.*

Démonstration. On prouve les différents cas d'une manière similaire : on suppose qu'il existe un sommet s qui distingue l'ensemble dont on veut prouver qu'il est un module et on aboutit à une contradiction, en utilisant les ensembles que l'on sait non-vides par hypothèse et le fait que X et Y sont des modules (ce qui induit qu'ils n'ont pas de sommet distinguant dans leur complémentaire). La figure 4.3 illustre certains de ces cas afin de faciliter le suivi de la preuve.

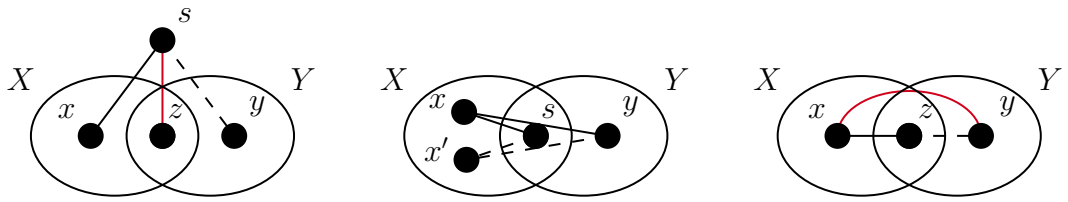
— $(X \cap Y)$ Supposons qu'il existe un sommet s qui distingue $X \cap Y$. On arrive à une contradiction en se demandant où se trouve s. S'il est dans $\overline{X}$, alors s distingue X. S'il est dans $\overline{Y}$, il distingue Y. Ainsi, s doit être dans $\overline{\overline{X} \cup \overline{Y}} = X \cap Y$, ce qui est interdit par la définition d'un sommet distinguant.

— $(X \cup Y)$ Supposons qu'il existe un sommet s qui distingue $X \cup Y$. Il existe donc deux sommets $x, y \in X \cup Y$ tels que (s, x) soit une arête et que (s, y) n'en soit pas une. Il n'est pas possible que x et y soient tous les deux dans X, car sinon s serait un sommet distinguant de X. De même pour Y. Sans perte de généralité, on peut supposer que $x \in X$ et $y \in Y$. On sait de plus que $X \cap Y$ est non-vide, il existe donc $z \in X \cap Y$. On arrive à une contradiction en se demandant s'il y a une arête entre s et z : si oui, s distingue Y. Si non, s distingue X.

— $(X \setminus Y)$ Supposons qu'il existe un sommet s qui distingue $X \setminus Y$. Le sommet s se trouve dans $X \cap Y$, sinon, s distinguerait X. Il existe donc deux sommets $x, x' \in X \setminus Y$ tels que (s, x) soit une arête et que (s, x') n'en soit pas une. Comme $Y \setminus X$ est non-vide, il existe $y \in Y \setminus X$. On regarde quelles sont les arêtes entre y et les autres sommets : il y a une arête entre y et x, sinon x distingue Y. De même, il n'y a pas d'arête entre y et x'. Mais dans ce cas, y distingue X, ce qui est une contradiction.

— $(Y \setminus X)$ On raisonne comme pour le cas $X \setminus Y$.

— $(X \Delta Y)$ Supposons qu'il existe un sommet s qui distingue $X \Delta Y$. Si s se trouve dans $\overline{X \cup Y}$, alors s distingue $X \cup Y$. Ainsi, s se trouve dans $X \cap Y$. Il existe donc deux sommets $x, y \in X \cap Y$ tels que (s, x) soit une arête et que (s, y) n'en soit pas une. Il n'est pas possible que x et y soient tous les deux dans $X \setminus Y$, car sinon s serait un sommet distinguant de X. De même pour Y. Sans perte de généralité, on peut supposer que $x \in X \setminus Y$ et $y \in Y \setminus X$. On arrive à une contradiction en se demandant s'il y a une arête entre x et y : si oui, y distingue X. Si non, x distingue Y. □



(a) $X \cup Y$ (b) $X \setminus Y$ (c) $X \Delta Y$

FIGURE 4.3 – Illustration des trois cas $X \cup Y$, $X \setminus Y$ et $X \Delta Y$ de la preuve du lemme 4.2. Un trait plein représente une arête, un trait pointillé représente une non-arête et un trait rouge représente une arête contradictoire (c'est-à-dire qui ne peut être ni présente ni absente, entrainant une contradiction).

On remarque dans la preuve de ce lemme que toutes les hypothèses ne sont pas nécessaires à chaque fois. Pour l'intersection, il ne faut aucune hypothèse, sauf si on impose à un module d'être non-vide, auquel cas il suffit juste que l'intersection soit non-vide. Pour l'union, il suffit que l'intersection soit non-vide. Pour les différences, il suffit qu'elles soient toutes les deux non-vides : l'une pour la preuve, l'autre dans le cas où on dit que $\emptyset$ n'est pas un module. Pour le cas de la différence symétrique, la preuve nécessite que l'intersection soit non-vide pour la preuve, et si on demande à ce que $X \Delta Y$ soit non-vide, il faut que l'une des deux différences soit non-vide.

4.1.2 Famille partitive et orthogonalité

Famille partitive

On a vu que les modules vérifiaient certaines propriétés de stabilité. On veut maintenant abstraire la notion de module : on définit un objet mathématique qui vérifie les mêmes propriétés de stabilité que les modules afin de voir ce que l'on peut déduire.

Définition 4.4 (Famille partitive [CHM81])**.** *Soit E un ensemble et soit $\mathcal{F}$ une famille de sous-ensembles de E. On dit que $\mathcal{F}$ est partitive quand :*

— *les ensembles $\emptyset$ et E appartiennent à $\mathcal{F}$,*

— *chaque singleton* $\{x\}, x \in E$ *appartient à* $\mathcal{F}$
— *si* X *et* Y *sont deux ensembles appartenant à* $\mathcal{F}$ *tels que* $X \bigcirc\!\!\!\!\bigcirc Y$*, alors les ensembles* $X \cup Y$, $X \cap Y$, $X \setminus Y$, $Y \setminus X$ *et* $X \Delta Y$ *appartiennent aussi à* $\mathcal{F}$.

Comme pour les modules, il y a un désaccord sur l'ensemble vide entre les différents articles, certains demandant à ce qu'une famille partitive contienne l'ensemble vide [CHM81], d'autres non [HP10, BXHR12]. Certains encore n'y accordent pas d'importance [CDMR12]. Dans la continuité de mon choix pour les modules, je considère que l'ensemble vide doive faire partie de toute famille partitive. La volonté est la suivante : étant donné une famille quelconque, je veux pouvoir la compléter en famille partitive en ajoutant de nouveaux ensembles, en utilisant par exemple un opérateur de clôture. Si on interdit à un ensemble d'être présent dans la famille, cette propriété n'est pas possible pour toutes les familles qui le contiendraient.

Les conditions de stabilité peuvent aussi être altérées [CHM81], mais cela ne change pas la définition d'une famille partitive, comme le montre le lemme suivant :

Lemme 4.3. *Soit* E *un ensemble et soit* $\mathcal{F}$ *une famille de sous-ensembles de* E*. Alors* $\mathcal{F}$ *est une famille partitive si et seulement si elle vérifie les conditions suivantes :*

— *les ensembles* $\emptyset$ *et* E *appartiennent à* $\mathcal{F}$,
— *chaque singleton* $\{x\}, x \in E$ *appartient à* $\mathcal{F}$
— *si* X *et* Y *sont deux ensembles appartenant à* $\mathcal{F}$, $X \cap Y$ *appartient aussi à* $\mathcal{F}$.
— *si* X *et* Y *sont deux ensembles appartenant à* $\mathcal{F}$ *tels que* $X \bigcirc\!\!\!\!\bigcirc Y$*, alors les ensembles* $X \cup Y$ *et* $X \Delta Y$ *appartiennent aussi à* $\mathcal{F}$.

Démonstration. On démontre ce qui diffère entre la définition 4.4 et cette caractérisation.

($\Longrightarrow$) Il suffit de considérer deux ensembles X et Y et de démontrer que $X \cap Y$ appartient à $\mathcal{F}$ même si on n'a pas l'hypothèse $X \bigcirc\!\!\!\!\bigcirc Y$. Ainsi, d'après l'observation 4.1, les cas possibles pour X et Y sont $X = Y$, $X \cap Y = \emptyset$, $X \subset Y$ ou $X \supset Y$. Respectivement, $X \cap Y$ vaut X, $\emptyset$, X, Y. Dans tous les cas, $X \cap Y \in \mathcal{F}$.

($\Longleftarrow$) Soit deux ensembles X et Y tels que $X \bigcirc\!\!\!\!\bigcirc Y$. On a $X \Delta Y \in \mathcal{F}$ et donc $(X \Delta Y) \cap Y \in \mathcal{F}$, ce qui montre que $X \setminus Y \in \mathcal{F}$. On fait de même pour montrer que $Y \setminus X \in \mathcal{F}$. □

Orthogonale d'une famille

Soit $\mathcal{F}$ une famille de sous-ensembles de E. La définition 4.2 définit une relation (au sens de la section 2.3.2) sur les éléments de $\mathcal{F}$ par $X \perp Y \iff X \cap Y = \emptyset \vee X \subseteq Y \vee Y \subseteq X$. On peut ainsi considérer l'opérateur de clôture associé à cette relation, que l'on note $\mathrm{cl}_\perp(\cdot)$. On rappelle que la fonction $\mathrm{cl}_\perp(\cdot)$ prend en argument une famille $\mathcal{F}$ de sous-ensembles de E et renvoie une famille plus grande de sous-ensembles de E. Dans notre cas, en reprenant la notation $\mathcal{F}^\perp := \{X \in 2^E \mid \forall Y \in \mathcal{F}, X \perp Y\}$ de [CDMR12], cette fonction s'écrit $\mathrm{cl}_\perp(\mathcal{F}) = (\mathcal{F}^\perp)^\perp$ (cf. définition 2.11 en remarquant que la relation $\perp$ est symétrique, c'est-à-dire que $X \perp Y \iff Y \perp X$). Dans [CDMR12], il est montré que la famille de Moore associée à l'opérateur de clôture $\mathrm{cl}_\perp(\cdot)$, c'est-à-dire la famille des éléments de 2^E tels que $\mathrm{cl}_\perp(\mathcal{F}) = \mathcal{F}$ (cf. lemme 2.5 point 1), est la famille des familles partitives sur E.

Proposition 4.4. *Soit* E *un ensemble et soit* $\mathcal{F}$ *un famille de sous-ensembles de* E*. La famille* $\mathcal{F}$ *est partitive si et seulement* $\mathcal{F}$ *est clos pour* $cl_\perp(\cdot)$*, c'est-à-dire si* $cl_\perp(\mathcal{F}) = \mathcal{F}$.

Grâce à cette égalité, il est possible de calculer une représentation arborescente des modules en temps linéaire [CDMR12] et de faire de même pour les *splits*, qui sont une généralisation des modules à laquelle la prochaine section est dédiée.

4.2 Décomposition en splits

Les splits sont une généralisation des modules. Tout comme les modules, ils peuvent être définis à partir de sommets distinguants ou à partir d'une partition du complémentaire. Cependant, ils peuvent aussi être définis d'un point de vue purement algébrique, en oubliant la structure de graphe sous-jacente.

4.2.1 Définitions

Les splits ont été introduits par Cunningham et Edmonds dans [CE80] d'une façon extrêmement générale, puisqu'ils y étaient définis dans les hyper-graphes (ou famille d'ensemble) et dans un *cadre de décomposition*, qui est une généralisation abstraite à n'importe quel type d'objets. De plus, [CE80] contient bien une application des splits aux graphes, mais celle-ci est une notion différente de celle qui a perduré et qui nous intéresse ici, puisque cette première notion de split partitionnait les arêtes et non les sommets. Cependant, la notion de split telle qu'elle va être introduite dans la définition 4.5 pouvait être entrevue dans [CE80] en appliquant la définition des splits d'hyper-graphes à des hyper-graphes 2-réguliers, c'est-à-dire des graphes. C'est d'ailleurs ce qui a été fait deux ans plus tard par Cunningham seul dans [Cun82], où il donne la première définition moderne des splits, même si elle s'applique aussi aux graphes dirigés. De nombreuses définitions équivalentes ont été données par la suite, les voici :

Définition 4.5 (Split)**.** *Soit G un graphe. Un ensemble S de sommets de G est un* split *quand l'une des conditions équivalentes suivantes est vérifiée :*

- *([Cun82]) Si (x, y) et (x', y') sont deux arêtes du bord ∂S de S, alors (x, y') est aussi une arête de ∂S.*
- *([GP12]) Chaque sommet de $x \in S$ qui est voisin d'un sommet de $\overline{S}$ est adjacent à chaque sommet $y' \in \overline{S}$ qui est voisin d'un sommet de S.*
- *([CDMR12], définition courte) L'ensemble d'arêtes ∂S induit un graphe biparti complet.*
- *([BLS99]) Il existe $A \subseteq S$ et $A' \subseteq \overline{S}$ tel que $\{(x, y) \in E(G) \mid x \in S \text{ and } y \in \overline{S}\} = A \times A'$, c'est-à-dire que l'ensemble des arêtes entre S et $\overline{S}$ est égal à l'ensemble de toutes les arêtes possibles entre A et A'.*
- *([CDMR12], définition longue) Il existe quatre ensembles A, B, A', B' tels que $S = A \cup B$, $\overline{S} = A' \cup B'$, $A \cap B = A' \cap B' = \emptyset$ et il y a toutes les arêtes entre A et A' et aucune autre arête entre S et $\overline{S}$.*
- *([PHST24]) Le rang de la coupe $(S, \overline{S})$ est au plus 1. (cf. définition 2.19)*

Démonstration. Pour que cette définition soit complète, on prouve par implication circulaire que ces six définitions sont équivalentes :

- ([Cun82] $\implies$ [GP12]) Soit un sommet $x \in S$ qui est voisin d'un sommet $y \in \overline{S}$ et soit un sommet $y' \notin S$ qui est voisin d'un sommet $x' \in S$. Ainsi, (x, y) et (x', y') sont deux arêtes du bord ∂S de S. Donc (x, y') est une arête de S.

— ([GP12] $\implies$ [CDMR12]) On doit montrer que le sous-graphe G' constitué des sommets qui sont les extrémités d'une arête de ∂S et des arêtes de ∂S est un graphe biparti complet. Les deux parties sont $V(G') \cap S$ et $V(G') \cap \overline{S}$. Soit un sommet $x \in V(G') \cap S$ dans la première partie et soit un sommet $y' \in V(G') \cap \overline{S}$ dans la seconde partie. On sait que x est dans S et qu'il est extrémité d'une arête de ∂S, il a donc un voisin dans $\overline{S}$. Il est donc adjacent à chaque sommet de $\overline{S}$ qui est voisin d'un sommet S, notamment y'. Ainsi, G' est bien biparti complet.

— ([CDMR12], définition courte $\implies$ [BLS99]) On pose A l'ensemble des sommets de S qui ont un voisin dans $\overline{S}$ et A' l'ensemble des sommets de $\overline{S}$ qui ont un voisin dans S. On sait donc que le graphe induit par les arêtes entre A et A' est biparti complet. Il reste à montrer qu'il n'y a pas d'autre arêtes entre S et $\overline{S}$, ce qui est le cas par définition de A et de A', à savoir qu'un sommet qui n'est pas dans ces ensembles n'a pas de voisin de l'autre côté du bord ∂S.

— ([BLS99] $\implies$ [CDMR12], définition longue) On pose $B = S \setminus A$ et $B' = \overline{S} \setminus A'$. Par définition, $S = A \cup B$, $\overline{S} = A' \cup B'$ et $A \cap B = A' \cap B' = \emptyset$. On sait que l'ensemble des arêtes entre S et $\overline{S}$ est égal à $A \times A'$, ce qui signifie qu'il y a toutes les arêtes entre A et A' et aucune autre arête entre S et $\overline{S}$.

— ([CDMR12], définition longue $\implies$ [PHST24]) On s'intéresse à la matrice d'adjacence de $G[S, \overline{S}]$. Comme S se partitionne en A et B et $\overline{S}$ en A' et B', cette matrice d'adjacence a naturellement une forme de matrice par bloc, ou les quatre blocs sont les matrices d'adjacence de $G[A, A']$, $G[A, B']$, $G[B, A']$ et $G[B, B']$. On sait qu'il y a toutes les arêtes entre A et A', donc le bloc $G[A, A']$ est rempli de 1. De plus, il n'y a aucune autre arête entre S et $\overline{S}$, donc les trois blocs $G[A, B']$, $G[B, A']$ et $G[B, B']$ sont remplis de 0. Ainsi, le rang de la matrice d'adjacence de $G[S, \overline{S}]$ est au plus 1. Ce rang est 0 dans le cas où A ou A' est vide.

— ([PHST24] $\implies$ [Cun82]) On prouve la contraposée, c'est-à-dire que le bord ∂S contient les arêtes (x, y), (x', y') mais pas (x, y'), alors le rang de la coupe $(S, \overline{S})$ est au moins égal à 2. En effet, dans ce cas, la matrice d'adjacence de $G[\{x, x'\}, \{y, y'\}]$ est la suivante, où l'étoile représente une valeur inconnue :

$$\begin{array}{c} \\ x \\ x' \end{array} \begin{array}{c} \begin{array}{cc} y & y' \end{array} \\ \begin{bmatrix} 1 & 0 \\ * & 1 \end{bmatrix} \end{array}$$

Quelle que soit la valeur de l'étoile, cette matrice a rang 2. Ainsi, puisque la matrice d'adjacence de $G[S, \overline{S}]$ contient cette matrice comme sous-matrice, son rang est au moins 2. □

Traditionnellement, on demande à un split que lui et son complémentaire soient de taille au moins 2 [Cun82, GP12, BLS99] et notamment que les quatre parties A, B, A', B' soient non-vides [CDMR12]. Cependant, la définition algébrique ([PHST24]) n'a pas besoin d'imposer cette condition. En revanche, en imposant que la condition algébrique soit « le rang de la coupe $(S, \overline{S})$ est exactement 1 », en retrouve la condition qu'un split et son complémentaire doivent être non-vides, et plus particulièrement que les ensembles A et A' doivent être non-vides. On définit alors un *split strict* :

Définition 4.6 (Split strict)**.** *On appelle* split strict *un split S dont le rang de la coupe $(S, \overline{S})$ est exactement 1, c'est-à-dire un split tel que ∂S est non-vide, ou encore un split tel que A est non-vide.*

On a vu que les modules avaient une notion de chevauchent (cf. définition 4.2). Le pendant de cette notion pour les splits est le *croisement* :

Définition 4.7 (Ensembles qui se croisent $X \bigcirc\!\!\bigcirc_\times Y$ et ensembles croix-orthogonaux $X \perp_\times Y$)**.** *On dit de deux ensembles X et Y qu'ils se* croisent *quand $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$ et $\overline{X \cup Y} \neq \emptyset$. Autrement dit, X et Y se croisent quand on a à la fois $X \bigcirc\!\!\bigcirc Y$ et $\overline{X} \bigcirc\!\!\bigcirc \overline{Y}$. On note $X \bigcirc\!\!\bigcirc_\times Y$. Quand deux ensembles X et Y ne se croisent pas, on dit qu'ils sont* croix-orthogonaux. *On note $X \perp_\times Y$.*

Tout comme les modules, les splits ont des propriétés de stabilité.

4.2.2 Stabilité

Point de vue algébrique

On utilise ici exclusivement le point [PHST24] de la définition 4.5. On a alors la proposition suivante :

Proposition 4.5. *Soit G un graphe connexe et soit X et Y deux splits de G qui se croisent ($X \bigcirc\!\!\bigcirc_\times Y$). Alors les ensembles de sommets $X \cap Y$, $X \cup Y$, $X \setminus Y$ et $Y \setminus X$ sont aussi des splits.*

Démonstration. Comme X et Y sont des splits, on a $\rho(X) \leq 1$ et $\rho(Y) \leq 1$. Comme G est connexe, on a aussi que tout ensemble Z qui n'est ni $\emptyset$ ni V vérifie $\rho(Z) \geq 1$. En particulier, vu que $X \bigcirc\!\!\bigcirc_\times Y$, $\rho(X \cap Y) \geq 1$, $\rho(\overline{X \cup Y}) \geq 1$, $\rho(X \setminus Y) \geq 1$ et $\rho(Y \setminus X) \geq 1$. Par sous-modularité et symétrie de ρ, on a $\rho(X \cap Y) + \rho(\overline{X \cup Y}) \leq \rho(X) + \rho(Y)$. Ainsi, $\rho(X \cap Y) \leq \rho(X) + \rho(Y) - \rho(\overline{X \cup Y}) \leq 1 + 1 - 1 \leq 1$. De même, $\rho(\overline{X \cup Y}) \leq 1$. Ainsi, $X \cap Y$ et $X \cup Y$ sont des splits. On remplace Y par $\overline{Y}$ et par symétrie de la fonction ρ, on a $\rho(X \setminus Y) + \rho(Y \setminus X) \leq \rho(X) + \rho(Y)$, ce qui permet de conclure que $X \setminus Y$ et $Y \setminus X$ sont des splits. □

Point de vue graphe

Dans cette partie, on adopte un point de vue purement combinatoire, c'est-à-dire que l'on s'autorise tous les points de la définition 4.5 excepté le point [PHST24]. On va notamment utiliser la caractérisation suivante, qui est pour les splits ce que la définition 4.3 est pour les modules.

Lemme 4.6. *Soit S un split d'un graphe G. Il existe deux ensembles A et B tels que $A \cup B = S$, $A \cap B = \emptyset$ et tels que :*

- *aucun sommet de $\overline{S}$ ne distingue A.*
- *aucun sommet de $\overline{S}$ n'est voisin avec un sommet de B.*

Si S est un split strict, alors les ensembles A et B sont uniques. Dans le cas contraire, n'importe quelle paire d'ensembles (A, B) vérifiant $A \cup B = S$ et $A \cap B = \emptyset$ fonctionne.

Démonstration. Il s'agit d'une conséquence directe de la définition 4.5, point [CDMR12], définition longue.

Concernant l'unicité, si S est un split strict, on sait par définition que ∂S est non-vide. Ainsi, S est non-vide, $\overline{S}$ est non-vide et il existe une arête entre un sommet $x \in S$ et un sommet $y \in \overline{S}$. Supposons que l'on n'ait pas l'unicité de A et B, c'est-à-dire qu'il existe deux partitions A_1, B_1 et A_2, B_2 de S qui sont différentes et qui satisfont tous les deux

les conditions du lemme. Puisque ces partitions sont différentes, il existe un sommet x' qui est dans A_i et dans B_j avec $i \neq j$. Puisque $x' \in B_j$, x' n'est voisin d'aucun sommet de $\overline{S}$ et en particulier y. Ainsi, y est voisin de x mais pas de x', c'est donc un sommet distinguant de A_i, ce qui est une contradiction.

Enfin, si S n'est pas strict, c'est-à-dire si ∂S est vide, vu qu'il n'y a aucune arête entre S et $\overline{S}$, aucun sommet de $\overline{S}$ ne distingue S et aucun sommet de $\overline{S}$ n'est voisin avec un sommet de S, ce qui signifie que tous les sommets de S vérifient à la fois la condition de A et celle de B, ce qui montre que n'importe quelle partition convient. □

Le premier point du lemme 4.6 (« aucun sommet de $\overline{S}$ ne distingue A ») est appelé la *A-uniformité* du split S. Le second point (« aucun sommet de $\overline{S}$ n'est voisin avec un sommet de B ») est appelé la *B-uniformité* du split S. On appelle cette décomposition la *(A, B)-décomposition* du split S.

Cette caractérisation permet d'obtenir une propriété de stabilité supplémentaire :

Proposition 4.7. *Soit G un graphe connexe et soit X et Y deux splits de G qui se croisent ($X \between Y$). Alors $X \Delta Y$ est aussi un split.*

Cela se fait par une disjonction de cas.

Lemme 4.8. *Soit G un graphe connexe et soit X et Y deux splits stricts de G qui se croisent. Soit (A, B) la (A, B)-décomposition de X et soit (A', B') la (A, B)-décomposition de Y. Alors un des quatre cas suivant est vrai :*

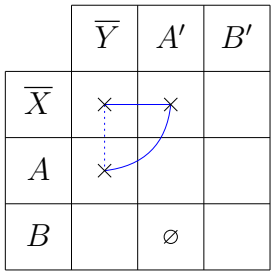
- $\boldsymbol{K}_{\neq}$ *(Cas symétrique non-dégénéré) Il y a une arête entre $\overline{X} \cap \overline{Y}$ et $A \cap A'$, et $A \neq A'$.*
- $\boldsymbol{K}_{=}$ *(Cas symétrique dégénéré) Il y a une arête entre $\overline{X} \cap \overline{Y}$ et $A \cap A'$, et $A = A'$.*
- $\boldsymbol{K}_X$ *(Cas asymétrique X) Il y a une arête entre $\overline{X} \cap \overline{Y}$ et $A' \setminus X$.*
- $\boldsymbol{K}_Y$ *(Cas asymétrique Y) Il y a une arête entre $\overline{X} \cap \overline{Y}$ et $A \setminus Y$.*

Démonstration. Le graphe G est connexe. Il y a donc une arête entre $\overline{X} \cap \overline{Y}$ et le reste du graphe. Cependant, il ne peut pas y avoir d'arête entre $\overline{X}$ et B par B-uniformité de X, et il ne peut pas y avoir d'arête entre $\overline{Y}$ et B' par B-uniformité de Y. Ainsi, l'arête sortant de $\overline{X} \cap \overline{Y}$ ne peut aller que vers l'ensemble $(\overline{B} \cap \overline{B'}) \setminus \overline{X} \cap \overline{Y}$, qui est égal à $(A \cap A') \cup (A' \setminus X) \cup (A \setminus Y)$, d'où les trois cas *symétrique*, *asymétrique X* et *asymétrique Y*. Dans le cas symétrique, on conclut par dichotomie entre $A \neq A'$ et $A = A'$. □

Il est parfois difficile de se représenter ce genre de preuves. Ainsi, j'introduis un outil permettant de suivre visuellement ces preuves. L'idée est qu'un split partitionne les sommets V en trois ensembles A, B et $\overline{X}$. Ainsi, étant donné deux splits X et Y, on a deux partitions différentes de V : d'un côté la partition $(A, B, \overline{X})$ et de l'autre une partition que l'on note $(A', B', \overline{Y})$. Il est naturel de représenter les neuf ensembles issus de l'intersection entre un des trois ensembles de la première partition et un des trois ensembles de la seconde partition. On obtient ce que j'appelle un *plateau d'intersections*. On peut ensuite représenter pour chacun des neuf ensembles s'il est vide ou non-vide, et on peut aussi y représenter des existences et des absences d'arêtes. Un exemple de plateau d'intersections ainsi que la façon dont il se lit sont représentés sur la figure 4.4.

Le but est maintenant de partir de chacun des quatre cas possibles ($\boldsymbol{K}_{\neq}$, $\boldsymbol{K}_{=}$, $\boldsymbol{K}_X$ et $\boldsymbol{K}_Y$) et de voir ce qui peut être déduit comme information, en utilisant notamment les plateaux d'intersections.

Avant de commencer, il peut être utile de voir comment les hypothèses peuvent être utilisées sur le plateau d'intersection. On va considérer les quatre hypothèses suivantes :



Une croix dans une case signifie que l'ensemble correspond est non-vide. Ainsi, dans le plateau d'intersection ci-contre, $\overline{X} \cap \overline{Y} \neq \emptyset$, $\overline{X} \cap A' \neq \emptyset$ et $A \cap \overline{Y} \neq \emptyset$. Un trait entre deux ensembles (non-vides) signifie qu'il existe au moins une arête entre ces deux ensembles. Ici, il existe une arête entre un sommet de $\overline{X} \cap \overline{Y}$ et un sommet de $\overline{X} \cap A'$. De même, il y a une arête entre un sommet de $\overline{X} \cap A'$ et un sommet de $A \cap \overline{Y}$. En revanche, rien n'indique qu'il s'agit du même sommet de $\overline{X} \cap A'$ dans les deux cas. Un trait en pointillés entre deux ensembles signifie qu'il n'y a pas d'arêtes entre ces deux ensembles Ainsi, il n'y a aucune arête entre $A \cap \overline{Y}$ et $\overline{X} \cap \overline{Y}$. Le symbole $\emptyset$ dans une case signifie que l'ensemble correspondant est vide. Ici, $B \cap A' = \emptyset$.

FIGURE 4.4 – Exemple et explication d'un plateau d'intersection.

la B-uniformité d'un split, la connectivité de G, la A-uniformité d'un split et le fait que les deux splits se croisent :

— (B-uniformité) La B-uniformité du split X signifie qu'il n'y a pas d'arête entre B et $\overline{X}$. Sur un plateau d'intersections, cela se traduit par l'absence d'arête entre une case de la ligne $\overline{X}$ et une case de la ligne B. De même, la B-uniformité du split Y se traduit par l'absence d'arête entre une case de la colonne $\overline{Y}$ et une case de la colonne B'. Ainsi, les seules arêtes autorisées sur un plateau d'intersections sont des arêtes entre deux cases voisines.

— (Connectivité) Le fait que le graphe soit connexe signifie qu'étant donné une partition (V_1, V_2) des sommets du graphe, il doit y avoir une arête entre un sommet de V_1 et un sommet de V_2. Sur un plateau d'intersections, en prenant V_1 égal à une seule case c, on déduit qu'il doit y avoir une arête entre c et une des autres cases. Cependant, en utilisant également la propriété de B-uniformité décrite au point précédant, cette autre case doit être une case voisine de c, ce qui limite en général grandement les arêtes possibles.

— (A-uniformité) La A-uniformité du split X signifie que s'il y a une arête entre un sommet v de $\overline{X}$ et un sommet de A, alors il doit y avoir une arête entre v et chaque sommet de A, de sorte à ce que v ne distingue pas A. Sur une plateau d'intersections, cela se traduit directement par le fait que s'il y a une arête entre une case c de la ligne $\overline{X}$ et une case de la ligne A, alors il y a une arête entre la case c et chaque case *non-vide* de la ligne A. Cependant, comme la B-uniformité interdit la présence de certaines arêtes, cette propriété peut être utilisée avec la propriété de B-uniformité pour montrer par l'absurde qu'une case doit être vide ou qu'une arête entre deux cases est impossible.

— (Croisement) Comme les splits X et Y se croisent, certains ensembles de cases ne peuvent pas être tous simultanément vides. On a déjà utilisé cette propriété pour dire que la case $\overline{X} \cap \overline{Y}$ était non-vide. On peut aussi déduire que les quatre cases en bas à droite ne peuvent pas être toutes les quatre vides (parce que $X \cap Y \neq \emptyset$), que les deux cases en bas de la première colonne ne peuvent pas être toutes les deux vides (parce que $X \setminus Y \neq \emptyset$), et que les deux cases les plus à droite de la première ligne ne peuvent pas être toutes les deux vides (parce que $Y \setminus X \neq \emptyset$).

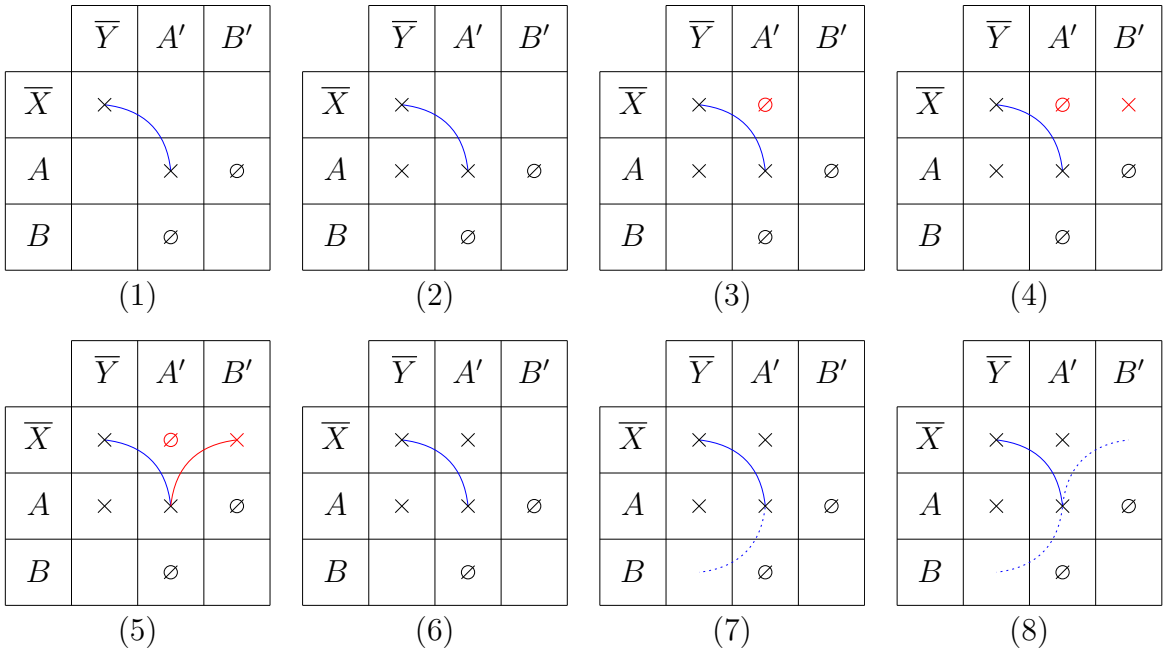


FIGURE 4.5 – Cas correspondant à $\boldsymbol{K}_{\neq}$.

On commence par traiter le cas $\boldsymbol{K}_{\neq}$. Dans ce cas-là, il y a une arête entre entre $\overline{X} \cap \overline{Y}$ et $A \cap A'$, et $A \neq A'$. D'abord, on montre que la case $A \cap B'$ est vide par l'absurde : si elle ne l'est pas, on utilise la propriété de A-uniformité (s'il y a une arête entre une case c et une autre case de la ligne A, alors il y a une arête entre c et chaque case non-vide de la ligne A), il doit y avoir une arête entre $\overline{X} \cap \overline{Y}$ et $A \cap B'$, ce qui est impossible par B-uniformité (il est impossible d'avoir une arête entre la première ligne et la troisième ligne). De même, on montre que $B \cap A'$ est vide, d'où le plateau (1) de la figure 4.5.

Ensuite, comme $A \neq A'$, cela signifie que la case $A \cap \overline{Y}$ ou la case $A' \cap \overline{X}$ est non-vide. On va montrer que si l'une est non-vide, alors l'autre aussi. Sans perte de généralité, on considère que c'est $A \cap \overline{Y}$ qui est non-vide et on suppose que $A' \cap \overline{X}$ est vide, afin de déduire une contradiction. Cela est représenté sur les plateaux (2) et (3). En utilisant la propriété *Croisement*, les deux cases les plus à droite de la première ligne ne peuvent pas être toutes les deux vides. Ainsi, $\overline{X} \cap B'$ est non-vide, comme représenté sur le plateau (4) Finalement, par connectivité, il y a une arête entre $\overline{X} \cap B'$ et $A \cap A'$, comme représenté sur le plateau (5), puis par A-uniformité, il y a une arête entre $\overline{X} \cap B'$ et $A \cap \overline{Y}$, ce qui est interdit par B-uniformité. On aboutit à une contradiction, montrant que si $A \cap \overline{Y}$ est non-vide, alors $A' \cap \overline{X}$ aussi. Un raisonnement similaire montre que si $A' \cap \overline{X}$ est non-vide, alors $A \cap \overline{Y}$ aussi. Ainsi, on obtient le plateau (6).

Enfin, on démontre par l'absurde qu'il n'y a pas d'arête entre $A \cap A'$ et $B \cap \overline{Y}$: s'il y en a une, alors il y a aussi une arête entre $\overline{X} \cap A'$ et $B \cap \overline{Y}$ par A-uniformité, mais une telle arête ne peut exister par B-uniformité, d'où une contradiction et le plateau (7). On raisonne de la même manière pour montrer qu'il n'y a pas d'arête entre $A \cap A'$ et $\overline{X} \cap B'$, d'où le plateau (8).

On traite maintenant le cas $\boldsymbol{K}_{=}$. Dans ce cas-là, il y a une arête entre entre $\overline{X} \cap \overline{Y}$ et $A \cap A'$, et $A = A'$, comme représenté sur le plateau (1) de la figure 4.6. En utilisant la propriété *Croisement*, les deux cases les plus en bas de la première colonne ne peuvent pas être toutes les deux vides. Ainsi, $B \cap \overline{Y}$ est non-vide. Puis, par connectivité, il y a

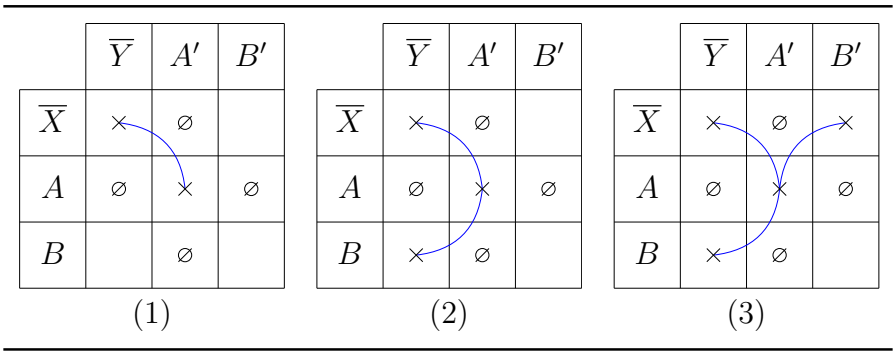


FIGURE 4.6 – Cas correspondant à $\boldsymbol{K}_{=}$.

une arête entre $B \cap \overline{Y}$ et $A \cap A'$, comme représenté sur le plateau (2). Exactement de la même manière, on déduit que $\overline{X} \cap B'$ est non-vide et qu'il y a une arête entre cette case et $A \cap A'$, ce qui permet d'aboutir au plateau (3).

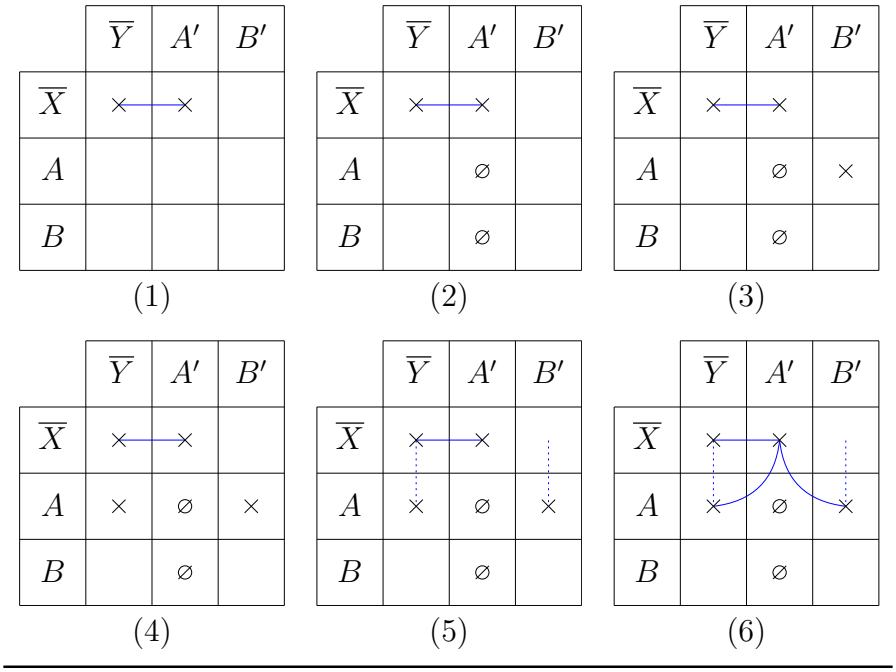


FIGURE 4.7 – Cas correspondant à $\boldsymbol{K}_{X}$.

Enfin, on traite le cas $\boldsymbol{K}_{X}$, c'est-à-dire quand il y a une arête entre $\overline{X} \cap \overline{Y}$ et $A' \setminus X$, comme représenté sur le plateau (1) de la figure 4.7.

D'abord, on peut supposer la case $A \cap A'$ vide. En effet, si elle ne l'est pas, il y a une arête entre elle et $\overline{X} \cap \overline{X'}$ par A-uniformité, ce qui implique que l'on est en fait dans le cas $\boldsymbol{K}_{\neq}$. On montre que la case $B \cap A'$ est vide par l'absurde : si elle ne l'est pas, il y a une arête entre elle et $\overline{X} \cap \overline{X'}$ par A-uniformité, ce qui est impossible par B-uniformité. On obtient alors le plateau (2).

On montre maintenant par l'absurde que $A \cap B'$ est non-vide : si elle l'était, alors $B \cap B'$ ne peut pas être vide par propriété de *Croisement* (qui dit que les quatre cases en bas à droite d'un plateau ne peuvent pas toutes être vides). Ensuite, par Connectivité, il doit y avoir une arête entre la case $B \cap B'$ et l'une des ses cases voisines. Mais celles-ci sont toutes vides, ce qui est donc impossible, d'où la contradiction. On obtient donc le

plateau (3).

D'une manière similaire, on montre par l'absurde que $A \cap \overline{X'}$ est non-vide : si elle l'était, alors $B \cap \overline{Y}$ ne peut pas être vide par propriété de *Croisement* (qui dit aussi que les deux cases en bas de la première colonne ne peuvent pas être toutes les deux vides). Ensuite, par Connectivité, il doit y avoir une arête entre la case $B \cap \overline{Y}$ et l'une des ses cases voisines. Mais celles-ci sont toutes vides, ce qui est donc impossible, d'où la contradiction. On obtient alors le plateau (4).

On montre maintenant par l'absurde qu'il n'y a pas d'arête entre les cases $\overline{X} \cap B'$ et $A \cap B'$: s'il y en a une, par A-uniformité, il y a une arête entre $\overline{X} \cap B'$ et $A \cap \overline{Y}$, ce qui est impossible par B-uniformité. On montre de la même manière qu'il n'y a pas d'arête entre $\overline{X} \cap \overline{Y}$ et $A \cap \overline{Y}$, d'où le plateau (5).

Finalement, afin de connecter la première ligne du plateau au reste, on a nécessairement la présence de deux arêtes supplémentaires : une entre les cases $\overline{X} \cap A'$ et $A \cap \overline{Y}$, et une entre les cases $\overline{X} \cap A'$ et $A \cap B'$. On obtient alors le plateau (6).

En inversant le rôle des deux splits du cas $\boldsymbol{K}_X$, on obtient le cas $\boldsymbol{K}_Y$. Finalement, les quatre cas possibles sont représentés sur la figure 4.8.

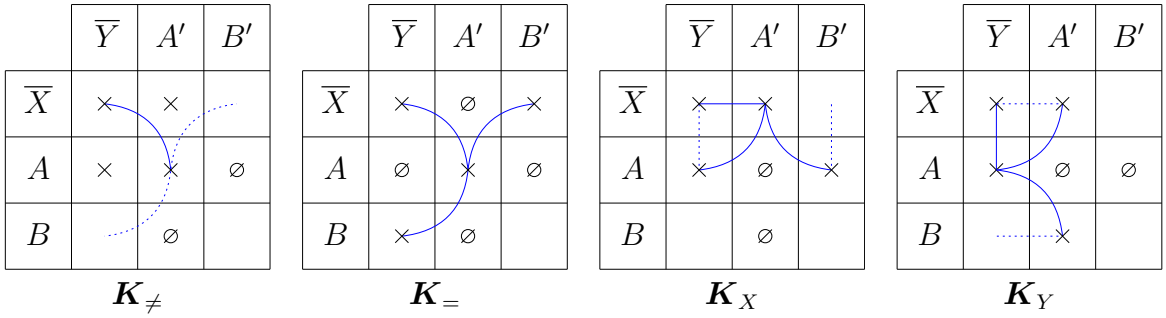


FIGURE 4.8 – Les quatre cas possibles.

À partir de ces quatre cas, on peut enfin démontrer la proposition 4.7.

Démonstration (de la proposition 4.7). Soit G un graphe connexe et soit X et Y deux splits de G qui se croisent, de (A, B)-décomposition respective (A, B) et (A', B'). On traite les quatre cas possibles. Dans chacun des cas, on commence par définir deux ensembles A_Δ et B_Δ, puis on montre que $X \,\Delta\, Y$ (ou $\overline{X \,\Delta\, Y}$) est un split de (A, B)-décomposition (A_Δ, B_Δ). Pour cela, il suffit de montrer la A-uniformité et la B-uniformité de $X \,\Delta\, Y$ (ou de $\overline{X \,\Delta\, Y}$). Pour montrer la A-uniformité, on procède toujours de la même manière : on suppose qu'il existe un sommet qui distingue A_Δ et on aboutit à l'existence d'un sommet qui distingue A ou A', ce qui constitue une contradiction.

— ($\boldsymbol{K}_{\neq}$) On pose $A_\Delta = A \,\Delta\, A'$ et $B_\Delta = B \,\Delta\, B'$. D'abord, on remarque que $A_\Delta = (A \cap \overline{Y}) \cup (\overline{X} \cap A')$ car les ensembles $A \cap B'$ et $B \cap A'$ sont vides. On suppose qu'il existe un sommet $s \notin X \,\Delta\, Y$ qui distingue A_Δ. L'ensemble $\overline{X \,\Delta\, Y}$ représente 5 cases du plateau (celle en haut à gauche et les quatre en bas à droite), mais s ne peut se trouver que dans $\overline{X} \cap \overline{Y}$ ou $A \cap A'$. En effet, les cases $A \cap B'$ et $B \cap A'$ sont vides, et la case $B \cap B'$ ne peut pas avoir d'arête vers $A \,\Delta\, A'$ par B-uniformité. Par définition d'un sommet distinguant, il existe $x, y \in A_\Delta$ tels que (s, x) est une arête et (s, y) non. On se retrouve avec quatre cas possibles :

1. $s \in \overline{X} \cap \overline{Y}$ et x et y sont tous les deux (sans perte de généralité) dans $A \cap \overline{Y}$.

2. $s \in \overline{X} \cap \overline{Y}$ et (sans perte de généralité), $x \in A \cap \overline{Y}$ et $y \in \overline{X} \cap A'$.
3. $s \in A \cap A'$ et x et y sont tous les deux (sans perte de généralité) dans $A \cap \overline{Y}$.
4. $s \in A \cap A'$ et (sans perte de généralité), $x \in A \cap \overline{Y}$ et $y \in \overline{X} \cap A'$.

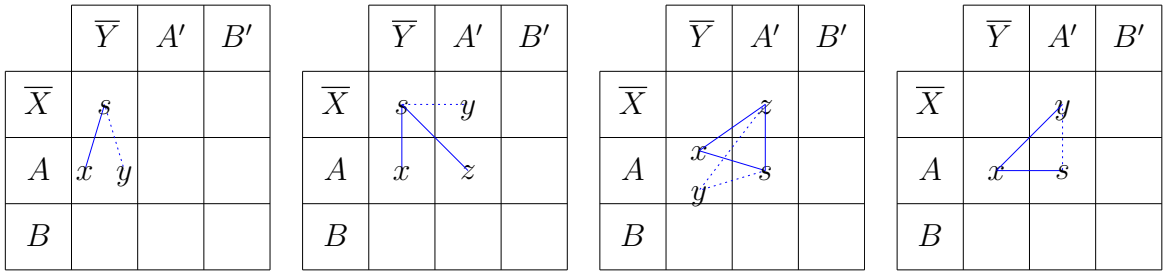


FIGURE 4.9 – Les quatre cas de $\boldsymbol{K}_{\neq}$.

La figure 4.9 illustre les quatre cas. Dans le cas 1, s est un sommet distinguant de Y, ce qui est impossible. Dans le cas 2, on prend un sommet z dans $A \cap A'$ qui est non-vide. L'arête entre s et x implique qu'il y a une arête entre s et z par A-uniformité, ce qui fait de s un sommet distinguant de Y à cause de de y et z. Dans le cas 3, on prend un sommet z dans $\overline{X} \cap A'$ qui est non-vide. L'arête entre x et s implique une arête entre x et z par A-uniformité, qui implique à son tour une arête entre z et s par A-uniformité. De même, la non-arête entre y et s implique une non-arête entre y et z par A-uniformité, ce qui fait de z un sommet distinguant de X à cause de y et s. Dans le cas 4, l'arête entre x et s implique une arête entre x et y par A-uniformité, ce qui fait de y un sommet distinguant de X à cause de x et s.

Pour la B-uniformité, le plateau montre qu'il n'y a pas d'arête entre B_Δ et $\overline{X \,\Delta\, Y}$.

— ($\boldsymbol{K}_{=}$) On pose $A_\Delta = A = A'$ et $B_\Delta = (B \cap B') \cup (\overline{X} \cap \overline{Y})$. On montre ici que c'est le complémentaire de $X \,\Delta\, Y$ qui a (A_Δ, B_Δ) pour (A, B)-décomposition. Pour la A-uniformité de $\overline{X \,\Delta\, Y}$, soit $s \in X \,\Delta\, Y$ un sommet distinguant de A_Δ. Par symétrie et sans perte de généralité, on peut supposer que $s \in B \cap \overline{Y}$. On a immédiatement que s distingue A' car $A_\Delta = A'$ et $s \notin Y$.

Pour la B-uniformité, il n'y a pas d'arête entre les cases placées aux quatre coins du plateau par la propriété de B-uniformité de X et Y.

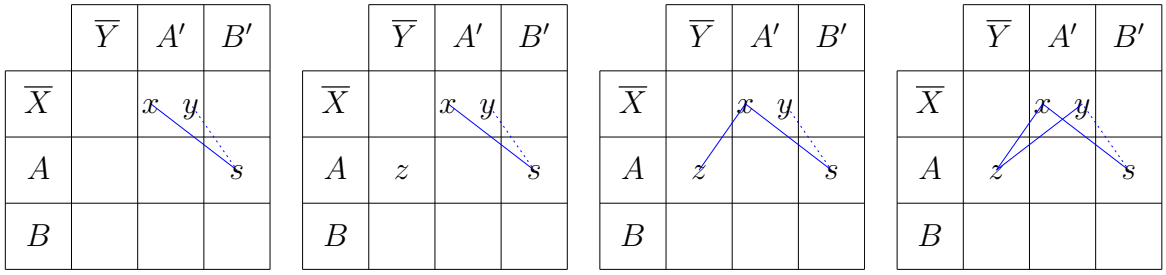


FIGURE 4.10 – Schéma de la preuve du cas $\boldsymbol{K}_X$.

— ($\boldsymbol{K}_X$) On pose $A_\Delta = A'$ et $B_\Delta = X \Delta B'$. Pour la A-uniformité, soit $s \notin X \Delta Y$ un sommet distinguant de A_Δ. Le sommet s ne peut pas appartenir à $\overline{X} \cap \overline{Y}$ car il s'agirait aussi d'un sommet distinguant de X et Y. Il ne peut pas appartenir à $A \cap A'$ ou $B \cap A'$ car ces ensembles sont vides. Il ne peut pas non plus appartenir à $B \cap B'$ par B-uniformité. Ainsi, $s \in A \cap B'$ et il existe x et y appartenant à $A_\Delta = \overline{X} \cap A'$, comme montré sur le plateau (1) de la figure 4.10. Soit $z \in A \cap \overline{Y}$ qui est non-vide. L'arête entre x et s implique l'existence d'une arête entre x et z par A-uniformité. Puis l'arête entre z et x implique l'existence d'une arête entre z et y par A-uniformité. Cependant, cela fait de y un sommet distinguant de A à cause de z et s.

Pour la B-uniformité,

— ($\boldsymbol{K}_Y$) On traite ce cas comme le cas $\boldsymbol{K}_X$, en inversant le rôle des deux splits. □

Point de vue mixte

Au final, les splits qui se croisent sont stables par union, intersection, différence et différence symétrique. Une question que l'on peut se poser est si l'est possible de démontrer toutes ces propriétés avec un seul point de vue. Avec le point de vue graphe, c'est possible en utilisant la disjonction de cas du lemme 4.8. Avec le point de vue algébrique, nous ne sommes pas parvenu à prouver la stabilité par différence symétrique.

4.2.3 Famille symétrique traversante

Dans le but d'abstraire les splits, on peut définir la notion de *famille symétrique traversante*, de sorte à ce que l'ensemble des splits d'un graphe connexe en soit une.

Définition 4.8 (Famille symétrique traversante, [EG77])**.** *Soit E un ensemble et soit $\mathcal{F}$ une famille de sous-ensembles de E. On dit que $\mathcal{F}$ est symétrique traversante quand :*

— *(basée) l'ensemble $\emptyset$ et les singletons $\{x\}$ pour tout $x \in E$ appartiennent à $\mathcal{F}$,*

— *(symétrique) si X appartient à $\mathcal{F}$ alors $E \setminus X$ appartient à $\mathcal{F}$,*

— *(traversante) si X et Y sont deux ensembles appartenant à $\mathcal{F}$ tels que $X \bigcirc\!\!\!\!\bigcirc_\times Y$, alors les ensembles $X \cup Y$, $X \cap Y$, $X \setminus Y$ et $Y \setminus X$ appartiennent aussi à $\mathcal{F}$.*

De la même manière que pour les modules, il existe des définitions alternatives. Par exemple, [FJ99] demande à ce que les ensembles $\emptyset$ et E n'appartiennent pas à une famille symétrique traversante. Il existe aussi des façons équivalentes de caractériser la condition « traversante ». Dans [FJ99], cette condition est seulement « si X et Y sont deux ensembles appartenant à $\mathcal{F}$ tels que $X \bigcirc\!\!\!\!\bigcirc_\times Y$, alors les ensembles $X \cup Y$ et $X \cap Y$ appartiennent aussi à $\mathcal{F}$ ». En effet, grâce à la condition « symétrique », si X et Y sont deux ensembles appartenant à $\mathcal{F}$ tels que $X \bigcirc\!\!\!\!\bigcirc_\times Y$, alors X et $\overline{Y} = E \setminus Y$ aussi, donc $X \cap \overline{Y} = X \setminus Y$ appartient bien à $\mathcal{F}$. En réalité, il est possible de simplifier encore davantage cette condition :

Lemme 4.9. *Soit E un ensemble et soit $\mathcal{F}$ une famille de sous-ensembles de E. Alors $\mathcal{F}$ est une famille symétrique traversante si et seulement si elle vérifie les conditions suivantes :*

— *(basée) l'ensemble $\emptyset$ et les singletons $\{x\}$ pour tout $x \in E$ appartiennent à $\mathcal{F}$,*

— *(symétrique) si X appartient à $\mathcal{F}$ alors $E \setminus X$ appartient à $\mathcal{F}$,*

— *(union jointe) si X et Y sont deux ensembles appartenant à $\mathcal{F}$ tels que $X \cap Y \neq \emptyset$, alors l'ensemble $X \cup Y$ appartient aussi à $\mathcal{F}$.*

Démonstration. On démontre ce qui diffère entre la définition 4.8 et cette caractérisation.

($\Longrightarrow$) Il suffit de considérer deux ensembles X et Y et de démontrer que $X \cup Y$ appartient à $\mathcal{F}$, même si on n'a que l'hypothèse $X \cap Y \neq \emptyset$ à la place de $X \bigcirc\!\!\!\!\times Y$. Si $X \setminus Y = \emptyset$, alors $X \cup Y = Y \in \mathcal{F}$. Si $Y \setminus X = \emptyset$, alors $X \cup Y = X \in \mathcal{F}$. Si $\overline{X \cup Y} = \emptyset$, alors $X \cup Y = E \in \mathcal{F}$. Si aucune de ces trois conditions n'est vraie, et vu que l'on a l'hypothèse $X \cap Y \neq \emptyset$, alors X et Y vérifient $X \bigcirc\!\!\!\!\times Y$ par définition et on a donc bien $X \cup Y \in \mathcal{F}$ via la propriété « traversante ».

($\Longleftarrow$) Soit deux ensembles X et Y tels que $X \bigcirc\!\!\!\!\times Y$. On montre que $X \cup Y$, $X \cap Y$, $X \setminus Y$ et $Y \setminus X$ appartiennent aussi à $\mathcal{F}$. Comme $X \bigcirc\!\!\!\!\times Y$ implique $X \cap Y \neq \emptyset$, on a déjà $X \cup Y \in \mathcal{F}$ via la propriété « union jointe ».

Pour montrer $X \cap Y \in \mathcal{F}$, on remarque que $X \bigcirc\!\!\!\!\times Y$ implique $\overline{X \cup Y} \neq \emptyset$, c'est-à-dire $\overline{X} \cap \overline{Y} \neq \emptyset$. Ainsi, par propriété « union jointe », $\overline{X} \cup \overline{Y} \in \mathcal{F}$, c'est-à-dire que $\overline{X \cap Y} \in \mathcal{F}$, et donc $X \cap Y \in \mathcal{F}$ par propriété « symétrique ».

D'un manière similaire, on montre $X \setminus Y \in \mathcal{F}$: $X \bigcirc\!\!\!\!\times Y$ implique $Y \setminus X \neq \emptyset$, c'est-à-dire $\overline{X} \cap Y \neq \emptyset$. Par propriété « union jointe », on a $\overline{X} \cup Y = \overline{X \setminus Y} \in \mathcal{F}$, puis $X \setminus Y \in \mathcal{F}$ par propriété « symétrique ». Le fait que $Y \setminus X \in \mathcal{F}$ se montre de la même manière en inversant les rôles de X et Y. □

La caractérisation des familles symétriques traversantes du lemme 4.9 m'a permis de bien les généraliser. En effet, pour généraliser ces familles, il faut notamment généraliser la propriété « traversante ». Or, cela n'a pas mené à des résultats concluants. En revanche, en partant de la propriété « union disjointe » (qui est équivalente à la propriété « traversante » pour les familles symétriques traversantes) et en la généralisant, on obtient un autre genre de famille qu'en généralisant la propriété « traversante ». Cette nouvelle famille est définie dans le chapitre 5, définition 5.4 et a mené à des résultats concluants.

4.2.4 Famille sans croisement

Parmi les familles symétriques traversantes, il en existe une catégorie particulière, appelées *familles sans croisement* qui ont la particularité de vérifier trivialement la propriété « traversante », dans le sens où pour tout ensemble X et Y de la famille, X et Y ne se croisent pas.

Définition 4.9 (Famille sans croisement [EG77])**.** *Soit E un ensemble et soit $\mathcal{F}$ une famille de sous-ensembles de E. On dit que $\mathcal{F}$ est sans croisement quand :*

— *(basée) l'ensemble $\emptyset$ et les singletons $\{x\}$ pour tout $x \in E$ appartiennent à $\mathcal{F}$,*

— *(symétrique) si X appartient à $\mathcal{F}$ alors $E \setminus X$ appartient à $\mathcal{F}$,*

— *(trivialement traversante) si X et Y sont deux ensembles appartenant à $\mathcal{F}$, alors $X \perp_\times Y$ (X et Y ne se croisent pas).*

Dans ce contexte, ce n'est pas équivalent de demander que la propriété « union disjointe » soit vérifiée trivialement (c'est-à-dire que $X \cap Y = \emptyset$ pour tout X et Y de la famille). En effet, si une famille $\mathcal{F}$ est symétrique et vérifie $X \cap Y = \emptyset$ pour tout X et Y, alors $\mathcal{F}$ doit être vide ou de la forme $\mathcal{F} = \{X, \overline{X}\}$, c'est-à-dire n'avoir que deux éléments.

Ainsi, bien que les propriétés « traversante » et « union disjointe » sont équivalentes en ce qui concerne la définition des familles symétriques traversantes, elles ne le sont pas dès qu'il s'agit de définir d'autres familles connexes :

— pour généraliser les familles symétriques traversantes, il vaut mieux utiliser la propriété « union disjointe »,

— mais pour restreindre les familles symétriques traversantes aux familles sans croisement, il vaut mieux utiliser la propriété « traversante ».

Une famille sans croisement contient au plus un nombre linéaire d'élément (c'est-à-dire que si $|E| = n$, alors une famille sans croisement sur E contient au plus $\mathcal{O}(n)$ éléments) et peut ainsi être représentée par un arbre [EG77, BXHR12].

Proposition 4.10. *[EG77] Soit $\mathcal{F}$ une famille sans croisement sur E. Il existe un (unique) arbre T sur E tel que :*

— *les feuilles de T soient les éléments de E,*

— *un ensemble X appartient à $\mathcal{F}$ si, et seulement si, il existe un nœud u de T telle que X soit l'ensemble des feuilles de zéro, un, tous sauf un ou tous les sous-arbres connexes de T obtenus en enlevant le nœud u.*

En général, on trouve plutôt la deuxième condition formulée : un ensemble X qui n'est ni vide ni E appartient à $\mathcal{F}$ si, et seulement si, il existe *une arête* e de T telle que X soit l'ensemble des feuilles d'un des deux sous-arbres de T obtenu en enlevant l'arête e. On se convainc facilement que cette formulation est équivalente. Cependant, elle présente certains inconvénients. Déjà, l'ensemble vide et l'ensemble E ne peuvent pas être représentés. De plus, afin de pouvoir généraliser par le suite, le fait d'indiquer clairement les quantités « zéro », « un », « tous sauf un » et « tous » induit une généralisation naturelle, qui consiste à changer les quantités autorisées. Par exemple, on peut autoriser n'importe quelle quantité de sous-arbres ou n'autoriser que les quantités inférieures à un nombre fixé.

Exemple 4.2. La figure 4.11a représente un exemple d'arbre T correspondant à une famille $\mathcal{F}$ sans croisement sur un ensemble $E = \{a, b, c, d, e, f, g\}$. On peut retrouver la famille $\mathcal{F}$ grâce à la proposition 4.10. On considère chaque nœud interne (ici, u_1, u_2, u_3 et u_4), on le retire de l'arbre, on regarde les différents sous-arbres connexes formés, et chaque ensemble de $0, 1, d-1$ ou d sous-arbres correspond à un élément de $\mathcal{F}$, où d est le degré du nœud enlevé. Par exemple, en enlevant u_1, on obtient la figure 4.11b. On a trois sous-arbres connexes : l'arbre ayant a comme feuille, celui ayant b comme feuille et celui ayant c, d, e, f, g comme feuilles. Ainsi, les ensembles suivants font partie de $\mathcal{F}$:

— l'ensemble $\emptyset$ en tant qu'union de zéro sous-arbre,

— les ensembles $\{a\}$, $\{b\}$ et $\{c, d, e, f, g\}$ en tant qu'union d'un seul sous-arbre,

— les ensembles $\{a, b\}$, $\{b, c, d, e, f, g\}$ et $\{a, c, d, e, f, g\}$ en tant qu'union de tous les sous-arbres sauf un

— et l'ensemble E en tant qu'union de tous les sous-arbres.

En enlevant u_3, on obtient la figure 4.11c, dans laquelle les sous-arbres connexes ont pour ensemble de feuilles $\{a, b\}$, $\{c, d\}$, $\{e\}$ et $\{f, g\}$. Les ensembles suivants font donc partie de $\mathcal{F}$:

— l'ensemble $\emptyset$ en tant qu'union de zéro sous-arbre,

— les ensembles $\{a, b\}$, $\{c, d\}$, $\{e\}$ et $\{f, g\}$ en tant qu'union d'un seul sous-arbre,

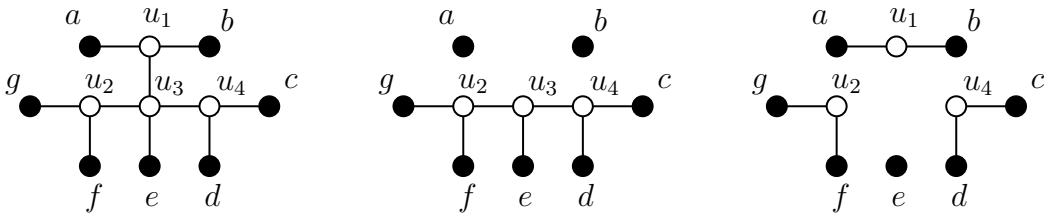
— les quatre ensembles $\{a, b, c, d, e\}$, $\{c, d, e, f, g\}$, $\{a, b, e, f, g\}$ et $\{a, b, c, d, f, g\}$ en tant qu'union de tous les sous-arbres sauf un

— et l'ensemble E en tant qu'union de tous les sous-arbres.

En faisant de même pour u_2 et u_4, on obtient finalement que l'ensemble $\mathcal{F}$ est égal à l'union :

— de tous les sous-ensembles de $0, 1, n-1$ et n éléments de E, où $n = |E|$

— et des ensembles $\{a, b\}$, $\{c, d\}$, $\{f, g\}$ ainsi que de leur complémentaire.



(a) Arbre T. (b) T sans u_1. (c) T sans u_3.

FIGURE 4.11 – Les feuilles sont en noir et sont indexées par des éléments de $E = \{a, b, c, d, e, f, g\}$. Les nœuds internes sont en blancs et sont nommés u_1, u_2, u_3, u_4.

Étant donné une famille symétrique traversante $\mathcal{F}$, il est possible d'étendre cette représentation en arbre. Pour cela, on a besoin de la *famille sans croisement induite par* $\mathcal{F}$. Il s'agit d'une famille obtenue en ne gardant que les éléments de $\mathcal{F}$ qui ne croisent aucun autre élément. Ainsi, cette famille est garantie d'être sans croisement.

Définition 4.10 (Famille sans croisement induite)**.** *Soit $\mathcal{F}$ une famille symétrique traversante. La famille sans croisement induite par $\mathcal{F}$ est la famille $\{X \in \mathcal{F} \mid \forall Y \in \mathcal{F}, X \perp_\times Y\}$.*

La propriété forte est qu'une famille symétrique traversante $\mathcal{F}$ peut être décomposée en arbre et que cet arbre est le même que celui de sa famille sans croisement induite, auquel on ajoute une annotation et un ordre sur chaque nœud. J'ai repris la dénomination des annotations de la thèse de Bùi Xuân [BX08]. On trouve par exemple une autre dénomination chez [CE80].

Proposition 4.11. *[CE80] Soit $\mathcal{F}$ une famille symétrique traversante sur E. Il existe un arbre T tel que :*

— *les feuilles de T soient les éléments de E,*

— *chaque nœud interne est annoté par* premier, dégénéré *ou* circulaire,

— *chaque nœud interne possède un ordre circulaire de ses voisins,*

— *un ensemble X appartient à $\mathcal{F}$ si et seulement si il existe un nœud u de T tel que l'un des cas suivants soit vérifié :*

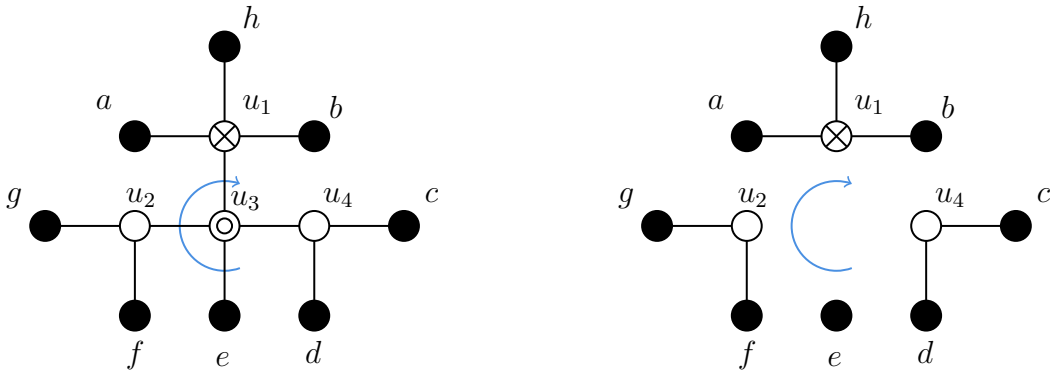
  — *u est annoté par* premier *et X est l'ensemble des feuilles de zéro, un, tous sauf un ou tous les sous-arbres connexes de T obtenu en enlevant le nœud u.*

  — *u est annoté par* complet *et X est l'ensemble des feuilles de* n'importe quel nombre *de sous-arbres connexes de T obtenus en enlevant le nœud u.*

  — *u est annoté par* circulaire *et X est l'ensemble des feuilles de n'importe quel nombre de sous-arbres connexes de T* qui sont contigus dans l'ordre circulaire autour de *u et obtenus en enlevant le nœud u.*

De plus, cet arbre est unique si on s'autorise à changer les annotations des nœuds de degré 3 et à inverser les ordres circulaires des arêtes autour des nœuds.

Le fait d'autoriser le changement d'annotation pour les nœuds de degré 3 vient du fait que pour ces nœuds, les trois annotations « premier », « complet » et « circulaire » sont équivalentes : changer l'annotation ne change pas les ensembles de sous-arbres connexes que l'on peut choisir.



(a) Arbre T. (b) T sans u_3.

FIGURE 4.12 – Les feuilles sont en noir et sont indexées par des éléments de $E = \{a, b, c, d, e, f, g, h\}$. Les nœuds internes sont en blancs et sont nommés u_1, u_2, u_3 et u_4. L'étiquette de chaque nœud interne est représentée par le motif au sein du nœud : vide pour un nœud premier, avec une croix pour un nœud complet et avec deux rond concentriques pour un nœud circulaire. Tacitement, l'ordre circulaire des voisins d'un nœud interne est l'ordre qui tourne dans le sens horaire autour du nœud, de sorte que les composantes contigües selon cet ordre soient celles qui sont côte à côte sur la figure. La flèche bleue explicite cet ordre pour u_3.

Exemple 4.3. La figure 4.12a représente un exemple d'arbre T correspondant à une famille $\mathcal{F}$ symétrique traversante sur un ensemble $E = \{a, b, c, d, e, f, g, h\}$. On peut retrouver la famille $\mathcal{F}$ grâce à la proposition 4.11. On considère chaque nœud interne (ici, u_1, u_2, u_3 et u_4), on le retire de l'arbre et on regarde les différents sous-arbres connexes formés. En fonction de l'étiquette du nœud considéré, certaines unions de sous-arbres connexes correspondent à un élément de $\mathcal{F}$. Par exemple, en enlevant u_3, on obtient la figure 4.11c. On a quatre sous-arbres connexes : l'arbre ayant a, b, h comme feuilles, celui ayant c, d comme feuilles, celui ayant e et celui ayant f, g. Ainsi, les ensembles suivants font partie de $\mathcal{F}$:

— l'ensemble $\emptyset$ en tant qu'union de zéro sous-arbre,

— les ensembles $\{a, b, h\}$, $\{c, d\}$, $\{e\}$ et $\{f, g\}$ en tant qu'union d'un seul sous-arbre,

— les ensembles $\{a, b, c, d, h\}$, $\{c, d, e\}$, $\{e, f, g\}$ et $\{a, b, f, g, h\}$ en tant qu'union de deux sous-arbres *contigus suivant l'ordre des nœuds de l'arbre*,

— les ensembles $\{a, b, c, d, e, h\}$, $\{c, d, e, f, g\}$, $\{a, b, e, f, g, h\}$ et $\{a, b, c, d, f, g, h\}$ en tant qu'union de tous les sous-arbres sauf un

— et l'ensemble E en tant qu'union de tous les sous-arbres.

En enlevant u_1, on obtient les sous-arbres connexes qui ont pour ensemble de feuilles $\{a\}$, $\{b\}$, $\{h\}$ et $\{c, d, e, f, g\}$. Les ensembles suivants font donc partie de $\mathcal{F}$:

— l'ensemble $\emptyset$ en tant qu'union de zéro sous-arbre,

— les ensembles $\{a\}$, $\{b\}$, $\{h\}$ et $\{c, d, e, f, g\}$ en tant qu'union d'un seul sous-arbre,

— les ensembles $\{a, b\}$, $\{a, h\}$, $\{b, h\}$, $\{a, c, d, e, f, g\}$, $\{b, c, d, e, f, g\}$ et $\{c, d, e, f, g, h\}$ en tant qu'union de deux sous-arbres,

— les ensembles $\{b, c, d, e, f, g, h\}$, $\{a, c, d, e, f, g, h\}$, $\{a, b, c, d, e, f, g\}$ et $\{a, b, h\}$ en tant qu'union de tous les sous-arbres sauf un

— et l'ensemble E en tant qu'union de tous les sous-arbres.

En faisant de même pour u_2 et u_4, on obtient finalement que l'ensemble $\mathcal{F}$ est égal à l'union :

— de tous les sous-ensembles de $0, 1, n-1$ et n éléments de E, où $n = |E|$

— et des ensembles $\{a, b\}$, $\{a, h\}$, $\{b, h\}$, $\{a, b, h\}$, $\{c, d\}$, $\{c, d, e\}$, $\{e, f, g\}$, $\{f, g\}$ ainsi que de leur complémentaire.

L'arbre de décomposition d'une famille symétrique traversante $\mathcal{F}$ est appelé *l'arbre de décomposition d'Edmonds-Giles* de $\mathcal{F}$, en référence à [EG77] qui présentait un arbre de décomposition pour les familles sans croisement.

Cependant, on peut déduire que l'ensemble des splits d'un graphe connexe est une famille symétrique traversante. En effet, on sait que l'ensemble vide et les singletons sont des splits. De plus, le complémentaire d'un split est un split au vu de la symétrie de sa définition (cf. définition 4.5). Enfin, la section 4.2.2 a permis d'établir le fait que dans un graphe connexe, si X et Y sont deux splits tels que $X \bigcirc\!\!\!\!\bigcirc_{\times} Y$, alors les ensembles $X \cup Y$, $X \cap Y$, $X \setminus Y$ et $Y \setminus X$ sont aussi des splits. On a même pu obtenir la propriété supplémentaire que $X \Delta Y$ est aussi un split.

Lemme 4.12. *L'ensemble des splits d'un graphe connexe forme une famille symétrique traversante.*

Étant donné qu'une famille symétrique traversante peut être représentée par un arbre, il est naturel de vouloir représenter l'ensemble des splits dans un arbre. Comme les splits ont une propriété supplémentaire de stabilité par rapport aux familles symétriques traversantes, à savoir que la différence symétrique de deux splits qui se croisent est aussi un split, l'arbre de décomposition des splits est plus rigide. Notamment, le cas d'un nœud circulaire n'est plus possible [CE80]. En revanche, il est possible de représenter à l'intérieur de cet arbre, en plus de l'ensemble des splits de G, la structure même de G. Cela forme *l'arbre de décomposition de Cunningham* [GP12].

Chapitre 5

Une nouvelle structure de graphe : Les r-splits

Sommaire

Dans ce chapitre, j'utiliserai les concepts définis dans la section 2.3. Dans la section 5.1, je commence par introduire les r-splits qui sont les objets principaux de ce chapitre. Il s'agit d'une généralisation des splits étudiés dans le chapitre 4. Je présente ensuite quelques propriétés de stabilité des r-splits qui généralisent celles des splits. Ces propriétés invitent à considérer l'ensemble des r-splits d'un graphe donné, comme cela était fait pour les splits. On obtient alors l'*hypergraphe des r-splits*. On abstrait cette notion afin d'obtenir une classe d'hypergraphe $\mathbb{K}_r(n)$, de la même façon que la notion de famille de splits avait été abstraite en la notion de famille symétrique traversante (cf. section 4.2.3). Celle-ci permet de représenter les r-splits d'un graphe. Enfin, en suivant ce qui a été fait avec les splits, on introduit une notion d'orthogonalité et d'hypergraphe sans r-croisement. Ces notions permettent d'obtenir un certain nombre de théorèmes qui sont tous une généralisation de ce qui a été obtenu pour les splits. Tout d'abord, il est possible de représenter l'intégralité des r-splits d'un graphe dans une structure de taille polynomiale (cf. section 5.2) et de calculer cette structure en temps polynomial (cf. section 5.4). De plus, on montre que cette structure est quasiment optimale en exposant une borne inférieure pour les hypergraphes sans r-croisement.

5.1 Introduction

Dans cette section, on reprend le cheminement de la décomposition en splits et on la généralise. On introduit ainsi les r-splits et on présente diverses notions et propriétés qui étendent celles des splits.

5.1.1 Définitions

Un split peut être défini comme une coupe de rang au plus 1 (cf. définition 4.5). Cette définition a l'avantage de pouvoir être généralisée facilement et naturellement. Ainsi, j'introduis la notion de r-split en tant que coupe de rang au plus r. On rappelle que le range d'une coupe X est le rang de la matrice d'adjacence de $G[X, \overline{X}]$ (cf définition 2.19).

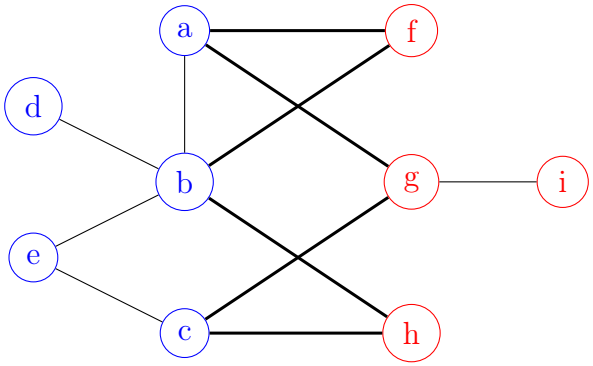
Définition 5.1 (r-split)**.** *Soit G un graphe et X un ensemble de sommets de G. X est un* r-split *si $\rho(X) \leq r$, c'est-à-dire si la coupe $(X, \overline{X})$ a un rang d'au plus r.*

Exemple 5.1. Dans le graphe de la figure 5.1, l'ensemble $\{a, b, c, d, e\}$ est un 2-split. C'est aussi un r-split pour tout $r \geq 2$. En revanche, ce n'est ni un 1-split ni un 0-split.

Correspondance avec les voisinages

Le but est d'interpréter la définition des r-splits en termes de voisinages, c'est-à-dire d'obtenir une propriété de la forme « un ensemble X est un r-split si, et seulement si, il existe r voisinages de X tels qu'une propriété soit vérifiée. » Dans cette propriété, un voisinage de X est un ensemble de sommets de $\overline{X}$. Dans le cas des splits (donc des 1-splits), on peut reformuler la définition 4.5 en « un ensemble X est un split si, et seulement si, il existe un voisinage N de X tel que tout sommet de X ait pour voisinage extérieur N ou $\emptyset$. » On rappelle que le voisinage extérieur d'un sommet x de X est $N(x) \cap \overline{X}$. Dans le cas des r-splits, on a la caractérisation suivante :

Lemme 5.1. *Un ensemble X est un r-split si, et seulement si, il existe r voisinages $N_1, \ldots, N_r$ de X tels que tout sommet de X a pour voisinage extérieur une union disjointe de n'importe quel nombre de voisinages $N_1, \ldots, N_r$.*



$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{array} \begin{array}{c} \begin{array}{ccccccccc} a & b & c & d & e & f & g & h & i \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 1 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$A(G)[\{a,b,c,d,e\},\{f,g,h,i\})] = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccc} f & g & h & i \end{array} \\ \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \end{array}$$

FIGURE 5.1 – Un graphe ayant une coupe de rang 2. Les sommets bleus représentent un côté de la coupe et les sommets rouges l'autre côté. Les arêtes en gras sont les arêtes qui traversent la coupe. La matrice d'adjacence du graphe est représentée en utilisant le même code couleur. Ainsi, l'ensemble $\{a,b,c,d,e\}$ est un 2-split.

Une union disjointe de n'importe quel nombre de voisinages $N_1,\ldots,N_r$ est de la forme $N = N_{i_1} \,\Delta\, N_{i_2} \,\Delta\, \ldots \,\Delta\, N_{i_k}$, avec $0 \leq k \leq r$. Cela signifie qu'un sommet appartient à N si, et seulement si, il appartient à un nombre impair de voisinages parmi $N_{i_1}, N_{i_2}, \ldots, N_{i_k}$.

Démonstration. Si X est un r-split, la matrice d'adjacence A de $G[X,\overline{X}]$ a un rang d'au plus r. Il existe donc r vecteurs $v_1,\ldots,v_r$ de $\mathbb{F}_2$ dont les sommets sont indexés par $\overline{X}$ tels que toute colonne de A s'écrit comme une combinaison linéaire de ces vecteurs. Comme cette combinaison linéaire s'effectue dans $\mathbb{F}_2$, les seuls coefficients possibles sont 0 et 1, et l'addition est effectuée modulo 2. Soit N_i le sous-ensemble de sommets de $\overline{X}$ tel que $y \in N_i$ si et seulement si le coefficient correspondant à y dans v_i est égal à 1. Il reste à montrer que chaque sommet x de X a un voisinage extérieur $N(x)\cap\overline{X}$ qui peut s'exprimer comme l'union disjointe de n'importe quel nombre de voisinages $N_1,\ldots,N_r$. Soit x un sommet de X et v le vecteur correspondant à la colonne de A indexée par x. On sait que v peut s'écrire comme la combinaison linéaire dans $\mathbb{F}_2$ de $v_1,\ldots,v_r$, c'est-à-dire qu'il existe $v_{i_1},\ldots,v_{i_k}$ tels que $v = v_{i_1} + \ldots + v_{i_k} \pmod 2$. Ainsi, on a la suite d'équivalences $y \in N(x)\cap\overline{X} \iff v[y] = 1 \iff v_{i_1}[y] + \ldots + v_{i_k}[y] = 1 \iff y \in N_{i_1} \,\Delta\, \ldots \,\Delta\, N_{i_k}$.

On utilise un raisonnement similaire pour montrer l'autre sens. S'il existe r voisinages $N_1,\ldots,N_r$ de X tel que tout sommet de X ait pour voisinage extérieur une union disjointe de n'importe quel nombre de voisinages $N_1,\ldots,N_r$, alors on définit v_i pour $1 \leq i \leq r$ tel que v_i soit indexé par les sommets de $\overline{X}$ et tel que $v_i[y] = 1$ si $y \in N_i$ et 0 sinon. Soit v le vecteur correspondant à une colonne de la matrice d'adjacence A de $G[X,\overline{X}]$ et soit x l'indice de v dans A. On montre que v s'écrit comme une combinaison linéaire dans $\mathbb{F}_2$ de $v_1,\ldots,v_r$, c'est-à-dire qu'il existe $v_{i_1},\ldots,v_{i_k}$ tels que $v = v_{i_1} + \ldots + v_{i_k} \pmod 2$. On sait que pour chaque x, il y a des voisinages extérieurs $N_{i_1},\ldots,N_{i_k}$ tels que

$N(x) \cap \overline{X} = N_{i_1} \Delta \ldots \Delta N_{i_k}$. Par définition des v_i, on a donc bien $v = v_{i_1} + \ldots + v_{i_k} \pmod 2$. □

5.1.2 Propriétés de base

Tout d'abord, deux propriétés découlent directement de la définition d'un r-split et des propriétés du rang de coupe ρ.

Lemme 5.2. *Soit G un graphe et X un ensemble de sommets.*

1. *Si $|X| \leq r$ alors X et $\overline{X}$ sont des r-splits.*
2. *Si X est un r-split alors $V(G) \setminus X$ est aussi un r-slit.*

Démonstration. On montre les deux points :

1. Comme le rang d'une matrice est toujours inférieur ou égal à son nombre de lignes et à son nombre de colonnes, on sait que $\rho(X) \leq \min(|X|, |\overline{X}|)$. Ainsi, X et $\overline{X}$ sont des r-splits.
2. D'après la proposition 2.13, on sait que $\rho(\overline{X}) = \rho(X)$, d'où le résultat. □

On voit dans cette preuve que $\rho(X) \leq \min(|X|, |\overline{X}|)$. Cela nous motive à nous intéresser au cas où cette inégalité est en réalité une égalité.

Définition 5.2 (r-split trivial)**.** *Un r-split X est dit* trivial *si $\rho(X) = \min(|X|, |\overline{X}|)$.*

En d'autres termes, le r-split X est trivial si la matrice d'adjacence de $G[X, \overline{X}]$ a rang maximal (dans $\mathbb{F}_2$).

On sait de plus que la fonction ρ est une fonction de connectivité (cf. proposition 2.13). Cela signifie en particulier que $\rho(X \cap Y) + \rho(X \cup Y) \leq \rho(X) + \rho(Y)$. On aimerait utiliser cette propriété pour montrer que l'union et l'intersection de deux r-splits sont aussi des r-splits. En effet, si X et Y sont des r-splits, cela signifie que $\rho(X) \leq r$ et $\rho(Y) \leq r$. Ainsi, $\rho(X \cap Y) + \rho(X \cup Y) \leq \rho(X) + \rho(Y) \leq 2r$. Pour pouvoir conclure, il faudrait que cette borne $\rho(X \cap Y) + \rho(X \cup Y) \leq 2r$ soit répartie équitablement entre $\rho(X \cap Y)$ et $\rho(X \cup Y)$ (de façon à déduire $\rho(X \cap Y) \leq r$ et $\rho(X \cup Y) \leq r$), ce qui signifie qu'on ne pourrait pas avoir par exemple $\rho(X \cap Y) \geq r + 1$ et $\rho(X \cup Y) \leq r - 1$. Pour ce faire, on va imposer que $\rho(X \cap Y) \geq r$ et $\rho(X \cup Y) \geq r$ sous certaines conditions. Ainsi, si on a $\rho(X \cap Y) + \rho(X \cup Y) \leq 2r$ et $\rho(X \cap Y) \geq r$ par exemple, alors on déduit que $\rho(X \cup Y) \leq r$.

On peut résoudre ce problème en se concentrant sur la motivation de l'introduction des r-splits Un des buts des 2-splits est de pouvoir décomposer un graphe qui est premier pour les 1-splits, c'est-à-dire qui ne contient aucun 1-split (excepté ceux qui sont triviaux). De même, on peut imaginer que les r-splits permettent de décomposer les graphes qui sont premiers pour les $(r-1)$-splits, c'est-à-dire qui ne contiennent aucun $(r-1)$-split (excepté ceux qui sont triviaux). Cela signifie que l'on peut supposer sans trop perdre de généralité que notre graphe G ne contient aucun $(r-1)$-split. Cette condition s'appelle la *r-rang connectivité*. Elle est définie formellement de la façon suivante :

Définition 5.3 (r-rang connecté [Oum23])**.** *Un graphe G est* r-rang connecté *si pour tout $m \leq r$, l'inégalité $\rho(X) < m$ implique l'inégalité $\min(|X|, |\overline{X}|) < m$.*

On montre facilement que cette définition est équivalente à la caractérisation suivante :

Lemme 5.3. *Un graphe G est* r-rang connecté *si et seulement si chaque $(r-1)$-split est trivial.*

Démonstration. Pour alléger les notations, on note $t(X) = \min(|X|, |\overline{X}|)$.

($\Rightarrow$) Soit G un graphe r-rang connecté. Soit X un $(r-1)$-split. Ainsi, On a $\rho(X) < r$. Soit $m = \rho(X) + 1$. On a donc $\rho(X) = m - 1 < m$. Par r-rang connectivité, on a $t(X) < m$ et donc $t(X) \leq \rho(X)$. Cependant, pour chaque ensemble X, on sait que $\rho(X) \leq t(X)$. Cela signifie donc que $t(X) = k$ et donc que X est trivial.

($\Leftarrow$) Soit G un graphe tel que chaque $(r-1)$-split est trivial (c'est-à-dire que $t(X) = \rho(X)$). Soit $m \leq r$ et X tels que $\rho(X) < m$. Ainsi, $\rho(X) < r$ et X est un $(r-1)$-split. Finalement, $t(X) = \rho(X) < m$, ce qui signifie que G est r-rang connecté. □

Bien que la caractérisation d'un graphe r-rang connecté porte sur les $(r-1)$-splits triviaux, on peut déduire une propriété qui caractérise les r-splits triviaux d'un graphe r-rang connecté.

Lemme 5.4. *Soit G un graphe r-rang connecté et X un ensemble de sommets de G. Alors X est un r-split trivial si et seulement si* $\min(|X|, |\overline{X}|) \leq r$.

Démonstration. Soit G un graphe r-rang connecté et X un ensemble de sommets de G.

($\Rightarrow$) On sait que X est un r-split trivial. Le côté trivial signifie que $\rho(X) = \min(|X|, |\overline{X}|)$. De plus, par définition d'un r-split, on a $\rho(X) \leq r$. Ainsi, $\min(|X|, |\overline{X}|) \leq r$. On remarque que cette partie de la preuve reste vraie même si G n'est pas r-rang connecté.

($\Leftarrow$) On sait que la taille de X vérifie $\min(|X|, |\overline{X}|) \leq r$. De plus, on sait que la fonction ρ vérifie $\rho(X) \leq \min(|X|, |\overline{X}|)$, ce qui montre que $\rho(X) \leq r$ et donc que X est un r-split. Si $\rho(X) = r$, alors $\rho(X) = \min(|X|, |\overline{X}|) = r$ et X est un r-split trivial. Sinon, $\rho(X) < r$, ce qui signifie que X est en fait un $(r-1)$-split. Puisque G est r-rang connecté, X est trivial. C'est donc un $(r-1)$-split trivial et à fortiori un r-split trivial. □

Réciproquement, dans un graphe r-rang connecté, les ensembles X dont la taille vérifie $r \leq |X| \leq |V| - r$ ont forcément un rang de coupe $\rho(X)$ qui vérifie $\rho(X) \geq r$.

Lemme 5.5. *Soit G un graphe r-rang connecté et X un ensemble de sommets de G vérifiant $r \leq |X| \leq |V| - r$. Alors $\rho(X) \geq r$.*

Démonstration. Par l'absurde, supposons que $\rho(X) < r$. Ainsi, X est un $(r-1)$-split. Par r-rang connectivité, X est trivial. Par définition d'être trivial, $\rho(X) = \min(|X|, |\overline{X}|)$. Ainsi, $\min(|X|, |\overline{X}|) < r$, ce qui contredit le fait que $r \leq |X| \leq |V| - r$. □

Ainsi, on a montré que dans un graphe r-rang connecté, dès qu'on a un ensemble de sommets X dont la taille vérifie $r \leq |X| \leq |V| - r$, on peut déduire une borne inférieure sur le rang de coupe de X. C'est exactement le genre de condition que l'on cherchait plus haut et qui permettait de conclure sur la stabilité des r-splits par union et intersection.

Lemme 5.6. *Si X et Y sont deux r-splits d'un graphe r-rang connecté G et si $|X \cap Y| \geq r$, alors $X \cup Y$ est aussi un r-split.*

Démonstration. On a trois cas possibles :

1. Si $|X \cup Y| \leq r$ ou $|\overline{X \cup Y}| \leq r$, alors $\rho(X \cup Y) \leq \min(|X \cup Y|, |\overline{X \cup Y}|) \leq r$, ce qui signifie que $X \cup Y$ est un r-split.
2. Si $|\overline{X \cap Y}| \leq r$, alors on a $|\overline{X \cup Y}| \leq |\overline{X \cap Y}| \leq r$ et on est ramené au premier cas, ce qui montre que $X \cup Y$ est un r-split.

3. Sinon, le nombre de sommets de $X \cup Y$ et $X \cap Y$ sont tous les deux entre r et $|V| - r$, car on a traité le cas où $|X \cup Y|$ n'est pas dans cette fourchette dans le premier cas, on a traité le cas où $|X \cap Y|$ est trop grand dans le deuxième cas et on sait que $|X \cap Y|$ ne peut être trop petit par hypothèse. Ainsi, d'après le lemme 5.5, $\rho(X \cup Y) \geq r$ et $\rho(X \cap Y) \geq r$. Vu que X et Y sont des r-splits, on sait que $\rho(X) \leq r$ et $\rho(Y) \leq r$. Par sous-modularité, on déduit que $\rho(X \cup Y) + \rho(X \cap Y) \leq 2r$. Ainsi, $\rho(X \cup Y) = \rho(X \cap Y) = r$ et $X \cup Y$ est un r-split. □

Ce lemme permet d'étendre la propriété de stabilité des splits aux r-splits.

Corollaire 5.7. *Soit G un graphe r-rang connecté et soit X et Y deux r-splits de G. On a :*

— *Si $|X \cap Y| \geq r$, alors $X \cup Y$ est un r-split.*

— *Si $|X \cup Y| \geq r$, alors $X \cap Y$ est un r-split.*

— *Si $|X \setminus Y| \geq r$, alors $Y \setminus X$ est un r-split.*

— *Si $|Y \setminus X| \geq r$, alors $X \setminus Y$ est un r-split*

Démonstration. Le premier point est prouvé par le lemme 5.6. En remplaçant X par $\overline{X}$ et/ou Y par $\overline{Y}$, on obtient les trois autres points en utilisant le fait que ρ est symétrique. □

Lien avec les splits

On rappelle que la définition des r-splits est telle qu'un 1-split est un split. Prenons alors $r = 1$ dans le corollaire 5.7 et vérifions que l'on retrouve bien la proposition 4.5, qui disait que dans un graphe connexe, si deux splits se croisent, alors leur union, leur intersection et leurs différences sont aussi des splits.

Tout d'abord, un graphe G est connexe si, et seulement si, il est 1-rang connecté. En effet, être 1-rang connecté signifie d'après le lemme 5.3 que tout 0-split est trivial, c'est-à-dire que toute coupe $(X, \overline{X})$ de rang 0 vérifie $\min(X, \overline{X}) = 0$. Or, une coupe de rang 0 est une coupe telle que la matrice d'adjacence de $G[X, \overline{X}]$ est nulle, c'est-à-dire qu'il n'y a pas d'arête entre X et $\overline{X}$, c'est-à-dire que cette coupe est faite entre deux parties de G qui ne sont pas connectées. De plus, les seules coupes qui vérifient $\min(X, \overline{X}) = 0$ sont $(V(G), \emptyset)$ et $(\emptyset, V(G))$. Ainsi, un graphe est 1-rang connecté s'il n'a pas deux parties qui sont déconnectées, c'est-à-dire si le graphe est connexe.

Ensuite, deux splits X et Y se croisent si $|X \cap Y| \geq 1$, $|X \cup Y| \geq 1$, $|X \setminus Y| \geq 1$ et $|Y \setminus X| \geq 1$. On remarque que cela correspond bien aux conditions du corollaire 5.7, mais que toutes ces conditions sont imposées simultanément dans le cas des splits, alors qu'elles sont demandées séparément dans le cas des r-splits. On peut donc penser qu'en utilisant le corollaire 5.7 avec $r = 1$, on peut déduire davantage de splits qu'en utilisant la proposition 4.5. Il se trouve que non. Dès qu'une condition parmi $|X \cap Y| \geq 1$, $|X \cup Y| \geq 1$, $|X \setminus Y| \geq 1$ et $|Y \setminus X| \geq 1$ n'est pas vérifiée, les ensembles qui peuvent être déduits comme split parmi $X \cap Y$, $X \cup Y$, $X \setminus Y$ et $Y \setminus X$ via le corollaire 5.7 peuvent être déduits comme split via d'autres propriétés. Par exemple, si on a bien $|X \cup Y| \geq 1$, $|X \setminus Y| \geq 1$ et $|Y \setminus X| \geq 1$ mais que $|X \cap Y| = 0$, on déduit via le corollaire 5.7 avec $r = 1$ que $X \cap Y$, $X \setminus Y$ et $Y \setminus X$ sont des splits (ce qu'on n'aurait pas pu déduire via la proposition 4.5). Cependant, $X \cap Y = \emptyset$ par hypothèse, $X \setminus Y = X$ et $Y \setminus X = Y$. On a donc déduit que l'ensemble vide, X et Y sont des splits, ce que l'on savait déjà.

Finalement, le corollaire 5.7 pris avec $r = 1$ est bien équivalent à la proposition 4.5.

5.1.3 L'hypergraphe des r-splits

Soit G un graphe r-rang connecté qui a pour ensemble de sommets $V(G) = [n]$. Notons $H_r(G)$ l'hypergraphe dont les sommets sont les mêmes que G (c'est-à-dire que $V(H_r(G)) = [n]$) et dont l'ensemble des hyper-arêtes $\mathcal{E}(H_r(G))$ est l'ensemble des r-splits de G (c'est-à-dire les ensembles de sommets $X \subseteq V(G)$ tels que $\rho(X) \leq r$). D'après les lemmes 5.2 et 5.6, on sait que $H_r(G)$ satisfait les propriétés suivantes : (1) si $A \in \mathcal{E}(H_r(G))$, alors $V(G) \setminus A \in \mathcal{E}(H_r(G))$; (2) pour chaque ensemble de sommets $X \subseteq V(G)$, si $|X| \leq r$, alors $X \in \mathcal{E}(H_r(G))$; (3) si $A, B \in \mathcal{E}(H_r(G))$ et si $|A \cap B| \geq r$, alors $A \cup B \in \mathcal{E}(H_r(G))$. Ainsi, on considère les hypergraphes qui vérifient ces propriétés.

Dans cette section, on ne travaillera qu'avec des hypergraphes dont l'ensemble des sommets est $V(H) = [n]$. Ainsi, on identifie l'hypergraphe H avec l'ensemble des ses hyper-arêtes $\mathcal{E}(H)$. Cela signifie par exemple que l'on écrira indifféremment $A \in \mathcal{E}(H)$ ou $A \in H$ pour désigner l'hyper-arête A de H. De même, on notera $\{A, B\}$ l'hypergraphe H tel que $V(H) = [n]$ et $\mathcal{E}(H) = \{A, B\}$.

Définition 5.4 (Notation $\mathbb{K}_r(n)$)**.** *Soit $\mathbb{K}_r(n)$ la classe des hypergraphes H ayant pour ensemble de sommets $V(H) = [n]$, tels que l'ensemble des hyper-arêtes $\mathcal{E}(H)$ vérifie les trois règles suivantes :*

$\boldsymbol{R}_{\leq}$: *Pour chaque ensemble de sommets $X \subseteq V(H)$, si $|X| \leq r$, alors $X \in \mathcal{E}(H)$.*

$\boldsymbol{R}_{\neg}$: *Si $A \in \mathcal{E}(H)$, alors $V(H) \setminus A \in \mathcal{E}(H)$.*

$\boldsymbol{R}_{\cup}$: *Si $A, B \in \mathcal{E}(H)$ et $|A \cap B| \geq r$, alors $A \cup B \in \mathcal{E}(H)$.*

En combinant les trois règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$, on déduit les trois propriétés suivantes :

Lemme 5.8. *Soit H un hypergraphe appartenant à la classe $\mathbb{K}_r(n)$. Alors H vérifie les trois propriétés suivantes :*

$\boldsymbol{P}_{\geq}$: *Pour chaque ensemble de sommets $X \subseteq V(H)$, si $|X| \geq n - r$, alors $X \in H$.*

$\boldsymbol{P}_{\cap}$: *Si $A, B \in H$ et $|\overline{A \cup B}| \geq r$, alors $A \cap B \in H$.*

$\boldsymbol{P}_{\setminus}$: *Si $A, B \in H$ et $|A \setminus B| \geq r$, alors $B \setminus A \in H$.*

Démonstration. La propriété $\boldsymbol{P}_{\geq}$ est obtenue en combinant les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$, tandis que les propriétés $\boldsymbol{P}_{\cap}$ et $\boldsymbol{P}_{\setminus}$ sont obtenues en combinant les règles $\boldsymbol{R}_{\cup}$ et $\boldsymbol{R}_{\neg}$ plusieurs fois chacune.

Tout d'abord, montrons la propriété $\boldsymbol{P}_{\geq}$. Soit X un ensemble de sommets tel que $|X| \geq n - r$. Ainsi, $\overline{X}$ vérifie $|\overline{X}| \leq r$, et d'après la règle $\boldsymbol{R}_{\leq}$, on a $\overline{X} \in H$. Enfin, en appliquant la règle $\boldsymbol{R}_{\neg}$ à $\overline{X} \in H$, on a $X \in H$.

Ensuite, montrons la propriété $\boldsymbol{P}_{\cap}$. Soit $A, B \in H$ tels que $|\overline{A \cup B}| \geq r$. D'après la règle $\boldsymbol{R}_{\neg}$, $\overline{A} \in H$ et $\overline{B} \in H$. On sait que $|\overline{A} \cap \overline{B}| = |\overline{A \cup B}| \geq r$, donc d'après la règle $\boldsymbol{R}_{\cup}$, $\overline{A} \cup \overline{B} = \overline{A \cap B} \in H$. Enfin, en appliquant la règle $\boldsymbol{R}_{\neg}$ à nouveau à l'ensemble $\overline{A \cap B} \in H$, on obtient que $A \cap B \in H$.

De la même manière, on montre la propriété $\boldsymbol{P}_{\setminus}$. Soit $A, B \in H$ tels que $|A \setminus B| \geq r$. D'après la règle $\boldsymbol{R}_{\neg}$, $\overline{B} \in H$. On sait que $|A \cap \overline{B}| = |A \setminus B| \geq r$, donc d'après la règle $\boldsymbol{R}_{\cup}$, $A \cup \overline{B} = \overline{B \setminus A} \in H$. Enfin, en appliquant la règle $\boldsymbol{R}_{\neg}$ à nouveau à l'ensemble $\overline{B \setminus A} \in H$, on obtient que $B \setminus A \in H$. □

De plus, l'union de plus de deux hyper-arêtes de H peut aussi être une hyper-arête de H lorsque certaines conditions sont vérifiées à propos de leurs intersections.

Lemme 5.9. *Soit $H \in \mathbb{K}_r(n)$ et soit $A_1, \ldots, A_k$ des hyper-arêtes de H. Si pour chaque $1 \leq i < k$ on a $|A_i \cap A_{i+1}| \geq r$, alors $\bigcup_{i=1}^{k} A_i$ est aussi une hyper-arête de H.*

Démonstration. On montre ce lemme par récurrence. Pour $k = 1$ il n'y a rien à prouver et pour $k = 2$, le lemme correspond à la règle $\boldsymbol{R}_\cup$. Supposons que le lemme soit vrai pour k hyper-arête et prouvons qu'il l'est pour $k+1$. Soit $A' = A_1 \cup \ldots \cup A_k$. A' est une hyper-arête de H par hypothèse de récurrence. Il reste à montrer que $A' \cup A_{k+1}$ est une hyper-arête de H. Puisque $A_k \subseteq A'$, on a $A_k \cap A_{k+1} \subseteq A' \cap A_{k+1}$, ce qui signifie que $|A' \cap A_{k+1}| \geq |A_k \cap A_{k+1}| \geq r$. Enfin, d'après la règles $\boldsymbol{R}_\cup$, $A' \cup A_{k+1}$ est une hyper-arête de H. □

On rappelle que l'on a défini la classe $\mathbb{K}_r(n)$ de sorte que pour chaque graphe r-rang connecté G d'ordre n, l'hypergraphe $H_r(G)$ constitué de tous les r-splits de G appartient à la classe $\mathbb{K}_r(n)$.

La classe $\mathbb{K}_r(n)$ a la propriété d'être une *famille de Moore* (cf. définition 2.16). Cela signifie donc que : (1) l'hypergraphe $2^{[n]}$ qui possède toutes les hyper-arête appartient à $\mathbb{K}_r(n)$; (2) si on prend deux hypergraphes $H_1, H_2 \in \mathbb{K}_r(n)$, alors leur intersection $H_1 \cap H_2$ est aussi dans $\mathbb{K}_r(n)$. On rappelle que l'intersection de deux hypergraphes H_1 et H_2 qui ont le même ensemble de sommets est l'hypergraphe $H_1 \cap H_2$ tel que $V(H_1 \cap H_2) = V(H_1) = V(H_2)$ et tel que $\mathcal{E}(H_1 \cap H_2) = \mathcal{E}(H_1) \cap \mathcal{E}(H_2)$. Cette définition d'intersection est naturelle quand on identifie un hypergraphe et son ensemble d'hyper-arêtes.

Lemme 5.10. *La classe $\mathbb{K}_r(n)$ est une famille de Moore.*

Démonstration. Tout d'abord, comme la définition 5.4 demande uniquement à ce que certaines hyper-arêtes soient présentes (et ne demande pas à ce que des hyper-arêtes soient absentes), on en déduit directement que l'hypergraphe $2^{[n]}$ qui contient toutes les hyper-arêtes est dans la classe $\mathbb{K}_r(n)$. Ensuite, montrons que si $H_1, H_2 \in \mathbb{K}_r(n)$, alors $H_1 \cap H_2 \in \mathbb{K}_r(n)$. Pour ce faire, soit A, B deux hyper-arêtes de $H_1 \cap H_2$ et soit X un ensemble de sommets de $V(H_1) = V(H_2)$. Montrons que ces hyper-arêtes vérifient les règles $\boldsymbol{R}_\leq, \boldsymbol{R}_\neg, \boldsymbol{R}_\cup$ de la définition 5.4 :

- Pour $\boldsymbol{R}_\leq$: Si $|X| \leq r$, alors X est une hyper-arête de H_1 d'après la règle $\boldsymbol{R}_\leq$ appliquée à X de $H_1 \in \mathbb{K}_r(n)$ et X est une hyper-arête de H_2 d'après la règle $\boldsymbol{R}_\leq$ appliquée à X de $H_2 \in \mathbb{K}_r(n)$. Ainsi, X est une hyper-arête de $H_1 \cap H_2$.
- Pour $\boldsymbol{R}_\neg$: Puisque A est une hyper-arête de $H_1 \cap H_2$, A est aussi une hyper-arête de H_1, et $\overline{A}$ est aussi une hyper-arête de H_1 d'après la règle $\boldsymbol{R}_\neg$ appliquée à A de $H_1 \in \mathbb{K}_r(n)$. De même, $\overline{A}$ est une hyper-arête de H_2. Ainsi, $\overline{A}$ est une hyper-arête de $H_1 \cap H_2$.
- Pour $\boldsymbol{R}_\cup$: Si $|A \cap B| \geq r$, alors $A \cup B$ est une hyper-arête de H_1d'après la règle $\boldsymbol{R}_\cup$ appliquée à A de $H_1 \in \mathbb{K}_r(n)$. De même, $A \cup B$ est une hyper-arête de H_2. Ainsi, $A \cup B$ est une hyper-arête de $H_1 \cap H_2$.

Finalement, $H_1 \cap H_2$ satisfait bien la définition 5.4, ce qui montre que $H_1 \cap H_2 \in \mathbb{K}_r(n)$. □

Le principal intérêt d'avoir une famille de Moore est que cela induit un *opérateur de clôture*, comme énoncé dans le lemme 2.5. Dans notre cas, l'opérateur de clôture associé est le suivant :

Lemme 5.11. *Soit H un hypergraphe avec $V(H) = [n]$ pour ensemble de sommets. La* clôture *de H dans $\mathbb{K}_r(n)$, notée $<H>_r$, est l'hypergraphe qui est l'intersection de tous*

les hypergraphes qui contiennent H et qui appartiennent à $\mathbb{K}_r(n)$. Un hypergraphe H tel que $<H>_r = H$ est appelé un hypergraphe clos pour $<\cdot>_r$*, un hypergraphe r-clos, ou simplement un hypergraphe clos quand il n'y a aucune ambigüité. Un hypergraphe à n sommets est clos si, et seulement si, il appartient à $\mathbb{K}_r(n)$*

Démonstration. Il s'agit d'une conséquence directe du deuxième point du lemme 2.5. □

En d'autres termes, un ensemble de sommets A est une hyper-arête de $<H>_r$ si et seulement si A est une hyper-arête de chaque hypergraphe de $\mathbb{K}_r(n)$ qui a H comme sous-hypergraphe. Afin de mieux appréhender l'opérateur de clôture $<\cdot>_r$, regardons un exemple.

Exemple 5.2. Prenons $n = 8$ et $r = 2$. Soit H l'hypergraphe ayant pour ensemble de sommets $V(H) = [8]$ et pour ensemble d'hyper-arêtes $\mathcal{E}(H) = \{\{1,2,3\},\{2,3,4,5\}\}$. L'hypergraphe H et sa clôture $<H>_r$ sont représentés sur la figure 5.2. Dans ce cas, l'hypergraphe $<H>_r$ est constitué des hyper-arêtes suivantes :

— tous les ensembles de 0, 1 ou 2 sommets, d'après la règle $\boldsymbol{R}_{\leq}$,

— tous les ensembles de 6, 7 ou 8 sommets, d'après la propriété $\boldsymbol{P}_{\geq}$,

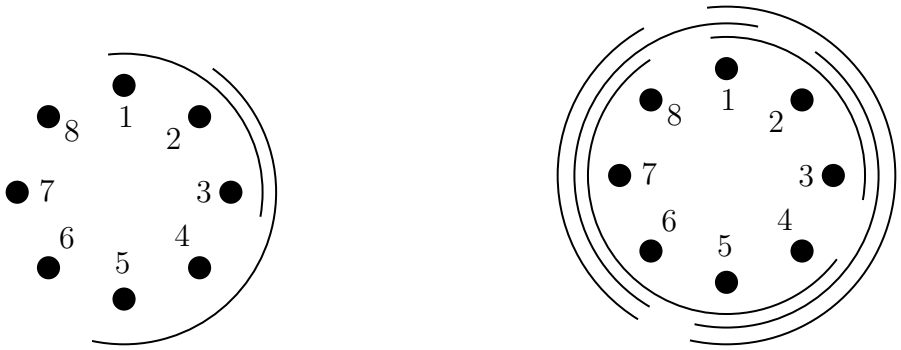
— $\{1,2,3\}$ et $\{2,3,4,5\}$, car H est un sous-hypergraphe de $<H>_r$,

— $\{4,5,6,7,8\}$ et $\{1,6,7,8\}$, d'après la règle $\boldsymbol{R}_{\neg}$ appliquée à chacun des ensembles du point précédent,

— $\{1,2,3,4,5\}$ d'après la règle $\boldsymbol{R}_{\cup}$ appliquée aux ensembles $\{1,2,3\}$ et $\{2,3,4,5\}$,

— $\{6,7,8\}$ d'après la règle $\boldsymbol{R}_{\neg}$ appliquée à $\{1,2,3,4,5\}$.

On prouve ainsi facilement que $<H>_r$ contenaient nécessairement toutes ces hyper-arêtes. Pour montrer que ce sont les seules, il suffit de vérifier que cet ensemble d'hyper-arêtes satisfait les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$.



(a) Hypergraphe H.

(b) Hypergraphe $<H>_r$. Toutes les hyper-arêtes ayant 0, 1, 2, 6, 7 ou 8 sommets sont présentes mais ne sont pas représentées.

FIGURE 5.2 – Hypergraphe H à 8 sommets et sa clôture pour $r = 2$. Chaque hyper-arête est représentée par un arc de cercle qui longe les sommets qu'elle contient.

Par définition d'un opérateur de clôture (cf. définition 2.14), $<\cdot>_r$ est *extensif* (pour tout hypergraphe H, on a $H \subseteq <H>_r$), *monotone* (pour tout hypergraphes H, H', si $H \subseteq H'$, alors $<H>_r \subseteq <H'>_r$) et *idempotent* (pour tout hypergraphe $H \in \mathbb{K}_r(n)$, on a $<H>_r = H$).

L'idée derrière cette opérateur de clôture est de pouvoir déduire toute l'information d'un hypergraphe clos H à partir d'un plus petit hypergraphe H' en considérant la clôture

$<H'>_r$ de H'. Dans ce cas, on dit que H' *représente* H, dans le sens où on peut obtenir toute l'information de H via H'.

Définition 5.5 (*H' représente H*)**.** *Si H et H' sont deux hypergraphes tels que $<H'>_r = H$, on dit que H'* représente *H.*

5.1.4 Hypergraphes sans r-croisement

On veut maintenant généraliser la notion d'orthogonalité définie dans le cas des familles partitives (cf. définition 4.2) et des familles symétriques traversantes (cf. définition 4.7). Cela permettra de définir des hyper-arêtes r-orthogonales ainsi que des hypergraphes sans r-croisement. La difficulté a été de comprendre comment généraliser cette notion. En effet, des ensembles orthogonaux sont des ensembles disjoints ou inclus et des ensembles croix-orthogonaux sont des ensembles A et B tels qu'au moins l'un des quatre ensembles $A \cap B$, $A \cup B$, $A \setminus B$, $A \setminus B$ soit vide.

Le point essentiel qui permet de mieux comprendre ces notions est celui selon lequel ces ensembles ont comme propriété fondamentale de ne pas contribuer à leur famille, dans le sens suivant. Dans une famille partitive, si on prend deux arêtes A et B de la famille, alors on peut déduire de nouvelles arêtes (comme $A \cup B$, $A \cap B$, etc.). Cependant, si A et B sont orthogonales, alors on ne peut déduire aucune nouvelle arête, car il est demandé explicitement à A et B de se chevaucher (c'est-à-dire de ne pas être orthogonales) pour pouvoir déduire de nouvelles arêtes. De même, dans le cas des familles symétriques traversantes, il faut que les arêtes se croisent (c'est-à-dire qu'elles ne soient pas croix-orthogonales) pour pouvoir déduire de nouvelles arêtes. C'est cette idée qui a permis de développer une notion d'hyper-arêtes r-orthogonales qui fonctionne, c'est-à-dire qui puisse être utilisée pour démontrer des théorèmes similaires à ceux démontrés dans le cas des familles partitives ou symétriques traversantes.

Dans le cas des hypergraphes clos, c'est la règle $\boldsymbol{R}_\cup$ qui permet de construire de nouvelles hyper-arêtes à partir de deux hyper-arêtes qui s'intersectent en au moins r sommets. Cependant, on sait que combinée à la règle $\boldsymbol{R}_\neg$, cette règle permet de déduire les propriétés $\boldsymbol{P}_\cap$ et $\boldsymbol{P}_\setminus$, avec lesquelles on peut construire de nouvelles hyper-arêtes. Afin de ne pas oublier de façon de construire de nouvelles hyper-arêtes, on demande explicitement à ce qu'il soit impossible de construire une nouvelle hyper-arêtes si on omet la règle $\boldsymbol{R}_\cup$.

Ainsi, étant donné deux hyper-arêtes A et B telles que $A \cup B$ soit aussi une hyper-arête. Soit $A \cup B$ peut être obtenu en utilisant uniquement les règles $\boldsymbol{R}_\leq$ et $\boldsymbol{R}_\neg$, soit la règle $\boldsymbol{R}_\cup$ est nécessaire pour construire $A \cup B$. Dans le premier cas, on dit que les hyper-arêtes A et B sont r-orthogonales. Dans le deuxième cas, on dit qu'elles se r-croisent. Pour formaliser ces idées, on introduit une nouvelle classe d'hypergraphes en utilisant uniquement les règles $\boldsymbol{R}_\leq$ et $\boldsymbol{R}_\neg$:

Définition 5.6 (Notation $\mathbb{K}_r^\circ(n)$)**.** *Soit $\mathbb{K}_r^\circ(n)$ la classe des hypergraphes H dont l'ensemble des sommets est $V(H) = [n]$ et qui vérifient :*

$\boldsymbol{R}_\leq$: *Pour chaque ensemble de sommets $X \subseteq V(H)$, si $|X| \leq r$, alors $X \in H$.*

$\boldsymbol{R}_\neg$: *Si $A \in H$, alors $\overline{A} \in H$.*

Un hypergraphe de la classe $\mathbb{K}_r^\circ(n)$ vérifie donc les deux mêmes premières règles qu'un hypergraphe dans $\mathbb{K}_r(n)$, mais n'a pas besoin de vérifier la troisième règle $\boldsymbol{R}_\cup$. Tout comme la classe $\mathbb{K}_r(n)$, la classe $\mathbb{K}_r^\circ(n)$ est une famille de Moore.

Lemme 5.12. *La classe* $\mathbb{K}_r^\circ(n)$ *est une famille de Moore.*

Démonstration. La preuve est la même que celle du lemme 5.10, excepté qu'il n'y a pas besoin de prouver le point qui concerne la règle $\boldsymbol{R}_\cup$. □

On a donc l'opérateur de clôture suivant :

Lemme 5.13. *Soit* H *un hypergraphe avec* $V(H) = [n]$ *pour ensemble de sommets. La* $\mathbb{K}_r^\circ$-clôture *de* H*, notée* $<H>_r^\circ$*, est l'hypergraphe qui est l'intersection de tous les hypergraphes qui contiennent* H *et qui appartiennent à* $\mathbb{K}_r^\circ(n)$*. Un hypergraphe* H *tel que* $<H>_r^\circ = H$ *est appelé un* hypergraphe clos pour $<\cdot>_r^\circ$.

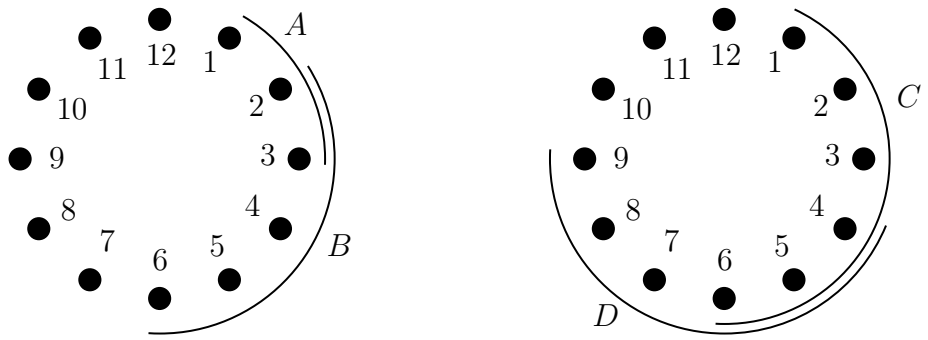
Démonstration. Il s'agit d'une conséquence directe du deuxième point du lemme 2.5. □

On peut maintenant définir la notion de r-orthogonalité :

Définition 5.7 (Arêtes r-orthogonales $A \perp_r B$)**.** *Soit* H *un hypergraphe et soit* $A, B \subseteq V(G)$ *deux hyper-arêtes de* H*. Les hyper-arêtes* A *et* B *sont dites* r-orthogonales *si* $<\{A, B\}>_r = <\{A, B\}>_r^\circ$*. On le note* $A \perp_r B$*. On rappelle que* $\{A, B\}$ *fait ici référence à l'hypergraphe qui a* $[n]$ *pour ensemble de sommets et* $\{A, B\}$ *pour ensemble d'hyper-arêtes.*

En d'autres termes, deux hyper-arêtes A et B d'un hypergraphe à n sommets sont r-orthogonales si le plus petit hypergraphe à n sommets qui les contient et qui vérifie les règles $\boldsymbol{R}_\leq$ et $\boldsymbol{R}_\neg$ est le même hypergraphe que le plus petit hypergraphe à n sommets qui les contient et qui vérifie les règles $\boldsymbol{R}_\leq$, $\boldsymbol{R}_\neg$ et $\boldsymbol{R}_\cup$. C'est une façon de garantir que la règle $\boldsymbol{R}_\cup$ est inutile du point de vue de A et B. Voyons quelques exemples d'arêtes r-orthogonales.

Exemple 5.3. Prenons un hypergraphe H avec $n = 12$ sommets (qui sont les entiers de 1 à n) et fixons $r = 3$. Soit $A = \{1, 2, 3\}$, $B = \{2, 3, 4, 5, 6\}$, $C = \{1, 2, 3, 4, 5, 6\}$ et $D = \{4, 5, 6, 7, 8, 9\}$ quatre hyper-arêtes de H (cf. figure 5.3). On peut montrer que $A \perp_r B$ et $C \perp_r D$. Notons que si on change $n = 12$ en $n = 13$, on a toujours $A \perp_r B$ mais on n'a plus $C \perp_r D$. On ne montrera pas cela dans cet exemple, car utiliser le lemme 5.13 pour faire cela est en général assez long. Ainsi, la section 5.3.1 sera dédiée à trouver une caractérisation plus simple pour la notion de r-orthogonalité et nous prouverons cet exemple à ce moment.



(a) H et ses hyper-arêtes A et B. (b) H et ses hyper-arêtes C et D.

FIGURE 5.3 – Hypergraphe H à 12 sommets. Chaque hyper-arête est représentée par un arc de cercle qui longe les sommets qu'elle contient.

On peut finalement introduire les hypergraphes sans r-croisement :

Définition 5.8 (Hypergraphe sans r-croisement)**.** *Un hypergraphe* H *est dit* sans r-croisement *si chaque paire d'hyper-arêtes* (A, B) *est* r*-orthogonale, c'est-à-dire* $A \perp_r B$.

5.1.5 Principaux théorèmes

La suite de ce chapitre est consacrée à la preuve des quatre théorèmes suivants.

Théorème 5.1. *Soit G un graphe r-rang connecté d'ordre n. Il existe un hypergraphe H avec $\mathcal{O}(n^{r+1})$ hyper-arêtes tel que $<H>_r = H_r(G)$.*

Théorème 5.2. *Soit H un hypergraphe sans r-croisement avec n sommets. Alors le nombre d'hyper-arêtes de H est au plus $\mathcal{O}(n^r)$.*

Théorème 5.3. *Soit $r > 0$ un entier fixé. Il existe un ensemble infini d'hypergraphes sans r-croisements $\{H_n\}$, chacun avec n sommets, telle que pour chaque hypergraphe H_n et pour chaque hypergraphe H' à n sommets, si $<H'>_r = <H_n>_r$, alors H' a au moins $\Omega(n^r)$ hyper-arêtes (avec r fixé et n qui tend vers l'infini).*

Théorème 5.4. *Étant donné un graphe G r-rang connecté d'ordre n, il est possible de construire un hypergraphe H avec $\mathcal{O}(n^{r+1})$ hyper-arêtes qui vérifie $<H>_r = H_r(G)$ en temps $\mathcal{O}(n^{2r+5+\omega} \log n)$, où $\omega \approx 2.37$ est lié à la multiplication de matrices.*

Le théorème 5.1 montre que l'ensemble des r-splits d'un graphe (c'est-à-dire l'hypergraphe $H_r(G)$) peut être représenté (via $<H>_r$) par un hypergraphe ayant un nombre polynomial d'hyper-arêtes. Autrement dit, si un graphe possède un nombre exponentiel de r-splits, seul un nombre polynomial d'entre eux est nécessaire pour en déduire tous les autres. Cela généralise le cas des splits, où seul un nombre linéaire d'entre eux est nécessaire pour déduire tous les autres, via l'arbre de décomposition de Cunningham. Les théorèmes 5.2 et 5.3 permettent de déduire que les hypergraphes r-clos sans r-croisements ont nécessairement $\Theta(n^r)$ hyper-arêtes, ce qui généralise le cas des familles partitives sans croisement. Enfin, le théorème 5.4 permet de s'assurer que la construction de H décrite dans le théorème 5.1 puisse se faire en temps polynomial, comme c'est le cas pour les splits.

5.2 Représentation polynomiale

La preuve du théorème 5.1 ressemble à cela : étant donné un hypergraphe $H \in \mathbb{K}_r$, on définit une notion d'hyper-arête essentielle de H, de sorte que le sous-hypergraphe H' constitué uniquement des hyper-arêtes essentielles ait une clôture égale à H. Une première approche naïve aboutit à $\mathcal{O}(n^{2r+1})$ arêtes essentielles. En étant plus minutieux et en considérant une seconde définition d'une arête essentielle, on parvient à n'avoir que $\mathcal{O}(n^{r+1})$ arêtes essentielles.

Ensuite, on montre que chaque hyper-arête de l'hypergraphe original H peut être obtenue en combinant les hyper-arêtes essentielle via les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$. Formellement, cela signifie que pour chaque hyper-arête A de H, il existe un hypergraphe H' constitué d'hyper-arêtes essentielles tel que A est une hyper-arête de $<H'>_r$.

Enfin, on conclut que l'hypergraphe H' constitué de toutes les hyper-arêtes essentielles vérifie $H \subseteq <H'>_r$, et comme H' est un sous-hypergraphe de H, on a une égalité. Finalement, comme $H_r(G) \in \mathbb{K}_r$, on peut prouver le théorème.

5.2.1 Arêtes essentielles, première approche

L'intérêt d'une hyper-arête essentielle est que toute hyper-arête puisse s'exprimer comme une union d'hyper-arêtes essentielles. On veut donc une certaine stabilité par union des hyper-arêtes essentielles. Comme les r-splits sont stables par union dès lors qu'ils s'intersectent en au moins r éléments (règle $\boldsymbol{R}_\cup$), cela motive à faire en sorte qu'une hyper-arête essentielle contiennent au moins r éléments. Plus précisément, comme il faut que l'intersection de deux hyper-arêtes ait au moins r éléments, il faut que chaque hyper-arête contiennent au moins $r+1$ éléments, de sorte qu'il soit possible que leur intersection en contienne r sans que l'une soit inclue dans l'autre.

Le but serait maintenant d'associer à chaque ensemble de $r+1$ éléments une unique hyper-arête essentielle qui les contienne. Pour que chaque hyper-arête puisse s'exprimer comme union d'hyper-arêtes essentielles, il faut qu'une hyper-arête essentielle soit la plus petite possible, afin de ne pas omettre certaines hyper-arêtes non-essentielles.

Il faudrait donc définir la plus petite hyper-arête contenant un ensemble de $r+1$ sommets. Pour que cela ait du sens, si X est un ensemble de $r+1$ sommets, il faut que l'intersection de deux hyper-arête contenant X soit aussi une hyper-arêtes. Il faut donc une certaine stabilité par intersection. Cependant, on n'a pas de propriété aussi forte, on a seulement que l'intersection de deux hyper-arêtes dont l'union évite au moins r sommets est aussi une hyper-arête (règle $\boldsymbol{P}_\cap$). Ainsi, il faut également fixer un ensemble Y de r sommets et considérer la plus petite hyper-arête qui contient X et évite Y.

Lemme 5.14. *Soit $H \in \mathbb{K}_r(n)$. Soit $X, Y \subseteq V(H)$ deux ensembles de sommets tels que $|Y| = r$ et $X \cap Y = \emptyset$. Soit $\mathcal{A}$ l'ensemble de toutes les hyper-arêtes A de H qui satisfont $X \subseteq A \subseteq \overline{Y}$. Alors l'intersection de toutes les hyper-arêtes de $\mathcal{A}$ est aussi une hyper-arête de $\mathcal{A}$.*

Démonstration. Tout d'abord, $\mathcal{A}$ est non-vide puisque $\overline{Y} \in \mathcal{A}$. En effet, $|\overline{Y}| = n - r$ donc $\overline{Y}$ est une hyper-arête de H d'après la propriété $\boldsymbol{P}_\geq$, et $X \cap Y = \emptyset$ implique $X \subseteq \overline{Y}$.

Ensuite, l'intersection de deux éléments de $\mathcal{A}$ est dans $\mathcal{A}$: soit $A, B \in \mathcal{A}$. On sait que $A, B \in H$ et $A \cup B \subseteq \overline{Y}$ donc $|A \cup B| \leq |\overline{Y}| = n - r$ et donc $|\overline{A \cup B}| \geq r$. Ainsi, d'après la propriété $\boldsymbol{P}_\cap$, $A \cap B \in H$. De plus, $X \subseteq A \cap B \subseteq \overline{Y}$, d'où $A \cap B \in \mathcal{A}$.

Ainsi, par récurrence, l'intersection de toutes les hyper-arêtes de $\mathcal{A}$ est aussi une hyper-arête de $\mathcal{A}$. □

On peut maintenant définir ce qu'est une hyper-arête essentielle.

Définition 5.9 (Hyper-arête essentielle, première définition)**.** *Soit $H \in \mathbb{K}_r(n)$. Une* hyper-arête essentielle *est une hyper-arête de la forme $\varphi_H(X, Y)$ pour certains ensembles de sommets $X \subseteq V(H)$ et $Y \subseteq V(H)$, où la fonction φ_H est définie de la façon suivante : elle prend en argument un ensemble de sommets $X \subseteq V(H)$ de taille $|X| = r+1$ et un ensemble de sommets $Y \subseteq V(H)$ de taille $|Y| = r$ tels que $X \cap Y = \emptyset$ et elle renvoie une hyper-arête A de H telle que $X \subseteq A \subseteq \overline{Y}$ et telle que A soit la plus petite hyper-arête avec cette propriété.*

La fonction φ est bien définie car la notion de « plus petite telle hyper-arête » existe graĉe au lemme 5.14. La figure 5.4 montre un exemple schématique d'une hyper-arête essentielle.

Remarque 5.1. On s'aperçoit que $\mathcal{A}$ est une famille de Moore sur $\overline{Y}$ et que la fonction $X \to \varphi_H(X, Y)$ est une fonction de clôture. En effet, la preuve du lemme 5.14 montre que

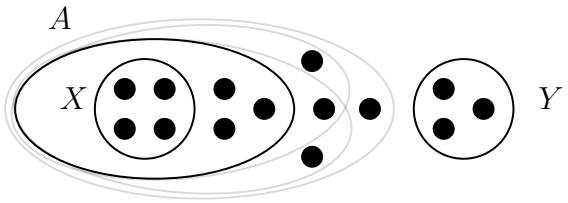


FIGURE 5.4 – En trait noir : les ensembles X, Y et $A = \varphi_H(X, Y)$. En trait gris : d'autres hyper-arêtes de H qui contiennent X et évitent Y. L'hyper-arête A est la plus petite des hyper-arêtes grises. Chaque point noir représente un sommet de H.

$\mathcal{A}$ contient $\overline{Y}$ et est stable par intersection (cf. définition 2.16). Ainsi, d'après le lemme 2.5, $X \to \varphi_H(X, Y)$ est une fonction de clôture. Finalement, on se rend compte que l'on manipule deux opérateurs de clôture :

— un qui clôture un hypergraphe : $H \to <H>_r$

— et un qui clôture un ensemble de sommets X de H : $X \to \varphi_H(X, Y)$.

Chaque hyper-arête est l'union de certaines hyper-arêtes essentielles

Dans cette section, on montre que si A est une hyper-arête de H, alors il existe un hypergraphe H' constitué de certaines hyper-arêtes essentielles de H tel que A soit une hyper-arête de $<H'>_r$. Si on prend une hyper-arête A de taille $|A| \leq r$ ou de taille $|A| \geq n - r$, on peut prendre $H' = \emptyset$ grâce à la propriété $\boldsymbol{R}_{\leq}$ ou $\boldsymbol{P}_{\geq}$. Sinon, on a $r+1 \leq |A| \leq n-r-1$, ce qui signifie que l'on peut trouver des ensembles X_i constitués de $r+1$ sommets et inclus dans A et que l'on peut trouver un ensemble Y de taille r qui évite A. Cela permet de considérer les arêtes essentielles $\varphi_H(X_i, Y)$ qui sont toutes des sous-ensembles de A par minimalité et dont l'union fait A. Montrons tout cela formellement.

On commence par le cas facile dans lequel $|A| \leq r$ ou $|A| \geq n - r$:

Lemme 5.15. *Soit $H \in \mathbb{K}_r(n)$. Soit A une hyper-arête de H telle que $|A| \leq r$ ou $|A| \geq n - r$. Alors $A \in <\emptyset>_r$, où $\emptyset$ représente l'hypergraphe sans hyper-arête.*

Démonstration. Si $|A| \leq r$, d'après la règle $\boldsymbol{R}_{\leq}$ appliquée à l'hypergraphe vide $\emptyset$, on a $A \in <\emptyset>_r$. Si $|A| \geq n - r$, d'après la propriété $\boldsymbol{P}_{\geq}$ appliquée à l'hypergraphe vide $\emptyset$, on a $A \in <\emptyset>_r$. □

Il reste le cas général dans lequel $r + 1 \leq |A| \leq n - r - 1$:

Lemme 5.16. *Soit $H \in \mathbb{K}_r(n)$. Soit A une hyper-arête de H telle que $r + 1 \leq |A| \leq n - r - 1$. Alors il existe une liste d'ensembles de sommets $X_1, \ldots, X_k$, un ensemble de sommets Y et un hypergraphe H' tels que les hyper-arêtes de H' sont exactement $\varphi_H(X_1, Y), \ldots, \varphi_H(X_k, Y)$ et tels que $A \in <H'>_r$.*

Démonstration. Écrivons A sous la forme $A = \{u_1, \ldots, u_{|A|}\}$. Pour chaque $1 \leq i \leq |A|-r$, prenons $X_i = \{u_i, \ldots, u_{i+r}\}$. Pour chaque X_i, on a donc $|X_i| = r + 1$. Soit Y un ensemble de r sommets qui ne sont pas dans A.

On montre que $A \in <H'>_r$ en appliquant le lemme 5.9 à l'hypergraphe $<H'>_r$ et aux k hyper-arêtes $\varphi_H(X_1, Y), \ldots, \varphi_H(X_k, Y)$. Pour pouvoir appliquer ce lemme, il faut vérifier que pour tout i, on ait bien $|\varphi_H(X_i, Y) \cap \varphi_H(X_{i+1}, Y)| \geq r$. Cela est bien le cas, car $X_i \subseteq \varphi_H(X_i, Y)$ et car $X_i \cap X_{i+1} = \{u_{i+1}, \ldots u_{i+r}\}$, qui est bien un ensemble de r sommets. Ainsi, $|\varphi_H(X_i, Y) \cap \varphi_H(X_{i+1}, Y)| \geq |X_i \cap X_{i+1}| = r$. D'où $A \in <H'>_r$. □

Ainsi, étant donné un hypergraphe $H \in \mathbb{K}_r(n)$ et une hyper-arête A de H, il existe une liste (éventuellement vide) d'ensembles de sommets $X_1, \ldots, X_k$, un ensemble de sommets Y et un hypergraphe H' tels que les hyper-arêtes de H' sont exactement les hyper-arêtes $\varphi_H(X_1, Y), \ldots, \varphi_H(X_k, Y)$ et tels que $A \in <H'>_r$.

5.2.2 Arêtes essentielles, seconde approche

Le problème de notre première approche est qu'elle donne au plus $\mathcal{O}(n^{2r+1})$ hyper-arêtes essentielles. Cela vient du fait que l'espace des entrées de φ_H est inclus dans $V^{r+1} \times V^r$, où V^{r+1} représente le choix de X et V^r celui de Y. On se rend compte que le choix de Y n'est pas vraiment nécessaire. On en a eu besoin seulement pour garantir que $\varphi_H(X, Y)$ soit bien défini. Or, dans les preuves, ce sont les choix de X qui sont utilisés. Le but est de voir s'il est possible de se passer du choix de Y et de faire en sorte que la fonction φ_H ne prenne qu'un ensemble X de taille $r + 1$ comme argument.

L'astuce est d'utiliser à bon escient la règle $\boldsymbol{R}_\neg$. En effet, vu que le complémentaire d'une hyper-arête est aussi une hyper-arête, il suffit de ne considérer que la moitié des hyper-arêtes, et plus particulièrement les hyper-arêtes de taille inférieure à $n/2$. L'autre moitié peut ensuite être retrouvée en utilisant la règle $\boldsymbol{R}_\neg$. L'idée est désormais la suivante :

On choisit un ensemble X de $r + 1$ sommets. S'il existe une hyper-arête A de H qui contient X telle que $|A| \leq n/2$, alors on associe X à la plus petite telle hyper-arête (qui est alors A ou un sous-ensemble de A). Sinon, cela signifie que chaque hyper-arête de H qui contient X a plus de $n/2$ sommets. On décide alors dans ce cas de n'associer X à aucune hyper-arête et on retire X de l'ensemble de définition de φ_H.

Dans un premier temps, on doit montrer que la notion de plus petite hyper-arête A de H qui contient X telle que $|A| \leq n/2$ a un sens. Pour cela, on sait que dans un hypergraphe $H \in \mathbb{K}_r(n)$, la propriété $\boldsymbol{P}_\cap$ garantit que l'intersection de deux hyper-arêtes A et B de H est aussi une hyper-arête de H sous réserve que $|\overline{A \cup B}| \geq r$. On montre que l'on peut remplacer cette condition par le fait que A et B soient de taille inférieure ou égale à $n/2$.

Lemme 5.17. *Soit $H \in \mathbb{K}_r(n)$. Soit A et B deux hyper-arêtes de H telles que $|A| \leq n/2$ et $|B| \leq n/2$. Alors $A \cap B$ est une hyper-arête de H.*

Démonstration. Si $|A \cap B| \leq r$, alors $A \cap B$ est une hyper-arête de H d'après la règle $\boldsymbol{R}_\leq$. Sinon, on a $|A \cap B| > r$. Ainsi, $|A \cup B| = |A| + |B| - |A \cap B| \leq n/2 + n/2 - r \leq n - r$, c'est-à-dire $|\overline{A \cup B}| \geq r$. D'après la propriété $\boldsymbol{P}_\cap$, $A \cap B$ est une hyper-arête de H. □

De façon naturelle, on peut étendre ce lemme à plus de deux arêtes :

Corollaire 5.18. *Soit $H \in \mathbb{K}_r(n)$. Soit $X \subseteq V(H)$ un ensemble de sommets. Soit $\mathcal{A}$ l'ensemble des hyper-arêtes A de H qui satisfont $X \subseteq A$ et $|A| \leq n/2$. Alors, si $\mathcal{A}$ est non-vide, l'intersection de toutes les hyper-arêtes de $\mathcal{A}$ est aussi une hyper-arête de $\mathcal{A}$.*

Démonstration. Soit A et B deux hyper-arêtes de $\mathcal{A}$. D'après le lemme 5.17, $A \cap B$ est une hyper-arête of H. Comme $A \cap B$ vérifie $X \subseteq A \cap B$ et $|A \cap B| \leq n/2$, c'est une hyper-arête de $\mathcal{A}$. On conclut par récurrence sur le nombre d'arêtes de $\mathcal{A}$. □

En d'autres termes, ce corollaire signifie que l'ensemble $\mathcal{A}$ possède un minimum du moment qu'il est non-vide. Grâce à cela, on peut définir ce qu'est une arête essentielle d'après notre seconde approche.

Définition 5.10 (Hyper-arête essentielle, seconde définition)**.** *Soit $H \in \mathbb{K}_r(n)$. Une* hyper-arête essentielle *est une hyper-arête de la forme $\varphi_H(X)$ pour certains ensembles de sommets $X \subseteq V$, où la fonction φ_H est définie de la façon suivante. Elle prend en argument un ensemble de sommets $X \subseteq V(H)$ de taille $|X| = r+1$. Si l'ensemble $\mathcal{A}$ des hyper-arêtes A de H qui satisfont $X \subseteq A$ et $|A| \leq n/2$ est vide, $\varphi_H(X)$ n'est pas défini et on retire X de l'ensemble de départ de φ_H. Si ce $\mathcal{A}$ est non-vide, il admet un minimum $\min \mathcal{A}$ d'après le corollaire 5.18. On pose alors $\varphi_H(X) = \min \mathcal{A}$.*

L'ensemble de départ de φ_H est donc l'ensemble des ensemble de sommets $X \subseteq V(H)$ de taille $|X| = r+1$ tels que $\{A \in V(H) \mid X \subseteq A \text{ et } |A| \leq n/2\} \neq \emptyset$.

Chaque hyper-arête est l'union de certaines hyper-arêtes essentielles

Dans cette section, on montre que si A est une hyper-arête de H, alors il existe un hypergraphe H' constitué de certaines hyper-arêtes essentielles de H tel que A soit une hyper-arête de $<H'>_r$. Pour montrer cela, on fait une disjonction de cas sur le nombre de sommets de l'hyper-arête A. Le cas principal concerne les hyper-arêtes vérifiant $r+1 \leq |A| \leq n/2$. Dans ce cas, le fait que $|A| \geq r+1$ garantit que l'on peut choisir un ensemble de sommets X inclus dans A de taille $r+1$ afin de le donner en paramètre à φ_H, et le fait $|A| \leq n/2$ garantit que X appartient bien au domaine de φ_H. Les autres cas se traitent facilement une fois le cas principal fait.

Dans un premier temps, prouvons formellement le cas principal, quand A vérifie $r+1 \leq |A| \leq n/2$:

Lemme 5.19. *Soit $H \in \mathbb{K}_r(n)$. Soit A une hyper-arête de H telle que $r+1 \leq |A| \leq n/2$. Alors il existe une liste d'ensembles $X_1, \ldots, X_k$ de sommets de H et un hypergraphe H' tel que les hyper-arêtes de H' soient exactement $\varphi_H(X_1), \ldots, \varphi_H(X_k)$ et tel que $A \in <H'>_r$.*

Autrement dit, une telle hyper-arête A peut être engendrée à partir d'une liste d'hyper-arêtes de H qui sont toutes de la forme $\varphi_H(X_i)$, c'est-à-dire que se sont toutes des hyper-arêtes essentielles.

Démonstration. Écrivons A sous la forme $A = \{u_1, \ldots, u_{|A|}\}$. Pour $1 \leq i \leq |A| - r$, posons $X_i = \{u_i, \ldots, u_{i+r}\}$. Pour chaque X_i, on a $|X_i| = r+1$. De plus, X_i est bien dans le domaine de φ_H, car il existe une hyper-arête de H qui contient X_i et qui a au plus $n/2$ sommets, à savoir A.

On montre ensuite que $A \in <H'>_r$ en appliquant le lemme 5.9 à l'hypergraphe $<H'>_r$ et aux k hyper-arêtes $\varphi_H(X_1), \ldots, \varphi_H(X_k)$. Pour pouvoir appliquer ce lemme, on doit montrer pour tout i que $|\varphi_H(X_i) \cap \varphi_H(X_{i+1})| \geq r$. Cela est vrai puisque $X_i \subseteq \varphi_H(X_i)$ et puisque $X_i \cap X_{i+1} = \{u_{i+1}, \ldots u_{i+r}\}$, qui est un ensemble de r sommets. Ainsi, $|\varphi_H(X_i) \cap \varphi_H(X_{i+1})| \geq |X_i \cap X_{i+1}| = r$.

Finalement, on a bien $A \in <H'>_r$. □

En utilisant la règle $\boldsymbol{R}_\neg$, on peut déduire le cas $n/2 \leq |A| \leq n-r-1$:

Corollaire 5.20. *Soit $H \in \mathbb{K}_r(n)$. Soit A une hyper-arête de H telle que $n/2 \leq |A| \leq n-r-1$. Alors il existe une liste d'ensembles $X_1, \ldots, X_k$ de sommets de H et un hypergraphe H' tel que les hyper-arêtes de H' soient exactement $\varphi_H(X_1), \ldots, \varphi_H(X_k)$ et tel que $A \in <H'>_r$.*

Démonstration. Soit $B = \overline{A}$. Ainsi, $r+1 \leq |B| \leq n/2$. D'après le lemme 5.19, il existe une liste d'ensembles de sommets $X_1, \ldots, X_k$ et un hypergraphe H' tel que les hyper-arêtes de H' sont exactement $\varphi_H(X_1), \ldots, \varphi_H(X_k)$ et tel que $B \in <H'>_r$. D'après la règle $\boldsymbol{R}_\neg$ appliquée à B, on a $A = \overline{B} \in <H'>_r$. □

Pour couvrir toutes les tailles possibles de l'hyper-arête A, il reste le cas dans lequel $|A| \leq r$ ou $|A| \geq n - r$. Celui-ci est déjà traité dans le lemme 5.15.

Finalement, étant donné une hypergraphe $H \in \mathbb{K}_r(n)$ et une hyper-arête A de H, il existe une liste (éventuellement vide) d'ensemble de sommets $X_1, \ldots, X_k$ et un hypergraphe H' tel que les hyper-arêtes de H' sont exactement $\varphi_H(X_1), \ldots, \varphi_H(X_k)$ et tel que $A \in <H'>_r$.

5.2.3 Résultat polynomial

Pour prouver le théorème 5.1, il suffit d'appliquer les résultats de la section 5.2.2 à chaque hyper-arête de l'hypergraphe H.

Lemme 5.21. *Étant donné un hypergraphe $H \in \mathbb{K}_r(n)$, il existe une liste d'ensembles de sommets $X_1, \ldots, X_k$ et un hypergraphe H' tel que toutes les hyper-arêtes de H' sont exactement $\varphi_H(X_1), \ldots, \varphi_H(X_k)$ et tel que $H = <H'>_r$.*

Démonstration. Pour chaque hyper-arête A_i de H, on sait d'après la section 5.2.2 qu'il existe un hypergraphe H_i tel que les hyper-arêtes de H_i sont de la forme $\varphi_H(X_{i_1})$, …, $\varphi_H(X_{i_k})$ et tel que $A_i \in <H_i>_r$. Soit H' l'hypergraphe défini comme l'union de tous les hypergraphes H_i. Montrons que $H = <H'>_r$.

Tout d'abord, on a :

$$H = \bigcup_i \{A_i\} \subseteq \bigcup_i <H_i>_r \underset{(1)}{\subseteq} <H'>_r.$$

L'inclusion (1) est due à la monotonie de l'opérateur de clôture $<\cdot>_r$ (cf. définition 2.14) appliquée à $H_i \subseteq H'$ pour tout i. Cela signifie que pour tout i, on a $<H_i>_r \subseteq <H'>_r$. Ainsi, l'union des $<H_i>_r$ est inclue dans $<H'>_r$.

De plus, on a :

$$H' = \bigcup_i H_i \underset{(2)}{\subseteq} \bigcup_i H = H.$$

L'inclusion (2) est due à l'extensivité de l'opérateur de clôture $<\cdot>_r$ (cf. définition 2.14).

On a donc $H' \subseteq H$, et par monotonie de $<\cdot>_r$, on déduit $<H'>_r \subseteq <H>_r$, c'est-à-dire $<H'>_r \subseteq H$ puisque $H \in \mathbb{K}_r(n)$.

Finalement, par double inclusion, on a bien $H = <H'>_r$. □

Corollaire 5.22. *Étant donné un hypergraphe $H \in \mathbb{K}_r(n)$, il existe un hypergraphe H' tel que le nombre d'hyper-arêtes de H' soit borné par $\mathcal{O}(n^{r+1})$ et tel que $H = <H'>_r$.*

Démonstration. D'après le lemme 5.21, il existe un hypergraphe H' constitué d'hyper-arêtes toutes de la forme $\varphi_H(X_i)$. Le nombre de telles hyper-arêtes est borné par la taille du domaine de φ_H, qui est inclus dans $V(H)^{r+1}$ et donc de taille n^{r+1}. □

Remarque 5.2. La première approche des hyper-arêtes essentielles conduit au même genre de résultat polynomial, mais avec une moins bonne borne. En effet, dans ce cas-là, les hyper-arêtes essentielles sont de la forme $\varphi_H(X_i, Y_i)$ et le domaine est maintenant de la forme $V(H)^{r+1} \times V(H)^r$, ce qui ne donnait qu'une borne en $\mathcal{O}(n^{2r+1})$.

On peut maintenant prouver le théorème 5.1. On rappelle qu'il s'énonce ainsi :

Théorème 5.1. *Soit G un graphe r-rang connecté d'ordre n. Il existe un hypergraphe H avec $\mathcal{O}(n^{r+1})$ hyper-arêtes tel que $<H>_r = H_r(G)$.*

Démonstration. Soit G un graphe r-rang connecté avec n sommets. Ainsi, $H_r(G) \in \mathbb{K}_r(n)$. D'après le corollaire 5.22, il existe un hypergraphe H' tel que le nombre d'hyper-arêtes H' soit borné par $\mathcal{O}(n^{r+1})$ et tel que $H_r(G) = <H'>_r$. □

5.2.4 Lien entre les deux approches

Il est possible de lier les deux notions d'hyper-arête essentielle.

Lemme 5.23. *Soit $H \in \mathbb{K}_r(n)$. Soit X un ensemble de $r+1$ sommets et soit Y un ensemble de r sommets tel que $X \cap Y = \emptyset$. Si $|\varphi_H(X,Y)| \leq n/2$, alors $\varphi_H(X,Y) = \varphi_H(X)$.*

Démonstration. On fait une disjonction de cas selon si $\varphi_H(X) \cap Y$ est vide ou non :

— Si $\varphi_H(X) \cap Y = \emptyset$, alors $\varphi_H(X) = \varphi_H(X,Y)$ et $|\varphi_H(X,Y)| \leq n/2$. En effet, $\varphi_H(X) \supseteq \varphi_H(X,Y)$ car $\varphi_H(X)$ est une hyper-arête de H qui contient X et qui évite Y, et $\varphi_H(X,Y)$ est la plus petite hyper-arête vérifiant ces propriétés. Ainsi, $|\varphi_H(X,Y)| \leq |\varphi_H(X)| \leq n/2$. De plus, $\varphi_H(X) \subseteq \varphi_H(X,Y)$ car $\varphi_H(X,Y)$ est une hyper-arête qui contient X et de taille inférieure ou égale à $n/2$, et $\varphi_H(X)$ est la plus petite d'entre elles.

— Sinon, $\varphi_H(X) \cap Y \neq \emptyset$, et on a alors $|\varphi_H(X,Y)| > n/2$. En effet, dans le cas contraire, si on suppose que $|\varphi_H(X,Y)| \leq n/2$, $\varphi_H(X,Y)$ est une arête qui contient X et de taille inférieure ou égale à $n/2$. Comme $\varphi_H(X)$ est la plus petite d'entre elles, $\varphi_H(X) \subseteq \varphi_H(X,Y)$. Or, puisque $\varphi_H(X) \cap Y \neq \emptyset$, il existe un sommet $y \in Y$ qui appartient à $\varphi_H(X)$ et donc à $\varphi_H(X,Y)$, ce qui contredit le fait que $\varphi_H(X,Y)$ évite Y. □

De plus, si tous les $\varphi_H(X,Y)$ vérifient $|\varphi_H(X,Y)| > n/2$, cela signifie que $\varphi_H(X)$ n'est pas défini, puisqu'il n'existe aucune hyper-arête contenant X de taille inférieure ou égale à $n/2$.

5.3 Borne sur le nombre d'hyper-arêtes

5.3.1 Hyper-arêtes orthogonales

On rappelle la définition de deux hyper-arêtes orthogonales (cf. définition 5.7) : A et B sont orthogonales (notée $A \perp_r B$) si les deux hypergraphes $<\{A,B\}>_r$ et $<\{A,B\}>_r^\circ$ sont égaux.

Ainsi, on commence par étudier l'opérateur de clôture $<\cdot>_r^\circ$ et son lien avec l'opérateur $<\cdot>_r$. Une application directe des définitions des deux opérateurs de clôture nous donne le lemme suivant :

Lemme 5.24. *Soit $V = [n]$ et soit $\emptyset$ l'hypergraphe ayant V pour ensemble de sommets et $\emptyset$ pour ensemble d'hyper-arêtes. Alors $<\emptyset>_r^\circ = <\emptyset>_r = \{A \subseteq V \mid |A| \leq r \text{ ou } |\overline{A}| \leq r\}$.*

Démonstration. Soit H l'hypergraphe dont l'ensemble d'hyper-arêtes est $\{A \subseteq V \mid |A| \leq r \text{ ou } |\overline{A}| \leq r\}$. D'après les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$ appliquées à l'hypergraphe $\emptyset$, on sait que $H \subseteq <\emptyset>_r$ et $H \subseteq <\emptyset>_r^\circ$. De plus, H est à la fois dans $\mathbb{K}_r$ et $\mathbb{K}_r^\circ$. Ainsi, $<H>_r = <H>_r^\circ = H$. Puisque $\emptyset \subseteq H$ et puisque qu'un opérateur de clôture est monotone, on a $<\emptyset>_r \subseteq <H>_r = H$ et $<\emptyset>_r^\circ \subseteq <H>_r^\circ = H$. □

Lemme 5.25. *Soit $V = [n]$ et soit $A \subseteq V$ et $B \subseteq V$ deux ensembles de sommets. On a $<\{A, B\}>_r^\circ = <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\}$.*

Démonstration. Soit $H = \{A, B\}$ et $H' = <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\}$. D'abord, montrons que $<H>_r^\circ \subseteq H'$. Puisque $<H>_r^\circ$ est le plus petit hypergraphe qui a H comme sous-hypergraphe et qui satisfait les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$, il suffit de montrer que H' possède H comme sous-hypergraphe et que H' satisfait les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$:

— H' possède les hyper-arêtes A et B, ce qui signifie que H' possède H comme sous-hypergraphe.

— H' satisfait la règle $\boldsymbol{R}_{\leq}$ puisque $<\emptyset>_r \subseteq H'$ et puisque $<\emptyset>_r$ satisfait la règle $\boldsymbol{R}_{\leq}$ par définition.

— H' satisfait la règle $\boldsymbol{R}_{\neg}$ puisque $<\emptyset>_r$ satisfait la règle $\boldsymbol{R}_{\neg}$ par définition et puisque $\{A, B, \overline{A}, \overline{B}\}$ satisfait la règle $\boldsymbol{R}_{\neg}$ par construction.

Maintenant, prouvons que $H' \subseteq <H>_r^\circ$. On sait que A et B sont des hyper-arêtes de $<H>_r^\circ$ puisque $<H>_r^\circ$ possède H comme sous-hypergraphe. Ensuite, $\overline{A}$ et $\overline{B}$ sont des hyper-arêtes de $<H>_r^\circ$ puisque $<H>_r^\circ$ satisfait le règle $\boldsymbol{R}_{\neg}$. Finalement, $<\emptyset>_r = <\emptyset>_r^\circ \subseteq <H>_r^\circ$. □

Lemme 5.26. *Soit H un hypergraphe. Alors $<H>_r^\circ \subseteq <H>_r$.*

Démonstration. On sait que l'hypergraphe $<H>_r$ est un hypergraphe qui a H comme sous-hypergraphe et qui satisfait les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$ (et aussi la règle $\boldsymbol{R}_{\cup}$, mais ce n'est pas utile ici). L'hypergraphe $<H>_r^\circ$ est le plus petit hypergraphe qui a H comme sous-hypergraphe et qui satisfait les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$. Ainsi, $<H>_r^\circ \subseteq <H>_r$. □

Le but est maintenant d'obtenir une formule logique qui exprime sous quelles conditions deux hyper-arêtes sont r-orthogonales.

Lemme 5.27. *Soit $r \geq 0$ et $V = [n]$. Soit $A \subseteq V$ et $B \subseteq V$ deux ensembles de sommets. On a l'équivalence suivante.*

$$\begin{aligned} A \perp_r B \iff & (|A \cap B| \geq r \implies A \cup B = A \vee A \cup B = B \vee |\overline{A} \cap \overline{B}| \leq r) \wedge \\ & (|\overline{A} \cap \overline{B}| \geq r \implies \overline{A} \cup \overline{B} = \overline{A} \vee \overline{A} \cup \overline{B} = \overline{B} \vee |A \cap B| \leq r) \wedge \\ & (|\overline{A} \cap B| \geq r \implies \overline{A} \cup B = \overline{A} \vee \overline{A} \cup B = B \vee |A \cap \overline{B}| \leq r) \wedge \\ & (|A \cap \overline{B}| \geq r \implies A \cup \overline{B} = A \vee A \cup \overline{B} = \overline{B} \vee |\overline{A} \cap B| \leq r). \end{aligned}$$

Démonstration. On prouve l'équivalence par double implication.

($\Longrightarrow$) On a $A \perp_r B$, c'est-à-dire $<\{A, B\}>_r^\circ = <\{A, B\}>_r$. En combinant cette égalité avec le lemme 5.25, on obtient $<\{A, B\}>_r = <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\}$. À partir de cette égalité, on va pouvoir traiter les quatre implications une par une. On prouve la première en détail et nous donnons seulement une ébauche pour les trois autres cas, qui se traitent d'une façon très similaire. Par soucis de concision, posons $H := \{A, B\}$.

— Si $|A \cap B| \geq r$, d'après la règle $\boldsymbol{R}_{\cup}$, $A \cup B \in <H>_r$. Regardons les valeurs possibles pour $A \cup B$ parmi les hyper-arêtes de $<H>_r = <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\}$. Si $A \cup B \in <\emptyset>_r$, alors d'après le lemme 5.24, $|A \cup B| \leq r$ ou $|\overline{A \cup B}| \leq r$. Si $A \cup B = \overline{A}$, alors $A = \emptyset$ et $B = V$, ce qui signifie $A \cup B = B$. Si $A \cup B = \overline{B}$, alors $A = V$ et $B = \emptyset$, ce qui signifie $A \cup B = A$. Sinon, $A \cup B = A$ ou $A \cup B = B$. En résumé :

$$\begin{aligned} |A \cap B| \geq r &\implies |A \cup B| \leq r \vee |\overline{A \cup B}| \leq r \vee A \cup B = A \vee A \cup B = B \\ &\implies |\overline{A} \cap \overline{B}| \leq r \vee A \cup B = A \vee A \cup B = B \end{aligned}$$

Pour obtenir la dernière implication, on a remplacé $|\overline{A \cup B}| \leq r$ par $|\overline{A} \cap \overline{B}| \leq r$ et on a supprimé $|A \cup B| \leq r$ car si $|A \cap B| \geq r$ et $|A \cup B| \leq r$, alors $A = B$, ce qui implique $A \cup B = A$.

— Si $|\overline{A} \cap \overline{B}| \geq r$, alors on a $|\overline{A \cup B}| \geq r$. D'après la propriété $\boldsymbol{P}_{\cap}$, on a $A \cap B \in <H>_r$, et d'après la règle $\boldsymbol{R}_{\neg}$, on a $\overline{A \cap B} = \overline{A} \cup \overline{B} \in <H>_r$. Avec le même raisonnement que le point précédent, en remplaçant A par $\overline{A}$, B par $\overline{B}$ et inversement, on conclut que $(|\overline{A} \cap \overline{B}| \geq r \implies \overline{A} \cup \overline{B} = \overline{A} \vee \overline{A} \cup \overline{B} = \overline{B} \vee |A \cap B| \leq r)$.

— Si $|\overline{A} \cap B| \geq r$, alors on a $|B \setminus A| \geq r$. D'après la propriété $\boldsymbol{P}_{\setminus}$, on a $A \setminus B \in <H>_r$, et d'après la règle $\boldsymbol{R}_{\neg}$, on a $\overline{A \setminus B} = \overline{A} \cup B \in <H>_r$. Avec le même raisonnement que le premier point, en remplaçant A par $\overline{A}$ et inversement, on conclut que $(|\overline{A} \cap B| \geq r \implies \overline{A} \cup B = \overline{A} \vee \overline{A} \cup B = B \vee |A \cap \overline{B}| \leq r)$.

— Si $|A \cap \overline{B}| \geq r$, alors on a $|A \setminus B| \geq r$. D'après la propriété $\boldsymbol{P}_{\setminus}$, on a $B \setminus A \in <H>_r$, et d'après la règle $\boldsymbol{R}_{\neg}$, on a $\overline{B \setminus A} = A \cup \overline{B} \in <H>_r$. Avec le même raisonnement que le premier point, en remplaçant B par $\overline{B}$ et inversement, on conclut que $(|A \cap \overline{B}| \geq r \implies A \cup \overline{B} = A \vee A \cup \overline{B} = \overline{B} \vee |\overline{A} \cap B| \leq r)$.

($\Longleftarrow$) On doit montrer $A \perp_r B$. On pose à nouveau $H := \{A, B\}$ et on définit $H' := <H>_r^{\circ} = <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\}$. D'après le lemme 5.26, on sait que $<H>_r^{\circ} \subseteq <H>_r$. Il reste à prouver que $<H>_r \subseteq H'$. Pour cela, il suffit de montrer que H' possède H comme sous-hypergraphe et que H' satisfait les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$. On sait déjà que H' possède H comme sous-hypergraphe d'après les définitions de H et H' et on sait aussi que H' satisfait les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$ car $H' = <H>_r^{\circ}$. Il reste donc à prouver que H' satisfait la règle $\boldsymbol{R}_{\cup}$.

Soit C et D deux hyper-arêtes de H' telles que $|C \cap D| \geq r$. Montrons que $C \cup D$ est aussi une hyper-arête de H'.

D'abord, si C est une hyper-arête telle que $|C| \leq r$, alors $|C \cap D| \leq |C| \leq r$, ce qui implique $|C \cap D| = r$ et $|C| = r$, et on a donc $C \subseteq D$. Ainsi, $C \cup D = D$ est une hyper-arête de H'. De plus, si $|C| \geq n - r$, alors $|C \cup D| \geq n - r$, et $C \cup D$ est une hyper-arête de $<\emptyset>_r$ et donc de H'. Finalement, si $C \in <\emptyset>_r$, alors $C \cup D \in H'$; et par symétrie, si $D \in <\emptyset>_r$, alors $C \cup D \in H'$.

Il ne reste plus qu'à traiter les cas où $\{C, D\} \subseteq \{A, B, \overline{A}, \overline{B}\}$. Si $C = D$, cela est trivial car $C \cup D = C \in H'$. Si $C = \overline{D}$, cela signifie que $\{C, D\} = \{A, \overline{A}\}$ ou $\{C, D\} = \{B, \overline{B}\}$ et on a donc $C \cup D = V \in H'$. En prenant en compte la symétrie entre C et D, il ne reste plus que quatre cas à traiter, qui sont tous les quatre très similaires :

— Si $C = A$ et $D = B$, puisque $|C \cap D| \geq r$, alors $|A \cap B| \geq r$. Par hypothèse, $(|A \cap B| \geq r \implies A \cup B = A \vee A \cup B = B \vee |\overline{A} \cap \overline{B}| \leq r)$. Si $A \cup B = A$, alors $C \cup D = A \cup B = A \in H'$. Si $A \cup B = B$, alors $C \cup D = A \cup B = B \in H'$. Si $|\overline{A} \cap \overline{B}| \leq r$, alors $|\overline{A \cup B}| \leq r$, et $\overline{A \cup B} \in H'$ d'après la règle $\boldsymbol{R}_{\leq}$, et $A \cup B \in H'$ d'après la règle $\boldsymbol{R}_{\neg}$, c'est-à-dire $C \cup D \in H'$.

— Si $C = \overline{A}$ et $D = \overline{B}$, puisque $|C \cap D| \geq r$, alors $|\overline{A} \cap \overline{B}| \geq r$. Par hypothèse, $(|\overline{A} \cap \overline{B}| \geq r \implies \overline{A} \cup \overline{B} = \overline{A} \vee \overline{A} \cup \overline{B} = \overline{B} \vee |A \cap B| \leq r)$. On effectue le même raisonnement en remplaçant A par $\overline{A}$, B par $\overline{B}$ et inversement, ce qui donne $C \cup D \in H'$.

— Si $C = \overline{A}$ et $D = B$, puisque $|C \cap D| \geq r$, alors $|\overline{A} \cap B| \geq r$. Par hypothèse, $(|\overline{A} \cap B| \geq r \implies \overline{A} \cup B = \overline{A} \vee \overline{A} \cup B = B \vee |A \cap \overline{B}| \leq r)$. On effectue le même raisonnement que pour le premier cas en remplaçant A par $\overline{A}$ et inversement, ce qui donne $C \cup D \in H'$.

— Si $C = A$ et $D = \overline{B}$, puisque $|C \cap D| \geq r$, alors $|A \cap \overline{B}| \geq r$. Par hypothèse, $(|A \cap \overline{B}| \geq r \implies A \cup \overline{B} = A \vee A \cup \overline{B} = \overline{B} \vee |\overline{A} \cap B| \leq r)$. On effectue le même raisonnement que pour le premier cas en remplaçant B par $\overline{B}$ et inversement, ce qui donne $C \cup D \in H'$.

Par double implication, on a l'équivalence désirée. □

Le lemme 5.27 permet de ne raisonner qu'en termes d'expression logique. Grâce à lui, on peut simplifier l'expression logique obtenue et obtenir de nouvelles équivalences à la relation $A \perp_r B$.

Corollaire 5.28. *Soit $V = [n]$ et soit $A \subseteq V$ et $B \subseteq V$. On a l'équivalence suivante.*

$$
\begin{aligned}
&A \perp_r B \\
&\iff \\
&(|A \cap B| < r \vee A \subseteq B \vee B \subseteq A \vee |\overline{A \cup B}| < r \vee |A \cap B| = |\overline{A \cup B}| = r) \wedge \\
&(|A \setminus B| < r \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee |B \setminus A| < r \vee |A \setminus B| = |B \setminus A| = r).
\end{aligned}
$$

Démonstration. On part de l'équivalence du lemme 5.27 et on applique les lois de De Morgan ainsi que des équivalences basiques sur les ensembles telles que $A \cup B = A \iff B \subseteq A$ et $\overline{A} \cup B = \overline{A} \iff A \cap B = \emptyset$. Par soucis de concision, posons $R_1 \equiv (|A \cap B| < r \vee A \subseteq B \vee B \subseteq A \vee |\overline{A \cup B}| < r)$ et $R_2 \equiv (|A \setminus B| < r \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee |B \setminus A| < r)$. Rappelons que pour n'importe quelles propositions X, Y, Z, on a $(X \implies Y) \iff (\neg X \vee Y)$ ainsi que $(Z \vee X) \wedge (Z \vee Y) \iff Z \vee (X \wedge Y)$. Ainsi, on déduit :

$$
\begin{aligned}
&A \perp_r B \\
&\iff \\
&(|A \cap B| < r \vee A \subseteq B \vee B \subseteq A \vee |\overline{A \cup B}| \leq r) \wedge \\
&(|\overline{A \cup B}| < r \vee A \subseteq B \vee B \subseteq A \vee |A \cap B| \leq r) \wedge \\
&(|A \setminus B| < r \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee |B \setminus A| \leq r) \wedge \\
&(|B \setminus A| < r \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee |A \setminus B| \leq r) \\
&\iff \\
&(R_1 \vee |\overline{A \cup B}| = r) \wedge (R_1 \vee |A \cap B| = r) \wedge (R_2 \vee |B \setminus A| = r) \wedge (R_2 \vee |A \setminus B| = r) \\
&\iff \\
&(R_1 \vee |A \cap B| = |\overline{A \cup B}| = r) \wedge (R_2 \vee |A \setminus B| = |B \setminus A| = r),
\end{aligned}
$$

ce qui conclut. □

Dans le cas où $r = 1$ et $n > 4$, le corollaire 5.28 se simplifie en $A \perp_1 B \iff (A \subseteq B \vee B \subseteq A \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset)$. En effet, dans ce cas, les relations R_1 et R_2 de la preuve deviennent égales et nous pouvons factoriser la proposition logique par R_1, ce qui donne $R_1 \vee (|A \cap B| = |\overline{A \cup B}| = |A \setminus B| = |B \setminus A| = 1)$. Ensuite, puisque $n > 4$, la condition $|A \cap B| = |\overline{A \cup B}| = |A \setminus B| = |B \setminus A| = 1$ est nécessairement fausse et on peut la supprimer de l'équivalence. Ainsi, la relation $\perp_r$ généralise la notion d'hyper-arêtes qui ne se croisent pas, comme introduit dans [CE80].

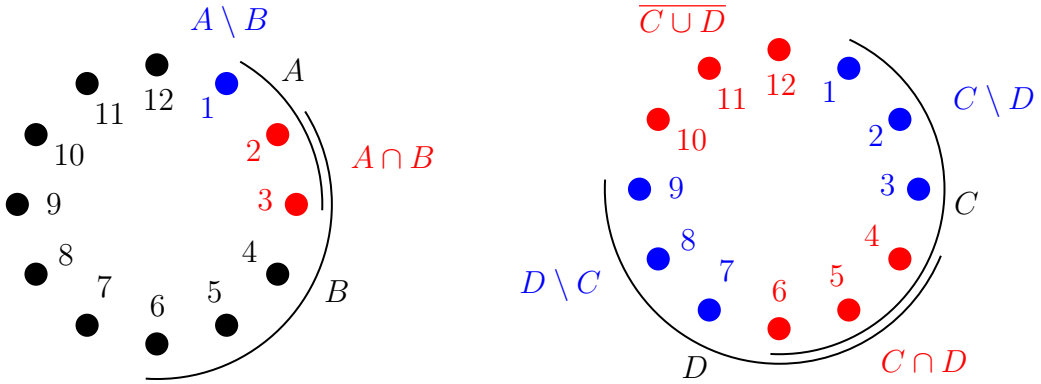
On peut maintenant revenir à l'exemple 5.3 qui exhibait quelques hyper-arêtes orthogonales non-triviales et utiliser le corollaire 5.28 pour montrer facilement qu'elles sont bien orthogonales :

Exemple 5.4. Prenons un hypergraphe H avec $n = 12$ sommets (qui sont les entiers de 1 à n) et fixons $r = 3$. Soit $A = \{1, 2, 3\}$, $B = \{2, 3, 4, 5, 6\}$, $C = \{1, 2, 3, 4, 5, 6\}$ et $D = \{4, 5, 6, 7, 8, 9\}$ quatre hyper-arêtes de H (cf. figure 5.3). On peut maintenant montrer que $A \perp_r B$ et $C \perp_r D$ via le corollaire 5.28 :

— On a $|A \cap B| = |\{2, 3\}| < 3$ et $|A \setminus B| = |\{1\}| < 3$, ce qui implique $A \perp_3 B$.

— On a $|C \cap D| = |\overline{C \cup D}| = 3$ et $|C \setminus D| = |D \setminus C| = 3$, ce qui implique $C \perp_3 D$.

Changeons maintenant $n = 12$ en $n = 13$. La relation $A \perp_3 B$ reste vraie mais la relation $C \perp_3 D$ est désormais fausse. En effet, en ce qui concerne $A \perp_3 B$, les inégalités $|A \cap B| < 3$ et $|A \setminus B| < 3$ ne dépendent pas de n. La relation reste donc vraie. Cependant, concernant $C \perp_3 D$, on a utilisé le fait que $|C \cap D| = 3$ et $|\overline{C \cup D}| = 3$. Or, l'opération de complémentarité dépend de l'ensemble V et donc de n. On a maintenant $|\overline{C \cup D}| = 4$. Comme $|C \cap D| = \{4, 5, 6\} \leq 3$, $C \not\subseteq D$, $D \not\subseteq C$ et $|\overline{C \cup D}| = |\{10, 11, 12, 13\}| \geq 3$, il n'est pas possible de satisfaire la première partie de la proposition logique du corollaire 5.28, ce qui montre que l'on n'a pas $C \perp_3 D$ dans le cas où $n = 13$.



(a) H et ses hyper-arêtes A et B. (b) H et ses hyper-arêtes C et D.

FIGURE 5.5 – Hypergraphe H à 12 sommets. Chaque hyper-arête est représentée par un arc de cercle qui longe les sommets qu'elle contient. Les sommets colorés sont les ensembles de sommets importants pour démontrer l'orthogonalité des deux hyper-arêtes considérées.

Avant de passer à la section suivante, voici quelques propriétés basiques concernant les hyper-arêtes r-orthogonales.

Lemme 5.29. *Soit H un hypergraphe, soit A et B deux hyper-arêtes de H et soit $r \geq 0$. On a :*

1. *Si $|A| \leq r$, alors $A \perp_r B$.*

2. $A \perp_r A$.
3. $A \perp_r \overline{A}$.
4. *Si* $A \perp_r B$, *alors* $B \perp_r A$.
5. *Si* $A \perp_r B$, *alors* $A' \perp_r B'$ *pour* $A' \in \{A, \overline{A}\}$ *et* $B' \in \{B, \overline{B}\}$.
6. *Si* $A \perp_r B$, *alors* $A \perp_{r+1} B$.

Démonstration. On prouve chaque point en utilisant l'équivalence du corollaire 5.28. On rappelle qu'elle s'énonce comme suit :

$$A \perp_r B \iff (|A \cap B| < r \vee A \subseteq B \vee B \subseteq A \vee |\overline{A \cup B}| < r \vee |A \cap B| = |\overline{A \cup B}| = r)$$
$$\wedge (|A \setminus B| < r \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee |B \setminus A| < r \vee |A \setminus B| = |B \setminus A| = r).$$

1. Il y a trois cas. Si $|A \cap B| = r$, alors $A \setminus B = \emptyset$, ce qui signifie que $A \perp_r B$. Si $|A \setminus B| = r$, alors $A \cap B = \emptyset$, ce qui signifie que $A \perp_r B$. Sinon, $|A \cap B| < r$ et $|A \setminus B| < r$, ce qui signifie que $A \perp_r B$.
2. Si $r > 0$, alors $A \subseteq A$ et $|A \setminus A| < r$. Pour le cas $r = 0$, on a $A \subseteq A$ et $|A \setminus A| = |A \setminus A| = r$.
3. $|A \cap \overline{A}| < r$ et $A \cap \overline{A} = \emptyset$.
4. La définition 5.7 est symétrique en A et B.
5. Le corollaire 5.28 est stable par changement de B en $\overline{B}$.
6. Si un ensemble à une taille d'au plus r, il a aussi une taille d'au plus $r + 1$.

Ainsi, chaque point est prouvé. □

5.3.2 Hypergraphes sans r-croisement

Le but de cette section est d'étudier les hypergraphes sans r-croisement afin de pouvoir en déduire des bornes sur le nombre d'hyper-arêtes d'un tel hypergraphe. Plus précisément, on veut déduire des relations entre le nombre d'hyper-arêtes d'un hypergraphe sans r-croisement H et le nombre d'hyper-arêtes de sa clôture $<H>_r$.

On sait qu'un hypergraphe sans r-croisement H ne contient que des hyper-arêtes qui sont orthogonales entre elles. Informellement, cela signifie qu'on peut obtenir sa clôture sans avoir besoin d'appliquer la règle $\boldsymbol{R}_\cup$. Ainsi, on a besoin d'appliquer que les règles $\boldsymbol{R}_\leq$ et $\boldsymbol{R}_\neg$. Or, la règle $\boldsymbol{R}_\leq$ ne peut ajouter qu'un maximum de $\mathcal{O}(n^r)$ hyper-arêtes (celles qui ont moins de r sommets et celles qui ont plus de $n - r$ sommets) et la règle $\boldsymbol{R}_\neg$ peut au maximum doubler le nombre d'hyper-arêtes. Ainsi, on peut borner le nombre d'hyper-arêtes de $<H>_r$ en fonction du nombre d'hyper-arêtes de H, comme on le montrera dans le lemme 5.33. De plus, le fait que la règle $\boldsymbol{R}_\neg$ puisse au maximum doubler le nombre d'hyper-arêtes est présenté formellement à travers le lemme 5.34. Un dernier détail important est le fait que prendre la clôture d'un hypergraphe sans r-croisement n'introduit pas de r-croisement, ce qui permet de garantir que la règle $\boldsymbol{R}_\cup$ ne peut pas devenir utile. Cela est présenté dans le lemme 5.35.

Afin de prouver ces trois lemmes, on montre d'abord trois lemmes élémentaires.

Lemme 5.30. *Soit* H *un hypergraphe qui a pour ensemble de sommets* $V(H) = [n]$ *et qui n'a qu'une seule hyper-arête* A, *c'est-à-dire que* $\mathcal{E}(H) = \{A\}$. *On rappelle que l'on s'autorise à noter* $\{A\}$ *l'hypergraphe* H. *Alors* $<\{A\}>_r = <\emptyset>_r \cup \{A, \overline{A}\}$.

Démonstration. L'hypergraphe $<\emptyset>_r \cup \{A, \overline{A}\}$ est un hypergraphe qui contient $\{A\}$ en tant que sous-hypergraphe et qui satisfait les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$. Puisque $<\{A\}>_r$ est le plus petit hypergraphe vérifiant ces contraintes, on a $<\{A\}>_r \subseteq <\emptyset>_r \cup \{A, \overline{A}\}$.

Puisque $\emptyset \subseteq \{A\}$, alors $<\emptyset>_r \subseteq <\{A\}>_r$ par monotonie. De plus, A est une hyper-arête de $<\{A\}>_r$ vu que $<\{A\}>_r$ contient $\{A\}$ comme sous-hypergraphe. Ainsi, puisque $<\{A\}>_r$ satisfait la règle $\boldsymbol{R}_{\neg}$, $\overline{A}$ est une hyper-arête de $<\{A\}>_r$, ce qui implique que $<\emptyset>_r \cup \{A, \overline{A}\} \subseteq <\{A\}>_r$, concluant la preuve. □

Le lemme suivant est assez intéressant car il aurait pu être une définition de la r-orthogonalité. Il donne en effet une équivalence à la relation $A \perp_r B$ en fonction uniquement de l'opérateur de clôture $<\cdot>_r$. Il ne permet plus de voir directement que la r-orthogonalité est liée à la non-utilisation de la règle $\boldsymbol{R}_{\cup}$, mais il permet de lier la r-orthogonalité aux opérateurs de clôture de Kuratowski [LL12]. Un opérateur de clôture $\mathrm{cl}(\cdot)$ est dit « de Kuratowski » lorsqu'il vérifie (entre autres) la condition $\mathrm{cl}(A \cup B) = \mathrm{cl}(A) \cup \mathrm{cl}(B)$. Le lemme suivant montre que $A \perp_r B$ si et seulement si $<\{A, B\}>_r = <\{A\}>_r \cup <\{B\}>_r$. De manière informelle, cela signifie que dans un hypergraphe sans r-croisement, l'opérateur de clôture $<\cdot>_r$ se comporte sur les hyper-arêtes en partie comme un opérateur de Kuratowski.

Lemme 5.31. *Soit $V = [n]$. Soit $A \subseteq V$ et $B \subseteq V$ deux ensembles de sommets. On a l'équivalence suivante : $A \perp_r B \iff <\{A, B\}>_r = <\{A\}>_r \cup <\{B\}>_r$.*

Démonstration. D'après la définition 5.7, on a $A \perp_r B \iff <\{A, B\}>_r = <\{A, B\}>_r^{\circ}$. D'après les lemmes 5.25 et 5.30, on a les égalités suivantes :

$$\begin{aligned} <\{A, B\}>_r^{\circ} &= <\emptyset>_r \cup \{A, B, \overline{A}, \overline{B}\} \\ &= (<\emptyset>_r \cup \{A, \overline{A}\}) \cup (<\emptyset>_r \cup \{B, \overline{B}\}) \\ &= <\{A\}>_r \cup <\{B\}>_r. \end{aligned}$$

D'où le résultat. □

Le lemme suivant permet d'exprimer la clôture d'un hypergraphe sans r-croisement en fonction de la clôture de chacune de ses hyper-arêtes.

Lemme 5.32. *Soit H un hypergraphe sans r-croisement ayant comme ensemble de sommets $V(H) = [n]$ et ayant au moins une hyper-arête. Pour chaque hyper-arête A de H, on note $\{A\}$ l'hypergraphe ayant $[n]$ comme ensemble de sommets et ayant uniquement A comme hyper-arête. Alors :*

$$<H>_r = \bigcup_{A \in H} <\{A\}>_r$$

Démonstration. On procède par double inclusion :

($\supseteq$) Pour chaque hyper-arête A, on a $\{A\} \subseteq H$. Ainsi, par monotonie de l'opérateur de clôture, $<\{A\}>_r \subseteq <H>_r$ et donc $\bigcup_{A \in H} <\{A\}>_r \subseteq <H>_r$.

($\subseteq$) Prouvons maintenant que $<H>_r$ est inclus dans l'union $U := \bigcup_{A \in H} <\{A\}>_r$. Pour ce faire, on prouve que U satisfait les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$.

- La règle $\boldsymbol{R}_{\leq}$ est satisfaite car H possède au moins une hyper-arête A et on a donc $<\emptyset>_r \subseteq <A>_r \subseteq U$.
- La règle $\boldsymbol{R}_{\neg}$ est satisfaite car chaque arête A de U appartient à un certain $<A'>_r$ qui contient aussi $\overline{A}$ d'après la règle $\boldsymbol{R}_{\neg}$ appliqué à $<A'>_r$.

— Pour montrer que U satisfait la règle $\boldsymbol{R}_{\cup}$, on prend A, B deux hyper-arêtes de U telles que $|A \cap B| \geq r$ et on montre que $A \cup B \in U$. D'après la définition de U, il existe $A' \in H$ et $B' \in H$ telles que $A \in <\{A'\}>_r$ et $B \in <\{B'\}>_r$.

D'après le lemme 5.30, $<\{A'\}>_r = <\emptyset>_r \cup \{A', \overline{A'}\}$. Si $A \in <\emptyset>_r$, d'après le lemme 5.24, on a $|A| \leq r$ ou $|A| \geq n - r$. Si $|A| \leq r$, puisque $|A \cap B| \geq r$, on a $|A| = |A \cap B| = r$, et donc $A \subseteq B$, ce qui implique $A \cup B = B \in U$. Si $|A| \geq n - r$, $|A \cup B| \geq n - r$ et $A \cup B \in <\emptyset>_r \subseteq U$. Il reste le cas où $A \subseteq \{A', \overline{A'}\}$. Avec la même analyse appliquée à B, on peut supposer que $B \subseteq \{B', \overline{B'}\}$. Puisque H est sans r-croisement, on sait que $A \perp_r B$. D'après le point 5 du lemme 5.29, on déduit $A' \perp_r B'$. Ensuite, avec le lemme 5.31, on a $<\{A'\}>_r \cup <\{B'\}>_r = <\{A', B'\}>_r$. Ainsi, $A, B \in <\{A', B'\}>_r$, et puisque $|A \cap B| \geq r$, d'après la règle $\boldsymbol{R}_{\cup}$ appliqué à $<\{A', B'\}>_r$, on a $A \cup B \in <\{A', B'\}>_r$. Enfin, vu que $<\{A', B'\}>_r = <\{A'\}>_r \cup <\{B'\}>_r$, on a $A \cup B \in <\{A'\}>_r \cup <\{B'\}>_r \subseteq U$.

Finalement, l'hypergraphe U satisfait les règles $\boldsymbol{R}_{\leq}$, $\boldsymbol{R}_{\neg}$ et $\boldsymbol{R}_{\cup}$ et possède H comme sous-hypergraphe. Puisque $<H>_r$ est le plus petit de ces hypergraphes, on a l'inclusion $<H>_r \subseteq U$. □

On peut maintenant comprendre la structure d'un hypergraphe sans r-croisement et donner des bornes sur son nombre d'hyper-arêtes.

Lemme 5.33. *Soit H un hypergraphe sans r-croisement avec n sommets. On a :*

$$<H>_r = <\emptyset>_r \cup \bigcup_{A \in H} \{A, \overline{A}\}.$$

Par conséquent, on a l'inégalité $|<H>_r| \leq 2(r+1)n^r + 2|H|$.

Démonstration. D'après les lemmes 5.32 et 5.30 appliqués à H, on a :

$$<H>_r = \bigcup_{A \in H} <H_A>_r = \bigcup_{A \in H} (<\emptyset>_r \cup \{A, \overline{A}\}) = <\emptyset>_r \cup \bigcup_{A \in H} \{A, \overline{A}\}.$$

Puis, vu que $<\emptyset>_r = \{A \subseteq V \mid |A| \leq r \text{ ou } |\overline{A}| \leq r\}$, on a :

$$|<H>_r| \leq \sum_{i=0}^{r} \binom{n}{i} + \sum_{i=0}^{r} \binom{n}{n-i} + 2|H| \leq 2(r+1)n^r + 2|H|.$$ □

En conséquence, lorsque H est sans r-croisement, H et $<H>_r$ ont approximativement le même nombre d'hyper-arête, à un terme additif en $\mathcal{O}(n^r)$ près. Vu que $H \subseteq <H>_r$, on a aussi $|H| \leq |<H>_r|$. On a finalement l'encadrement $|H| \leq |<H>_r| \leq 2(r+1)n^r + 2|H|$. Cela traduit l'idée informelle que les règles $\boldsymbol{R}_{\leq}$ et $\boldsymbol{R}_{\neg}$ ne peuvent que doubler le nombre d'hyper-arêtes et en ajouter un nombre de l'ordre de $\mathcal{O}(n^r)$. Cet encadrement peut être rendu plus précis en supprimant $<\emptyset>_r$ de chaque hypergraphe présent dans l'inégalité et ainsi formaliser l'idée que la règle $\boldsymbol{R}_{\neg}$ ne peut que doubler le nombre d'hyper-arêtes.

Lemme 5.34. *Soit H un hypergraphe sans r-croisement avec n sommets. On a :*

$$|H \setminus <\emptyset>_r| \leq |<H>_r \setminus <\emptyset>_r| \leq 2|H \setminus <\emptyset>_r|.$$

Démonstration. On pose les quatre hypergraphes suivants (qui ont tous pour ensemble de sommets celui de H) : soit $H' = H \setminus <\emptyset>_r$, soit $J = H \cap <\emptyset>_r$, soit $U = \bigcup_{A\in H}\{A, \overline{A}\}$ et soit $U' = \bigcup_{A\in H'}\{A, \overline{A}\}$. On a alors :

$$U = \bigcup_{A\in H}\{A, \overline{A}\} = \bigcup_{A\in J}\{A, \overline{A}\} \cup \bigcup_{A\in H'}\{A, \overline{A}\} = \bigcup_{A\in J}\{A, \overline{A}\} \cup U' \subseteq <\emptyset>_r \cup U'.$$

D'après le lemme 5.33,$<H>_r = <\emptyset>_r \cup U$. Ainsi, $<H>_r \subseteq <\emptyset>_r \cup U'$. Vu que $U' \subseteq U$, on a aussi $<\emptyset>_r \cup U' \subseteq <\emptyset>_r \cup U = <H>_r$, d'où $<H>_r = <\emptyset>_r \cup U'$.

D'après le lemme 5.24, on sait que $<\emptyset>_r = \{A \subseteq V \mid |A| \leq r \text{ ou } |A| \geq n - r\}$. Vu que $H' = H \setminus <\emptyset>_r$, chaque hyper-arête A de H' satisfait $r < |A| < n - r$. Puisque U' est constitué uniquement d'hyper-arêtes A telles que $A \in H'$ ou $\overline{A} \in H'$, chaque hyper-arête A de U' satisfait $r < |A| < n-r$. Ainsi, U' et $<\emptyset>_r$ sont disjoints, ce qui implique que l'égalité $<H>_r = <\emptyset>_r \cup U'$ peut être réécrite en $<H>_r \setminus <\emptyset>_r = U'$. D'après la définition de U', on a $|H'| \leq |U'| \leq 2|H'|$, ce que l'on réécrit en $|H \setminus <\emptyset>_r| \leq |<H>_r \setminus <\emptyset>_r| \leq 2|H \setminus <\emptyset>_r|$, d'où le résultat. □

Après avoir prouvé ces encadrements, on vérifie qu'être sans r-croisement est une propriété qui est conservée quand on ajoute ou enlève l'opérateur de clôture $<\cdot>_r$.

Lemme 5.35. *Soit H un hypergraphe avec n sommets. Alors H est sans r-croisement si et seulement si $<H>_r$ est sans r-croisement.*

Démonstration. On montre ce lemme par double implication.

($\Longrightarrow$) Si H est sans r-croisement, d'après le lemme 5.33, il satisfait :

$$<H>_r = <\emptyset>_r \cup \bigcup_{A\in H}\{A, \overline{A}\}. \qquad (*)$$

On doit montrer que chaque paire d'hyper-arêtes de $<H>_r$ est r-orthogonale. Soit A, B deux hyper-arêtes de $<H>_r$. Si $A \in <\emptyset>_r$, alors d'après le lemme 5.24, on a $|A| \leq r$ ou $|\overline{A}| \leq r$. Ainsi, avec le point 1 du lemme 5.29, $A \perp_r B$. Le même raisonnement permet de déduire que $B \in <\emptyset>_r$ implique $A \perp_r B$. Il reste donc le cas où $A \notin <\emptyset>_r$ et $B \notin <\emptyset>_r$. Grâce à l'égalité $(*)$, on sait qu'il existe $A' \in H$ tel que $A = A'$ ou $\overline{A'}$ et qu'il existe $B' \in H$ tel que $B = B'$ ou $\overline{B'}$. Puisque H est sans r-croisement, $A' \perp_r B'$. En utilisant le point 5 du lemme 5.29, on déduit $A \perp_r B$.

($\Longleftarrow$) Si $<H>_r$ est sans r-croisement, chaque paire d'hyper-arêtes de $<H>_r$ est r-orthogonale. Puisque $H \subseteq <H>_r$, chaque paire d'hyper-arêtes de H est r-orthogonale et H est sans r-croisement. □

5.3.3 Borne supérieure

Le but de cette section est de borner le nombre maximal d'hyper-arêtes que peut avoir un hypergraphe sans r-croisement à n sommets. En calculant la dimension de Vapnik-Chervonenkis d'un tel hypergraphe, on obtient facilement la borne $\mathcal{O}(n^{2r+1})$. En utilisant l'idée que toutes les hyper-arêtes d'un hypergraphe sans r-croisement sont essentielles, on obtient la même borne que celle sur le nombre d'hyper-arêtes essentielles d'un hypergraphe clos, à savoir $\mathcal{O}(n^{r+1})$. Finalement, en étudiant précisément la définition de la r-orthogonalité, on peut déduire la borne $\mathcal{O}(n^r)$.

Première borne, via la dimension VC

La difficulté de cette approche est de se familiariser avec la dimension VC [VC71]. Ainsi, nous introduisons cette définition progressivement et nous présentons le lemme de Sauer-Shelah, qui permet de relier le nombre d'hyper-arêtes d'un hypergraphe à sa dimension VC.

Définition 5.11 (Restriction d'hypergraphe $H|_C$)**.** *Soit H un hypergraphe et soit C un ensemble de sommets de $V(H)$. La* restriction de H à C*, notée $H|_C$, est l'hypergraphe qui a pour ensemble de sommets C et pour ensemble d'hyper-arêtes $\{A \cap C, A \in \mathcal{E}(H)\}$.*

L'hypergraphe $H|_C$ est en fait la façon la plus naturelle de restreindre l'hypergraphe H à un ensemble de sommets plus petit (ici C). Pour ce faire, on prend chaque hyper-arête de H est on ne garde que les sommets de l'hyper-arête qui sont dans C.

Définition 5.12 (Pulvériser)**.** *Soit H un hypergraphe et soit C un ensemble de sommets de $V(H)$. On dit que H* pulvérise *C si la restriction $H|_C$ est l'hypergraphe complet, c'est-à-dire ayant les $2^{|C|}$ hyper-arêtes possibles.*

Définition 5.13 (Dimension de Vapnik-Chervonenkis [VC71])**.** *Soit H un hypergraphe. La* dimension VC *de H est la taille du plus grand ensemble de sommets C que H pulvérise.*

Autrement dit, si un hypergraphe à une dimension VC d'au moins k, alors il existe une ensemble C de taille k que H pulvérise. En reformulant la notion de pulvérisation, on obtient le lemme suivant :

Lemme 5.36. *Soit H un hypergraphe de dimension VC au moins k. Alors il existe un ensemble C de taille k tel que pour toute bi-partition (C_+, C_-) de C, il existe une hyper-arête $A \in \mathcal{E}(H)$ vérifiant :*
- *l'hyper-arête A contient C_+, c'est-à-dire que $C_+ \subseteq A$ et*
- *l'hyper-arête A évite C_-, c'est-à-dire que $A \cap C_- = \emptyset$.*

Démonstration. Soit H un hypergraphe. Soit d sa dimension VC. Par définition de la dimension VC, il existe un ensemble B de taille d que H pulvérise. On remarque que si H pulvérise un ensemble, il pulvérise aussi tout sous-ensemble de cet ensemble. Soit C un sous-ensemble de B de taille $k \leq d$. On montre que le fait que H pulvérise C est bien équivalent à la condition demandée par le lemme. Par définition de la pulvérisation, $H|_C$ est un hypergraphe complet, c'est-à-dire que pour tout ensemble de sommets $C_+ \subseteq C$, $C_+ \in \mathcal{E}(H|_C)$. Par définition de $H|_C$, il existe une hyper-arête $A \in \mathcal{E}(H)$ telle que $A \cap C = C_+$. Cela signifie que $C_+ \subseteq A$. De plus, en posant $C_- = C \setminus C_+$, on a :

$$A \cap C_- = A \cap (C \setminus C_+) = A \cap (C \cap \overline{C_+}) = (A \cap C) \cap \overline{C_+} = C_+ \cap \overline{C_+} = \emptyset,$$

ce qui conclut. □

Le lemme de Sauer-Shelah permet de borner le nombre d'hyper-arêtes d'un hypergraphe en fonction de sa dimension VC.

Proposition 5.37 (Lemme de Sauer-Shelah [Sau72, She72])**.** *Soit H un hypergraphe à n sommets de dimension VC au plus k. On a l'inégalité :*

$$|\mathcal{E}(H)| \leq \sum_{i=0}^{k} \binom{n}{i}.$$

Avec le lemme 5.36 et lemme de Sauer-Shelah (proposition 5.37), on peut facilement prouver la proposition suivante :

Proposition 5.38. *Soit H un hypergraphe sans r-croisement avec n sommets. Alors le nombre d'hyper-arêtes de H est au plus $\mathcal{O}(n^{2r+1})$.*

Démonstration. Avant toute chose, cette proposition cherche à montrer un comportement asymptotique vis-à-vis de n, avec r fixé. On peut donc supposer n grand devant r, à savoir $n \geq 2r+3$.

On montre d'abord par l'absurde que la dimension VC de H est au plus $2r+1$. On suppose pour cela que H a une dimension VC d'au moins $2r+2$. D'après le lemme 5.36, il existe un ensemble C de taille $2r+2$ vérifiant certaines propriétés. Sans perte de généralité, notons $C = \{1, \ldots, 2+2\}$, c'est-à-dire que C est constitué des entiers entre 1 et $2r+2$. Prenons dans un premier temps la bipartition $(C_+ = \{1, \ldots, r+1\}, C_- = \{r+2, \ldots, 2r+2\})$. D'après le lemme 5.36, il existe une hyper-arête $A \in \mathcal{E}(H)$ qui contient C_+ (c'est-à-dire les sommets entre 1 et $r+1$) et évite C_- c'est-à-dire les sommets entre $r+2$ et $2r+2$. Prenons dans un second temps la bipartition $(C_+ = \{2, \ldots, r+2\}, C_- = \{1\} \cup \{r+3, \ldots, 2r+2\})$. D'après le lemme 5.36, il existe une hyper-arête $B \in \mathcal{E}(H)$ qui contient C_+ (c'est-à-dire les sommets entre 2 et $r+2$) et évite C_- c'est-à-dire le sommet 1 ainsi que les sommets entre $r+3$ et $2r+2$. La disposition des hyper-arêtes A et B est représentée sur la figure 5.6. Finalement, on montre que les hyper-arêtes A et B ne sont pas r-orthogonales :

- On a $A \cap B \supseteq \{2, \ldots, r+1\}$ et donc $|A \cap B| \geq r$.
- On a $A \not\subseteq B$ car le sommet 1 est dans A mais pas dans B.
- On a $B \not\subseteq A$ car le sommet $r+2$ est dans B mais pas dans A.
- On a $\overline{A \cup B} \supseteq \{r+3, \ldots, 2r+2\}$ et donc $|\overline{A \cap B}| \geq r$.
- Si $|A \cap B| = |\overline{A \cup B}| = r$ alors C est l'ensemble total de sommets de H, ce qui signifie que $n = 2r+2$, ce qui est exclu par hypothèse.

Ainsi, d'après le corollaire 5.28, les hyper-arêtes A et B ne sont pas r-orthogonales, ce qui contredit le fait que H est sans r-croisement. Par l'absurde, on a donc que la dimension VC de H est au plus $2r+1$.

Finalement, on conclut grâce au lemme de Sauer-Shelah, qui nous dit ici que le nombre d'hyper-arêtes de H est au plus $\sum_{i=0}^{2r+1} \binom{n}{i} = \mathcal{O}(n^{2r+1})$. □

Deuxième borne, via les hyper-arêtes essentielles

Il est possible d'améliorer sensiblement cette borne en utilisant les lemmes que l'on a prouvés dans la section 5.3.2.

Proposition 5.39. *Soit H un hypergraphe sans r-croisement avec n sommets. Alors le nombre d'hyper-arêtes de H est au plus $\mathcal{O}(n^{r+1})$.*

Démonstration. On considère $<H>_r$, qui est un hypergraphe dans $\mathbb{K}_r(n)$ par définition. D'après le théorème 5.1, il existe un hypergraphe H' ayant au plus $\mathcal{O}(n^{r+1})$ hyper-arête et tel que $<H>_r = <H'>_r$. D'après le lemme 5.35, vu que H est sans r-croisement, $<H>_r$ est aussi sans r-croisement, de même que H'. On peut donc appliquer le lemme 5.33 à H', qui donne l'inégalité $|<H'>_r| \leq 2(r+1)n^r + 2|H'|$. Or, $<H'>_r = <H>_r$ et $H \subseteq <H>_r$, ce qui donne en l'inégalité $|H| \leq |<H'>_r|$. En combinant les deux inégalités précédentes, on obtient $|H| \leq 2(r+1)n^r + 2|H'|$. Or, on a dit que $|H'| = \mathcal{O}(n^{r+1})$, ce qui permet de déduire que $|H| \leq \mathcal{O}(n^{r+1})$, d'où le résultat. □

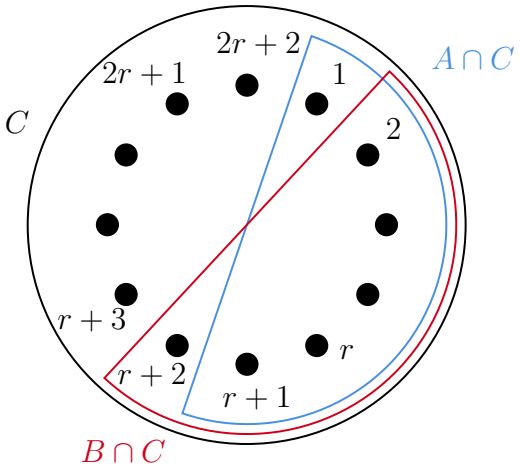


FIGURE 5.6 – Ensemble C avec les hyper-arêtes A et B dont on a représenté que l'intersection avec C. Les sommets de C sont numérotés de 1 à $2r+2$ et sont disposés en cercle sur la figure, en croissant dans le sens horaire (comme sur une horloge). On ne sait pas comment se comportent A et B en dehors de C, mais les sommets qu'elles contiennent et évitent à l'intérieur de C suffisent pour déduire que A et B ne sont pas r-orthogonales.

Troisième borne, via l'orthogonalité

On peut avoir une meilleure borne en étant plus précis. En développant le corollaire 5.28, on obtient :

Lemme 5.40. *Soit $V = [n]$ et soit $A \subseteq V$ et $B \subseteq V$. On a l'équivalence suivante.*

$$
\begin{aligned}
&A \perp_r B \\
&\quad \iff A \subseteq B \vee B \subseteq A \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee \\
&\qquad (|A \cap B| < r \wedge |A \setminus B| < r) \vee \\
&\qquad (|A \cap B| < r \wedge |B \setminus A| < r) \vee \\
&\qquad (|A \cap B| < r \wedge |A \setminus B| = |B \setminus A| = r) \vee \\
&\qquad (|\overline{A \cup B}| < r \wedge |A \setminus B| < r) \vee \\
&\qquad (|\overline{A \cup B}| < r \wedge |B \setminus A| < r) \vee \\
&\qquad (|\overline{A \cup B}| < r \wedge |A \setminus B| = |B \setminus A| = r) \vee \\
&\qquad (|A \setminus B| < r \wedge |A \cap B| = |\overline{A \cup B}| = r) \vee \\
&\qquad (|B \setminus A| < r \wedge |A \cap B| = |\overline{A \cup B}| = r) \vee \\
&\qquad |A \setminus B| = |B \setminus A| = |A \cap B| = |\overline{A \cup B}| = r.
\end{aligned}
$$

En se concentrant sur la taille des ensembles, on déduit ce qui suit.

Corollaire 5.41. *Soit $V = [n]$ et soit $A \subseteq V$ et $B \subseteq V$. On a l'implication suivante.*

$$\begin{aligned} & A \perp_r B \\ & \quad \Longrightarrow A \subseteq B \vee B \subseteq A \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset \vee \\ & \qquad |A| < 2r - 1 \vee \\ & \qquad |B| < 2r - 1 \vee \\ & \qquad (|A| < 2r \wedge |B| < 2r) \vee \\ & \qquad |B| > n - 2r + 1 \vee \\ & \qquad |A| > n - 2r + 1 \vee \\ & \qquad (|A| > n - 2r \wedge |B| > n - 2r) \vee \\ & \qquad (|A| < 2r \wedge |B| > n - 2r) \vee \\ & \qquad (|A| > n - 2r \wedge |B| < 2r) \vee \\ & \qquad n = 4r. \end{aligned}$$

Démonstration. On montre la façon dont on déduit l'implication suivante.

$$(|A \cap B| < r \wedge |A \setminus B| < r) \implies |A| < 2r - 1.$$

Puisque $A = (A \cap B) \cup (A \setminus B)$, on a $A \leq |A \cap B| + |A \setminus B| \leq (r-1) + (r-1) < 2r - 1$. On procède de la même manière pour chaque terme de l'équivalence du lemme 5.40. □

Cette implication détaillée révèle le fait que si A et B sont r-orthogonales, alors soit une des quatre conditions parmi $A \subseteq B$, $B \subseteq A$, $A \cap B = \emptyset$ ou $\overline{A \cup B} = \emptyset$ est vraie, soit l'un des deux ensembles A ou B est contraint en taille. Notamment, cette taille doit être petite (inférieur à $2r$) ou grande (supérieure à $n - 2r$). Dès lors, si l'on s'intéresse uniquement aux ensembles A et B de taille ni trop petite ni trop grande, on déduit le corollaire suivant.

Corollaire 5.42. *Soit $V = [n]$ et soit $A \subseteq V$ et $B \subseteq V$. Si $n > 4r$, $2r \leq |A| \leq n - 2r$ et $2r \leq |B| \leq n - 2r$, alors on a :*

$$A \perp_r B \iff A \subseteq B \vee B \subseteq A \vee A \cap B = \emptyset \vee \overline{A \cup B} = \emptyset.$$

Démonstration. On procède par double inclusion.

($\Longrightarrow$) Si A et B sont r-orthogonales, d'après le corollaire 5.41, on sait que l'on a $A \subseteq B$, $B \subseteq A$, $A \cap B = \emptyset$, $\overline{A \cup B} = \emptyset$, une condition sur la taille de A ou B, ou le fait que $n = 4r$. Comme on a pris pour hypothèse le fait que $2r \leq |A| \leq n - 2r$, $2r \leq |B| \leq n - 2r$ et $n > 4r$, il ne reste que les possibilités $A \subseteq B$, $B \subseteq A$, $A \cap B = \emptyset$ et $\overline{A \cup B} = \emptyset$.

($\Longleftarrow$) Si l'une des quatre conditions est vraies, à fortiori, l'une des conditions du membre droit de l'équivalence du lemme 5.40 est vraie, d'où $A \perp_r B$. □

Grâce à ce corollaire, on déduit une borne sur le nombre d'hyper-arêtes de chaque taille d'un hypergraphe sans r-croisement H, si tant est que le nombre de sommets de H est suffisamment grand devant r.

Lemme 5.43. *Soit r un entier non nul fixé et soit $n \geq 5r$. Soit $V = [n]$. Soit H un hypergraphe sans r-croisement (c'est-à-dire que pour tout $A, B \in \mathcal{E}(H), A \perp_r B$). On définit également les cinq sous-ensembles d'hyper-arêtes de H suivants :*

— *Soit $\mathcal{E}_1$ l'ensemble des hyper-arêtes A de H telles que $0 \leq |A| \leq r$.*

— *Soit $\mathcal{E}_2$ l'ensemble des hyper-arêtes A de H telles que $r+1 \leq |A| \leq 2r-1$.*
— *Soit $\mathcal{E}_3$ l'ensemble des hyper-arêtes A de H telles que $2r \leq |A| \leq n-2r$.*
— *Soit $\mathcal{E}_4$ l'ensemble des hyper-arêtes A de H telles que $n-2r+1 \leq |A| \leq n-r-1$.*
— *Soit $\mathcal{E}_5$ l'ensemble des hyper-arêtes A de H telles que $n-r \leq |A| \leq n$.*

Alors on a l'inégalité $|\mathcal{E}_i| \leq 2n^r$ pour chaque i.

Démonstration. On procède en trois points, selon la valeur de i :

— Pour $i = 1$ et $i = 5$, $\mathcal{E}_i$ est un ensemble dont la taille est majorée par le nombre de sous-ensembles d'un ensemble à r éléments (il s'agit des éléments que l'on choisit concernant $\mathcal{E}_1$ et des éléments que l'on évite concernant $\mathcal{E}_5$), c'est-à-dire 2^r. À fortiori, $2^r \leq 2n^r$.

— Pour $i = 2$, on prouve d'abord par l'absurde qu'étant donné un ensemble V_+ de r sommets, il y au plus $r-1$ hyper-arêtes de $\mathcal{E}_2$ qui contiennent tous les sommets de V_+. Supposons qu'il y ait au moins r hyper-arêtes de cette forme. Puisque que chacune des hyper-arêtes de $\mathcal{E}_2$ a une taille comprise entre $r+1$ et $2r-1$, au moins deux hyper-arêtes différentes de $\mathcal{E}_2$ qui contiennent tous les sommets de V_+ ont la même taille. Appelons A et B ces deux hyper-arêtes de même taille. Par choix de A et B, on a $|A \cap B| \geq |V_+| = r$. Comme A et B sont des ensembles différents de même taille, ils ne peuvent pas être inclus l'un dans l'autre. On a donc $A \not\subseteq B$ et $B \not\subseteq A$. De plus, $|A \cup B| \leq |A| + |B| \leq 4r-2$. Ainsi, $|\overline{A \cup B}| \geq n - 4r + 2 \geq 5r - 4r + 2 = r + 2$, ce qui implique à la fois $|\overline{A \cup B}| \geq r$ et $|\overline{A \cup B}| \neq r$. Finalement, on a $|A \cap B| \geq r \wedge A \not\subseteq B \wedge B \not\subseteq A \wedge |\overline{A \cup B}| \geq r \wedge |\overline{A \cup B}| \neq r$. D'après le corollaire 5.28, cela implique que A et B ne sont pas r-orthogonales, ce qui est une contradiction. Ainsi, il y au plus $r-1$ hyper-arêtes de $\mathcal{E}_2$ qui contiennent tous les sommets de V_+.

Comme une hyper-arête de $\mathcal{E}_2$ a au moins $r+1$ sommets, il existe $r+1$ ensembles V_+ différents de r sommets qui sont inclus dans une hyper-arête de $\mathcal{E}_2$ donnée. Alors, si on construit une liste L en concaténant pour chaque ensemble V_+ de r sommets la liste des hyper-arêtes de $\mathcal{E}_2$ qui contiennent V_+, chaque hyper-arête apparaitra au moins $r+1$ fois dans L. Comme il y a $\binom{n}{r} \leq n^r$ choix de V_+, il y a au total au plus $(r-1)n^r$ hyper-arêtes dans L. Ainsi, le nombre d'hyper-arêtes différentes dans L est au plus $(r-1)n^r/(r+1) \leq 2n^r$ et chaque hyper-arête de $\mathcal{E}_2$ apparait bien dans L (puisqu'elle apparait au moins $r+1$ fois). D'où $|\mathcal{E}_2 \leq 2n^r|$.

— Pour $i = 4$, on effectue le même raisonnement que pour $i = 2$, en remplaçant l'ensemble V_+ de r sommets qu'il faut contenir par un ensemble V_- de r sommets qu'il faut éviter.

— Pour $i = 3$, le corollaire 5.42 implique que l'ensemble des hyper-arêtes de $\mathcal{E}_3$ ont une structure arborescente, ce qui signifie que la taille de $\mathcal{E}_3$ est linéaire en n. On a précisément $|\mathcal{E}_3| \leq 2n \leq 2n^r$. □

Théorème 5.2. *Soit H un hypergraphe sans r-croisement avec n sommets. Alors le nombre d'hyper-arêtes de H est au plus $\mathcal{O}(n^r)$.*

Démonstration. On partitionne l'ensemble des arêtes de H en $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5$ comme dans le lemme 5.43. D'après ce lemme, on sait que si $n > 5r$, on a $|\mathcal{E}_i| \leq 2n^r$ pour chaque i. Ainsi, on a bien $|\mathcal{E}(H)| = |\mathcal{E}_1| + |\mathcal{E}_2| + |\mathcal{E}_3| + |\mathcal{E}_4| + |\mathcal{E}_5| = \mathcal{O}(n^r)$ quand n tend vers l'infini. □

Cette borne est atteinte, comme montré dans la section suivante.

5.3.4 Borne inférieure

On sait qu'un hypergraphe sans r-croisement a au plus $\mathcal{O}(n^r)$ hyper-arêtes. Dans cette section, on prouve que certains hypergraphes sans r-croisement peuvent avoir au moins $\Omega(n^r)$ hyper-arêtes. Par conséquent, grâce au lemme 5.34 qui induit que le nombre d'hyper-arêtes d'un hypergraphe r-clos sans r-croisement est approximativement le même que le nombre nombre d'hyper-arêtes de n'importe quel hypergraphe dont il est la clôture, on montre qu'il existe des hypergraphes r-clos H qui nécessitent au moins $\Omega(n^r)$ hyper-arêtes pour être représentés (c'est-à-dire que tout hypergraphe dont la clôture est H a nécessairement au moins $\Omega(n^r)$ hyper-arêtes). Ainsi, on obtient une borne inférieure sur le nombre d'hyper-arêtes nécessaires pour représenter certains hypergraphes r-clos en utilisant les hypergraphes sans r-croisement.

Lemme 5.44. *Pour tout $r \geq 1$, il existe un ensemble infini d'hypergraphes sans r-croisements $\{H_n\}$ ayant chacun n sommets et un nombre d'hyper-arêtes supérieur à $\Omega(n^r)$ (avec n qui tend vers l'infini).*

Démonstration. Soit $r \geq 1$. Soit n un multiple de $r+1$, c'est-à-dire qu'on a $n = k(r+1)$ pour un certain entier k. On construit un hypergraphe avec n sommets tels que chaque paire d'hyper-arêtes est r-orthogonale. Pour ce faire, on considère l'ensemble de sommets $V := \mathbb{Z}/k\mathbb{Z} \times [r+1]$, où $\mathbb{Z}/k\mathbb{Z}$ dénote l'ensemble des entiers modulo k. On interprète l'ensemble V ainsi : à chaque sommet est assignée une valeur entre 0 et $k-1$ ainsi qu'une couleur parmi $r+1$ couleurs possibles. La valeur est utilisée modulo k dans les équations qui suivent, d'où la nécessité de la prendre dans l'anneau $\mathbb{Z}/k\mathbb{Z}$ plutôt que dans l'ensemble $\{0, \ldots, k-1\}$. On considère maintenant l'ensemble suivant d'hyper-arêtes :

$$\mathcal{E} := \left\{ A = \{(v_i, c_i)\}_{i=1}^{r+1} \in \binom{V}{r+1} \;\middle|\; \{c_i\}_{i=1}^{r+1} = [r+1] \text{ et } \sum_{i=1}^{r+1} v_i \equiv 0 \mod k \right\}$$

En français, $\mathcal{E}$ est la famille de tous les ensembles de $r+1$ sommets de V tels que :

— les sommets ont des couleurs différentes deux à deux et

— la somme de la valeur de chacun des sommets vaut 0 modulo k.

Puisqu'il y a $r+1$ couleurs au total et $r+1$ sommets dans une hyper-arête, chaque hyper-arête contient exactement un sommet de chaque couleur. On considère maintenant l'hypergraphe H_n tel que $V(H_n) = V$ et $\mathcal{E}(H_n) = \mathcal{E}$.

D'abord, on remarque que si l'on fixe r sommets qui ont des couleurs distinctes deux à deux, il n'y a qu'une seule hyper-arête qui contient ces r sommets. En effet, notons ces r sommets (v_i, c_i), avec $1 \leq i \leq r$. Soit (v_{r+1}, c_{r+1}) un sommet. Pour que $\{(v_i, c_i)\}_{i=1}^{r+1}$ soit une hyper-arête, c_{r+1} doit être la couleur qui manque dans $\{(v_i, c_i)\}_{i=1}^{r}$ et v_{r+1} doit être égal à $-\sum_{i=1}^{r} v_i \mod k$. Puis, si $A, B \in \mathcal{E}$ sont deux hyper-arêtes de H telles que $|A \cap B| = r$, alors $A = B$. On montre maintenant que pour chaque paire (A, B) d'hyper-arêtes distinctes, $A \perp_r B$. On sait que désormais $|A \cap B| < r$ puisque $A \neq B$. Ensuite, on a soit $|A \setminus B| < r$, ce qui implique $A \perp_r B$ avec $|A \cap B| < r$ et le corollaire 5.28, ou soit $|A \setminus B| = r$, ce qui implique $|A \cap B| = 1$, puis $|B \setminus A| = r$ et donc $A \perp_r B$ via le même corollaire 5.28. Par conséquent, H_n est un hypergraphe sans r-croisement.

Maintenant, calculons la cardinalité de $\mathcal{E}$. Sans perte de généralité, on peut réordonner les sommets de chaque hyper-arête de sorte que $c_i = i$. Il n'y a donc aucun choix à faire concernant les couleurs de chaque sommet d'une hyper-arête. Pour $1 \leq i \leq r$, la valeur v_k peut être choisie parmi k valeurs différentes. Pour v_{r+1}, il n'y a qu'un seul

choix possible : $v_{r+1} = -\sum_{i=1}^{r} v_i \mod k$. Ainsi, $|\mathcal{E}| = k^r$. Puisque $n = k(r+1)$, on a $|\mathcal{E}| = k^r = (r+1)^{-r}\, n^r = \Omega(n^r)$, vu que r est fixé et n tend vers l'infini. □

Théorème 5.3. *Soit $r > 0$ un entier fixé. Il existe un ensemble infini d'hypergraphes sans r-croisements $\{H_n\}$, chacun avec n sommets, telle que pour chaque hypergraphe H_n et pour chaque hypergraphe H' à n sommets, si $<H'>_r = <H_n>_r$, alors H' a au moins $\Omega(n^r)$ hyper-arêtes (avec r fixé et n qui tend vers l'infini).*

Démonstration. Soit $\{H_n\}$ l'ensemble infini d'hypergraphes définie dans le lemme 5.44. Soit H' un hypergraphe tel que $<H'>_r = <H_n>_r$. L'hypergraphe H_n est sans r-croisement, donc d'après le lemme 5.35, $<H_n>_r$ et H' sont aussi sans r-croisement. Ainsi, on peut appliquer le lemme 5.34 à l'hypergraphe H', ce qui s'écrit $2|H' \setminus <\emptyset>_r| \geq |<H'>_r \setminus <\emptyset>_r|$. On a donc $2|H'| \geq |<H'>_r \setminus <\emptyset>_r|$, c'est-à-dire $2|H'| \geq |<H_n>_r \setminus <\emptyset>_r|$. De plus, $H_n \subseteq <H_n>_r \setminus <\emptyset>_r$ puisque $H_n \subseteq <H_n>_r$ et puisque chaque hyper-arête A de H_n a une taille vérifiant $r < |A| < n-r$, ce qui signifie que $H_n \cap <\emptyset>_r = \emptyset$. Ainsi, $|<H_n>_r \setminus <\emptyset>_r| \geq |H_n|$. On rappelle que le nombre d'hyper-arêtes de H_n satisfait $\Omega(n^r)$. Ainsi, puisque $2|H'| \geq |H_n|$, le nombre d'hyper-arêtes de H' est au moins $\Omega(n^r)$. □

Si on compare les différentes bornes sur les hypergraphes r-clos, cela signifie que certains hypergraphes r-clos nécessitent au moins $\Omega(n^r)$ pour être représentés, là où le corollaire 5.22 garantit que l'on peut toujours le faire en moins de $\mathcal{O}(n^{r+1})$ hyper-arêtes.

5.4 Algorithmique

Afin de pouvoir utiliser les sections précédentes en pratique, on se demande quelle est la complexité pour calculer l'ensemble des r-splits essentiels d'un graphe G. Pour cela, on considère la complexité de calculer un r-split essentiel. Soit X un ensemble de sommets de G et soit H l'hypergraphe des r-splits de G, on cherche à calculer $\varphi_H(X)$, c'est-à-dire $\min\{A \in \mathcal{E}(H) \mid X \subseteq A, |A| \leq n/2\}$. Comme une hyper-arête de H est un ensemble de sommets A tel que $\rho(A) \leq r$, avec ρ une fonction sous-modulaire, et que ρ ne peut pas valoir moins de r sur l'ensemble $\{A \in \mathcal{E}(H) \mid X \subseteq A, |A| \leq n/2\}$, on déduit que $\varphi_H(X)$ est égal au plus petit ensemble A appartenant à $\{A \in \mathcal{E}(H) \mid X \subseteq A, |A| \leq n/2\}$ qui minimise $\rho(A)$.

On est donc amené à un problème de minimisation de fonction sous-modulaire (ou SFM, pour *Submodular Function Minimization*), qui est un problème très étudié. Voir [McC05] pour une vue d'ensemble et [Kra10] pour des implémentations. Le problème SFM consiste à trouver la valeur minimale d'une fonction sous-modulaire f. De plus, il peut-être aussi demandé de trouver l'argument qui minimise cette valeur de f. Cependant, étant donné deux arguments A et B qui minimisent f, par sous-modularité, $A \cup B$ et $A \cap B$ minimisent aussi f. Ainsi, il existe un argument minimal et un argument maximal qui minimisent f. On demande alors en général de trouver l'argument maximal qui minimise f. Dans [IO09], l'algorithme `SFMwave` permet de trouver l'argument maximal en temps faiblement polynomial.

Proposition 5.45 ([IO09])**.** *Soit $f : 2^E \to \mathbb{N}$ une fonction sous-modulaire sur un ensemble E de taille n. Soit EO la complexité pour évaluer la fonction f. Soit M la valeur maximale que peut prendre f. Alors il est possible de calculer l'unique solution maximale A qui minimise $f(A)$ en temps $\mathcal{O}((n^4 EO + n^5)\log(nM))$.*

Cet algorithme est dit faiblement polynomial car il dépend de façon polynomiale de $\log M$; un algorithme fortement polynomial serait un algorithme qui ne dépend pas du tout de M ; en revanche, un algorithme qui dépend de façon polynomiale de M est appelé quasi-polynomial [McC05].

Dans notre cas, $M \leq n$ car une matrice de taille n par n a un rang inférieur ou égal à n. De plus, le rang d'une matrice n par n peut être calculé en temps $\mathcal{O}(n^\omega)$ en utilisant l'élimination Gaussienne [BH74, IMH82], ce qui donne une borne sur EO.

Il est possible de restreindre l'espace de recherche à un ensemble de la forme $\{A \mid X \subseteq A \subseteq \overline{Y}\}$, où X et Y sont deux ensembles disjoints fixés. En effet, étant donné une fonction sous-modulaire $f : 2^E \to \mathbb{Z}$ et deux ensembles disjoints X et Y, on déduit que la fonction $g : 2^{E\setminus(X\cup Y)} \to \mathbb{Z}$ définie par $g(A) = f(X \cup A) - f(X)$ est aussi sous-modulaire. Elle est appelée le *résidu de f par rapport à X et Y*. Le fait de soustraire $f(X)$ n'est pas nécessaire pour que le résidu soit sous-modulaire. Cela permet simplement de faire en sorte que $g(\emptyset) = 0$, c'est-à-dire que g soit normalisée. Ainsi, en appliquant un algorithme SFM au résidu de f, on restreint la recherche du minimum à l'ensemble $\{A \mid X \subseteq A \subseteq \overline{Y}\}$.

On peut donc calculer $\varphi_H(X, Y)$ pour tout Y de taille r via un algorithme SFM et ensuite calculer $\varphi_H(X)$ en utilisant le lemme 5.23. Finalement, cette approche permet de calculer un r-split essentiel $\varphi_H(X)$ en temps $\mathcal{O}(n^{r+4+\omega} \log n)$ et permet donc de calculer l'ensemble des $\mathcal{O}(n^{r+1})$ r-splits essentiels en temps $\mathcal{O}(n^{2r+5+\omega} \log n)$, qui est alors la complexité pour calculer le graphe H du théorème 5.1.

Théorème 5.4. *Étant donné un graphe G r-rang connecté d'ordre n, il est possible de construire un hypergraphe H avec $\mathcal{O}(n^{r+1})$ hyper-arêtes qui vérifie $<H>_r = H_r(G)$ en temps $\mathcal{O}(n^{2r+5+\omega} \log n)$, où $\omega \approx 2.37$ est lié à la multiplication de matrices.*

5.5 Perspectives

Structure d'arbre. J'ai pu obtenir une structure polynomiale permettant de représenter les r-splits. Cependant, les théorèmes sur les splits indiquent qu'il est possible de représenter les splits sous une forme arborescente. On peut donc se demander s'il en est de même pour les r-splits. Dans le cas des splits, l'idée est de ne représenter dans l'arbre que les splits qui sont orthogonaux avec tous les autres splits (appelés *splits forts*) et de déduire les autres splits par union de certains splits forts. Les splits forts forment une famille sans croisement, ce qui permet de les représenter facilement sous forme d'arbre. Dans le cas des r-splits, les r-splits forts (qui sont les r-splits r-orthogonaux avec tous les autres r-splits du graphe) forment un hypergraphe sans r-croisement. Le corollaire 5.42 montre que l'ensemble des r-splits forts dont la taille n'est ni trop petite ni trop grande a une structure d'arbre au même titre que les splits. Pour en déduire une structure d'arbre pour les r-splits, il resterait à considérer les splits forts dont la taille est inférieure à $2r$ ou supérieure à $n - 2r$ et il faudrait également pouvoir exprimer tout r-split comme une combinaison de r-splits forts.

Application à la compression. Un r-split X étant associée à une coupe $(X, \overline{X})$ dont le rang est inférieur ou égal à r, il permet une certaine compression du graphe G. En effet, la matrice de $G[X, \overline{X}]$ peut être représentée par $r(n - r)$ bits, ce qui est toujours mieux que la représentation explicite en $|X|(n - |\overline{X}|)$ bits, étant donné que $r \leq |X| \leq n - r$. Cependant, la facteur de compression dépend de la taille de X : plus celle-ci est loin de r et $n-r$, meilleur est la compression. Dans le meilleur des cas, obtenu pour $|X| = |\overline{X}| = n/2$,

on représente avec $r(n-r)$ bits une matrice de taille $n^2/4$. On dit alors qu'un r-split X est *équilibré* si $|X| \approx |\overline{X}|$. L'idée est alors de chercher les r-splits équilibrés de G. Il serait alors intéressant de voir comment la théorie développée dans ce chapitre pourrait permettre de les chercher.

Complexité. Dans la section 5.4, j'ai présenté une façon de calculer en temps polynomial une représentation de l'hypergraphe des r-splits d'un graphe donné. Une idée alternative pour calculer cette représentation est de s'inspirer de ce qui est fait pour la décomposition modulaire et pour les splits. Dans ce cas, on sait que l'opérateur de clôture associé à la relation orthogonale est le même que l'opérateur de clôture qui rend une famille partitive (cf. proposition 4.4). De même, l'opérateur de clôture associé à la relation croix-orthogonale est le même que l'opérateur de clôture qui rend une famille symétrique traversante. On peut se demander si l'opérateur clôture induit par la relation de r-orthogonalité (au sens du lemme 2.11) est le même que l'opérateur de r-clôture. On peut montrer que non avec un exemple minimal constitué d'un hypergraphe H ayant deux hyper-arêtes qui ne sont pas r-orthogonales. La r-clôture $<H>_r$ est strictement incluse dans $\mathrm{cl}_{\perp_r}(H)$, qui est l'hypergraphe H auquel on a appliqué l'opérateur de clôture induit par la relation de r-orthogonalité. On se demande alors s'il existe une relation permettant d'exprimer la r-clôture comme l'opérateur de clôture induit par cette relation, sachant que ce n'est pas toujours possible (cf. lemme 2.12). Si oui, est-ce qu'il permettrait de calculer la représentation de l'hypergraphe des r-splits d'un graphe donné plus rapidement ?

Chapitre 6

Recherche de motifs dans les graphes ordonnés

Sommaire

Dans ce chapitre, je présente un travail fait avec trois autres chercheurs : Michel Habib, Laurent Feuilloley et Guillaume Ducoffe. Ce travail est en cours de publication à SIDMA.

Nous nous intéressons ici à la recherche de motifs dans les graphes ordonnés. Il s'agit de la continuité de ce qui a pu être présenté dans les chapitres précédents, excepté que nous nous intéressons ici à des graphes ordonnés, c'est-à-dire qui ont leurs sommets ordonnés. Là où nous cherchions des structures dans le chapitre 3, des modules et des splits dans le chapitre 4 et des r-splits dans le chapitre 5, nous cherchons ici des *motifs* qui sont des sous-graphes ordonnés. Ils imposent des contraintes qui sont à mi-chemin entre celles de la recherche de sous-graphe et celles de la recherche de sous-graphe induit.

La section 6.1 présente les définitions précises d'un motif et d'un graphe ordonné puis expose la motivation du problème ainsi qu'un aperçu de l'ensemble des résultats qui seront prouvés dans ce chapitre. Pour commencer, la section 6.2 s'intéressera aux motifs n'ayant que 3 ou 4 sommets afin de mieux comprendre les techniques de recherche de motif. Les travaux de cette section ont été faits originalement par Michel Habib. Je les présente par soucis d'exhaustivité tout en proposant des preuves légèrement différentes des siennes. La section 6.3 se concentre sur la recherche de plusieurs motifs simultanément, ce qui est dans certains cas plus facile que la recherche d'une seul motif. Il s'agit d'un travail fait en collaboration avec Michel Habib. Ensuite, nous regarderons les motifs de taille arbitraire mais appartenant à une certaine classe. Cela sera fait dans la section 6.4, en élargissant progressivement la classe de motifs considérée. Cela a été étudié avec l'aide de Laurent Feuilloley. Enfin, la section 6.5 introduit un nouveau paramètre de graphe permettant de mesurer la complexité nécessaire pour rechercher un motif quelconque dans un graphe ordonné. Nous utiliserons pour cela des tenseurs, dont la définition ainsi que la présentation de certains de leurs produits est rappelé dans la section 2.4. Ce travail a aussi été fait en collaboration avec Laurent Feuilloley.

6.1 Introduction

6.1.1 Définitions

Graphe ordonné

Dans ce chapitre, on s'intéresse aux graphes *ordonnés*. Dans ce cadre, l'ensemble des sommets est muni d'une relation d'ordre total, de sorte qu'étant donné deux sommets distincts, on puisse dire que l'un des sommets est plus grand que l'autre.

Définition 6.1 (Ordre)**.** *Soit E un ensemble fini. Un* ordre τ *de E est une liste des différents éléments de E. Si x et y sont deux éléments de E, on dit que x est plus petit que y si x apparait avant y dans la liste τ. On note $x <_\tau y$.*

Ainsi, l'ordre τ est une façon d'induire et de représenter un ordre total sur l'ensemble E. Voici des exemples d'ordre.

Exemple 6.1. Soit $E = \{1, 2, 3, 4\}$. L'ordre $\tau = [1, 2, 3, 4]$ représente l'ordre habituel sur les entiers $1, 2, 3, 4$. Ainsi, on a les inégalités $1 <_\tau 2 <_\tau 3 <_\tau 4$, ainsi que l'inégalité $1 <_\tau 3$, l'inégalité $2 <_\tau 4$ et l'inégalité $1 <_\tau 4$.

On a également besoin de la notion d'*ordre miroir*, d'*ordre restreint* et de *sous-ordre* :

Définition 6.2 (Ordre miroir $\texttt{miroir}(\tau)$)**.** *Soit E un ensemble et τ ordre sur E. L'ordre miroir de τ, noté $\texttt{miroir}(\tau)$ est l'ordre tel que pour tout $x, y \in E$, $x <_{\texttt{miroir}(\tau)} y \iff y <_\tau x$.*

Exemple 6.2. Soit $E = \{1, 2, 3, 4\}$ et $\tau = [1, 2, 3, 4]$. L'ordre $\texttt{miroir}(\tau)$ est l'ordre $[4, 3, 2, 1]$.

Définition 6.3 (Ordre restreint $\tau|_E$)**.** *Soit deux ensembles $E_1 \subset E_2$. Si τ_2 un ordre sur E_2 alors $\tau_2|_{E_1}$ est l'ordre sur E_1 obtenu en ne gardant que les éléments de E_1.*

Définition 6.4 (Sous-ordre ou ordre étendu $\tau_1 \prec \tau_2$)**.** *Soit deux ensembles $E_1 \subset E_2$. Si τ_1 un ordre sur E_1 et τ_2 un ordre E_2, on dit que l'ordre τ_1 est* un sous-ordre *de l'ordre τ_2 (ou que l'ordre τ_2* étend *l'ordre τ_1) si $\tau_2|_{E_1} = \tau_1$, c'est-à-dire si on obtient la liste τ_1 en prenant la liste τ_2 et en enlevant les éléments qui ne sont pas dans E_1. On note $\tau_1 \prec \tau_2$.*

Exemple 6.3. Soit $E_1 = \{2, 3, 4, 5\}$ et $E_2 = \{1, 2, 3, 4, 5, 6, 7\}$. L'ordre $\tau_1 = [2, 5, 3, 4]$ est un sous-ordre de l'ordre $\tau_2 = [2, 5, 6, 1, 3, 7, 4]$ car si on enlève les éléments qui ne sont pas dans E_1 (c'est-à-dire 1, 6 et 7), il reste $[2, 5, \not{6}, \not{1}, 3, \not{7}, 4] = [2, 5, 3, 4]$.

Finalement, on peut définir un graphe ordonné :

Définition 6.5 (Graphe ordonné)**.** *Un* graphe ordonné *est une paire (G, τ), où G est un graphe (au sens habituel) et où τ est un ordre sur les sommets de G.*

Remarque 6.1. On rappelle que tous les graphes considérés dans ce manuscrit sont symétriques, ce qui signifie que si (x, y) est une arête de G alors (y, x) l'est aussi. Ainsi, quand on décrira un graphe ordonné, on se permettra de ne lister que l'une des deux arêtes.

En général, quand cela est possible, on choisit comme ensemble de sommets du graphe les entiers de 1 à n et on prend comme ordre τ la liste des entiers de 1 à n dans l'ordre croissant, de sorte que la relation d'ordre $<_\tau$ induite par τ soit la relation d'ordre classique sur les entiers $<$. Pour représenter un graphe ordonné (G, τ), en général, on aligne les différents sommets $V(G)$ sur une ligne et on les ordonne dans le même ordre que τ, de sorte à ce que le sommet le plus petit soit tout à gauche et le plus grand tout à droite. La figure 6.1 montre un exemple de graphe ordonné.

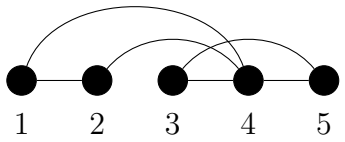


FIGURE 6.1 – Exemple de graphe ordonné. Les arcs représentent les arêtes du graphe. Afin de faciliter la lecture de la représentation, les arêtes reliant deux sommets consécutifs (comme 1 et 2 sur cet exemple) sont représenté par un segment plutôt qu'un arc de cercle. Ce graphe a donc 6 arêtes qui sont $(1, 2), (1, 4), (2, 4), (3, 4), (3, 5)$ et $(4, 5)$. On ne liste pas les arêtes symétriques (à savoir $(2, 1), (4, 1), (4, 2), (4, 3), (5, 3)$ et $(5, 4)$), comme dit dans la remarque 6.1.

Motif

Étant donné un graphe ordonné, on se demande si l'on peut y trouver des sommets qui respectent un certain nombre de contraintes. On s'intéresse à trois types de contraintes :

— les sommets cherchés sont dans un certain ordre,

— certaines paires de sommets sont reliées par une arête,

— et certaines paires de sommets ne sont pas reliées par une arête.

Exemple 6.4. Étant donné un graphe ordonné (G, τ), on se demande s'il est possible de trouver trois sommets a, b, c tels que :

— l'ordre de ces sommets est $a <_\tau b <_\tau c$,

— il y a une arête entre a et c,

— et il n'y a pas d'arête entre a et b.

Ces contraintes motivent la définition d'un motif, qui est une façon concise et visuelle de représenter un tel ensemble de contraintes. Cela est représenté en figure 6.2.

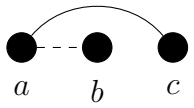


FIGURE 6.2 – Représentation visuelle des contraintes de l'exemple 6.4. L'ordre des sommets indique que $a <_\tau b <_\tau c$. Le trait plein entre a et c demande à ce qu'il y ait une arête entre a et c. Le trait en pointillés entre a et b demande à ce qu'il n'y ait pas d'arête entre a et b.

Définition 6.6 (Motif)**.** *Un* motif P *est un objet constitué :*

— *d'un ensemble de sommets* $V(P)$*,*

— *d'un ensemble d'arêtes obligatoires* $M(P)$*,*

— *d'un ensemble d'arêtes interdites* $F(P)$*,*

— *d'un ordre sur les sommets* $\tau(P)$*,*

tel que $M(P) \cap F(P) = \emptyset$*. On note* $E(P) = M(P) \cup F(P)$ *l'ensemble des arêtes (obligatoires ou interdites) du motif.*

Le fait que $M(P) \cap F(P) = \emptyset$ permet d'assurer qu'une arête ne peut pas à la fois être obligatoire et interdite. Tout comme un graphe ordonné, un motif peut être représenté visuellement.

Remarque 6.2. Dans le continuité de la remarque 6.1, comme on ne considère que des graphes symétriques, si une arête (x, y) appartient au motif P, alors non seulement l'arête symétrique (y, x) appartient à P, mais surtout les arêtes (x, y) et (y, x) sont du même type, c'est-à-dire soit toutes les deux obligatoires, soit toutes les deux interdites.

Exemple 6.5. L'exemple 6.4 donnait un ensemble de contraintes à respecter. On peut maintenant utiliser un motif pour représenter ces contraintes :

— Comme on s'intéresse à trois sommets a, b, c, on prend $V(P) = \{a, b, c\}$.

— Vu que l'on veut $a <_\tau b <_\tau c$, on prend $\tau(P) = [a, b, c]$.

— Puisqu'on veut une arête entre a et c, on prend $M(P) = \{(a, c)\}$.

— Puisqu'on ne veut pas d'arête entre a et b, on prend $F(P) = \{(a, b)\}$.

Tout cela se représente sous forme visuelle, comme cela a été fait sur la figure 6.2. Un trait plein représente une arête obligatoire (c'est-à-dire dans $M(P)$). Un trait en pointillés représente une arête interdite (c'est-à-dire dans $F(P)$). L'absence de trait entre b et c signifie qu'il peut y avoir une arête ou qu'il peut ne pas y en avoir entre b et c.

Étant donné un motif, on peut définir le *motif miroir* et le *motif complémentaire* de ce motif. Le premier est le motif obtenu en prenant le miroir de l'ordre et le second est le motif obtenu en inversant arête et non-arête :

Définition 6.7 (Motif miroir $\texttt{miroir}(P)$)**.** *Soit P un motif. Le motif miroir de P, noté $\textit{miroir}(P)$, est le motif tel que $V(\textit{miroir}(P)) = V(P)$, $M(\textit{miroir}(P)) = M(P)$, $F(\textit{miroir}(P)) = F(P)$, $\tau(\textit{miroir}(P)) = \textit{miroir}(\tau(P))$.*

Définition 6.8 (Motif complémentaire $\overline{P}$)**.** *Soit P un motif. Le motif complémentaire de P, noté $\overline{P}$, est le motif tel que $V(\overline{P}) = V(P)$, $M(\overline{P}) = F(P)$, $F(\overline{P}) = M(P)$, $\tau(\overline{P}) = \tau(P)$.*

Il ne reste plus qu'à définir formellement le fait qu'un graphe ordonné respecte les contraintes induites par un motif, c'est-à-dire définir ce que signifie contenir ou éviter un motif.

Définition 6.9 ((G, τ) contient ou évite P, f réalise P dans (G, τ))**.** *Soit (G, τ) un graphe ordonné et soit P un motif. On dit que (G, τ)* contient *P lorsqu'il existe une fonction f des sommets $V(P)$ du motif vers les sommets $V(G)$ du graphe telle que :*

- *la fonction f conserve l'ordre : pour tous sommets $x, y \in V(P)$, on a $x <_{\tau(P)} y$ implique $f(x) <_\tau f(y)$,*
- *la fonction f respecte les arêtes obligatoires : pour toute arête $(x, y) \in M(P)$, on a $(f(x), f(y)) \in E(G)$,*
- *la fonction f respecte les arêtes interdites : pour toute arête $(x, y) \in F(P)$, on a $(f(x), f(y)) \notin E(G)$.*

Dans ce cas, on dit que f réalise *P dans (G, τ). Quand il n'existe pas de telle fonction f, on dit que le graphe ordonné (G, τ)* évite *le motif P.*

Exemple 6.6. On considère le graphe ordonné (G, τ) et le motif P de la figure 6.3. On rappelle que ces représentations signifient que :

- $V(G) = \{1, 2, 3, 4, 5\}$ et $V(P) = \{a, b, c\}$,
- $\tau = [1, 2, 3, 4, 5]$ et $\tau(P) = [a, b, c]$,
- $E(G) = \{(1, 2), (1, 4), (2, 4), (3, 4), (3, 5), (4, 5)\}$ et $E(P) = \{(a, b), (a, c)\}$,
- parmi les arêtes de P, il y a deux types d'arêtes, qui sont $M(P) = \{(a, c)\}$ et $F(P) = \{(a, b)\}$.

On considère la fonction $f : V(P) \to V(G)$ définie par $f(a) = 1$, $f(b) = 3$ et $f(c) = 4$. On vérifie que f réalise P dans (G, τ) en vérifiant les trois points de la définition 6.9 :

- La fonction f conserve l'ordre puisque $a <_{\tau(P)} b <_{\tau(P)} c$ et $f(a) <_\tau f(b) <_\tau f(c)$. Autrement dit, la fonction f est croissante.
- La fonction f respecte les arêtes obligatoires puisque la seule arête obligatoire de P est (a, c) et on a bien que $(f(a), f(c)) = (1, 4)$ est une arête de G.
- La fonction f respecte les arêtes interdites puisque la seule arête interdite de P est (a, b) et on a bien que $(f(a), f(b)) = (1, 3)$ n'est pas une arête de G.

Ainsi, le graphe ordonné (G, τ) contient le motif P. D'ailleurs, il le contient à un autre endroit, puisqu'il existe une autre fonction f qui réalisent P dans (G, τ), à savoir la fonction f définie par $f(a) = 2$, $f(b) = 3$ et $f(c) = 4$.

Lemme 6.1. *Si f réalise P dans (G, τ) alors f est injective.*

Démonstration. Soit $x, y \in V(P)$ tels que $f(x) = f(y)$. Si $x \neq y$, sans perte de généralité, on a $x <_{\tau(P)} y$ et donc $f(x) <_\tau f(y)$, ce qui contredit le fait que $f(x) = f(y)$. □

(a) Graphe ordonné (G, τ). (b) Motif P.

FIGURE 6.3 – Le graphe ordonné (G, τ) que l'on avait déjà représenté en figure 6.1 et le motif P que l'on avait déjà représenté en figure 6.2.

Lemme 6.2. *Si f réalise P dans (G, τ) alors $x <_{\tau(P)} y \iff f(x) <_\tau f(y)$,*

Démonstration. On sait que l'implication $x <_{\tau(P)} y \implies f(x) <_\tau f(y)$ est vraie par définition. On montre $x <_{\tau(P)} y \impliedby f(x) <_\tau f(y)$ par disjonction de cas. On suppose que l'on a $f(x) <_\tau f(y)$. Puisque l'ordre $\tau(P)$ est total sur $V(P)$, on a soit $x = y$, soit $y <_{\tau(P)} x$, soit $x <_{\tau(P)} y$. Le premier cas implique que $f(x) = f(y)$ ce qui est faux. Le deuxième cas implique $f(y) <_{\tau(P)} f(x)$ car f conserve l'ordre, ce qui est faux. Par principe du tiers exclu, c'est le troisième cas qui est vrai. □

6.1.2 Motivation

Un graphe est *cordal* lorsque tout cycle de taille au moins 4 contient une *corde*, c'est-à-dire une arête reliant deux sommets qui ne sont pas adjacent dans le cycle. Une caractérisation des graphes cordaux et qu'un graphe G est cordal si, et seulement si, il admet un *ordre d'élimination simplicial*, c'est-à-dire qu'il existe un ordre τ des sommets de G tel que le voisinage à gauche (suivant l'ordre τ) de chaque sommet de G forme une clique [FG65]. Autrement dit, si l'on prend un sommet u de G ainsi que v_1 et v_2 deux voisins à gauche de u suivant l'ordre τ (c'est-à-dire tels que $v_1 \leq_\tau u$ et $v_2 \leq_\tau u$), alors v_1 et v_2 sont voisins. En terme de motif, cela implique immédiatement qu'un graphe G est cordal si, et seulement si, il existe un ordre τ tel que le graphe ordonné (G, τ) évite le motif P décrit dans la figure 6.4 [FH21b].

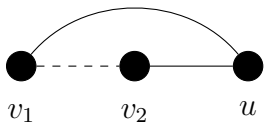


FIGURE 6.4 – Motif permettant de certifier qu'un graphe est cordal.

Il est montré dans [HMPV00] qu'étant donné un graphe G, il est possible de trouver en temps linéaire un ordre τ tel que G est cordal si et seulement si le graphe ordonné (G, τ) évite le motif de la figure 6.4. Il s'agit de l'ordre *Lex-BFS*. Ainsi, il n'a pas besoin de vérifier tous les ordres possibles pour déterminer si un graphe est cordal, il suffit de vérifier l'ordre Lex-BFS. Enfin, étant donné un graphe ordonné (G, τ), il est montré dans [HMPV00] que l'on peut déterminer en temps linéaire via du raffinement de partition si (G, τ) contient le motif de la figure 6.4 ou non. Finalement, [HMPV00] montre que l'on peut détecter si un graphe est cordal en deux étapes élémentaires :

1. on commence par déterminer un ordre τ,
2. puis on vérifie si le graphe ordonné (G, τ) contient ou évite ce motif.

Le fait que chacune des ces étapes puisse être faite en temps linéaire permet d'avoir un algorithme efficace pour détecter si un graphe est cordal.

On peut alors se poser la même question pour d'autres classes de graphe. Dans [FH21b], il est montré qu'un certain nombre de classes peuvent être caractérisées au moyen d'un motif interdit, comme pour la classe des graphes cordaux. Ces classes seront détaillées dans la section 6.2.2. Deux questions se posent alors pour chacune de ces classes :

1. Quelle est la complexité pour déterminer un ordre τ tel que G appartient à cette classe si et seulement si le graphe ordonné (G, τ) évite le motif associé à cette classe ?
2. Quelle est la complexité pour déterminer si le graphe ordonné (G, τ) évite le motif associé à cette classe ?

Le premier problème a été résolu dans [FH21b]. Si la classe de graphe est caractérisée par un motif interdit ayant trois sommets, il y a toujours un algorithme linéaire qui permet de calculer τ. Il reste alors la seconde question, à laquelle nous nous sommes intéressés : Étant donné un motif P, quelle est la complexité nécessaire pour détecter si un graphe ordonné contient ou évite ce motif ?

6.1.3 État de l'art

Avant de tenter de répondre à cette question, regardons ce qui a déjà état fait dans le domaine des graphes ordonnés ainsi que la question de la détection de motifs dans d'autres domaines.

Adaptation aux graphes ordonnés. Tout d'abord, la plupart des problèmes qui sont étudiés sur les graphes peuvent être adaptés pour les graphes ordonnés. Par exemple, la notion de *nombre chromatique* est adaptée en imposant que les différentes classes de couleur soient des intervalles du graphe ordonné. Cela rend le nombre chromatique ordonné plus facile à calculer que sa version non-ordonnée étant donné qu'il est calculable en temps linéaire via un algorithme glouton [PT06], alors que son homologue non-ordonné est connu pour être NP-complet. Cette notion est utilisée pour étendre la *théorie extrémale* aux graphes ordonnés [Tar18]. Cela consiste à se demander quel est le nombre d'arêtes que peut avoir au maximum un graphe qui évite un motif. Dans le cas non-ordonné, il existe une formule qui relie ce nombre au nombre chromatique du motif à éviter [ES46]. Dans le cas ordonné, la même formule est obtenue [PT06], à savoir $(1-1/(\chi-1))n^2/2+o(n^2)$, où χ est le nombre chromatique (éventuellement ordonné) du motif à éviter. Quand le nombre chromatique est 2, la formule donne $o(n^2)$ et une étude plus approfondie est nécessaire. Ainsi, il est prouvé dans [Tar18] que si le motif à éviter est de nombre chromatique 2 tout en étant un couplage, alors ce $o(n^2)$ est en réalité un $\mathcal{O}(n)$.

Éviter un motif : classe et ordre. Concernant les graphes qui évitent un motif donné, il a été montré pour les petits motifs qu'une classe de graphe évitant un motif était toujours une classe de graphe déjà connue. Ainsi, [Dam90] caractérise les classes de graphes qui évitent un motif complet à 3 sommets et [FH21b] caractérise celles qui évitent un ensemble de motifs à 3 sommets. Cela sera utilisé par la suite pour nommer les motifs à 3 sommets en fonction du nom de la classe de graphes induite. De plus, [FH21a] tente de caractériser certaines classes de graphes qui évitent un motif à 4 sommets en s'intéressant à celles obtenues par construction géométrique (c'est-à-dire une généralisation des graphes d'intervalle).

Une question connexe consiste, étant donné un motif fixé, à construire un algorithme qui prend en entré un graphe non-ordonné et produit en sortie un ordre des sommets, de sorte que s'il existe un ordre du graphe qui évite le motif, alors l'ordre produit est un de ses ordres. Par exemple, on a vu que si le motif est celui qui caractérise la classe des graphes cordaux, un algorithme de complexité linéaire existe en ordonnant les sommets suivant l'ordre Lex-BFS. Cependant, il n'existe pas toujours d'algorithme polynomial pour se faire. Par exemple, d'après le théorème de Gallai-Hasse-Roy-Vitaver, un graphe est k-colorable si, et seulement si, il existe un ordre des sommets qui évite le motif P_{k+1}, c'est-à-dire le chemin (non-induit) à $k+1$ sommets. Or, dès que $k \geq 3$, il est connu qu'il est NP-complet de savoir si un graphe est k-colorable. Ainsi, trouver un ordre qui évite le motif P_{k+1} pour $k \geq 3$ est NP-complet.

Recherche de motifs. Bien que la recherche de motif dans les graphes ordonnés soit assez nouvelle, elle a beaucoup été étudié dans d'autres domaines. Tout d'abord, elle l'a été dans le cas des *permutations*. Dans ce contexte, à la fois le graphe G et le motif P doivent être des permutations, c'est-à-dire des graphes ordonnés dans lesquels les arêtes doivent représenter un couplage où la première extrémité d'une arête doit être parmi la première moitié des sommets et la seconde extrémité parmi la seconde moitié des sommets. Dans ce cas très restreint, il existe toujours un algorithme en temps $2^{\mathcal{O}(k \log k)}n$, où k est le nombre de sommets de P et n celui de G [GM14]. Ainsi, si l'on fixe le motif P, on a un algorithme linéaire en G. Si on autorise G à être quelconque et P à être un couplage quelconque (pas nécessairement une permutation), les récents papiers sur la *largeur de jumeau* (*twin-width* en anglais) [BGOdM$^+$22, BGdMT22] suggèrent un algorithme quasi-quadratique en $\mathcal{O}(n^2 \log n)$.

Concernant la recherche de motifs dans les graphes non-ordonnés, il existe de nombreux résultats. Le cas général est connu pour être NP-complet si le motif est pris en entrée, avec une borne triviale en $\mathcal{O}(n^k)$. Pour certaines classes de motifs (arbres, graphes à seuil, chaines biparties, complémentaires de chaine bipartie), il existe des algorithmes polynomiaux [KOSU12], c'est-à-dire que l'exposant ne dépend pas de la taille k du motif. De même, pour certaines classes de graphes, on peut rendre le problème polynomial en n. Ainsi, si le graphe G de taille n est planaire et le motif P de taille k est quelconque, on peut détecter P dans G en temps $2^{\mathcal{O}(k)}n$ [Dor09]. Plus généralement, si le graphe G a une largeur de jumeau (*twin-width*) d'au plus d, il existe un algorithme en temps $2^{\mathcal{O}(k \log k)}d^{2k}n$, pourvu que l'on dispose de la décomposition associée à la largeur de jumeau [BGK$^+$20].

Si on fixe un motif P de taille k mais qu'on laisse le graphe G être aussi général que possible, on peut se demander si on peut faire mieux que $\mathcal{O}(n^k)$. Dans le cas général, surprenamment, oui. Il existe un algorithme en $\mathcal{O}(n^{\beta(k)})$ et, pour $k \geq 6$, en $\mathcal{O}(m^{\beta(k)/2})$, avec $\beta(k) := \omega(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)$ et où $\omega(a,b,c)$ désigne l'exposant de la multiplication de matrices rectangulaires [EG04]. Pour le cas des petits motifs, il est possible de faire mieux que ces bornes. Pour $k = 3$, il y a quatre motifs possibles : le motif triangle et son complémentaire peuvent être détectés en $\mathcal{O}(n^\omega)$ et $\mathcal{O}(m^{2\omega/(\omega+1)})$ [AYZ97] tandis que le motif chemin et son complémentaire peuvent être détectés en temps linéaire $\mathcal{O}(n+m)$ [HKSS13]. Pour $k = 4$, on peut détecter n'importe quel motif en temps $\mathcal{O}(n^\omega)$ et $\mathcal{O}(m^{2\omega/(\omega+1)})$ avec haute probabilité, excepté la clique K_4 et son complémentaire [WWWY14]. De plus, un chemin induit de taille 4 peut être détecté en temps linéaire $\mathcal{O}(n+m)$ en faisant un Lex-BFS [BCHP08]. Pour $k = 5$, certains motifs peuvent être détectés plus efficacement qu'en temps $\mathcal{O}(n^{\beta(5)})$ avec $\beta(5) = \omega(1,2,2) \leq 2\omega(0.5,1,1) \leq 2 \times 2.046681 < 4.09$ (cf. [LG12] pour la meilleure valeur

connue de $\omega(0.5, 1, 1)$). De fait, dans [FKLL15], il est proposé un algorithme probabiliste en $\mathcal{O}(n^4)$ pour certains motifs à 5 sommets. Dans le cas des motifs plus grands, [KLL13] propose un tableau récapitulatif pour les cycles, les chemins et les motifs de largeur arborescente bornée ou de stable maximal borné. Dans ces cas-là, un algorithme polynomial pour lequel l'exposant est indépendant de k existe.

6.1.4 Problématique et réponse naïve

Étant donné un motif P, on s'intéresse à la complexité nécessaire pour détecter si un graphe ordonné contient ce motif ou si au contraire il l'évite. Si on note $k = |V(P)|$ le nombre de sommets de P et $n = |V(G)|$ le nombre de sommets du graphe ordonné (G, τ) donné en paramètre, il y a un algorithme naïf qui répond à cette question en temps $\mathcal{O}(n^k)$.

L'idée de cet algorithme est de vérifier pour chaque ensemble de k sommets $1 \leq i_1 < \ldots < i_k \leq n$ de G si la fonction $f : V(P) \to V(G)$ telle que $f(1) = i_1$, $f(2) = i_2$, et ainsi de suite jusqu'à $f(k) = i_k$, est une réalisation de P dans (G, τ). Cela est fait en vérifiant que chaque arête obligatoire du motif P est présente dans (G, τ) et que chaque arête interdite du motif P est absente dans (G, τ).

Algorithme 2 : Recherche naïve d'un motif P de k sommets.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient P » ou « G évite P »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
1  Pour chaque ensemble de k entiers 1 ≤ i_1 < ... < i_k ≤ n faire
2      contient ← true
3      Pour chaque paire d'entiers 1 ≤ a < b ≤ k faire
4          Si (i_a, i_b) ∈ M(P) et (i_a, i_b) ∉ E(G) alors
5              contient ← false
6          Si (i_a, i_b) ∈ F(P) et (i_a, i_b) ∈ E(G) alors
7              contient ← false
8      Si contient = true alors
9          Retourner « G contient P »
10 Retourner « G évite P »
```

Correction. Si (G, τ) contient P, il existe une réalisation $f : V(P) \to V(G)$ de P dans (G, τ). En notant $f(V(P)) = \{i_1, i_2, \ldots, i_k\}$ avec $1 \leq i_1 < \ldots < i_k \leq n$, il existe une itération de la boucle de la ligne 1 qui vérifie l'ensemble de k entiers $1 \leq i_1 < \ldots < i_k \leq n$. Puisque f respecte l'ordre, cela signifie que $f(1) = i_1$, $f(2) = i_2$, et ainsi de suite jusqu'à $f(k) = i_k$. Il reste alors à vérifier que f respecte bien les arêtes obligatoires et les arêtes interdites de P, ce qui est fait via la boucle de la ligne 3 et les conditions aux lignes 4 et 6. Si (G, τ) évite P, il n'existe aucune des ces réalisations, et la boucle de la ligne 1 les vérifient toutes.

Complexité. Concernant la complexité de l'algorithme 2, on peut supposer que $k \geq 2$ (autrement le motif P est juste un sommet et seul le graphe vide ne contient pas le motif P). On effectue alors un pré-calcul de la matrice d'adjacence de G en temps

quadratique afin de pouvoir effectuer les tests $(i_a, i_b) \notin E(G)$ et $(i_a, i_b) \in E(G)$ en temps constant. À propos des tests $(i_a, i_b) \in M(P)$ et $(i_a, i_b) \in F(P)$, l'algorithme ne prend pas P en argument mais peut, au contraire, être réécrit en fonction de P (on parle de méta-algorithme). On peut alors déplier la boucle de la ligne 3 en la remplaçant par une suite de tests de la forme $(i_a, i_b) \notin E(G)$ ou $(i_a, i_b) \in E(G)$, dans lesquels chaque paire (a, b) est remplacée par une paire concrète de valeurs. Cela signifie que les tests $(i_a, i_b) \in M(P)$ et $(i_a, i_b) \in F(P)$ ne sont pas faits pendant l'exécution de l'algorithme 2, mais qu'ils sont réalisés pendant la transformation du méta-algorithme en algorithme. On a alors bien un algorithme avec une complexité de $\mathcal{O}(n^k)$ puisque la boucle de la ligne 1 est effectuée $\binom{n}{k}$ fois et puisque le contenu de cette boucle s'effectue en temps constant.

Méta-algorithme. Ce concept de méta-algorithme va revenir tout au long de ce chapitre. L'idée précise est d'avoir un méta-algorithme qui, étant donné un motif P ayant k sommets, produit un algorithme $\mathcal{A}_P$. Cet algorithme $\mathcal{A}_P$ prend en argument un graphe ordonné (G, τ) ayant n sommets et renvoie un booléen indiquant si (G, τ) contient P. Ce qui nous intéresse dans ce chapitre est la complexité de $\mathcal{A}_P$ en fonction de n (et éventuellement du nombre d'arêtes m de G). La complexité pour passer du méta-algorithme à l'algorithme $\mathcal{A}_P$ n'est jamais évaluée. Dans tous les cas, cette complexité est une fonction de k (qui est une constante vis à vis de $\mathcal{A}_P$) et ne change donc pas la complexité asymptotique de $\mathcal{A}_P$ (pour laquelle c'est n et m qui tendent vers l'infini).

Motif miroir. On a le lemme suivant qui permet de passer au motif miroir en ajoutant seulement une complexité linéaire :

Lemme 6.3. *Soit $\mathcal{A}$ un algorithme qui permet de dire si un motif P apparait dans n'importe quel graphe ordonné (G, τ) donné en entrée. Soit $f(n, m)$ la complexité de $\mathcal{A}$, avec $n = |V(G)|$ et $m = |E(G)|$. Alors il existe un algorithme $\mathcal{A}'$ qui permet de dire si le motif $\texttt{miroir}(P)$ apparait dans n'importe quel graphe ordonné (G, τ) donné en entrée, avec une complexité $\mathcal{O}(n + m) + f(n, m)$.*

Démonstration. L'algorithme $\mathcal{A}'$, qui prend en argument un graphe ordonné (G, τ) est obtenu en prenant le miroir de l'ordre τ afin obtenir $(G, \texttt{miroir}(\tau))$ et en appliquant $\mathcal{A}$ à $(G, \texttt{miroir}(\tau))$. Autrement dit, au lieu de prendre le miroir de l'ordre de P, on prend le miroir de l'ordre de (G, τ), car cela est équivalent. Sans perte de généralité, on peut supposer que $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$. Ainsi, $(G, \texttt{miroir}(\tau))$ est obtenu en remplaçant dans toutes les arêtes le sommet i par le sommet $n - i + 1$, ce qui se fait en temps $\mathcal{O}(n + m)$. □

Motif complémentaire. On a aussi le lemme suivant qui permet de passer au motif complémentaire en ajoutant une complexité quadratique :

Lemme 6.4. *Soit $\mathcal{A}$ un algorithme qui permet de dire si un motif P apparait dans n'importe quel graphe ordonné (G, τ) donné en entrée. Soit $f(n, m)$ la complexité de $\mathcal{A}$, avec $n = |V(G)|$ et $m = |E(G)|$. Alors il existe un algorithme $\mathcal{A}'$ qui permet de dire si le motif $\overline{P}$ apparait dans n'importe quel graphe ordonné (G, τ) donné en entrée, avec une complexité $\mathcal{O}(n^2) + f(n, m)$.*

Démonstration. L'algorithme $\mathcal{A}'$, qui prend en argument un graphe ordonné (G, τ) est obtenu en prenant le complémentaire du graphe G afin obtenir $(\overline{G}, \tau)$ et en appliquant

$\mathcal{A}$ à $(\overline{G}, \tau)$. Autrement dit, au lieu de prendre le complémentaire de P, on prend le complémentaire de G, car cela est équivalent. Le graphe $(\overline{G}, \tau)$ est obtenu en calculant la matrice d'adjacence de (G, τ), en en prenant le complémentaire, puis en la parcourant afin d'obtenir la liste des arêtes de $(\overline{G}, \tau)$. Cela se fait en temps $\mathcal{O}(n^2)$. □

Concernant le passage au complémentaire, on verra qu'il est souvent possible d'avoir une meilleure complexité, en adaptant directement l'algorithme $\mathcal{A}$ pour qu'il travaille sur le complémentaire de G sans avoir besoin de le calculer, permettant ainsi de garder la complexité de $\mathcal{A}$ et de ne pas ajouter un terme quadratique.

Le but des sections suivantes est d'obtenir des algorithmes plus performants que l'algorithme naïf dans le cas où le motif P n'est pas quelconque.

6.1.5 Résultats

Au cours de cette thèse, nous avons pu établir un certain nombre de théorème concernant la recherche de motif. Voici un aperçu des théorèmes qui seront démontrés dans ce chapitre.

Petits motifs

Les premiers théorèmes concernent la recherche de petits motifs, pour lesquels nous nous sommes demandés s'il était possible d'obtenir des algorithmes linéaires. Nous avons commencé par étudier les motifs à trois sommets. Le premier théorème a été énoncé et prouvé par Michel Habib. Je mentionne ce théorème ici car j'ai participé à la rédaction détaillée de certaines preuves. Les preuves présentées dans ce manuscrit sont donc les miennes, même si elles trouvent leur inspiration dans certains travaux de Michel Habib.

Théorème 6.1. *Soit P un motif à trois sommets qui n'est pas* Triangle*, co-*Triangle*,* Comparability *ou co-*Comparability*. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n + m)$.*

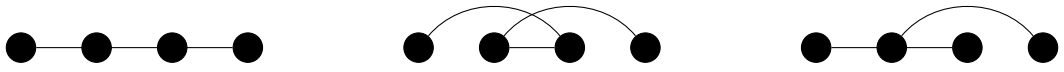
Dans ce théorème, les motifs Triangle, co-Triangle, Comparability ou co-Comparability font référence aux motifs de la figure 6.5 et appartiennent à une nomenclature plus large définie dans [FH21b] et rappelée dans la suite de ce chapitre

(a) Triangle (b) co-Triangle (c) Comparability (d) co-Comparability

Figure 6.5 – Les motifs exclus par le théorème 6.1.

Après avoir étudié les motifs à trois sommets, nous avons étudié ceux en ayant quatre. Cependant, là où il est possible de considérer l'ensemble des motifs à trois sommets à la main, il n'est plus envisageable de considérer ceux à quatre sommets. Nous nous sommets donc intéressé aux chemins, qui sont au nombre de huit quand on considère tous les ordres possibles. Cela mène au théorème 6.2, qui inclut également les étoiles en plus des chemins. La figure 6.6 montre des exemples de motifs concernés par ce théorème.

Théorème 6.2. *Soit P un motif à quatre sommets contenant trois arêtes obligatoires et aucune arête interdite. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n + m)$.*



(a) Un chemin. (b) Un autre chemin. (c) Une étoile.

FIGURE 6.6 – Exemple de motif concernés par le théorème 6.2.

Une autre question amenée par Michel Habib concerne les ensembles de motifs. Au lieu de se demander si un graphe ordonné contient ou évite un seul motif, on se demande s'il contient un des motifs d'un ensemble S ou s'il évite tous les motifs de S. L'algorithme naïf consisterait alors à tester un par un tous les motifs de S. On a voulu voir s'il était possible de faire mieux et on s'est intéressé au cas particulier où tous les motifs ont trois sommets. Dans ce cas, le théorème 6.3 indique que la réponse est affirmative. En effet, dans le cas où S contient un des motifs exclu par le théorème 6.1 (et rappelé en figure 6.5), un algorithme linéaire ne semble pas possible à première vue. Cependant, le théorème 6.3 montre qu'il est toujours possible d'avoir un algorithme linéaire.

L'idée est la suivante : si $S = \{P_1, P_2\}$ est constitué d'un motif P_1 de la figure 6.5 ainsi que d'un autre motif P_2 qui n'est pas dans la figure 6.5, on peut d'abord tester en temps linéaire si P_2 est présent dans (G, τ). Si oui, on sait que (G, τ) contient un motif de S. Si non, on peut utiliser le fait que (G, τ) évite P_2 et appliquer un algorithme linéaire permettant de détecter P_1 dans ce cas précis. De façon contre-intuitive, le théorème marche aussi quand les deux motifs sont dans la figure 6.5. Dans ce cas, soit P_1 et P_2 induisent une classe de graphe finie (c'est le cas si P_1 est TRIANGLE et P_2 co-TRIANGLE, car tout graphe ayant au moins 6 sommets contient soit TRIANGLE soit co-TRIANGLE d'après le théorème de Ramsey), soit la classe de graphe est très rigide (c'est le cas si P_1 est COMPARABILITY et P_2 co-COMPARABILITY, auquel cas un graphe évitant ces deux motifs simultanément doit être un graphe de permutation, ce qui est détectable en temps linéaire).

Théorème 6.3. *Soit S un ensemble de motifs à trois sommets qui n'est pas un singleton constitué de* TRIANGLE*,* co-TRIANGLE*,* COMPARABILITY *ou* co-COMPARABILITY*. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient au moins un motif de S ou s'il les évite tous en temps linéaire $\mathcal{O}(n + m)$.*

Motifs appartenant à une classe

Le but des théorèmes précédents était surtout de se familiariser avec les algorithmes de détection de motifs. À présent, nous pouvons énoncer des théorèmes généraux qui s'appliquent à des classes entières de motif. Un motif peut avoir certaines propriétés qui aident à les détecter dans un graphe. Tout d'abord, il peut être planaire extérieur, ce qui signifie que dans la représentation habituelle du motif (comme sur la figure 6.7), les arêtes ne se croisent pas. Un motif peut aussi être sans cycle, ce qui signifie que le graphe associé (en oubliant le caractère obligatoire ou interdit des arêtes) ne possède pas de cycle. Enfin, un motif peut être positif, c'est-à-dire qu'il ne possède aucune arête interdite. Quand on impose ces trois conditions, on peut détecter le motif en temps linéaire, comme le montre le théorème 6.4.

Théorème 6.4. *Soit P un motif planaire extérieur positif sans cycle. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n + m)$.*

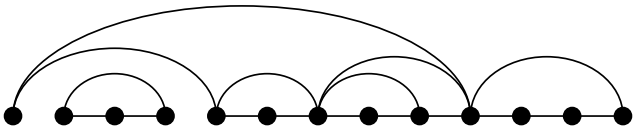


FIGURE 6.7 – Exemple d'un motif planaire extérieur positif avec 14 sommets et 17 arêtes. Chaque arête doit être représentée par une ligne droite ou un arc de cercle qui reste au-dessus des sommets et aucune de ces arêtes ne doit en croiser une autre.

Si on enlève la condition « positif », on peut détecter le motif en temps quadratique, comme énoncé dans le théorème 6.5. En effet, pour traiter les non-arêtes du graphe ordonné, on doit construire sa matrice d'adjacence, ce qui prend un temps quadratique.

Théorème 6.5. *Soit P un motif planaire extérieur sans cycle. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps quadratique $\mathcal{O}(n^2)$.*

Enfin, en enlevant en plus la condition « sans cycle », on peut détecter le motif en temps égal à celui de la multiplication de matrice, comme cela a été énoncé dans le théorème 6.6. On rappelle qu'étant donné deux matrices de taille n par n, ω représenté le nombre minimum tel que la multiplication de ces matrices se fasse en complexité temporelle $\mathcal{O}(n^\omega)$ [CW87]. La borne actuelle la plus précise est $\omega < 2.3728596$ [AW21]. Cette complexité semble optimale puisqu'il faut faire une multiplication de matrices pour détecter ne serait-ce qu'un triangle et que la classe des motifs planaires extérieurs contient le motif triangle.

Théorème 6.6. *Soit P un motif planaire extérieur. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps sous-cubique $\mathcal{O}(n^\omega)$.*

Largeur de fusion d'un motif

Il reste la question des motifs quelconques. J'ai alors introduit un paramètre, la largeur de fusion (ou *merge-width* en anglais), permettant de mesurer la difficulté d'un motif quelconque, dans le sens où on peut détecter un motif de largeur de fusion borné en temps polynomial, où le degré du polynomial est proportionnel à la largeur de fusion du motif. Cela est énoncé précisément dans le théorème 6.7.

Théorème 6.7. *Soit P un motif et soit $p = mw(P)$ sa largeur de fusion. On peut décider si un graphe ordonné (G, τ) à n sommets contient ou évite P en temps polynomial $\mathcal{O}(n^2 + n^{\omega p/2})$.*

La définition exacte de la largeur de fusion sera donnée au fil de la section 6.5. Elle ressemble à la largeur de clique dans le sens où on construit le motif à partir de motifs élémentaires et d'opérations entre les motifs. Ici, l'opération que l'on peut effectuer entre deux motifs est la fusion. Comme pour la largeur de clique qui cherche à minimiser le nombre de couleurs utilisées dans sa décomposition, la largeur de fusion cherche à minimiser le nombre d'ancres utilisées dans sa décomposition sous forme d'arbre. Une ancre est un marqueur sur un sommet d'un motif qui permet de préciser comment se fait la fusion de ce motif avec un autre. Intuitivement, plus il y a d'arêtes qui se croisent, plus il faudra d'ancres par motif. Comme toujours lorsque l'on introduit un nouveau paramètre de graphe, il convient de le comparer à d'autres paramètres existants. La largeur de fusion est facilement comparable avec la largeur arborescente (en anglais, *treewidth* [RS86], notée

tw). La largeur arborescente d'un motif est alors définie comme la largeur arborescente du graphe sous-jacent, c'est-à-dire en oubliant le caractère obligatoire ou interdit des arêtes et en oubliant l'ordre des sommets. On obtient alors le théorème 6.8.

Théorème 6.8. *Pour tout motif P, $tw(P) \leq \frac{3}{2}mw(P) - 1$, si $mw(P) \geq 2$.*

Cependant, on aimerait borner supérieurement la largeur de fusion, de sorte à pouvoir appliquer le théorème 6.7 en calculant un paramètre de graphe plus connu (mais peut-être moins précis). La largeur arborescente ne fonctionne pas dans ce sens à moins de faire intervenir un paramètre logarithmique. Nous avons donc opté pour la largeur de chemin (en anglais, pathwidth [RS83]). Cependant, pour avoir une borne supérieure qui ait du sens, il faut la comparer à un paramètre qui prenne en compte l'ordre des sommets du graphe. On modifie alors légèrement la définition de la largeur de chemin. Cela se fait naturellement car la définition de la largeur de chemin est un minimum sur tous les ordres possibles de sommet. Il suffit d'imposer l'ordre des sommets du motif. On obtient alors le théorème 6.9, où opw désigne la largeur de chemin ordonné (en anglais, *ordered pathwidth*).

Théorème 6.9. *Soit P un motif. On a $mw(P) \leq opw(P) + 2$.*

6.2 Recherche de petits motifs

On s'intéresse dans un premier temps aux motifs ayant 3 ou 4 sommets.

6.2.1 Algorithmique linéaire

Soit (G, τ) un graphe ordonné tel que $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$. On commence par rappeler les pré-traitement qui peuvent être effectué sur (G, τ) en temps linéaire $\mathcal{O}(n + m)$, où $m = |E(G)|$ est le nombre d'arêtes de G.

Tri des arêtes

Si G est représenté en mémoire par une liste de ses m arêtes données dans un ordre quelconque, on peut trier ses arêtes en temps linéaire $\mathcal{O}(n + m)$. Cela est plus efficace qu'un algorithme de tri classique qui demanderait un temps quasi-linéaire $\mathcal{O}(m \log m)$. Cela permet d'avoir une liste L des m arêtes de G triées dans l'ordre lexicographique. Pour construire cette liste, on applique deux fois l'algorithme 3. Il s'agit d'un algorithme qui s'applique normalement à une graphe orienté et qui inverse l'orientation des arêtes tout en les triant suivant la première composante. L'appliquer deux fois permet de trier les arêtes d'un graphe orienté. Cela fonctionne également pour un graphe non-orienté dans lequel chaque arête est présente dans les deux orientations possibles.

On peut ensuite ajouter une liste de taille n telle que la i-ième case contient un pointeur vers la première arête de la liste L ayant i comme extrémité gauche, c'est-à-dire vers l'arête $(i, \min N(i))$. Cela revient à découper la liste L suivante les listes `voisin`$[l]$. Cela permet d'accéder à l'arête $(i, \max N(i))$ en temps constant, car il s'agit de l'arête juste avant $(i+1, \min N(i+1))$ dans la liste L. On peut aussi vouloir accéder à l'arête $(i, \max N^-(i))$ en temps constant. Pour cela, on crée une autre liste de taille n telle que la i-ième case contient un pointeur vers l'arête $(i, \max N^-(i))$. Cela permet en plus d'accéder à l'arête $(i, \min N^+(i))$ en temps constant, car il s'agit de l'arête juste après $(i, \max^- N(i))$ dans la liste L.

Algorithme 3 : Demi-tri des arêtes.

Entrée : Une liste L des m arêtes d'un graphe ordonné (G, τ) de n sommets
Sortie : Une liste L des arêtes inversées et triées suivant l'une des composantes
Sans perte de généralité, on suppose $V(G) = \{1, \dots, n\}$ et $\tau = [1, \dots, n]$.

```
1 Créer n listes vides voisin[l] telle que la l-ième liste soit de taille N(l).
2 Pour l allant de 1 à m faire
3 |   (i, j) ← L[l]
4 |   Ajouter (j, i) à la fin de voisin[j]
5 Retourner L qui est la concaténation des listes voisin[l] pour l allant de 1 à n
```

Exemple 6.7. Soit L la liste $[(2,1),(2,3),(3,2),(4,1),(1,2),(1,4),(1,3),(3,1)]$. En appliquant l'algorithme 3, on obtient les listes :

- `voisin`$[1] = [(1,2),(1,4),(1,3)]$,
- `voisin`$[2] = [(2,3),(2,1)]$,
- `voisin`$[3] = [(3,2),(3,1)]$,
- `voisin`$[4] = [(4,1)]$.

Ce qui donne la liste à moitié triée $L = [(1,2),(1,4),(1,3),(2,3),(2,1),(3,2),(3,1),(4,1)]$. Après une deuxième application de l'algorithme 3, on obtient la liste triée $L = [(1,2),(1,3),(1,4),(2,1),(2,3),(3,1),(3,2),(4,1)]$. La liste annexe de pointeurs est $L_P = [0, 3, 5, 7]$. Le i-ème élément de L_P pointe vers le début de `voisin`$[i]$ dans L. Cette liste L_P peut être construite dès le début de l'algorithme en dénombrant les arêtes de L qui ont un sommet donnée comme extrémité gauche.

Pointeurs sur l'arête symétrique

On note que chaque arête (i, j) est présente deux fois dans la liste L : une fois sous la forme (i, j) et une fois sous la forme (j, i). Ainsi, il peut être pratique pour chaque arête (i, j) d'avoir un pointeur vers (j, i). Il est possible de calculer l'ensemble de ces pointeurs en temps linéaire $\mathcal{O}(n + m)$. Pour cela, on applique à nouveau l'algorithme de demi-tri des arêtes sur la liste déjà triée L, mais en ajoutant un pointeur au moment de l'inversion de l'arête, comme cela est montré dans l'algorithme 4.

Algorithme 4 : Demi-tri des arêtes (avec ajout de pointeur d'inversion).

Entrée : Une liste L triée des m arêtes d'un graphe ordonné (G, τ) de n sommets
Sortie : Une liste L triée dans laquelle chaque arête (i, j) a un pointeur vers l'arête (j, i)
Sans perte de généralité, on suppose $V(G) = \{1, \dots, n\}$ et $\tau = [1, \dots, n]$.

```
1 Créer n listes vides voisin[l] telles que la l-ième liste soit de taille N(l).
2 Pour l allant de 1 à m faire
3 |   (i, j) ← L[l]
4 |   Ajouter une structure contenant l'arête (j, i) et l'entier l à la fin de voisin[j]
5 Retourner L qui est la concaténation des listes voisin[l] pour l allant de 1 à n
```

La liste L est une liste de structures possédant chacune une arête (i, j) ainsi qu'un entier l qui est la position de (j, i) dans L.

Ainsi, étant donné une liste d'arêtes L dans laquelle chaque arête (i, j) a son arête symétrique (j, i) présente dans L, on peut construire en temps $\mathcal{O}(n + m)$ une liste L' de couples (arête (i, j), entier l) qui est triée suivant l'ordre lexicographique des arêtes et telle que l'entier l est l'indice de l'arête symétrique (j, i) dans L'. Cela se fait en appliquant deux fois l'algorithme 3 puis une fois l'algorithme 4.

Arêtes voisines

Étant donné une arête $e = (i, j)$ de (G, τ), on définit quatre autres arêtes liées à (i, j) :

— $e^+ = (i, j')$, où j' est le plus petit sommet plus grand que j tel que (i, j') est une arête de G.

— $e^- = (i, j')$, où j' est le plus grand sommet plus petit que j tel que (i, j') est une arête de G.

— ${}^+e = (i', j)$, où i' est le plus petit sommet plus grand que i tel que (i', j) est une arête de G.

— ${}^-e = (i', j)$, où i' est le plus grand sommet plus petit que i tel que (i', j) est une arête de G.

Ces quatre arêtes ont la propriété d'être voisines de (i, j) ou (j, i) dans la liste triée L des arêtes de G, ce qui permet d'y accéder en temps constant depuis (i, j) (car on rappelle qu'on a accès en temps constant à (j, i) depuis (i, j)).

Si $e = (i, j)$ est une arête, on note $\min e$ le plus petit sommet parmi i et j, et on note $\max e$ le plus grand sommet parmi i et j. Cette notation est utile quand on considère une arête e par son nom plutôt que par ses extrémités (i, j).

Finalement, les valeurs minimales et maximales des listes $N^-(i)$, $N^+(i)$ et des arêtes e^+, e^-, ${}^+e$, ${}^-e$ peuvent être obtenues en temps constant pour tout sommet i et toute arête e à partir d'un pré-calcul en temps linéaire $\mathcal{O}(n + m)$.

Non-arête

Il est également possible d'accéder à certaines non-arêtes en temps constant. Étant donné un sommet i, on peut pré-calculer la valeur minimale et maximale des listes de non-arêtes $\overline{N^-}(i) := \{1, \ldots, i - 1\} \setminus N^-(i)$ et $\overline{N^+}(i) := \{i + 1, \ldots, n\} \setminus N^+(i)$ en temps respectif $|N^-(i)|$, $|N^+(i)|$ en parcourant ces listes d'adjacence et en repérant le premier sommet qui n'y est pas. Ainsi, on pré-calcule en temps linéaire $\mathcal{O}(n + m)$ les valeurs minimales et maximales des listes $\overline{N^-}(i)$ et $\overline{N^+}(i)$. On peut alors les stocker dans des tableaux et y accéder ensuite en temps constant. De même, on peut tester si $\overline{N^-}(i)$ ou $\overline{N^+}(i)$ est vide ou non-vide en temps constant.

Inclusion de partition

Soit $f : V(G) \to V(G)$ une fonction que l'on peut évaluer en temps constant. Il est possible de tester en temps linéaire si pour tout $i \in V(G)$ on a $N(i) \subseteq N(f(i))$. Pour cela, soit $j \in V(G)$ et soit $f^{-1}(j)$ l'ensemble des nombres i tels que $f(i) = j$. Comme f n'est pas forcément une bijection, cet ensemble n'est pas nécessairement un singleton. L'idée est de vérifier simultanément pour tous les $i \in f^{-1}(j)$ si l'on a $N(i) \subseteq N(f(i)) = N(j)$. Notons $N(f^{-1}(j)) = \bigcup_{i \in f^{-1}(j)} N(i)$. Ainsi, il suffit de tester si l'on a $N(f^{-1}(j)) \subseteq N(j)$. Si oui, on a bien $N(i) \subseteq N(f(i))$ pour tout $i \in f^{-1}(j)$. Si non, il existe un i tel que $N(i) \not\subseteq N(f(i))$. Pour tester que l'on a $N(f^{-1}(j)) \subseteq N(j)$ pour un j donné, si on suppose que $N(f^{-1}(j))$

et $N(j)$ sont triés, cela se fait en temps $|N(f^{-1}(j))|+|N(j)| \leq (\sum_{i\in f^{-1}(j)} |N(i)|)+|N(j)|$. Ainsi, tester $N(f^{-1}(j)) \subseteq N(j)$ pour tous les j se fait en temps borné par :

$$\sum_{j=1}^{n}\left(\left(\sum_{i\in f^{-1}(j)}|N(i)|\right)+|N(j)|\right) = \sum_{j=1}^{n}\sum_{i\in f^{-1}(j)}|N(i)| + \sum_{j=1}^{n}|N(j)|$$
$$= \sum_{i=1}^{n}|N(i)| + \sum_{j=1}^{n}|N(j)| = 2n.$$

Il reste à montrer que l'on peut calculer et trier l'ensemble des $N(f^{-1}(j))$ en temps linéaire. Pour cela, il suffit de copier la liste L des arêtes en une liste L', de remplacer chaque arête (i,j) de L' par $(f(i),j)$, puis de trier L' en utilisant deux fois l'algorithme de demi-tri. On a alors le lemme suivant :

Lemme 6.5. *Soit $f : V(G) \to V(G)$ une fonction que l'on peut évaluer en temps constant. Il est possible de tester en temps linéaire la proposition « $\forall i \in V(G)$, $N(i) \subseteq N(f(i))$ ».*

Exactement de la même façon, on prouve le lemme suivant :

Lemme 6.6. *Soit $f : V(G) \to V(G)$ une fonction que l'on peut évaluer en temps constant. Il est possible de tester en temps linéaire la proposition « $\forall i \in V(G)$, $N(f(i)) \subseteq N(i)$ ».*

6.2.2 Motifs à trois sommets

Cette partie est la suite du travail de Michel Habib et Laurent Feuilloley fait en [FH21b] et reprend des éléments d'un travail commun avec Michel Habib et Laurent Feuilloley en cours de publication [DFHP23]. Cet article montre le théorème 6.1 :

Théorème 6.1. *Soit P un motif à trois sommets qui n'est pas* Triangle, *co-*Triangle, Comparability *ou co-*Comparability. *On peut décider si un graphe ordonné (G,τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n+m)$.*

Avant tout, décrivons les différents motifs à trois sommets. La figure 6.8 montre la liste exhaustive des motifs à trois sommets, avec leur nom tel que donné par [FH21b].

Le but est maintenant de montrer que pour chacun des motifs qui ne sont pas Triangle, co-Triangle, Comparability et co-Comparability, ce motif peut être détecté en temps linéaire.

Motifs ayant deux arêtes ou moins

Dans le cas où le motif considéré a deux arêtes ou moins (qu'elles soient obligatoires ou interdites), la preuve est assez simple. On sait alors qu'un sommet du motif est l'extrémité de toutes les arêtes du motif. On regarde pour chacun des sommets du graphe ordonné (G,τ) si la liste d'adjacence contient les arêtes demandées par le motif. Par exemple, pour le motif Interval, on parcourt tous les sommets i du graphe ordonné (G,τ) et on regarde si $\min \overline{N^+}(i) < \max N^+(i)$. Si oui, on a trouvé une non-arête et une arête qui satisfont la contrainte d'ordre imposée par le motif. Si non, il n'existe pas de réalisation du motif Interval dans (G,τ) tel que le sommet le plus à gauche de Interval corresponde au sommet i. Le test « $\min \overline{N^+}(i) < \max N^+(i)$ » est faisable en temps constant en utilisant les pré-calculs exposés dans la section 6.2.1.

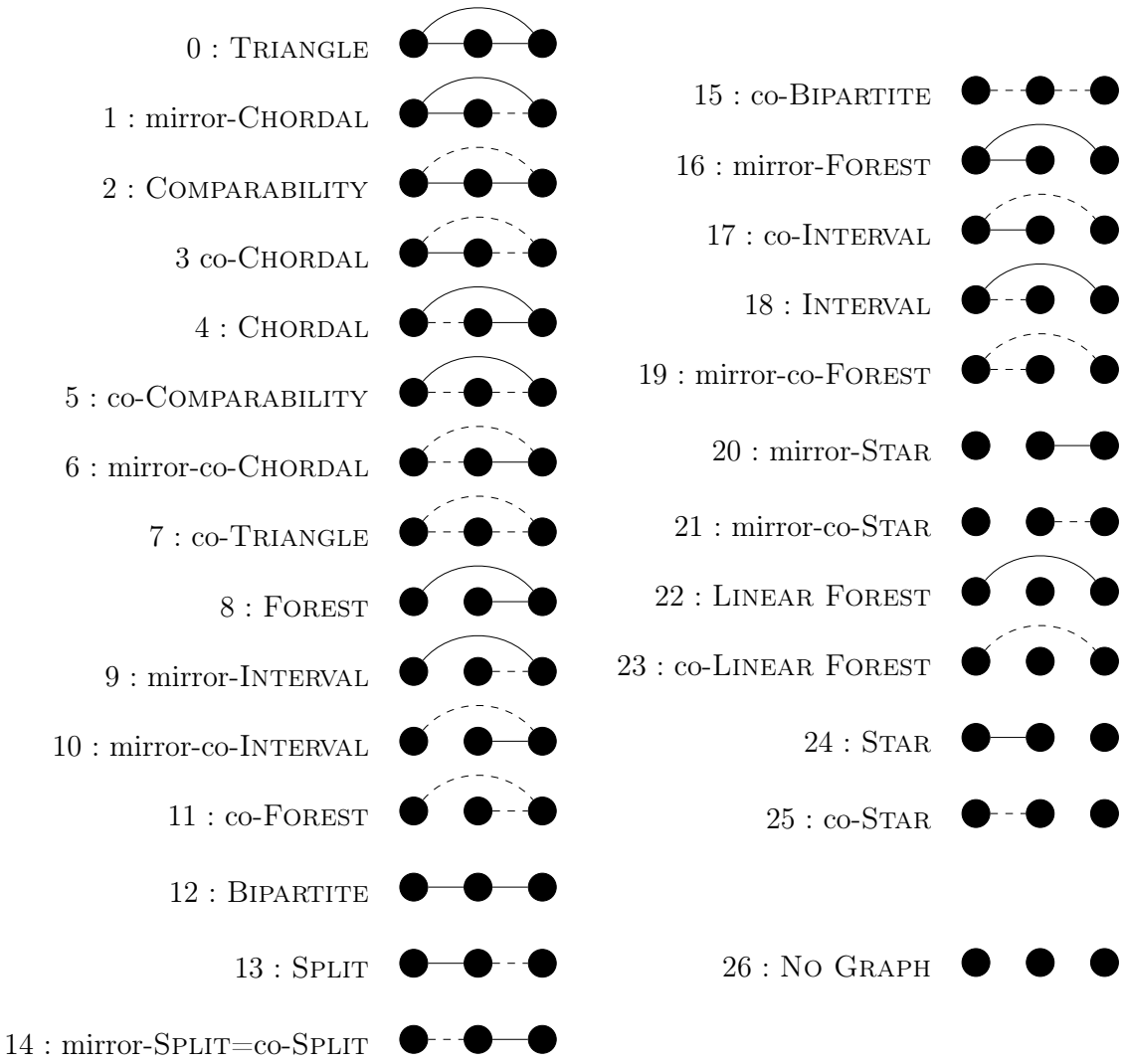


Figure 6.8 – Les 27 motifs à trois sommets.

Le but est maintenant d'expliciter ce test pour chaque motif, en faisant attention que ce test soit calculable en temps constant pour chaque sommet i. Plus formellement, il suffit de trouver pour chacun des motifs une fonction $\phi : V(G) \to \{0, 1\}$ (où 0 représente la valeur booléenne « faux » et 1 la valeur booléenne « vrai »), qui puisse être évaluée en temps constant (moyennant éventuellement un pré-calcul linéaire).

8. Forest : $\phi(i) =$ « $\min N^-(i) < \max N^-(i)$ »
9. mirror-Interval : $\phi(i) =$ « $\min N^-(i) < \max \overline{N^-}(i)$ »
10. mirror-co-Interval : $\phi(i) =$ « $\min \overline{N^-}(i) < \max N^-(i)$ »
11. co-Forest : $\phi(i) =$ « $\min \overline{N^-}(i) < \max \overline{N^-}(i)$ »
12. Bipartite : $\phi(i) =$ « $N^-(i) \neq \emptyset$ et $N^+(i) \neq \emptyset$ »
13. Split : $\phi(i) =$ « $N^-(i) \neq \emptyset$ et $\overline{N^+}(i) \neq \emptyset$ »
14. mirror-Split : $\phi(i) =$ « $\overline{N^-}(i) \neq \emptyset$ et $N^+(i) \neq \emptyset$ »
15. co-Bipartite : $\phi(i) =$ « $\overline{N^-}(i) \neq \emptyset$ et $\overline{N^+}(i) \neq \emptyset$ »
16. mirror-Forest : $\phi(i) =$ « $\min N^+(i) < \max N^+(i)$ »
17. co-Interval : $\phi(i) =$ « $\min N^+(i) < \max \overline{N^+}(i)$ »

18. Interval : $\phi(i) =$ « $\min \overline{N^+}(i) < \max N^+(i)$ »
19. mirror-co-Forest : $\phi(i) =$ « $\min \overline{N^+}(i) < \max \overline{N^+}(i)$ »
20. mirror-Star : $\phi(i) =$ « $i > 1$ et $N^+(i) \neq \emptyset$ »
21. mirror-co-Star : $\phi(i) =$ « $i > 1$ et $\overline{N^+}(i) \neq \emptyset$ »
22. Linear Forest : $\phi(i) =$ « $\max N^+(i) > i + 1$ »
23. co-Linear Forest : $\phi(i) =$ « $\max \overline{N^+}(i) > i + 1$ »
24. Star : $\phi(i) =$ « $N^-(i) \neq \emptyset$ et $i < n$ »
25. co-Star : $\phi(i) =$ « $\overline{N^-}(i) \neq \emptyset$ et $i < n$ »
26. No Graph : $\phi(i) =$ « $n < 3$ »

Algorithme 5 : Recherche d'un motif P ayant un numéro entre 8 et 26.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient P » ou « G évite P »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
Pour chaque i ∈ V(G) faire
    Si φ(i) alors
        Retourner « G contient P »
Retourner « G évite P »
```

Comme $\phi(i)$ se teste en temps constant sous réserve d'un pré-calcul en temps $\mathcal{O}(n + m)$ et qu'on effectue ce test pour chaque sommet $i \in V(G)$, on a bien un algorithme s'exécutant en temps linéaire $\mathcal{O}(n + m)$.

Motif ayant trois arêtes

Le théorème 6.1 que l'on cherche à montrer précise que seuls les motifs qui ne sont pas Triangle, co-Triangle, Comparability ou co-Comparability peuvent être détectés en temps linéaire. Ainsi, les seuls motifs à trois sommets et trois arêtes (obligatoires ou interdites) dont on doit prouver la détection linéaire sont Chordal and co-Chordal.

La preuve est moins évidente que pour les motifs à deux arêtes et est due à Michel Habib [DFHP23]. Je propose ici l'algorithme par souci d'exhaustivité.

Algorithme 6 : Recherche linéaire du motif Chordal.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient Chordal » ou « G évite Chordal »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
Pour i allant de 3 à n faire
    Si N⁻(i) ≠ ∅ alors
        j = max N⁻(i)
        Si NOT(N⁻(i) \ j ⊆ N⁻(j)) alors
            Retourner « G contient Chordal »
Retourner « G évite Chordal »
```

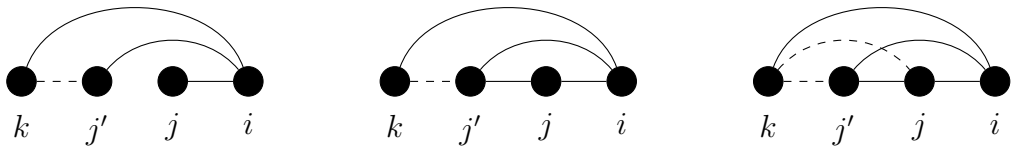
Correction. L'idée est la suivante : on ne va chercher le motif CHORDAL qu'à certaines positions précises de (G, τ) et montrer que si (G, τ) contient le motif CHORDAL, il contient forcément un motif CHORDAL à l'une des positions cherchées. Les positions cherchées sont les positions $k < j < i$ de G telles que j est le plus proche voisin à gauche de i, c'est-à-dire $j = \max N^-(i)$.

Supposons maintenant qu'il existe une réalisation de ce motif CHORDAL qui ne soit pas de cette forme. Il se réalise alors en $k < j' < i$ avec $j' < \max N^-(i)$. On suppose que cette réalisation est la plus petite possible parmi les réalisations qui ne sont pas de la forme voulue, dans le sens où $|i - k|$ est minimal. On pose alors $j = \max N^-(i)$. On a donc les arêtes présentes en figure 6.9a. On discute maintenant de la présence d'une arête entre j' et j :

— S'il n'y en a pas, on a trouvé un motif CHORDAL de la forme voulue à la position $j' < j < i$. On a donc prouvé ce qu'on voulait.

— S'il y en a une, on obtient la figure 6.9b. On déduit qu'il n'y a pas d'arête entre k et j'. En effet, s'il y en avait une, on aurait trouvé un motif CHORDAL en $k < j' < j$, qui aurait donc une taille $|j - k|$ strictement plus petite que $|i - k|$, contredisant la minimalité du motif CHORDAL en $k < j' < i$. On a donc la figure 6.9c. Dans cette figure, $k < j < i$ est un motif CHORDAL de la forme voulue.

Ainsi, pour vérifier que (G, τ) ne contient pas de CHORDAL, il suffit de vérifier pour tout sommet i de G que l'on a l'implication « si $(k, i) \in E(G)$ alors $(k, j) \in E(G)$ » pour tout $k < j$, avec $j = \max N^-(i)$. Si cette implication est fausse, on a trouvé une réalisation de CHORDAL dans (G, τ) en $k < j < i$. Si cette implication est vraie, aucune des positions décrites précédemment ne contient CHORDAL et CHORDAL n'apparait pas dans (G, τ). Cette implication pour un i donné équivaut à la formule $\phi(i) =$ « $N^-(i) \setminus \{j\} \subseteq N^-(j)$ », avec toujours $j = \max N^-(i)$. Le fait que l'on retire j de l'ensemble $N^-(i)$ vient du fait que l'on ne vérifie cette implication que pour $k < j$. Il ne faut donc regarder que les voisinages dans $[1, j-1]$ et ainsi retirer le sommet j. Il n'y a pas besoin de retirer d'autres sommets car j est le seul voisin à gauche de i entre i et k par définition de j.



(a) Hypothèses. (b) Arête entre j' et j. (c) Minimalité.

FIGURE 6.9 – Les différentes étapes de la démonstration qui prouve qu'il existe une réalisation de CHORDAL qui a la forme voulue.

Complexité. Tel quel, l'algorithme 6 n'est pas linéaire mais apparait plutôt quadratique. Cependant, il existe plusieurs façons pour l'exécuter de façon linéaire. Il est proposé dans [HMPV00] d'utiliser une structure de donnée adéquate et du raffinement de partition afin de ne pas parcourir plusieurs fois une même liste d'adjacence. De mon côté, je propose d'utiliser le lemme 6.5. En posant $f(i) = \max N^-(i)$, celui-ci nous dit que l'on peut tester la proposition « $\forall i \in V(G)$, $N(i) \subseteq N(f(i))$ » en temps linéaire. Dans notre cas, on veut tester une inclusion plus faible, dans le sens où on ne demande pas que tout $N(i)$ soit inclus dans $N(f(i))$ mais seulement $N^-(i) \setminus f(i)$. Comme la complexité donnée

par le lemme 6.5 vient de la somme des parcours des listes $N(i)$ pour tout i, en diminuant la taille de cette liste, on ne fait que baisser la complexité nécessaire. De même, si on ne demande pas $\forall i$ mais seulement $\forall i \in \{3..n\} \cap \{i \mid N^-(i) \neq \emptyset\}$, on ne fait que parcourir moins de listes, ce qui ne fait que diminuer la complexité. D'où une complexité linéaire $\mathcal{O}(n+m)$.

Concernant le motif co-CHORDAL, un raisonnement similaire prouve la correction et la linéarité de l'algorithme 7. On utilise cependant le lemme 6.6 à la place du lemme 6.5 pour prouver la linéarité. À propos de la correction, en adaptant celle de l'algorithme 6, on montre qu'il faut maintenant vérifier l'implication « si $(k,j) \in E(G)$ alors $(k,i) \in E(G)$ » pour tout $k < j$, avec $j = \max \overline{N^-}(i)$, à la place de « si $(k,i) \in E(G)$ alors $(k,j) \in E(G)$ » et $j = \max N^-(i)$ (on a échangé i et j et changé la définition de j). Cela se traduit en la formule $\phi(i) =$ « $N^-(j) \subseteq N^-(i)$ ».

Algorithme 7 : Recherche linéaire du motif co-CHORDAL.

Entrée : Un graphe ordonné (G,τ) de n sommets
Sortie : « G contient co-CHORDAL » ou « G évite co-CHORDAL »
Sans perte de généralité, on suppose $V(G) = \{1,\dots,n\}$ et $\tau = [1,\dots,n]$.

```
Pour i allant de 3 à n faire
    Si N⁻(i) ≠ ∅ alors
        j = max N⁻(i)
        Si NOT(N⁻(j) ⊆ N⁻(i)) alors
            Retourner « G contient co-CHORDAL »
Retourner « G évite co-CHORDAL »
```

Le théorème 6.1 est ainsi prouvé.

6.2.3 Quelques motifs à quatre sommets

Dans cette section, on s'intéresse aux motifs à quatre sommets contenant trois arêtes obligatoires et aucune arête interdite. Il y a deux façons d'agencer trois arêtes sur quatre sommets : en étoile ou en chemin. Dans le cas de l'étoile, à symétrie près, il n'y a que deux motifs possibles. Dans le cas du chemin, à symétrie près, il y a huit motifs possibles. Le but est de montrer le théorème suivant :

Théorème 6.2. *Soit P un motif à quatre sommets contenant trois arêtes obligatoires et aucune arête interdite. On peut décider si un graphe ordonné (G,τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n+m)$.*

Étoiles

Dans le cas des motifs en étoile, on a un sommet central et trois arêtes qui partent de ce sommet. Pour vérifier le premier cas, il suffit de regarder s'il existe un sommet i du graphe tel que $|N^+(i)| \geq 3$. Pour vérifier le second cas, il suffit de regarder s'il existe un sommet i du graphe tel que $|N^+(i)| \geq 2$ et $|N^-(i)| \geq 1$.

Chemins

Dans le cas des motifs en chemin, ils ont une arête centrale et deux arêtes qui partent des extrémités de cette arête centrale. Il y a huit motifs possibles. Je commence par

(a) Sommet i à l'extrémité du motif. (b) Sommet i au centre du motif.

FIGURE 6.10 – Les deux ordres possibles pour une étoile, en excluant les ordres miroirs.

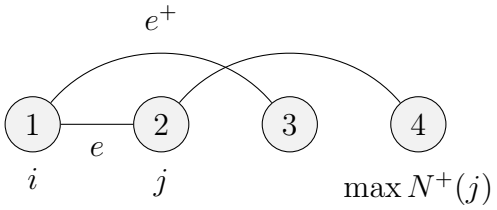
expliquer en détails un des huit motifs, puis je montre comment adapter l'algorithme pour les sept autres motifs. L'idée principale consiste à parcourir toutes les arêtes e du graphe et vérifier si le motif P apparait avec e comme arête centrale. La seconde idée est de ne vérifier qu'un seul emplacement potentiel du motif P dans le graphe ordonné (G, τ) pour chaque arête e, en s'assurant que si le motif n'est pas présent à cet emplacement, alors il ne peut pas être trouvé avec l'arête e comme arête centrale du motif.

Dans l'exemple ci-dessous, pour chaque arête e, on essaie de détecter le motif qui est extrémal dans le sens suivant :

— le sommet 3 du motif P est le sommet $\max e^+$ du graphe G,

— le sommet 4 du motif P est le sommet $\max N^+(j)$ du graphe G.

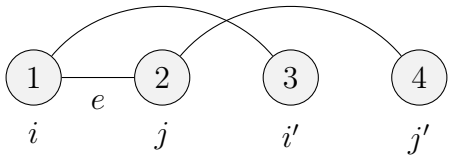
On prouve que cela est suffisant.



On a donc deux cas possibles :

— Si $\max e^+ < \max N^+(j)$, alors G contient P.

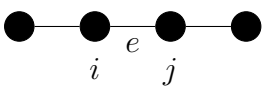
— Si $\max e^+ \geq \max N^+(j)$, alors il n'y a pas de motif P dans G tel que l'arête centrale de P soit l'arête e de G.



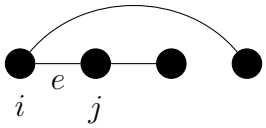
On démontre ce second cas par l'absurde. Supposons qu'il existe un tel motif P dans G tel que $\max e^+ \geq \max N^+(j)$. On note i' et j' les deux sommets de G tel que i' corresponde au sommet 3 de P et j' corresponde au sommet 4 de P. Ainsi, d'après les contraintes du motif P, cela signifie que $i' < j'$ et que (i, i') et (j, j') sont des arêtes de G. Puisque $j' \in N^+(j)$, on a $j' \leq \max N^+(j)$. Puisque i' est un sommet plus grand que j et tel que (i, i') soit une arête et puisque $\max e^+$ est le plus petit sommet plus grand que j tel que $(i, \max e^+)$ soit une arête, on a $\max e^+ \leq i'$. Ainsi, on a $j' \leq \max N^+(j) \leq \max e^+ \leq i'$, ce qui contredit $i' < j'$.

De cette manière, le motif P peut être détecté en considérant chaque arête e du graphe G et en vérifiant la condition $\max e^+ < \max N^+(j)$. Vu que cette condition peut être vérifiée en temps constant (moyennant un pré-calcul en $\mathcal{O}(n+m)$) et qu'il y a m arêtes à vérifier, le motif P peut être détecté en temps linéaire $\mathcal{O}(n+m)$.

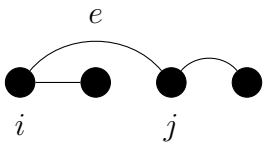
Afin de généraliser cette idée, notons $\phi : E(G) \to \mathbb{B}$ la fonction qui prend une arête de G en argument et qui renvoie le booléen « $\max e^+ < \max N^+(j)$ ». L'algorithme de détection s'écrit alors :



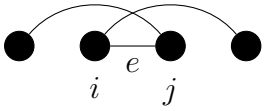
$\phi(e) =$ "$N^-(i) \neq \emptyset$ and $N^+(j) \neq \emptyset$"



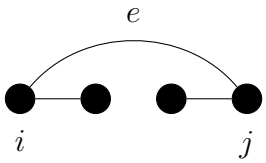
$\phi(e) =$ "$\min N^+(j) < \max N^-(i)$"



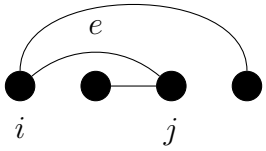
$\phi(e) =$ "$\min N^+(i) < j$ and $N^+(j) \neq \emptyset$"



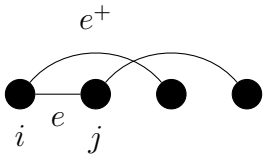
$\phi(e) =$ "$\min N^-(j) < i$ and $j < \max N^+(i)$"



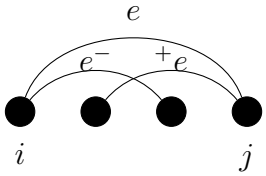
$\phi(e) =$ "$\min N^+(i) < \max N^-(j)$"



$\phi(e) =$ "$i < \max N^-(j)$ and $j < \max N^+(i)$"



$\phi(e) =$ "$\max e^+ < \max N^+(j)$"



$\phi(e) =$ "$\min {}^+e < \max e^-$"

FIGURE 6.11 – Les huit ordres possible d'un chemin, en excluant les ordres miroirs.

Pour montrer que les sept autres motifs qui correspondent à un chemin sont détectables en temps linéaire, il suffit de trouver une fonction $\phi : E(G) \to \mathbb{B}$ pour chacun des cas qui puisse être évaluée en temps constant (moyennant éventuellement un pré-calcul linéaire) et telle qu'on ait la propriété « Si $\phi(e)$ est faux, alors il n'y a pas de motif P dans G tel que

Algorithme 8 : Recherche d'un motif P chemin.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient P » ou « G évite P »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
Pour chaque e ∈ E(G) faire
    Si φ(e) alors
        Retourner « G contient P »
Retourner « G évite P »
```

l'arête centrale de P soit l'arête e de G. » La figure 6.11 représente une liste récapitulative des huit motifs qui correspondent à un chemin (en incluant le motif déjà traité). Les motifs obtenus en prenant le miroir de l'ordre τ ne sont pas représentés. Chaque motif est accompagné de sa fonction $\phi(e)$ correspondante. L'algorithme 8 présente l'algorithme capable de détecter les huit chemins de la figure 6.11 en temps linéaire $\mathcal{O}(n + m)$, en utilisant la fonction $\phi(e)$ associée.

6.3 Plusieurs motifs simultanément

Au lieu de se demander si un motif est présent ou non, on se demande si un ensemble de motifs est présent ou non. Étant donné S un ensemble de motifs et (G, τ) un graphe ordonné, on se demande si (G, τ) contient au moins un motif de S ou s'il les évite tous.

On s'intéresse dans cette section aux ensembles S constitués de n'importe quel nombre de motifs à 3 sommets.

6.3.1 Outils

On dit que deux ensembles S et S' de motifs sont *équivalents* si pour tous les graphes ordonnés (G, τ), on a l'équivalence (G, τ) évite tous les motifs de S si et seulement si (G, τ) évite tous les motifs de S'. Dit d'une autre manière, S et S' sont équivalents lorsque (G, τ) contient au moins un motif de S si et seulement si (G, τ) contient au moins un motif de S'. On note $S \simeq S'$. De plus, on dit qu'un motif P *se décompose* en un ensemble S si $\{P\} \simeq S$. Un ensemble S est *réduit* s'il n'existe pas S' tel que $|S'| < |S|$ et $S \simeq S'$.

Le complément terme à terme d'un ensemble de motifs S est l'ensemble $S^* := \{\overline{P}, P \in S\}$. On rappelle que $\overline{P}$ désigne le complément de P, c'est-à-dire le motif P' dans lequel on intervertit arêtes obligatoires et arêtes interdites, ce qui signifie que l'on a $V(P') = V(P)$, $M(P') = F(P)$, $F(P') = M(P)$, $\tau(P') = \tau(P)$. De même, le miroir terme à terme d'un ensemble de motifs S est l'ensemble $\texttt{miroir}(S) := \{\texttt{miroir}(P), P \in S\}$, où $\texttt{miroir}(P)$ désigne le motif P' obtenu en renversant l'ordre de P.

On a le lemme suivant :

Lemme 6.7. *Soit S_1 et S_2 deux ensembles de motifs tels que $S_1 \simeq S_2$. On a alors $S_1^* \simeq S_2^*$ et $\texttt{miroir}(S_1) \simeq \texttt{miroir}(S_2)$.*

Démonstration. Si (G, τ) est un graphe ordonné et P un motif, (G, τ) contient P si et seulement si $(\overline{G}, \tau)$ contient $\overline{P}$. En effet, si un ensemble de sommets $U \subseteq V(G)$ réalise P dans (G, τ), alors en inversant arêtes et non-arêtes, l'ensemble U réalise $\overline{P}$ dans (G, τ)

contient P. Ainsi, on a les équivalences :

$$\begin{aligned}
&\text{Pour tous les graphes ordonnés } (G,\tau),\ (G,\tau) \text{ évite tous les motifs de } S_1^*\\
\iff &\text{Pour tous les graphes ordonnés } (G,\tau),\ (\overline{G},\tau) \text{ évite tous les motifs de } S_1\\
\iff &\text{Pour tous les graphes ordonnés } (G,\tau),\ (\overline{G},\tau) \text{ évite tous les motifs de } S_2\\
\iff &\text{Pour tous les graphes ordonnés } (G,\tau),\ (G,\tau) \text{ évite tous les motifs de } S_2^*
\end{aligned}$$

D'où $S_1^* \simeq S_2^*$. De façon similaire, on prouve $\texttt{miroir}(S_1) \simeq \texttt{miroir}(S_2)$. en utilisant le fait que (G,τ) contient P si et seulement si $(G, \texttt{miroir}(\tau))$ contient $\texttt{miroir}(P)$. □

6.3.2 Stratégie

Soit S un ensemble de motifs. On commence par réduire S, c'est-à-dire que l'on cherche un motif équivalent S' qui ne puisse pas être réduit. Dans [FH21b], il a été montré qu'il existe une liste L_{87} de 87 ensembles de motifs tels que tout ensemble S est équivalent à l'un des ensembles de L_{87}, au complément terme à terme de l'un des ensembles de L_{87} ou au miroir terme à terme de l'un des ensembles de L_{87}.

Si S est maintenant constitué d'un seul motif, on est ramené au cas de la section 6.2.2, à savoir qu'un motif peut être certifié en temps linéaire, sauf s'il s'agit d'un motif parmi {Triangle, Comparability, co-Comparability, co-Triangle}, auquel cas il faut un temps $\mathcal{O}(n^\omega)$. Sinon, S contient plusieurs motifs. Dans le cas où aucun des motifs Triangle, Comparability, co-Comparabilityet co-Triangle ne figure dans S, on peut certifier en temps linéaire chacun des motifs de S les uns après les autres. Il ne reste donc que le cas où S contient plusieurs motifs dont au moins un parmi {Triangle, Comparability, co-Comparability, co-Triangle}. De plus, si la classe des graphes ordonnés évitant S est finie, il suffit de tester si (G,τ) est l'un de ces graphes, ce qui se fait donc en temps linéaire en la taille de G (nous n'avons pas le problème du test de l'isomorphisme de graphe vu qu'ils sont ordonnés). On s'intéresse aux motifs de L_{87} qui contiennent au moins un de ces quatre motifs et qui n'induisent pas une classe finie. Cela nous donne une liste plus petite, constituée de 26 ensembles. Notons cette liste L_{26}. La voici :

1. {Comparability, co-Comparability}
2. {mirror-Chordal, Comparability}
3. {Comparability, mirror-Interval}
4. {mirror-Chordal, Comparability, Chordal}
5. {Comparability, Chordal, mirror-Interval}
6. {Comparability, Split}
7. {Comparability, co-Comparability, Split}
8. {Comparability, Chordal, Split}
9. {Comparability, Split, Interval}
10. {Triangle, co-Comparability}
11. {Triangle, co-Chordal}
12. {Triangle, co-Chordal, mirror-co-Chordal}
13. {Triangle, co-Chordal, co-Comparability}

14. {Triangle, co-Chordal, co-Comparability, mirror-co-Chordal}
15. {co-Comparability, Bipartite}
16. {co-Chordal, co-Comparability, Bipartite}
17. {co-Chordal, co-Comparability, mirror-co-Chordal, Bipartite}
18. {co-Comparability, mirror-Forest}
19. {co-Comparability, mirror-co-Chordal, mirror-Forest}
20. {co-Chordal, co-Comparability, mirror-Forest}
21. {co-Chordal, co-Comparability, mirror-co-Chordal, mirror-Forest}
22. {Comparability, mirror-Forest}
23. {Comparability, co-Comparability, mirror-Forest}
24. {co-Comparability, mirror-co-Interval, mirror-Forest}
25. {co-Comparability, Star}
26. {co-Comparability, mirror-co-Chordal, Star}

En particulier, l'ensemble {Triangle, co-Triangle} ne figure pas dans cette liste car il induit une classe finie, vu que tout graphe d'ordre au moins 6 contient l'un de ces deux motifs d'après le théorème de Ramsey.

Ainsi, tout ensemble de motifs S est soit détectable en temps linéaire, soit équivalent à un ensemble de L_{26}, au miroir terme à terme d'un ensemble de L_{26}, ou au complément terme à terme d'ensemble de L_{26}.

Cette liste étant encore un peu longue à vérifier à la main, nous la réduisons encore davantage. On définit 12 ensembles de deux motifs, que l'on appelle les *paires fondamentales*, de sorte que chaque ensemble de motifs de L_{26} puisse s'exprimer comme l'union de paires fondamentales.

1. $S_1 = \{$Comparability, co-Comparability$\}$
2. $S_2 = \{$Comparability, Chordal$\}$
3. $S_2' = \{$Comparability, mirror-Chordal$\}$
4. $S_3 = \{$Comparability, mirror-Interval$\}$
5. $S_4 = \{$Comparability, Split$\}$
6. $S_4^* = \{$co-Comparability, Split$\}$
7. $S_5 = \{$Triangle, co-Comparability$\}$
8. $S_6 = \{$Triangle, co-Chordal$\}$
9. $S_7 = \{$co-Comparability, Bipartite$\}$
10. $S_8 = \{$Comparability, mirror-Forest$\}$
11. $S_8^* = \{$co-Comparability, mirror-Forest$\}$
12. $S_9 = \{$co-Comparability, Star$\}$

On prouve maintenant que chaque paire fondamentale peut être certifiée en temps linéaire. Pour ce faire, on commence par montrer certaines équivalences.

Lemme 6.8. *Les ensembles de motifs suivants sont équivalents :*

— $S_4 \simeq \{$Split, *co*-Interval$\}$

— $S_4^* \simeq \{$Split, *mirror*-Interval$\}$

— $S_8 \simeq \{$*mirror*-Forest, Bipartite$\}$

— $S_8^* \simeq \{$*mirror*-Forest, *mirror*-Interval$\}$

— $S_9 \simeq \{$Star, *mirror*-Interval$\}$

Démonstration. Soit (G, τ) un graphe ordonné. Pour chaque équivalence $S \simeq S'$ et pour chaque motif P dans S, on montre que si (G, τ) contient P alors il contient un motif dans S'. De même pour chaque motif P dans S', on montre que si (G, τ) contient P alors il contient un motif dans S. Puisqu'à chaque fois $|S| = 2$ et $|S'| = 2$, il y a $2 + 2 = 4$ motifs P à considérer. Quand un motif apparait à la fois dans S et S', cela est trivial et nous ne le mentionnons pas. De plus, on remarque que si P est inclus dans P', cela signifie que si (G, τ) contient P' alors il contient P. On ne mentionnera pas non plus ce cas trivial. On remarque alors que dans les cinq équivalences, il y a à chaque fois parmi les 4 motifs P à considérer deux motifs égaux et un inclus dans un autre. Il n'y a donc qu'un seul motif P à considérer à chaque fois.

— Pour prouver que $S_4 = \{$Comparability, Split$\} \simeq \{$Split, co-Interval$\}$, il suffit de montrer que si (G, τ) contient co-Interval alors il contient un motif dans S_4. Soit (a, b, c) les trois sommets qui réalisent le motif co-Interval dans (G, τ). Ainsi, (a, b) est une arête de G et (a, c) n'est pas une arête de G. Si (b, c) est une arête, alors (a, b, c) réalise le motif Comparability. Dans le cas contraire, (a, b, c) réalise Split.

— Pour prouver que $S_4^* = \{$co-Comparability, Split$\} \simeq \{$Split, mirror-Interval$\}$, il suffit de montrer que si (G, τ) contient mirror-Interval alors il contient un motif dans S_4^*. Soit (a, b, c) les trois sommets qui réalisent le motif mirror-Interval dans (G, τ). Ainsi, (a, c) est une arête de G et (b, c) n'est pas une arête de G. Si (a, b) est une arête, alors (a, b, c) réalise le motif Split. Dans le cas contraire, (a, b, c) réalise co-Comparability.

— Pour prouver que $S_8 = \{$Comparability, mirror-Forest$\} \simeq \{$mirror-Forest, Bipartite$\}$, il suffit de montrer que si (G, τ) contient Bipartite alors il contient un motif dans S_8. Soit (a, b, c) les trois sommets qui réalisent le motif Bipartite dans (G, τ). Ainsi, (a, b) et (b, c) sont toutes les deux des arêtes de G. Si (a, c) est une arête, alors (a, b, c) réalise le motif mirror-Forest. Dans le cas contraire, (a, b, c) réalise Comparability.

— Pour prouver que $S_8^* = \{$co-Comparability, mirror-Forest$\} \simeq \{$mirror-Forest, mirror-Interval$\}$, il suffit de montrer que si (G, τ) contient mirror-Interval alors il contient un motif dans S_8^*. Soit (a, b, c) les trois sommets qui réalisent le motif mirror-Interval dans (G, τ). Ainsi, (a, c) est une arête de G et (b, c) n'est pas une arête de G. Si (a, b) est une arête, alors (a, b, c) réalise le motif mirror-Forest. Dans le cas contraire, (a, b, c) réalise co-Comparability.

— Pour prouver que $S_9 = \{$co-Comparability, Star$\} \simeq \{$Star, mirror-Interval$\}$, il suffit de montrer que si (G, τ) contient mirror-Interval alors il contient un motif dans S_9. Soit (a, b, c) les trois sommets qui réalisent le motif mirror-Interval dans (G, τ). Ainsi, (a, c) est une arête de G et (b, c) n'est pas une arête de G. Si (a, b) est une arête, alors (a, b, c) réalise le motif Star. Dans le cas contraire, (a, b, c) réalise co-Comparability. □

6.3.3 Algorithmes

Pour chaque paire fondamentale restante, soit on écrit un algorithme capable de certifier cette paire en temps linéaire, soit on montre que cette paire est équivalente à une union d'autres paires fondamentales.

Proposition 6.9. *L'ensemble* $S_1 = \{$COMPARABILITY, *co-*COMPARABILITY$\}$ *peut être détecté en temps linéaire.*

Démonstration. On introduit l'ordre $\prec$ sur les sommets du graphe (G, τ) comme suit :

$$a \prec b \iff \begin{cases} a <_\tau b \text{ et } (a,b) \in E(G), \text{ ou} \\ b <_\tau a \text{ et } (a,b) \notin E(G) \end{cases}$$

On montre alors que (G, τ) évite S_1 si et seulement si $\prec$ est un ordre total sur $V(G)$. En effet, on sait déjà que $\prec$ est anti-réflexive (puisque $<_\tau$ est anti-réflexive), anti-symétrique (car sinon, on aurait à la fois $(a,b) \in E(G)$ et $(a,b) \notin E(G)$) et total (puisqu'on a soit $(a,b) \in E(G)$, soit $(a,b) \notin E(G)$, et donc selon si on a $a <_\tau b$ ou $b <_\tau a$, on a $a \prec b$ ou $b \prec a$). Il reste donc à montrer que (G, τ) évite S_1 si et seulement si $\prec$ est transitive :

- ($\Leftarrow$) Supposons que (G, τ) contienne COMPARABILITY en $a <_\tau b <_\tau c$. On a donc (a,b) et (b,c) qui sont des arêtes et (a,c) qui n'en est pas. Ainsi, on a $a \prec b \prec c$ mais $a \not\prec c$, ce qui contredit le fait que $\prec$ est transitive. De même, si on suppose que (G, τ) contient co-COMPARABILITY en $a <_\tau b <_\tau c$, on déduit que $c \prec b \prec a$ mais $c \not\prec a$. Ainsi, (G, τ) évite tous les motifs de S_1.
- ($\Rightarrow$) On suppose que l'on a $a \prec b \prec c$ et on prouve $a \prec c$. On regarde les six ordres possibles des sommets selon $<_\tau$ (cf. figure 6.12) :
  (a) Si $a <_\tau b <_\tau c$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \in E(G)$ et $(b,c) \in E(G)$. Ainsi, puisque (G, τ) évite COMPARABILITY, on déduit que $(a,c) \in E(G)$ et donc $a \prec c$.
  (b) Si $a <_\tau c <_\tau b$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \in E(G)$ et $(b,c) \notin E(G)$. Ainsi, puisque (G, τ) évite co-COMPARABILITY, on déduit que $(a,c) \in E(G)$ et donc $a \prec c$.
  (c) Si $b <_\tau a <_\tau c$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \notin E(G)$ et $(b,c) \in E(G)$. Ainsi, puisque (G, τ) évite co-COMPARABILITY, on déduit que $(a,c) \in E(G)$ et donc $a \prec c$.
  (d) Si $b <_\tau c <_\tau a$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \notin E(G)$ et $(b,c) \in E(G)$. Ainsi, puisque (G, τ) évite COMPARABILITY, on déduit que $(a,c) \notin E(G)$ et donc $a \prec c$.
  (e) Si $c <_\tau a <_\tau b$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \in E(G)$ et $(b,c) \notin E(G)$. Ainsi, puisque (G, τ) évite COMPARABILITY, on déduit que $(a,c) \notin E(G)$ et donc $a \prec c$.
  (f) Si $c <_\tau b <_\tau a$, puisque $a \prec b$ et $b \prec c$, on a $(a,b) \notin E(G)$ et $(b,c) \notin E(G)$. Ainsi, puisque (G, τ) évite co-COMPARABILITY, on déduit que $(a,c) \notin E(G)$ et donc $a \prec c$.

Si on se place dans le cas où $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$, cela signifie que (G, τ) évite S_1 si et seulement si $\prec$ est l'ordre $<$. Grâce au lemme 2.6, on peut vérifier facilement si $\prec$ est l'ordre $<$ ou non en dénombrant $\prec^{-1}(i)$ pour chaque sommet i de G. Or, $|\prec^{-1}(i)| = |N^-(i)| + |\overline{N^+}(i)| = |N^-(i)| + n - i - |N^+(i)|$. D'où l'algorithme 9. □

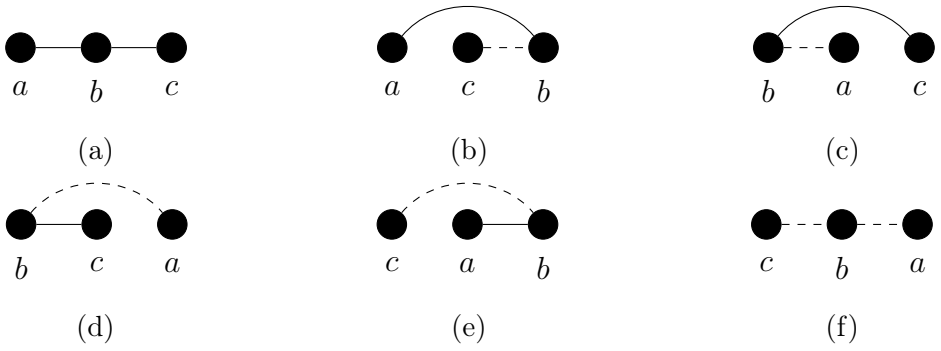


FIGURE 6.12 – Les différents cas pour S_1.

Algorithme 9 : Recherche linéaire de l'ensemble de motifs S_1.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient un des motifs de S_1 » ou « G évite tous les motifs de S_1 »
Sans perte de généralité, on suppose $V(G) = \{1, \dots, n\}$ et $\tau = [1, \dots, n]$.

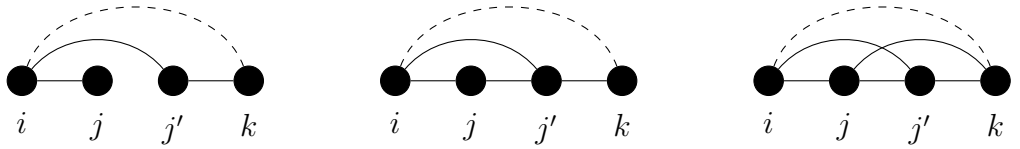
```
L ← une liste de taille n
Pour chaque i ∈ V(G) faire
    L[i] ← |N⁻(i)| + n − i − |N⁺(i)|
Si L est une permutation de {0, ..., n − 1} alors
    Retourner « G évite tous les motifs de S1 »
Sinon
    Retourner « G contient un des motifs de S1 »
```

Proposition 6.10. *Les deux ensembles $S_2 = \{$*Comparability*, Chordal$\}$ et $S_2' = \{$*Comparability*, mirror-*Chordal$\}$ peuvent être détectés en temps linéaire.*

Démonstration. On traite le cas de S_2', le cas S_2 se prouvant de la même manière en inversant l'ordre. On commence par vérifier en temps linéaire si mirror-Chordal est dans (G, τ) ou non via l'algorithme 6. Si oui, on a trouvé un motif de S_2' dans (G, τ). Si non, il reste à vérifier que (G, τ) ne contient pas Comparability en utilisant le fait que (G, τ) évite mirror-Chordal. L'idée est la suivante : on ne va chercher le motif Comparability qu'à certains endroits de (G, τ) et montrer que si (G, τ) contient Comparability tout en évitant mirror-Chordal, il contient forcément Comparability à l'une des positions cherchées. Les positions cherchées sont les positions $i < j < k$ de G telles que j est le premier voisin de i, c'est-à-dire $j = \min N^+(i)$.

Supposons maintenant qu'il existe une réalisation de ce motif Comparability qui ne soit pas de cette forme. Il se réalise alors en i, j', k avec $j' > \min N^+(i)$. On suppose que cette réalisation est la plus petite possible parmi les réalisations qui ne sont pas de la forme voulue, dans le sens où $|k - i|$ est minimal. On pose alors $j = \min N^+(i)$. On a donc les arêtes présentes en figure 6.13a. Comme mirror-Chordal n'apparait pas dans (G, τ), il y a une arête entre j et j', d'où la figure 6.13b. Enfin, comme on a supposé que la réalisation de Comparability en i, j', k' était minimale, il doit y avoir une arête entre j et k, sinon on aurait une réalisation de Comparability en j, j', k' et $|k - j| < |k - i|$. On obtient donc la figure 6.13c. Cependant, la réalisation de Comparability en i, j, k est de la forme voulue, à savoir que $j = \min N^+(i)$. On a donc prouvé que s'il existe une réalisation de Comparability dans (G, τ), alors il en existe une qui est à l'une des

positions voulues.



(a) Hypothèses. (b) Sans mirror-CHORDAL. (c) Minimalité.

FIGURE 6.13 – Les différentes étapes de la démonstration qui prouve qu'il existe une réalisation de COMPARABILITY qui a la forme voulue.

Ainsi, pour vérifier que (G, τ) ne contient pas de COMPARABILITY, il suffit de vérifier pour tout sommet i de G que l'on a l'implication « si $(j, k) \in E(G)$ alors $(i, k) \in E(G)$ » pour tout $k > j$, avec $j = \min N^+(i)$. Si cette implication est fausse, on a trouvé une réalisation de COMPARABILITY dans (G, τ) en i, j, k. Si cette implication est vraie, aucun des positions décrites précédemment ne contient COMPARABILITY et par conséquent COMPARABILITY n'apparait pas dans (G, τ). Cette implication pour un i donné équivaut à la formule $\phi(i) =$ « $N^+(j) \subseteq N^+(i) \setminus \{j\}$ », avec toujours $j = \min N^+(i)$. En utilisant le lemme 6.6 de la même façon que dans la preuve de la linéarité de l'algorithme 7, on peut tester l'ensemble des formules $\phi(i)$ pour tous les sommets i de G en temps linéaire.

Algorithme 10 : Recherche linéaire de l'ensemble de motifs S'_2.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient un des motifs de S'_2 » ou « G évite tous les motifs de S'_2 »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
Si (G, τ) contient mirror-CHORDAL (cf. algorithme 6) alors
    Retourner « G contient un des motifs de S'_2 »
Pour i allant de 3 à n faire
    Si N+(i) ≠ ∅ alors
        j = min N+(i)
        Si NOT(N+(j) ⊆ N+(i) \ {j}) alors
            Retourner « G contient un des motifs de S'_2 »
Retourner « G évite tous les motifs de S'_2 »
```

On procède de même pour S_2 en inversant l'ordre. On a alors $j = \max N^-(k)$ au lieu de $j = \min N^+(i)$. □

Proposition 6.11. *L'ensemble $S_3 =$ {COMPARABILITY, mirror-INTERVAL} peut être détecté en temps linéaire.*

Démonstration. On a $S_3 \simeq$ {COMPARABILITY, co-COMPARABILITY, mirror-CHORDAL} $= S_1 \cup S'_2$. □

Proposition 6.12. *L'ensemble $S_5 =$ {TRIANGLE, co-COMPARABILITY} peut être détecté en temps linéaire.*

Démonstration. D'après [FH21b], G est biparti si, et seulement si, il existe un ordre τ' tel que (G, τ') évite tous les motifs de S_5. On commence donc par tester si (G, τ) est biparti, ce qui se fait essayant de colorier G avec deux couleurs (disons bleu et rouge) en temps linéaire via un parcours BFS. On note V_B les sommets de G coloriés en bleu et V_R ceux qui sont coloriés en rouge. Si G n'est pas biparti, tout ordre τ' fait que (G, τ') contient au moins un motif de S_5, et en particulier, (G, τ) contient au moins un motif de S_5. Si G est biparti, on vérifie si (G, τ) contient chacun des motifs de S_5 en utilisant la coloration en rouge et bleu de G.

Tout d'abord, un graphe biparti n'a pas de cycle impair et donc le motif TRIANGLE ne pas apparaitre dans (G, τ). Concernant co-COMPARABILITY, on parcourt chaque arête $e = (i, k)$ de G pour vérifier si elle peut être l'arête du motif co-COMPARABILITY. Comme G est biparti, une extrémité de e est bleue tandis que l'autre est rouge. Sans perte de généralité, i est bleu et k est rouge. On teste pour chaque arête $e = (i, k)$ de G si la formule $\phi(e) =$ « $\min \overline{N^+}(i) \cap V_R < k$ ou $\max \overline{N^-}(k) \cap V_B > i$ » est vraie ou non. Si $\phi(e)$ est vraie, on a trouvé un sommet j entre i et k tel que i n'est pas voisin avec l'une des extrémités de e et est de la couleur de l'autre extrémité. Ainsi, i, j, k réalise co-COMPARABILITY. Si $\phi(e)$ est faux, il n'existe pas de j tel que i, j, k réalise co-COMPARABILITY. En effet, si c'était le cas, selon la couleur de j, une des inégalités serait vraie. S'il est bleu, on sait que j n'est pas voisin avec k et donc $j \in \overline{N^-}(k) \cap V_B$ avec $j > i$, ce qui implique $\max \overline{N^-}(k) \cap V_B > i$. Inversement, si j est rouge, on a $\min \overline{N^+}(i) \cap V_R < k$. On a donc l'algorithme suivant :

Algorithme 11 : Recherche linéaire de l'ensemble de motifs S_5.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient un des motifs de S_5 » ou « G évite tous les motifs de S_5 »
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.

```
Si G n'est pas biparti alors
    Retourner « G contient un des motifs de S_5 »
(V_B, V_R) ← décomposition des sommets de G en bleu et rouge
Calculer les valeurs de min N+(i) ∩ V_R pour tout sommet i ∈ V(G)
Calculer les valeurs de max N-(k) ∩ V_B pour tout sommet k ∈ V(G)
Pour chaque e = (i, k) ∈ E(G) faire
    Si min N+(i) ∩ V_R < k ou max N-(k) ∩ V_B > i alors
        Retourner « G contient un des motifs de S_5 »
Retourner « G évite tous les motifs de S_5 »
```

Pour que cet algorithme soit bien linéaire, il reste à vérifier que l'on peut calculer les valeurs de $\min \overline{N^+}(i) \cap V_R$ pour tout sommet $i \in V(G)$ en temps linéaire (et une méthode similaire fonctionnera pour calculer les valeurs de $\max \overline{N^-}(k) \cap V_B$). Comme à chaque fois que l'on veut parcourir $\overline{N^+}(i)$, on parcourt en réalité $N^+(i)$. Si on note L la liste des éléments de $N^+(i)$ et si on prend j un indice, la paire de valeur $L[j], L[j+1]$ représente l'ensemble des non-voisins de i entre $L[j]$ et $L[j+1]$, c'est-à-dire l'intervalle $\{L[j]+1, \ldots, L[j+1]-1\}$. Dès que cet intervalle est non-vide, on a trouvé l'élément minimal de $\overline{N^+}(i)$. Cependant, ici, on cherche l'élément minimal de $\overline{N^+}(i) \cap V_R$. Il faut donc vérifier en temps constant si l'intervalle $\{L[j]+1, \ldots, L[j+1]-1\}$ a une intersection non-vide avec V_R. Pour ce faire, on pré-calcule un tableau R tel que $R[j]$ contient le plus petit élément de V_R qui est strictement plus grand que j. Cela se fait en temps $\mathcal{O}(n)$ en parcourant V_R (que l'on a préalablement trié en temps $\mathcal{O}(n+m)$ comme décrit en section 6.2.1). Maintenant,

pour vérifier en temps constant si l'intervalle $\{L[j]+1,\ldots,L[j+1]-1\}$ a une intersection non-vide avec V_R, on regarde si $L[j]+1 \in V_R$ (faisable en temps constant via un tableau de booléens pré-calculé représentant V_R). Si oui, l'intervalle $\{L[j]+1,\ldots,L[j+1]-1\}$ a une intersection non-vide avec V_R. Si non, on compare $R[L[j]+1]$ avec $L[j+1]-1$. Si le premier est plus petit ou égal que le second, l'intervalle $\{L[j]+1,\ldots,L[j+1]-1\}$ a une intersection non-vide avec V_R. Dans le cas contraire, l'intervalle $\{L[j]+1,\ldots,L[j+1]-1\}$ a une intersection vide avec V_R. □

Proposition 6.13. *L'ensemble $S_6 = \{$Triangle, co-Chordal$\}$ peut être détecté en temps linéaire.*

Démonstration. En utilisant l'algorithme 7, on peut vérifier en temps linéaire si (G,τ) contient co-Chordal. Si oui, on a trouvé un motif de S_6 dans (G,τ). Si non, on sait que le complémentaire de G est cordal. On calcule un ordre d'élimination simplicial du complémentaire de G, ce qui peut se faire directement à partir de G sans passer au complémentaire via un LexBFS [HMPV00]. À partir de cet ordre, on peut calculer la taille du plus grand indépendant du complémentaire de G [RTL76]. S'il est de taille 3 ou plus, on a trouvé un indépendant de taille 3 dans le complémentaire de G et G contient donc le motif Triangle. Sinon, G ne contient pas le motif Triangle. □

Proposition 6.14. *L'ensemble $S_7 = \{$co-Comparability, Bipartite$\}$ peut être détecté en temps linéaire.*

Démonstration. On a $S_7 \simeq \{$Comparability, co-Comparability, Bipartite$\} = S_1 \cup \{$Bipartite$\}$. □

Ainsi, chaque paire fondamentale peut être certifiée en temps linéaire. On montre maintenant que le complémentaire terme à terme de chacune des ces paires peut être certifié en temps linéaire. D'abord, les équivalences restent vraies quand on passe au complémentaire terme à terme, comme montré par le lemme 6.7. Il reste donc à vérifier les complémentaires terme à terme des paires fondamentales qui n'utilisent pas d'équivalence, à savoir les paires S_1, S_2, S_5 et S_6. La paire S_1 est sa propre complémentaire terme à terme. Il ne reste donc que S_2, S_5 et S_6 :

— Pour S_2 (et S_2'), le raisonnement peut être adapté au complémentaire, de sorte que la formule $\phi(i) =$ « $\overline{N^+}(j) \subseteq \overline{N^+}(i)$ », avec $j = \min \overline{N^+}(i)$, permet de tester si co-Comparability apparait dans un graphe ne contenant pas mirror-co-Chordal. Le test $\phi(i)$ est équivalent au test $\phi'(i) =$ « $\overline{\overline{N^+}(i)} \subseteq \overline{\overline{N^+}(j)}$ ». Or, $\overline{\overline{N^+}(i)} = \{i+1,\ldots,n\} \cap \overline{N^+(i)} = \{1,\ldots,i\} \cup N^+(i)$. De même, $\overline{\overline{N^+}(j)} = \{1,\ldots,j\} \cup N^+(i)$. Ainsi, la formule $\phi'(i)$ est équivalente à « $N^+(i) \subseteq \{i+1,\ldots,j\} \cup N^+(j)$ » que l'on peut tester en temps $|N^+(i)| + |N^+(j)|$: on teste si les premiers éléments de $N^+(i)$ sont dans $\{i+1,\ldots,j\} \cup N^+(j)$ en regardant simplement s'ils sont dans $\{i+1,\ldots,j\}$, c'est-à-dire s'ils sont plus petits que j. Par la même analyse que celle du lemme 6.5, l'ensemble des tests $\phi(i)$ se fait en temps linéaire.

— Concernant S_5, on doit tester si le complémentaire du graphe est biparti. Or, un tel graphe doit avoir au moins de l'ordre de $n^2/4$ arêtes. Ainsi, si le graphe a moins d'arêtes que cette borne, on sait que son complémentaire ne peut pas être biparti et qu'il contient donc un élément du complémentaire terme à terme de S_5. Sinon, son nombre d'arêtes est quadratique, ce qui signifie tout algorithme quadratique en le nombre de sommets et en réalité linéaire en le nombre d'arêtes. On peut donc passer le graphe au complémentaire en temps linéaire. Il ne reste qu'à appliquer l'algorithme 11 sur le complémentaire pour conclure.

— Pour S_6, la preuve est la même car les techniques utilisées raisonnent déjà sur le complémentaire (sans pour autant le calculer). On calcule donc simplement le plus grand indépendant de G que l'on sait cordal via l'algorithme linéaire présenté dans [RTL76].

De plus, si un ensemble de motifs peut se certifier en temps linéaire, alors son miroir terme à terme aussi, en appliquant le lemme 6.3 à chacun des motifs de l'ensemble.

Récapitulons. Chacune des 12 paires fondamentales ainsi que son complémentaire et son miroir terme à terme peut être certifiée en temps linéaire. Chaque ensemble de la liste L_{26} peut s'écrire sous la forme d'une union de paires fondamentales. Dès lors, chaque ensemble de la liste L_{26} ainsi que son complémentaire et son miroir terme à terme peuvent être certifiés en temps linéaire. Étant donné un ensemble S de la liste L_{87}, il y a deux cas :

— Soit l'ensemble S ne contient aucun motif parmi {Triangle, Comparability, co-Comparability, co-Triangle}. Dans ce cas, chaque motif de S, son complémentaire et son miroir terme à terme peut être certifié en temps linéaire d'après la section 6.2.2.

— Sinon, S appartient à L_{26} et lui ainsi que son complémentaire et son miroir terme à terme peut être certifié en temps linéaire.

Étant donné un ensemble S de motifs à trois sommets, soit il s'agit d'un singleton contenant Triangle, Comparability, co-Comparability ou co-Triangle, auquel cas S peut être certifié en temps $\mathcal{O}(n^\omega)$; soit il est équivalent à un ensemble de L_{87}, au complémentaire ou au miroir terme à terme d'un ensemble de L_{87} et peut être certifié en temps linéaire.

6.4 Recherche de motifs appartenant à une classe

6.4.1 Motifs planaires extérieurs sans cycle

Dans cette section, on s'intéresse aux *motifs planaires extérieurs sans cycle* et plus particulièrement aux motifs planaires extérieurs *positifs* sans cycle

Définition 6.10 (Motif planaire extérieur, motif positif, motif sans cycle)**.** *Un motif P est* planaire extérieur positif sans cycle *s'il vérifie les trois conditions suivantes :*

— *(Positif) Le motif P n'a pas d'arête interdite, c'est-à-dire que $F(P) = \emptyset$.*

— *(Planaire extérieur) Étant donné deux arêtes (i, j) et (i', j'), on ne peut pas avoir $i < i' < j < j'$. Les comparaisons se font selon l'ordre $\tau(P)$.*

— *(Sans cycle) Le motif P n'a pas de cycle d'arêtes.*

Si un motif ne remplit pas la condition Positif, il est seulement planaire extérieur sans cycle.

Ainsi, l'ensemble des arêtes (obligatoires ou interdites) $E(P)$ est égal à l'ensemble des arêtes obligatoires $M(P)$. Comme on s'intéresse également aux motifs planaires extérieurs sans cycle qui ne sont pas positifs ($E(P) \neq M(P)$), on utilise la notation $E(P)$ pour parler des arêtes de P afin de pouvoir couvrir à la fois les motifs positifs et non-positifs.

On montre le théorème suivant :

Théorème 6.4. *Soit P un motif planaire extérieur positif sans cycle. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n + m)$.*

Un motif de cette classe est donc détectable en temps linéaire quelle que soit la taille du motif. Cette classe est assez naturelle car être planaire extérieure est une condition souvent imposée, comme pour la coloration de graphe où un graphe planaire extérieur peut toujours être colorié avec seulement 3 couleurs. Le fait d'être sans cycle signifie que le motif est une forêt. Enfin, le fait d'être positif permet à la fois de simplifier la compréhension en n'ayant qu'un seul type d'arête et surtout de mieux exploiter la structure du graphe d'entrée, qui est donné sous forme de sa liste d'arêtes. En effet, il est plus facile de tester si une arête obligatoire du motif est présente à une endroit du graphe que de tester si une arête interdite du motif est absente à un endroit du graphe.

De plus, ces trois conditions ont l'air nécessaires pour garantir une complexité linéaire :

— Sans la planarité extérieure, le motif peut avoir des arêtes qui se croisent énormément, ce qui rend notre technique de division de motif impossible à contrôler. Nous avons des exemples de motifs positifs et sans cycle que nous conjecturons difficile à détecter (cf. conjecture 6.1).

— Sans le côté positif, on peut adapter notre algorithme pour obtenir une complexité quadratique. On pense qu'il n'est pas possible de faire mieux dans le cas général car travailler avec des arêtes interdites revient à travailler sur le complémentaire du graphe, qui se calcule en temps quadratique.

— Si le motif a des cycles, on peut adapter notre algorithme pour être aussi efficace que la multiplication de matrices (Voir section 6.4.2). On pense qu'il n'est pas possible de pouvoir faire mieux, car le motif Triangle nécessite un temps égal à celui de la multiplication de matrices d'après les hypothèses de complexité usuelles [WW10].

Catégorisation des arêtes

La première étape consiste à utiliser la planarité du motif pour ranger les arêtes dans différentes catégories, selon leur position relative. Dans la suite, toutes les comparaisons entre sommets se font selon l'ordre $\tau(P)$ du motif P considéré. On peut supposer sans perte de généralité que $\tau(P) = [1, \ldots, k]$, c'est-à-dire que l'ordre est l'ordre habituel sur les entiers de 1 à k, où k est le nombre de sommets de P.

Soit deux arêtes $(i, j), (i', j')$ telles que $i < j$, $i' < j'$ (c'est-à-dire que les deux arêtes sont ordonnées dans le même sens que le motif) et $i \leq i'$ (c'est-à-dire que l'on peut supposer sans perte de généralité que la seconde arête a son extrémité gauche à droite de l'extrémité gauche de la première arête). Dans le cas d'un motif planaire extérieur, on a uniquement deux possibilités :

— $i \leq i' < j' \leq j$. On dit que les arêtes sont *emboitées*.

— $i < j \leq i' < j'$. On dit que les arêtes sont *côte à côte*.

Dans le cas général, on pourrait aussi avoir $i < i' < j < j'$, mais cela est interdit par le caractère planaire extérieur du motif. Quand deux arêtes sont côte à côte, on dit que (i, j) est l'arête gauche et (i', j') l'arête droite. Quand deux arêtes sont emboitées, on dit que l'arête intérieure (i', j') est emboitée à l'intérieur de l'arête extérieure (i, j). Dans ce cas, on peut encore davantage caractériser l'arête intérieure (i', j') :

— (i', j') est *profondément emboitée* à l'intérieur de (i, j) s'il y a une arête (i'', j'') telle (i', j') est emboitée à l'intérieur de (i'', j'') et que (i'', j'') est elle même emboitée à l'intérieur de (i, j).

— sinon, (i', j') est *directement emboité* à l'intérieur de (i, j).

Il existe aussi des arêtes qui ne sont emboitées à l'intérieur d'aucune autre. On dit que ces arêtes sont *extérieures*. Grâce à ces catégorisations, on peut partitionner l'ensemble des arêtes emboitées à l'intérieur d'une arête (i, j) donnée. D'abord, on peut partitionner facilement les arêtes directement emboitées à l'intérieur de (i, j) en fonction de la façon dont elles sont reliées à l'arête (i, j). Étant donné une arête (i', j'), elle peut soit être reliée à (i, j), soit ne pas l'être. Si elle est reliée, elle peut atteindre l'arête (i, j) soit en passant par i, soit par j. Cela donne donc trois catégories :

— une arête (i', j') directement emboitée qui n'est pas reliée à (i, j) est appelée *arête emboitée au centre*,

— une arête (i', j') directement emboitée qui est reliée à (i, j) via le sommet i est appelée *arête emboitée à gauche*,

— une arête (i', j') directement emboitée qui est reliée à (i, j) via le sommet j est appelée *arête emboitée à droite*,

Comme nous travaillons avec des motifs sans cycle, une arête ne peut être à la fois reliée à gauche et à droite.

Ensuite, concernante un arête (i', j') profondément emboitée à l'intérieur de (i, j), on sait qu'elle est emboitée à l'intérieur d'une arête (i'', j'') elle-même directement emboitée à l'intérieur de (i, j). On donne alors à (i', j') la même catégorie que (i'', j''), c'est-à-dire que (i', j') est une arête emboitée au centre si (i'', j'') est une arête emboitée au centre, (i', j') est une arête emboitée à gauche si (i'', j'') est une arête emboitée à gauche et (i', j') est une arête emboitée à droite si (i'', j'') est une arête emboitée à droite.

Ainsi, étant donné une arête (i, j), chaque arête (i', j') emboitée à l'intérieur de (i, j) est soit une arête emboitée au centre à l'intérieur de (i, j), soit une arête emboitée à gauche à l'intérieur de (i, j), ou soit une arête emboitée à droite à l'intérieur de (i, j). La figure 6.14 présente un exemple de cette catégorisation.

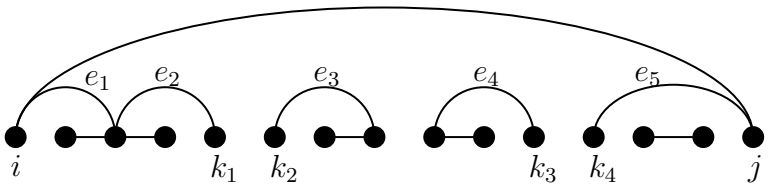


FIGURE 6.14 – Classification des arêtes emboitées à l'intérieur de (i, j). Excepté l'arête (i, j) qui n'est pas classifiée, les arêtes représentées par un arc courbe sont directement emboitées à l'intérieur de (i, j) et les arêtes représentées par une ligne droite sont profondément emboitées à l'intérieur de (i, j). Les arêtes e_1 et e_2 sont emboitées à gauche à l'intérieur de (i, j), les arêtes e_3 et e_4 sont emboitées au centre à l'intérieur de (i, j) et l'arête e_5 est emboitée à droite à l'intérieur de (i, j). En ce qui concerne les arêtes profondément emboitées à l'intérieur de (i, j), les arêtes emboitées à l'intérieur de e_1 ou e_2 sont emboitées à gauche à l'intérieur de (i, j), les arêtes emboitées à l'intérieur de e_3 ou e_4 sont emboitées au centre à l'intérieur de (i, j) et les arêtes emboitées à l'intérieur de e_5 sont emboitées à droite à l'intérieur de (i, j). Finalement, on peut résumer cette catégorisation en disant que chaque arête dont les deux extrémités sont entre le sommet i et le sommet k_1 est emboitée à droite à l'intérieur de (i, j), chaque arête dont les deux extrémités sont entre le sommet k_2 et le sommet k_3 est emboitée au centre à l'intérieur de (i, j) et chaque arête dont les deux extrémités sont entre le sommet k_4 et le sommet j est emboitée à droite à l'intérieur de (i, j). Il est possible de montrer qu'il existe toujours des sommets k_1, k_2, k_3 et k_4 qui permettent un tel partitionnement.

Décomposition d'un motif

Grâce à cette classification des arêtes, il est possible de décomposer un motif planaire extérieur (positif) sans cycle. Tout d'abord, étant donné un motif P planaire extérieur (positif) sans cycle, on définit une arête spéciale qui va jouer le rôle de l'arête (i, j) de la partie précédente et à partir de laquelle on va pouvoir parler d'arête emboitée à gauche, au centre ou à droite et d'arête côte à côte à gauche ou à droite. On définit aussi une seconde arête spéciale qui joue un rôle symétrique.

Définition 6.11 (Arêtes LOME et ROME)**.** *Soit P un motif planaire extérieur (positif) sans cycle. On définit les deux arêtes spéciales suivantes :*

- *Soit LOME l'arête extérieure la plus à gauche (Left-OuterMost Edge en anglais) du motif P, c'est-à-dire une arête (i, j) qui n'est pas emboitée à l'intérieur d'une autre et telle que le sommet i est minimal parmi toutes les arêtes extérieures (i, j).*
- *Soit ROME l'arête extérieure la plus à droite (Right-OuterMost Edge en anglais) du motif P, c'est-à-dire une arête (i, j) qui n'est pas emboitée à l'intérieur d'une autre et telle que le sommet j est maximal parmi toutes les arêtes extérieures (i, j).*

Le fait que l'on impose que LOME soit extérieure signifie que le sommet j est maximal parmi toutes les arêtes (i, j) avec i le plus petit sommet à être l'extrémité d'une arête. De même, le fait que l'on impose que ROME soit extérieur signifie que le sommet i est minimal parmi toutes les arêtes (i, j) avec j le plus grand sommet à être l'extrémité d'une arête.

À partir de l'arête LOME (respectivement ROME), on peut partitionner l'ensemble des autres arêtes du motif P en trois ensembles, comme montré sur la figrue 6.15.

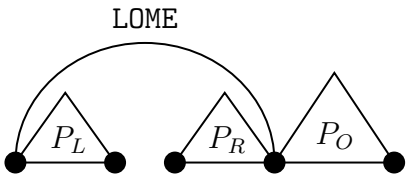


FIGURE 6.15 – Schéma des trois sous-motifs induits par l'arête LOME. Chaque triangle représente un sous-motif. Chacun des ces sous-motifs peut éventuellement être vide. De plus, les motifs P_L et P_O peuvent éventuellement ne pas être connectés à LOME. En revanche, si P_R est non-vide, d'après la définition d'une arête emboitée à droite, il est forcément connecté.

On discute d'abord du cas de LOME. Soit $\texttt{LOME} = (i_{\texttt{LOME}}, j_{\texttt{LOME}})$, avec $i_{\texttt{LOME}} < j_{\texttt{LOME}}$. D'après la catégorisation, toute arête (i, j) est soit côte à côte de LOME soit emboitée à l'intérieur de LOME. Dans le cas où (i, j) est côte à côte de LOME, elle ne peut être qu'à droite de LOME, car LOME est une arête la plus à gauche du motif P. Dans le cas où (i, j) est emboitée à l'intérieur de LOME, on a vu que (i, j) peut être soit emboitée à gauche, soit emboitée au centre, soit emboitée à droite.

On définit alors les trois sous-motifs suivants :

- Le motif P_L constitué des arêtes emboitées à gauche à l'intérieur de $(i_{\texttt{LOME}}, j_{\texttt{LOME}})$ et des arêtes emboitées au centre à l'intérieur de $(i_{\texttt{LOME}}, j_{\texttt{LOME}})$.
- Le motif P_R constitué des arêtes emboitées à droite de $(i_{\texttt{LOME}}, j_{\texttt{LOME}})$.
- Le motif P_O constitué des arêtes côte à côte à droite de $(i_{\texttt{LOME}}, j_{\texttt{LOME}})$.

On a bien une partition des arêtes de P autres que LOME, car chacune des ces arêtes est exactement dans l'un des sous-motifs parmi P_L, P_R ou P_O. Pour définir complètement les motifs P_L, P_R et P_O, il faut également parler de leur ensemble de sommets. Pour $i = L$, R ou O, l'ensemble des sommets $V(P_i)$ du sous-motif P_i est l'intervalle $[u, v] \subseteq v(P)$ avec u le plus petit sommet qui est une extrémité d'une arête de $E(P_i)$ et u le plus grand sommet qui est une extrémité d'une arête de $E(P_i)$. Autrement dit, $V(P_i)$ est l'ensemble des sommets induits par les arêtes $E(P_i)$ de P_i, auquel on a ajouté les éventuels sommets isolés qui se trouveraient au milieu, de sorte à ce que $V(P_i)$ soit un intervalle. Pour être complet, l'ordre $\tau(P_i)$ est égal à l'ordre τ restreint à $V(P_i)$: $\tau(P_i) = \tau|_{V(P_i)}$. Dans le cas où P_i n'a pas d'arête, on considère qu'il est constitué d'un sommet :

— Si P_L n'a pas d'arête, on pose $V(P_L) = \{i_{\texttt{LOME}}\}$.

— Si P_R n'a pas d'arête, on pose $V(P_R) = \{j_{\texttt{LOME}}\}$.

— Si P_O n'a pas d'arête, on pose $V(P_O) = \{j_{\texttt{LOME}}\}$.

Cette convention permet d'éviter de travailler avec un ensemble de sommets vide. À partir de ce découpage en sous-motifs, on définit 6 longueurs, qui sont représentées sur la figure 6.16 :

— La longueur $\texttt{ecart}(i-)$ qui est le nombre de sommets à gauche de $i_{\texttt{LOME}}$, $i_{\texttt{LOME}}$ exclu. Mathématiquement, cette longueur vaut $\texttt{ecart}(i-) := i_{\texttt{LOME}} - 1$.

— La longueur $\texttt{ecart}(iL)$ qui est le nombre de sommets entre $i_{\texttt{LOME}}$ et le sommet le plus à gauche de P_L, en incluant l'un et excluant l'autre. Ce nombre vaut 0 si P_L est relié à l'arête LOME (c'est-à-dire si P_L contient au moins une arête emboitée à gauche à l'intérieur de LOME) et vaut un nombre strictement positif s'il ne l'est pas (c'est-à-dire si P_L ne contient pas d'arête emboitée à gauche à l'intérieur de LOME). Mathématiquement, cette longueur vaut $\texttt{ecart}(iL) := \min V(P_L) - i_{\texttt{LOME}}$.

— La longueur $\texttt{ecart}(LR)$ qui est le nombre de sommets entre le sommet le plus à droite de P_L et le sommet le plus à gauche de P_R, en incluant l'un et excluant l'autre. Ce nombre ne peut pas être nul par définition. Mathématiquement, cette longueur vaut $\texttt{ecart}(LR) := \min V(P_R) - \max V(P_L)$.

— La longueur $\texttt{ecart}(RO)$ qui est le nombre de sommets entre $j_{\texttt{LOME}}$ et le sommet le plus à gauche de P_O, en incluant l'un et excluant l'autre. Ce nombre vaut 0 si P_O est relié à l'arête LOME et vaut un nombre strictement positif sinon. Mathématiquement, cette longueur vaut $\texttt{ecart}(RO) := \min V(P_O) - j_{\texttt{LOME}}$.

— La longueur $\texttt{ecart}(O+)$ qui est le nombre de sommets à droite du sommet le plus à gauche de P_O, ce sommet exclut. Mathématiquement, cette longueur vaut $\texttt{ecart}(O+) := k - \max V(P_O)$.

— La longueur $\texttt{ecart}(j+)$ qui est le nombre de sommets à droite de $j_{\texttt{LOME}}$, $j_{\texttt{LOME}}$ exclut. Mathématiquement, cette longueur vaut $\texttt{ecart}(j+) := k - j_{\texttt{LOME}}$.

En ce qui concerne ROME, on a un découpage identique, excepté pour le motif P_O :

— Le motif P_L constitué des arêtes emboitées à gauche à l'intérieur de $(i_{\texttt{ROME}}, j_{\texttt{ROME}})$ et des arêtes emboitées au centre à l'intérieur de $(i_{\texttt{ROME}}, j_{\texttt{ROME}})$.

— Le motif P_R constitué des arêtes emboitées à droite de $(i_{\texttt{ROME}}, j_{\texttt{ROME}})$.

— Le motif P_O constitué des arêtes côte à côte à *gauche* de $(i_{\texttt{ROME}}, j_{\texttt{ROME}})$.

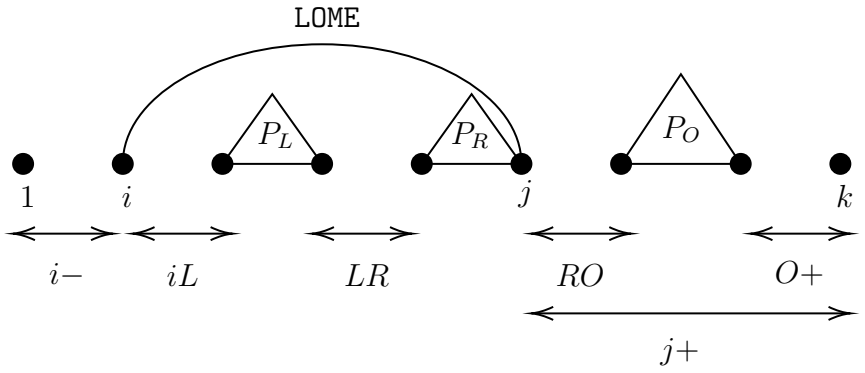


FIGURE 6.16 – Les différentes longueurs `ecart`$(\cdot)$ induites par l'arête `LOME`.

Sommets extrémaux

Étant donné un motif planaire extérieur positif sans cycle P et un graphe ordonné (G, τ), on s'intéresse aux sous-graphes ordonnés $(H, \tau|_H)$ de (G, τ) qui réalisent P, et plus particulièrement aux sommets extrémaux de H, à savoir le plus petit sommet de H (noté $\min H$) et le plus grand sommet de H (noté $\max H$). Ce minimum et ce maximum sont calculés en fonction de l'ordre τ, que l'on peut supposer être l'ordre habituel, c'est-à-dire que $\tau = [1, \ldots, n]$, où n est le nombre de sommets de G. On rappelle que $\tau|_H$ désigne l'ordre τ restreint aux sommets de H.

Notons $\mathcal{M}_P$ l'ensemble des paires de sommets (u, v) de (G, τ) telles qu'il existe un sous-graphe ordonné $(H, \tau|_H)$ de (G, τ) qui réalise P avec $u = \min H$ et $v = \max H$. Idéalement, on aimerait connaitre parfaitement l'ensemble $\mathcal{M}_P$. Cela permettrait de savoir où se trouve l'ensemble des sous-graphes qui réalisent P dans (G, τ) et donc d'avoir un algorithme de détection de P récursif en décomposant P en sous-motifs. Cependant, nous ne pouvons pas calculer toutes les paires (u, v) de $\mathcal{M}_P$ dans notre cas, car cela demanderait un temps au moins quadratique, car $\mathcal{M}_P$ peut avoir une taille quadratique. Heureusement, il est possible de calculer en temps linéaire des quantités liées à $\mathcal{M}_P$ qui suffisent pour détecter P.

On définit les quatre fonctions suivantes, qui associent chacune un sommet de G à un autre sommet de G en fonction de l'ensemble $\mathcal{M}_P$:

Définition 6.12 (Notations m_P^+, m_P^-, M_P^+ et M_P^-)**.** *On définit :*

- $m_P^+(u) = \inf\{v \in V(G) \mid (u, v) \in \mathcal{M}_P\}$,
- $m_P^-(v) = \sup\{u \in V(G) \mid (u, v) \in \mathcal{M}_P\}$,
- $M_P^+(u) = \inf\{v \in V(G) \mid \exists u' \geq u, (u', v) \in \mathcal{M}_P\}$,
- $M_P^-(v) = \sup\{u \in V(G) \mid \exists v' \leq v, (u, v') \in \mathcal{M}_P\}$.

Dans notre cas, inf est synonyme de min et sup est synonyme de max, car $\mathcal{M}_P$ est un ensemble fini. Il y a une exception cependant : $\inf \emptyset$ est parfaitement défini et vaut $+\infty$, et $\sup \emptyset$ est parfaitement défini et vaut $-\infty$. Nous utiliserons ces égalités dans notre algorithme.

Remarque 6.3. On a :

- $M_P^+(u) = \inf\{m_P^+(u') \mid u' \geq u\}$,
- $M_P^-(v) = \sup\{m_P^-(v') \mid v' \leq v\}$.

Cela implique que si l'on connait les valeurs de $m_P^+(u')$ et $m_P^-(v')$ pour tout $u' \in V(G)$ et pour tout $v' \in V(G)$, alors on peut en déduire facilement les valeurs de $M_P^+(u)$ et $M_P^-(v)$ pour tout $u \in V(G)$ et pour tout $v \in V(G)$.

Algorithme

On construit un tableau de taille n qui renseigne la valeur de $m_P^+(u)$ pour chaque sommet u du graphe ordonné (G, τ). On fait de même pour les fonctions m_P^-, M_P^+ et M_P^-. Cela peut se faire récursivement de façon linéaire, en décomposant le motif P suivant `LOME` ou `ROME` pour en déduire trois motifs P_L, P_R et P_O et en répétant le calcul du tableau sur ces trois motifs, comme cela est présenté dans le lemme 6.15.

Lemme 6.15. *Soit P un motif planaire extérieur positif sans cycle avec ℓ arêtes. Soit (G, τ) un graphe ordonné avec n sommets et m arêtes. Soit f une fonction qui est soit m_P^+ ou soit m_P^-. On peut construire un tableau T de taille n tel que $T[u] = f(u)$ pour tout sommet $u \in V(G)$ en temps linéaire en la taille du graphe $c\ell(n+m)$, pour une certaine constante c.*

Algorithme 12 : Recherche du motif P constitué de k sommets et de $\ell = 0$ arête.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : Un tableau T tel que $T[u] = m_P^+(u)$
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.
T est un tableau avec des indices allant 1 à n, où chaque case est initialisé à $+\infty$
Pour u allant de 1 à $n - k + 1$ **faire**
    $T[u] = u + k - 1$
Retourner T

Algorithme 13 : Recherche du motif P constitué de k sommets et de $\ell = 1$ arête.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : Un tableau T tel que $T[u] = m_P^+(u)$
Sans perte de généralité, on suppose $V(G) = \{1, \ldots, n\}$ et $\tau = [1, \ldots, n]$.
On suppose aussi que $V(P) = \{1, \ldots, k\}$, $\tau(P) = [1, \ldots, k]$
On note (a, b) l'unique arête de P
T est un tableau avec des indices allant 1 à n, où chaque case est initialisé à $+\infty$
Pour chaque $(u, v) \in E(G)$ **faire**
    Si $u \geq a$ et $v - u \geq b - a$ et $n - v \geq k - b$ **alors**
        $T[u - a] = \min(T[u - a], v + n - k + b)$
Retourner T

Démonstration. On montre ce lemme par récurrence sur le nombre d'arêtes ℓ du motif. Quand $\ell = 0$ ou $\ell = 1$, le tableau T se construit facilement en parcourant l'ensemble des arêtes du graphe G, comme cela est montré dans les algorithmes 12 et 13 dans le cas $f = m_P^+$.

Algorithme 14 : Recherche récursive d'un motif P ayant k sommets et ℓ arêtes.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : Un tableau T tel que $T[u] = m_P^+(u)$
Sans perte de généralité, on suppose $V(G) = \{1, \dots, n\}$ et $\tau = [1, \dots, n]$.
On suppose aussi que $V(P) = \{1, \dots, k\}$ et $\tau(P) = [1, \dots, k]$
On définit les sous-motifs P_L, P_R et P_O
On définit les longueurs $\texttt{ecart}(i-)$, $\texttt{ecart}(Li)$, $\texttt{ecart}(LR)$, $\texttt{ecart}(RO)$, $\texttt{ecart}(O+)$, $\texttt{ecart}(j+)$
On calcule récursivement les tableaux T_L, T_R et T_O tels que $T_L[u] = m_{P_L}^+(u)$, $T_R[v] = m_{P_R}^-(v)$ et $T_O[u] = m_{P_O}^+(u)$
Si $\texttt{ecart}(Li) > 0$ **alors**
    // On remplace le contenu de T_L pour que $T_L[u] = M_{P_L}^+(u)$
    Pour u allant de $n-1$ à 1 **faire**
        $T_L[u] = \min(T_L[u], T_L[u+1])$
Si $\texttt{ecart}(RO) > 0$ **alors**
    // On remplace le contenu de T_O pour que $T_O[u] = M_{P_O}^+(u)$
    Pour u allant de $n-1$ à 1 **faire**
        $T_O[u] = \min(T_O[u], T_O[u+1])$
T est un tableau avec des indices allant 1 à n, où chaque case est initialisé à $+\infty$
Pour chaque $(u, v) \in E(G)$ **faire**
    // On fait correspondre l'arête (u, v) de G avec l'arête LOME de P
    Si $u \geq \texttt{ecart}(i-)$ et $T_L[u + \texttt{ecart}(Li)] + \texttt{ecart}(LR) < T_R[v]$ et $T_O[v + \texttt{ecart}(RO)] + \texttt{ecart}(O+) \leq n$ **alors**
        $T[u - \texttt{ecart}(i-)] = \min(T[u - \texttt{ecart}(i-)], v + \texttt{ecart}(j+))$
Retourner T

Algorithme 15 : Recherche d'un motif planaire extérieur positif sans cycle P.

Entrée : Un graphe ordonné (G, τ) de n sommets
Sortie : « G contient P » ou « G évite P»
Sans perte de généralité, on suppose $V(G) = \{1, \dots, n\}$ et $\tau = [1, \dots, n]$.
Calculer T en utilisant l'algorithme 14
Pour u allant de 1 à n **faire**
    Si $T[u] < +\infty$ **alors**
        Retourner « G contient P »
Retourner « G évite P »

Pour le cas général, on suppose que le lemme est vrai pour tout motif ayant ℓ arêtes ou moins et on montre qu'il est vrai pour un motif ayant $\ell+1$ arêtes. On traite d'abord le cas $f = m_P^+$. On considère la décomposition de P suivant son arête `LOME` en trois sous-motifs P_L, P_R et P_O. Par récurrence, on peut obtenir toutes les valeurs de $m_{P_L}^+$ en temps $c|E(P_L)|(n+m)$, toutes les valeurs de $m_{P_R}^-$ en temps $c|E(P_R)|(n+m)$ et toutes les valeurs de $m_{P_O}^+$ en temps $c|E(P_O)|(n+m)$. À partir de $m_{P_L}^+$, on peut calculer $M_{P_L}^+$ en temps n via la formule $m_{P_L}^+(u) = \inf\{m_{P_L}^+(u') \mid u' \geq u\}$. De même, à partir de $m_{P_O}^+$, on peut calculer $M_{P_O}^+$ en temps n via la formule $m_{P_O}^+(u) = \inf\{m_{P_O}^+(u') \mid u' \geq u\}$. On a donc accès à toutes ces quantités en temps $c(|E(P_L)| + |E(P_R)| + |E(P_O)|)(n+m) + 2n = c\ell(n+m) + 2n$. Maintenant, comme ceci est décrit dans l'algorithme 14, on parcourt chaque arête (u, v) du graphe, on essaie d'identifier cette arête avec l'arête `LOME` du motif P et on utilise les valeurs de $m_{P_L}^+$, $m_{P_R}^-$, $m_{P_O}^+$, $M_{P_L}^+$ et $M_{P_O}^+$ pour calculer une borne sur la valeur de $m_P^+(u)$. En faisant cela pour toutes les arêtes, on aura déduit la valeur exacte de $m_P^+(u)$. On fait un calcul par arête, on a donc une complexité de m pour cette boucle. Finalement, la complexité totale est $c\ell(n+m) + 2n + m \leq c(\ell+1)(n+m)$ si tant est que $c \geq 2$.

Une analyse similaire permet de traiter le cas de $f = m_P^-$, en utilisant `ROME` au lieu de `LOME`. □

On peut remarquer que la décomposition du motif P en trois sous-motifs ainsi que le calcul des longueurs `ecart`$(\cdot)$ ne sont pas mentionné dans la complexité. En effet, le but est ici de créer un méta-algorithme qui étant donné un motif planaire extérieur positif sans cycle P, crée un algorithme permettant de détecter P dans un graphe (G, τ). Ainsi, la complexité nécessaire pour créer ce méta-algorithme n'est pas à prendre en compte dans l'algorithme de détection. Concrètement, ce méta-algorithme découpe récursivement P en trois sous-motifs et permet de savoir pour chaque arête de P s'il faudra la découper selon `LOME` ou `ROME`.

L'algorithme 15 présente l'algorithme linéaire capable de détecter un motif planaire extérieur positif sans cycle en utilisant le calcul de la fonction m_P^+ fait en temps linéaire, prouvant ainsi le théorème 6.4 :

Théorème 6.4. *Soit P un motif planaire extérieur positif sans cycle. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps linéaire $\mathcal{O}(n+m)$.*

Cet algorithme fonctionne en l'état pour un motif planaire extérieur sans cycle P qui ne serait pas positif. Seulement, la complexité change, car chaque boucle qui s'effectuait sur toutes les arêtes (interdites ou obligatoires) de G (**Pour chaque** $(u, v) \in E(G)$) nécessite alors un temps quadratique et non plus linéaire. On en déduit alors le théorème 6.5 :

Théorème 6.5. *Soit P un motif planaire extérieur sans cycle. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps quadratique $\mathcal{O}(n^2)$.*

6.4.2 Motifs planaires extérieurs avec cycle

On s'intéresse maintenant aux *motifs planaires extérieurs*. Il s'agit d'une généralisation des motifs présentés dans la section 6.4.1, puisque l'on n'impose plus que le motif soit sans cycle ou positif. On rappelle la définition 6.10 qui indique qu'un motif planaire extérieur est un motif tel qu'étant donné deux arêtes (i, j) et (i', j'), on ne peut pas avoir $i < i' < j < j'$.

Les comparaison se font selon l'ordre $\tau(P)$. De façon visuelle, un motif planaire extérieur est un motif P tel que si on dessine les sommets $V(P)$ sur une ligne droite selon l'ordre $\tau(P)$ et si l'on dessine les arêtes $E(P)$ par des demi-cercles au-dessus de cette ligne, alors les arêtes ne se croisent pas, comme cela est montré sur la figure 6.17.

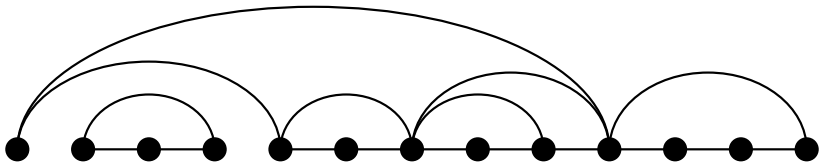


FIGURE 6.17 – Exemple d'un motif planaire extérieur avec 14 sommets et 17 arêtes. Pour simplifier la lisibilité, certaines arêtes sont dessinées via des ellipses ou des lignes droites plutôt que par des demi-cercles.

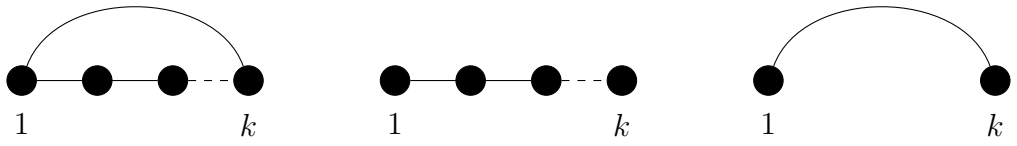
Le but de cette section est de démontrer le théorème suivant :

Théorème 6.6. *Soit P un motif planaire extérieur. On peut décider si un graphe ordonné (G, τ) à n sommets et m arêtes contient ou évite P en temps sous-cubique $\mathcal{O}(n^\omega)$.*

Pour ce faire, on commence par prouver le lemme 6.16 qui décrit récursivement la structure d'un motif planaire extérieur. Ensuite, le lemme 6.17 décrit l'algorithme de détection de motif planaire extérieur en utilisant leur structure récursive.

Lemme 6.16. *Un motif planaire extérieur P ayant pour ensemble de sommets $V(P) = \{1, \ldots, k\}$, avec $k > 1$, peut se décomposer d'une des deux façons suivantes :*

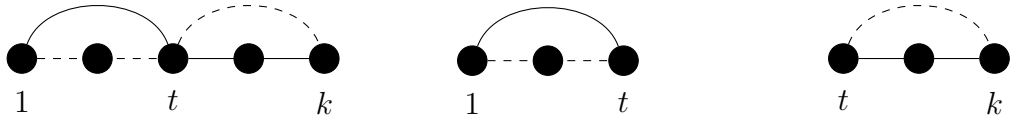
1. *(Arête recouvrante) Il y a l'arête $(1, k)$ dans P. On écrit $P \setminus (1, k)$ pour désigner le motif P privé de cette arête. $P \setminus (1, k)$ est un motif planaire extérieur. Cf. figure 6.18.*
2. *(Sommet séparateur) Il y a un sommet t, avec $1 < t < k$, tel qu'il n'y a aucune arête (i, j) avec $i < t < j$. On note $P[1, t]$ et $P[t, k]$ les deux motifs ayant respectivement pour ensemble de sommets $\{1, \ldots, t\}$ et $\{t, \ldots, k\}$. Cf. figure 6.19.*



(a) Motif P. (b) Motif $P \setminus (1, k)$. (c) Arête $(1, k)$.

FIGURE 6.18 – Décomposition d'un motif P qui a une arête recouvrante en un motif $P \setminus (1, k)$ et en une arête $(1, k)$.

Démonstration. On regarde le voisinage du sommet 1. S'il y a une arête entre 1 et k, on est dans le cas d'une *arête recouvrante*. Sinon, soit le sommet 1 n'a pas de voisin et on est dans le cas d'un *sommet séparateur* avec $t = 2$, soit il y a une arête entre le sommet 1 et d'autres sommets t avec $1 < t < k$. On prend dans ce cas t le plus grand de ces sommets. Il suffit alors de vérifier qu'il n'y a pas d'arête (i, j) avec $i < t < j$. Pour $i = 1$, puisque t est maximal, il n'y a pas d'arête $(1, j)$ avec $t < j$. Pour $i > 1$, cela signifie que les sommets vérifient $1 < i < t < j$. Or, puisque P est planaire extérieur et d'après la définition 6.10, il ne peut pas exister à la fois l'arête $(1, t)$ et l'arête (i, j). Vu que l'arête $(1, t)$ est présente, c'est (i, j) qui n'existe pas. Ainsi, on est dans le cas d'un *sommet séparateur*. □



(a) Motif P (b) Motif $P[1,t]$ (c) Motif $P[t,k]$

FIGURE 6.19 – Décomposition d'un motif P qui a un sommet séparateur t en un motif $P[1,t]$ et en un motif $P[t,k]$.

Lemme 6.17. *Soit P un motif planaire extérieur avec k sommets et ℓ arêtes. Soit (G,τ) un graphe ordonné avec n sommets. Soit T le tableau binaire bi-dimensionnel de taille $n \times n$ tel que pour toute paire de sommets $u, v \in V(G)$, $T[u][v] = 1$ si et seulement si (G,τ) contient P avec u qui correspond à 1 et v qui correspond à k. Cela signifie que dans la définition 6.9, $f(1) = u$ et $f(k) = v$. Alors on peut construire T en temps $c(k^2+\ell)n^\omega$, pour une certaine constante c.*

Démonstration. Tout d'abord, on peut construire la matrice d'adjacence de G en temps quadratique $\mathcal{O}(n^2)$. On montre ce lemme par récurrence sur le nombre de sommets k du motif. Si $k = 1$, $T[u][v] = 1$ pour toute paire de sommets $u, v \in V(G)$.

Pour le cas général, on suppose que le lemme est vrai pour tout motif ayant k sommets ou moins et pour tout motif ayant ℓ arêtes ou moins et on montre qu'il est vrai pour un motif P ayant $k+1$ sommets et $\ell+1$ arêtes. D'après le lemme 6.16, il y a deux cas possibles :

— Si P a une *arête recouvrante*, alors on peut considérer le motif $P' = P \setminus (1, k+1)$ qui possède ℓ arêtes et $k+1$ sommets. Par récurrence, on peut construire le tableau T' qui correspond au motif P' en temps $c((k+1)^2+\ell)n^\omega$. Si $(1,k+1)$ est une arête obligatoire, on a $T[u][v] = 0$ si (u,v) n'est pas une arête de G et $T[u][v] = T'[u][v]$ dans le cas contraire. Si $(1,k+1)$ est une arête interdite, on a $T[u][v] = 0$ si (u,v) est une arête de G et $T[u][v] = T'[u][v]$ dans le cas contraire. Dans les deux cas, pour toute paire de sommets $u, v \in V(G)$, on peut déterminer si (u,v) est une arête ou non via la matrice d'adjacence de G puis calculer $T[u][v]$ à partir de $T'[u][v]$ en temps constant, ce qui permet de construire T à partir de T' en temps quadratique n^2. Finalement, on construit T en temps $c((k+1)^2+\ell)n^\omega+n^2 \leq c((k+1)^2+\ell)n^\omega+cn^\omega = c((k+1)^2+\ell+1)n^\omega$.

— si P a un *sommet séparateur* t avec $1 < t < k+1$, alors on peut considérer les motifs $P_1 = P[1,t]$ et $P_2 = P[t,k+1]$ qui possèdent respectivement k_1 et k_2 sommets avec $1 < k_1 < k+1$, $1 < k_2 < k+1$ et $k_1 + k_2 = (t-1+1) + (k+1-t+1) = k+2$, et qui possèdent respectivement ℓ_1 et ℓ_2 arêtes avec $\ell_1 + \ell_2 = \ell+1$. Par récurrence, on peut construire le tableau T_1 qui correspond au motif P_1 en temps $c(k_1^2+\ell_1)n^\omega$ et le tableau T_2 qui correspond au motif P_2 en temps $c(k_2^2+\ell_2)n^\omega$.

Ensuite, on a $T[u][v] = 1$ si, et seulement si, il existe un w tel que $T_1[u][w] = 1$ et $T_2[w][v] = 1$. En effet, s'il existe un tel w, alors (G,τ) contient P_1 avec u qui correspond au sommet 1 de P_1 et w qui correspond au sommet k_1 de P_1, c'est-à-dire qu'il existe une fonction f_1 vérifiant les conditions de la définition 6.9 telle que $f_1(1) = u$ et $f_1(k_1) = w$; et (G,τ) contient P_2 avec w qui correspond au sommet 1 de P_2 et v qui correspond au sommet k_2 de P_2, c'est-à-dire qu'il existe une fonction f_2 vérifiant les conditions de la définition 6.9 telle que $f_2(1) = w$ et $f_2(k_2) = v$. Ainsi, on construit une fonction f vérifiant les conditions de la définition 6.9 telle

que $f(1) = u$ et $f(k+1) = v$. Pour cela on pose :

$$f(i) = \begin{cases} f_1(i) & \text{si } i \leq k_1 \\ f_2(i - k_1 + 1) & \text{si } i \geq k_1 \end{cases}$$

La fonction f est bien définie pour $i = k_1$ car la première formule donne $f(k_1) = f_1(k_1) = w$ et la seconde donne $f_2(k_1 - k_1 + 1) = f_2(1) = w$. On a bien $f(1) = f_1(1) = u$ et $f(k+1) = f_2(k-k_1+1+1) = f_2(k_1+k_2-2-k_1+1+1) = f_2(k_2) = v$. Il reste à montrer que f vérifie les conditions de la définition 6.9 :

— Soit $x, y \in V(P)$ tel que $x <_{\tau(P)} y$. Si $x <_{\tau(P)} k_1$ et $y <_{\tau(P)} k_1$, on a $f(x) = f_1(x)$ et $f(y) = f_1(y)$. Comme f_1 respecte l'ordre, on a $f_1(x) <_\tau f_1(y)$ et donc $f(x) <_\tau f(y)$. De même, si $x \geq_{\tau(P)} k_1$ et $y \geq_{\tau(P)} k_1$, puisque f_2 respecte l'ordre, on a $f_2(x) <_\tau f_2(y)$ et donc $f(x) <_\tau f(y)$. Si $x <_{\tau(P)} k_1$ et $y \geq_{\tau(P)} k_1$, alors $f(x) = f_1(x)$ et $f(y) = f_2(y-k_1+1)$. On ensuite $f(x) = f_1(x) < f_1(k_1) = w = f_2(1) \leq f_2(y - k_1 + 1) = f(y)$. Enfin, si $x \geq_{\tau(P)} k_1$ et $y <_{\tau(P)} k_1$, on a $x \geq_{\tau(P)} y$, ce qui contredit l'hypothèse $x <_{\tau(P)} y$. Ainsi, f conserve l'ordre.

— La fonction f respecte bien les arêtes obligatoires et interdites parce que f_1 et f_2 respectent les arêtes obligatoires et interdites et parce qu'il n'y a aucune arête entre P_1 et P_2.

Dans l'autre sens, si $T[u][v] = 1$, (G, τ) contient P avec u qui correspond à 1 et v qui correspond à $k+1$. Il existe donc une fonction f qui associe les sommets de P à certains sommets de (G, τ). Soit w le sommet de (G, τ) tel que $f(t) = w$. Ainsi, en posant $f_1(i) = f(i)$, on déduit que (G, τ) contient P_1 avec u qui correspond à 1 et w qui correspond à $t = k_1$. En posant $f_2(i) = f(i+t-1)$, on déduit que (G, τ) contient P_2 avec w qui correspond à 1 car $f_2(1) = f(1+t-1) = f(t) = w$ et v qui correspond à k_2 car $f_2(k_2) = f(k_2 + t - 1) = f(k_2 + k_1 - 1) = f(k + 2 - 1) = f(k + 1) = v$.
Cette propriété peut s'écrire sous la forme :

$$T[u][v] = \max_w T_1[u][w] \times T_2[w][v].$$

Ainsi, en notant M la multiplication matricielle de T_1 par T_2, on a $T[u][v] = \min(1, M_{u,v})$. On peut donc calculer T à partir de T_1 et T_2 en temps n^ω. Finalement, on construit T en temps $c(k_1^2+\ell_1)n^\omega + c(k_2^2+\ell_1)n^\omega + dn^\omega$ pour une certaine constante d correspondant à celle de la multiplication matricielle. On pose alors $c \geq d$. De plus, comme $k_1 + k_2 = k + 2$, on a $(k+1)^2 - (k_1^2 + k_2^2 + 1) = 2(k_1 + k_2 + k_1k_2 - 1)$. Puisque $k_1 + k_2 + k_1k_2 \geq 1$, on a $k_1^2 + k_2^2 + 1 \leq (k+1)^2$. On a alors :

$$\begin{aligned} & c(k_1^2 + \ell_1)n^\omega + c(k_2^2 + \ell_1)n^\omega + dn^\omega \\ & \leq c(k_1^2 + \ell_1 + k_2^2 + \ell_1 + 1)n^\omega \\ & = c(k_1^2 + k_2^2 + 1 + \ell)n^\omega \\ & \leq c((k+1)^2 + \ell)n^\omega. \end{aligned}$$

Ce qui conclut. □

6.5 Recherche de motif quelconque

On rappelle qu'un motif planaire extérieur peut être détecté en temps $\mathcal{O}(n^\omega)$ par divisions successives. Pour répondre à la question « Est-ce que (G, τ) contient P ? », on

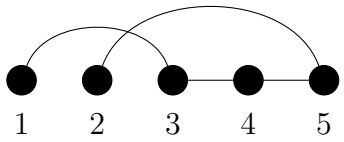
répond à la place à n^2 questions plus spécifiques. Étant donné $i \in V(G)$ et $j \in V(G)$, on se pose la question suivante : « Est-ce que (G, τ) contient P si on impose que le premier sommet de P corresponde à i et le dernier sommet de P corresponde à j ? » En faisant cela, on a certes davantage de questions, mais cela permet de diviser le motif P en motifs P_1 et P_2 plus petits. En effet, en ayant une matrice M_1 qui indique pour tout (i, k) si l'on peut trouver P_1 avec la condition « i est le plus petit sommet de P_1 et k le plus grand sommet de P_1 » et en ayant une matrice M_2 qui indique pour tout (k, j) si l'on peut trouver P_2 avec la condition « k est le plus petit sommet de P_2 et j le plus grand sommet de P_2 », on peut construire la matrice M telle que $M(i, j)$ réponde à la question « Est-ce que (G, τ) contient P si on impose que le premier sommet de P corresponde à i et le dernier sommet de P corresponde à j ? ». Pour cela, on regarde s'il existe un sommet k tel que $M_1(i, k) = M_2(k, j) = 1$. Si oui, on a trouvé un triplet (i, k, j) tel que P_1 apparaisse dans (G, τ) avec i comme premier sommet et k comme dernier sommet et tel que P_2 apparaisse dans (G, τ) avec k comme premier sommet et j comme dernier sommet. Ainsi, P apparait dans (G, τ) avec i comme premier sommet et j comme dernier sommet. Si non, alors le motif P ne peut pas être trouvé dans (G, τ) si l'on impose i comme premier sommet et j comme dernier sommet. Le but est maintenant de généraliser cette approche.

6.5.1 Réalisation partielle

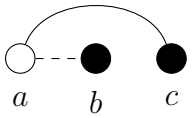
Si l'on a un motif P et un graphe ordonné (G, τ), on peut parler d'un ensemble de sommets de $S \subseteq V(G)$ qui réalise P. Il s'agit informellement de l'endroit où se trouve P dans G. Formellement, cela signifie qu'il existe une fonction $f : V(P) \to V(G)$ qui vérifie les conditions de la définition 6.9 et telle que $f(V(P)) = S$, c'est-à-dire que les sommets du motif P sont envoyés sur les sommets S de G. L'idée de la section précédente, c'est de se concentrer sur des réalisations partielles, c'est-à-dire qu'on demande à ce que certains sommets du motif P soient envoyés sur des sommets particuliers de G. Par exemple, dans le cas précédents, on choisissait deux sommets i, j de G et on voulait que le premier sommet du motif P corresponde à i et que le dernier sommet de P corresponde à j. Autrement dit, on se demandait s'il existait une réalisation de P dans G quand on impose une *réalisation partielle* qui consiste à envoyer le premier sommet de P sur i et le dernier sur j. Une autre façon de le voir, c'est de dire que l'on part d'une réalisation partielle du motif P et l'on se demande si on peut l'étendre à une réalisation complète. La difficulté est de formaliser ce concept d'une façon pratique et compréhensible. Afin d'avoir un aspect visuel, nous avons opté pour la formalisation suivante.

Définition 6.13 (Motif ancré)**.** *Un* motif ancré (P, A) *est un motif P accompagné d'un sous-ensemble de ses sommets $A \subseteq V(P)$. Soit (G, τ) un graphe ordonné et S un sous-ensemble des sommets de G de même taille que A, c'est-à-dire que $|S| = |A|$. Une* réalisation partielle en S *du motif ancré (P, A) dans un graphe ordonné (G, τ) est une fonction $f : V(P) \to V(G)$ qui réalise P dans (G, τ) (c'est-à-dire qu'elle vérifie les conditions de la définition 6.9) et telle que $f(A) = S$.*

Exemple 6.8. On considère le graphe ordonné (G, τ) et le motif ordonné (P, A) de la figure 6.20. Il y a une réalisation partielle en $\{1\}$ de (P, A) dans (G, τ) via la fonction f définie par $f(a) = 1, f(b) = 2$ et $f(c) = 3$. De même, il y a une réalisation partielle en $\{2\}$ de (P, A) dans (G, τ) via la fonction f définie par $f(a) = 2, f(b) = 3$ et $f(c) = 5$, mais aussi par la fonction f définie par $f(a) = 2, f(b) = 4$ et $f(c) = 5$. En revanche, il n'y a pas de réalisation partielle en $\{3\}$, en $\{4\}$ ou en $\{5\}$.



(a) Graphe ordonné (G, τ).



(b) Motif ancré (P, A). Les ancres sont représentées par des sommets blancs.

FIGURE 6.20 – Réalisation partielle d'un motif ancré dans un graphe ordonné.

Étant donné un motif ancré (P, A) et un graphe ordonné (G, τ), on aimerait savoir pour quels sous-ensembles de sommets $S \subseteq V(G)$ de taille $d := |A|$ le motif ancré (P, A) est une réalisation partielle en S. Il existe $\binom{n}{d}$ ensembles S possibles. Pour simplifier le stockage informatique, on va considérer tous les tuples possibles $(a_1, \ldots, a_d)$ de d sommets de G et prendre $S = \{a_1, \ldots, a_d\}$. Cela entraine de la redondance (car plusieurs tuples sont associés au même ensemble) et des ensembles inutiles (si plusieurs éléments du tuples sont égaux, l'ensemble S a alors moins de d éléments), mais cela permet de stocker les informations dans un tableau à d dimensions. De plus, cela n'entraine pas d'explosion combinatoire tant que d est petit devant n, car $\binom{n}{d} \sim n^d$ quand n tend vers l'infini. Ainsi, on définit le *tableau des réalisations partielles* du motif ancré (P, A) par :

Définition 6.14 (Tableau des réalisations partielles)**.** *Soit (G, τ) un graphe ordonné et soit (P, A) est un motif ancré. On note $n = |V(G)|$ et $d := |A|$. Le* tableau des réalisations partielles $\mathfrak{T}$ *du motif ancré (P, A) est un tenseur à d dimensions, où chaque dimension est de taille n et est étiquetée par une ancre a_i, tel que $\mathfrak{T}[a_1 : u_1, \ldots, a_d : u_d]$ vaut :*

- *1 s'il existe une réalisation partielle en $S = \{u_1, \ldots, u_d\}$ du motif ancré (P, A) dans le graphe ordonné (G, τ).*
- *0 sinon.*

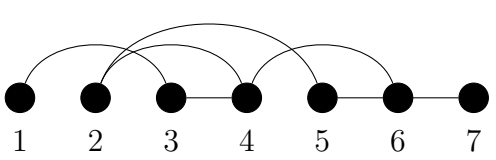
On note ici que $A = \{a_1, \ldots, a_d\}$ est un ensemble de sommets du motif P alors que $\{u_1, \ldots, u_d\}$ est un ensemble de sommets du graphe G. Le tableau des réalisations partielles permet de savoir s'il est possible de réaliser le motif P si on impose que les sommets $a_1, \ldots, a_d$ du motif P soient envoyés sur les sommets $u_1, \ldots, u_d$ du graphe G.

Exemple 6.9. On considère à nouveau le graphe ordonné (G, τ) et le motif ordonné (P, A) de la figure 6.20. On a vu dans l'exemple 6.8 que (P, A) avait une réalisation partielle en $\{1\}$ et en $\{2\}$, mais pas en en $\{3\}$, en $\{4\}$ ou en $\{5\}$. Le tableau des réalisations partielles $\mathfrak{T}$ du motif ancré (P, A) dans (G, τ) est un tableau à une dimension de taille 5 qui vaut donc $\mathfrak{T} = [1, 1, 0, 0, 0]$.

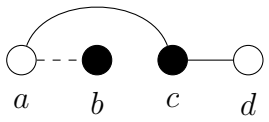
Exemple 6.10. On considère le graphe ordonné (G, τ) et le motif ordonné (P, A) de la figure 6.21. Le tableau des réalisations partielles $\mathfrak{T}$ du motif ancré (P, A) dans (G, τ) est un tableau à deux dimensions de taille 7. Il y a une réalisation partielle en $\{1, 4\}$ de (P, A) dans (G, τ) via la fonction f définie par $f(a) = 1, f(b) = 2$ et $f(c) = 3, f(d) = 4$. Il y a deux réalisations partielles en $\{2, 6\}$ de (P, A) dans (G, τ) via la fonction f définie par $f(a) = 2, f(b) = 3$ et $f(c) = 4, f(d) = 6$ et la fonction f définie par $f(a) = 2, f(b) = 3$ et $f(c) = 5, f(d) = 6$. Enfin, il y a une réalisation partielle en $\{4, 7\}$ de (P, A) dans (G, τ) via la fonction f définie par $f(a) = 4, f(b) = 5$ et $f(c) = 6, f(d) = 7$. Le tableau des

réalisations partielles $\mathfrak{T}$ vaut donc :

$$\mathfrak{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$



(a) Graphe ordonné (G, τ).



(b) Motif ancré (P, A). Les ancres sont représentés par des sommets blancs.

FIGURE 6.21 – Réalisation partielle d'un motif ancré dans un graphe ordonné.

Lemme 6.18. *Soit (G, τ) un graphe ordonné et soit (P, A) est un motif ancré constitué d'une seule arête (obligatoire ou interdite) dont l'une des deux extrémités est une ancre. Soit $d = |A|$. On peut calculer le tableau des réalisations partielle $\mathfrak{T}$ du motif ancré (P, A) en temps $\mathcal{O}(n^2 + n^d)$.*

Démonstration. On note $e = (a, b)$ l'unique arête de P. On traite le cas où e est une arête obligatoire, le cas où c'est une arête interdite se fait de la même manière. Sans perte de généralité, le sommet a est le sommet ancré. On note d_L le nombre d'ancres à gauche de a (exclus), on note d_M le nombre d'ancres entre a et b (exclus) et on note d_R le nombre d'ancres à droite de b (exclus). On calcule matrice d'adjacence M du graphe ordonné (G, τ) en temps quadratique. Ensuite, on supprime les arêtes de la matrice d'adjacence qui n'ont pas d_L sommets à gauche de l'arête, d_M entre les extrémités et d_R à droite. En effet, ces arêtes-là ne peuvent pas être utilisées pour réaliser P. Dans le cas où e est une non-arête, au lieu de supprimer des arêtes, on supprime des non-arêtes, c'est-à-dire qu'on ajoute des arêtes à la matrice d'adjacence. On note M' cette nouvelle matrice d'adjacence. Enfin, pour chaque sommet i de G, on vérifie s'il est l'extrémité gauche d'une arête de la nouvelle matrice d'adjacence M' de G. Pour un sommet, cela se fait en temps linéaire en parcourant la ligne correspondante dans M'. Cela se fait donc en temps quadratique pour tous les sommets.

Maintenant, pour chacune des n^d entrées du tableau des réalisations partielle $\mathfrak{T}$, on vérifie si cette entrée doit être un 0 ou un 1. On note $u_1, \ldots, u_d$ l'entrée que l'on vérifie et on pose $S = \{u_1, \ldots, u_d\}$.

Dans le cas où b est une ancre de (P, A), on note u_i le sommet de S qui doit correspondre à a et u_j le sommet de S qui doit correspondre à b. Cela signifie que $i = d_L + 1$ et $j = d_L + d_M + 2$. Si $M'[u_i, u_j]$ vaut 1, il y a une arête entre u_i et u_j et celle-ci a bien d_L sommets à gauche (auxquels on peut associer les ancres $a_1, \ldots, a_{d_L}$), d_M sommets entre ces extrémités (auxquels on peut associer les ancres $a_{d_L+2}, \ldots, a_{d_L+d_M+1}$) et d_R sommets à droite (auxquels on peut associer les ancres $a_{d_L+d_M+3}, \ldots, a_d$). On pose alors $\mathfrak{T}[a_1 : u_1, \ldots, a_d : u_d] = 1$. Sinon, il n'y a pas d'arête vérifiant ces propriétés et on pose $\mathfrak{T}[a_1 : u_1, \ldots, a_d : u_d] = 0$.

Dans le cas où b n'est pas une ancre de (P, A), on note juste u_i le sommet de S qui doit correspondre à a, c'est-à-dire tel que $i = d_L + 1$. Si u_i est l'extrémité gauche d'une arête de M', on a trouvé une réalisation partielle de (P, A) en S et on pose $\mathfrak{T}[a_1 : u_1, \ldots, a_d : u_d] = 1$. Sinon, on pose $\mathfrak{T}[a_1 : u_1, \ldots, a_d : u_d] = 0$. □

6.5.2 Fusion de deux motifs

On veut continuer de généraliser ce qui se faisait dans le cas des motifs planaires extérieurs. L'étape suivante est de fusionner deux petits motifs pour en obtenir un plus grand et de faire de même pour les matrices qui représentent les endroits où se trouvent ces motifs dans le graphe ordonné. L'idée pour les motifs planaires extérieurs est que si on a deux motifs P_1 et P_2, on peut les fusionner en identifiant le dernier sommet de P_1 avec le premier de P_2. Maintenant, afin de généraliser, il faut pouvoir fusionner deux motifs sur d'autres sommets, pas forcément aux extrémités.

Cependant, cela peut entrainer des ambigüités. Par exemple, si on veut fusionner deux motifs en identifiant leur premier sommet, dans quel ordre met-on les autres sommets des deux motifs ? On remarque qu'il y a donc des fusions qui peuvent être ambigües. On désire formaliser cela.

Définition 6.15 (Fusion de motifs)**.** *Soit P_1 et P_2 deux motifs. Un motif P est une* fusion *des motifs P_1 et P_2 lorsqu'il vérifie les conditions suivantes :*

- $V(P) = V(P_1) \cup V(P_2)$,
- $M(P) = M(P_1) \cup M(P_2)$,
- $F(P) = F(P_1) \cup F(P_2)$,
- $\tau(P_1) \prec \tau(P)$ *et* $\tau(P_2) \prec \tau(P)$ *(cf. définition 6.4).*

On rappelle qu'un motif P doit vérifier $M(P) \cap F(P) = \emptyset$. Pour que P puisse être la fusion de P_1 et P_2, il faut donc qu'il n'y ait pas d'arêtes qui se contredisent entre P_1 et P_2, c'est-à-dire qu'il faut $M(P_1) \cap F(P_2) = \emptyset$ et $F(P_1) \cap M(P_2) = \emptyset$.

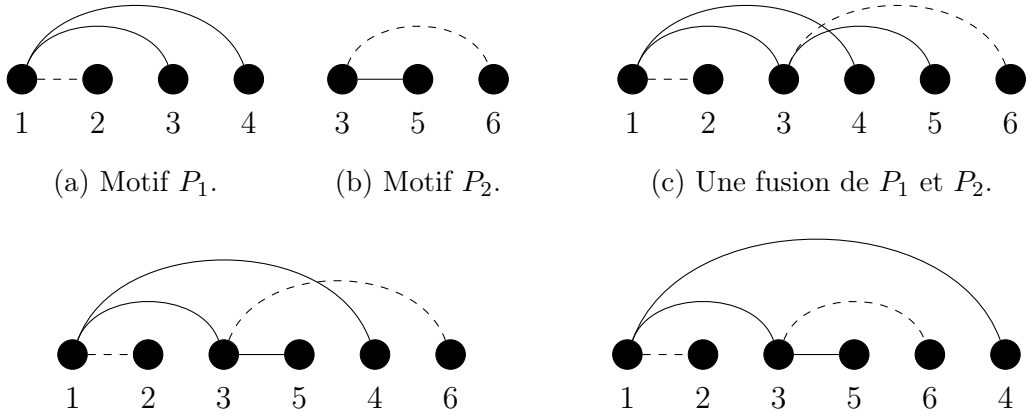
Exemple 6.11. On considère un motif P_1 tel que $V(P_1) = \{1, 2, 3, 4\}$ et tel que $\tau(P_1) = [1, 2, 3, 4]$ ainsi qu'un motif P_2 tel que $V(P_2) = \{3, 5, 6\}$ et tel que $\tau(P_2) = [3, 5, 6]$. Pour que P soit une fusion de P_1 et P_2, il faut que $\tau(P_1) \prec \tau(P)$ et $\tau(P_2) \prec \tau(P)$, c'est-à-dire que l'ordre $\tau(P)$ contienne $\tau(P_1) = [1, 2, 3, 4]$ et $\tau(P_2) = [3, 5, 6]$ comme sous-ordre. Il y a trois ordre $\tau(P)$ qui vérifient ces propriétés, à savoir les ordres $[1, 2, 3, 4, 5, 6]$, $[1, 2, 3, 5, 4, 6]$ et $[1, 2, 3, 5, 6, 4]$. On en déduit alors trois fusions possibles pour P_1 et P_2, une pour chacun de ces ordres. Des exemples de motifs P_1 et P_2 ainsi que de fusion possibles sont représentés dans la figure 6.22.

Une première conséquence de la condition $\tau(P_1) \prec \tau(P)$ et $\tau(P_2) \prec \tau(P)$ est que l'ordre des sommets en commun doit être le même dans P_1 et dans P_2.

Lemme 6.19. *Soit P_1 et P_2 deux motifs et soit P un motif qui est une fusion des motifs P_1 et P_2. Soit $I = V(P_1) \cap V(P_2)$. Alors $\tau(P_1)|_I = \tau(P_2)|_I$.*

Démonstration. Comme il existe une fusion P de P_1 et P_2, on a un ordre $\tau(P)$ des sommets de P qui vérifie $\tau(P_1) \prec \tau(P)$ et $\tau(P_2) \prec \tau(P)$. Par définition, on a $\tau(P_1) = \tau(P)|_{V(P_1)}$ et $\tau(P_2) = \tau(P)|_{V(P_2)}$. Ainsi :

$$\tau(P_1)|_I = \big(\tau(P)|_{V(P_1)}\big)\,|_I = \tau(P)|_{V(P_1)\cap V(P_2)} = \big(\tau(P)|_{V(P_2)}\big)\,|_I = \tau(P_2)|_I. \qquad \square$$



(a) Motif P_1. (b) Motif P_2. (c) Une fusion de P_1 et P_2.

(d) Une deuxième fusion de P_1 et P_2. (e) Une troisième fusion de P_1 et P_2.

FIGURE 6.22 – Exemple de fusion de motifs.

On remarque qu'il peut exister plusieurs fusions possibles de deux motifs, tout comme il peut ne pas en exister. On a la terminologie suivante :

Définition 6.16 (Fusion uniquement déterminée, fusion impossible, fusion ambigüe)**.** *Une fusion des motifs P_1 et P_2 est* uniquement déterminée *lorsqu'il existe un seul motif P qui est la fusion des motifs P_1 et P_2. S'il n'existe aucun motif, la fusion est* impossible*. S'il en existe plusieurs, elle est* ambigüe*.*

Les fusions possibles pour une paire de motifs ne dépendent que de l'ordre des sommets de chaque motif.

Lemme 6.20. *Soit P_1, P_2, P'_1 et P'_2 quatre motifs tels qu'il existe au moins une fusion de P_1 et P_2 et au moins une fusion de P'_1 et P'_2, et tels que pour $i = 1, 2$:*

- *les sommets de P_i et P'_i sont les mêmes : $V(P_i) = V(P'_i)$,*
- *l'ordre des sommets est le même entre P_i et P'_i : $\tau(P_i) = \tau(P'_i)$,*

Alors la fusion entre P_1 et P_2 est uniquement déterminée si et seulement si la fusion entre P'_1 et P'_2 l'est aussi.

Démonstration. On montre la contraposée : on suppose qu'il y a deux fusions différentes P_a et P_b de P_1 et P_2 et on montre qu'il existe deux fusions différentes P'_a et P'_b de P'_1 et P'_2. Cela signifie qu'il y a deux ordres distincts $\tau_a = \tau(P_a)$ et $\tau_b = \tau(P_b)$ qui vérifient tous les deux $\tau(P_1) \prec \tau_i$ et $\tau(P_2) \prec \tau_i$ pour $i = a, b$. On considère alors le motif P'_a défini par $V(P'_a) = V(P'_1) \cup V(P'_2)$, $F(P'_a) = F(P'_1) \cup F(P'_2)$, $M(P'_a) = M(P'_1) \cup M(P'_2)$, $\tau(P'_a) = \tau_a$. On vérifie que l'on a bien $F(P'_a) \cap M(P'_a) = \emptyset$. C'est le cas car il existe une fusion de P'_1 et P'_2 par hypothèse. On vérifie aussi que $\tau(P'_1) \prec \tau_a$ et $\tau(P'_2) \prec \tau_a$. C'est le cas puisque $\tau(P'_1) = \tau(P_1) \prec \tau_a$ et $\tau(P'_2) = \tau(P_2) \prec \tau_a$. On définit de la même manière le motif P'_b, avec pour seule différence avec le motif P'_a l'ordre $\tau(P'_b)$ qui vaut τ_b. On a donc deux motifs différents P'_a et P'_b de P'_1 et P'_2 qui diffèrent par leur ordre des sommets. □

On s'intéresse maintenant à la décomposition d'un motif :

Définition 6.17 (Décomposition d'un motif $P \to P_1, P_2$)**.** *Un motif P* se décompose *en P_1 et P_2 si la fusion de P_1 et P_2 est uniquement déterminée et si elle donne P. On note $P \to P_1, P_2$.*

On considère le problème suivant : étant donné un motif P, comment le décomposer en P_1 et P_2 ? Le plus simple est de prendre P_1 et P_2 tels que $V(P_1) = V(P_2) = V(P)$ et $\tau(P_1) = \tau(P_2) = \tau(P)$, et de partager les arêtes obligatoire $M(P)$ et les arêtes interdites $F(P)$ entre P_1 et P_2. Cependant, on aura besoin par la suite de minimiser la quantité $|V(P_1) \cap V(P_2)|$. Avec cet exemple naïf, cette quantité est maximale. Il faut donc faire mieux. Par exemple, dans le cas des motifs planaires extérieurs, on avait $|V(P_1) \cap V(P_2)| = 1$, vu que le seul sommet en commun était le dernier sommet de P_1 qui était aussi le premier sommet de P_2. Il s'agit du meilleur cas possible, puisque une fusion telle que $|V(P_1) \cap V(P_2)| = 0$ est forcément ambigüe. Bien que l'on est libre de choisir le partage des arêtes entre P_1 et P_2, on ne peut pas choisir les sommets de chaque motif P_1, P_2, car il faut absolument que chaque extrémité des arêtes choisies pour P_i apparaisse dans P_i, ce qui fait augmenter le nombre de sommets dans chaque P_i et donc la proportion de sommets dans $V(P_1) \cap V(P_2)$.

Décomposition de motifs

On a le lemme suivant :

Lemme 6.21. *Soit P un motif qui se décompose en P_1 et P_2 ($P \to P_1, P_2$). Soit τ un ordre de $V(P_1) \cup V(P_2)$ tels que $\tau(P_1) \prec \tau$ et $\tau(P_2) \prec \tau$. Alors $\tau = \tau(P)$.*

Démonstration. Si τ était un ordre différent de $\tau(P)$, alors le motif P' défini par :

- $V(P') = V(P_1) \cup V(P_2)$,
- $M(P') = M(P_1) \cup M(P_2)$,
- $F(P') = F(P_1) \cup F(P_2)$,
- $\tau(P') = \tau$

serait une fusion de P_1 et P_2 différente de P, puisque $\tau(P') \neq \tau(P)$. □

Soit P_1 et P_2 deux motifs dont P est une fusion. Soit A l'ensemble des sommets que P_1 et P_2 ont en commun : $A = V(P_1) \cap V(P_2)$. Soit $d = |A|$ le nombre de sommets en commun. On note $A = \{a_1, \ldots, a_d\}$ avec $a_1 <_{\tau(P)} \ldots <_{\tau(P)} a_d$. Pour $i = 1, 2$ et $j = 1, \ldots, d-1$, on note V_i^j l'ensemble des sommets de P_i qui sont strictement entre a_j et a_{j+1} suivant l'ordre τP_i. De même, on note V_i^0 l'ensemble des sommets de P_i qui sont inférieurs strictement à a_1 et V_i^d l'ensemble des sommets de P_i qui sont supérieurs strictement à a_d (suivant l'ordre τP_i). On peut caractériser l'unicité de la fusion en fonction des ensembles V_1^j et V_2^j pour tout j.

Lemme 6.22. *Soit P un motif qui se décompose en P_1 et P_2 ($P \to P_1, P_2$). Alors pour tout $j = 0, \ldots, d$, on a $V_1^j = \emptyset$ ou $V_2^j = \emptyset$.*

Démonstration. On suppose par l'absurde qu'il existe un j tel que $V_1^j \neq \emptyset$ et $V_2^j \neq \emptyset$. Dans un premier temps, on considère que j est compris entre 1 et $d-1$ et on note V^j l'ensemble des sommets de P qui sont entre a_j et a_{j+1}, c'est-à-dire que $V^j = V_1^j \cup V_2^j$. Par définition des V_i^j, cette union est disjointe, c'est-à-dire que les sommets appartiennent soit uniquement à P_1, soit uniquement à P_2. Par hypothèse, on sait qu'il y a au moins un sommet x_1 de P_1 et au moins un sommet x_2 de P_2. On considère les deux ordres suivants :

- τ_1 qui est le même que $\tau(P)$, excepté dans V^j où on met d'abord tous les sommets de P_1 puis ensuite tous les sommets de P_2.

— τ_2 qui est le même que $\tau(P)$, excepté dans V^j où on met d'abord tous les sommets de P_2 puis ensuite tous les sommets de P_1.

Ces deux ordres sont strictement différents car dans le premier, $x_1 < x_2$ et dans le second, $x_2 < x_1$. Ces deux ordres ont bien $\tau(P_1)$ et $\tau(P_2)$ comme sous-ordre. Ainsi, on peut construire P'_1 et P'_2 deux motifs différents qui sont des fusions de P_1 et P_2 et donc l'ordre des sommets est respectivement $\tau(P_1)$ et $\tau(P_2)$. Cela contredit le fait que la fusion de P_1 et P_2 est uniquement déterminée. On traite le cas où $j = 0$ et celui où $j = d$ de la même façon, en considérant $V^j = V_1^j \cup V_2^j$. □

6.5.3 Arbre de fusion

Le but est maintenant de partir d'un motif P et de le décomposer récursivement jusqu'à obtenir uniquement des motifs constitués de seulement deux sommets (car il est impossible de décomposer un motif de deux sommets en motifs strictement plus petits). Cela donne alors un arbre de décomposition (ou arbre de fusion) de P. Formellement, cella donne la définition suivante :

Définition 6.18 (Arbre de fusion)**.** *Soit P un motif. Un* arbre de fusion *de P est un arbre T tel que :*

— *chaque nœud de T est étiqueté par un motif,*

— *la racine de T est étiquetée par le motif P,*

— *les feuilles de T sont étiquetées par des motifs ayant une arête,*

— *chaque nœud interne u de T a exactement deux fils u_1, u_2 tels que le motif P qui étiquette u se décompose en P_1 qui étiquette u_1 et P_2 qui étiquette P_2.*

La figure 6.23 représente un exemple d'arbre de fusion d'un motif. C'est un arbre de fusion car les motifs qui étiquettent les feuilles contiennent bien exactement une arête (même si elles contiennent aussi des sommets en plus des extrémités de l'arête). On vérifie également que les fusions sont uniquement déterminées. En notant P_i le motif qui étiquette le nœud u_i, le motif P_2 se décompose bien en P_3 et P_4. Cela est dû à la présence du sommet b dans P_3 et du sommet c dans P_4. Grâce à eux, le seul ordre qui étende à la fois $[a, b, c]$ et $[b, c, d]$ est $[a, b, c, d]$. Sans eux, on aurait eu les ordres $[a, c]$ et $[b, d]$, qui sont étendus par plusieurs autres ordres tels que $[a, c, b, d]$ ou $[b, d, a, c]$.

Cet arbre permet de répondre à la question « Est-ce que (G, τ) contient P ? » de façon récursive, en se demandant à la place pour chacun des deux fils P_1, P_2 de P dans l'arbre « Est-ce que (G, τ) contient P_i ? », puis en concluant à partir de la réponse à ces deux sous-questions. Comme la fusion est uniquement déterminée, le fait de trouver P_1 et P_2 dans (G, τ) devrait permettre de déduire que P est aussi dans (G, τ). Cependant, il est possible que P_1 et P_2 se trouvent à des endroits totalement différents de (G, τ), de sorte que P ne soit pas dans (G, τ) malgré la présence de P_1 et P_2, comme montré dans la figure 6.24

Remarque 6.4. Un motif P a toujours au moins un arbre de fusion. En effet, un motif P peut systématiquement se décomposer en deux motifs P_1 et P_2. Pour ce faire, on pose $V(P_1) = V(P_2) = V(P)$ et on partitionne les arêtes de P arbitrairement entre P_1 et P_2. Il est clair que la fusion de P_1 et P_2 est uniquement déterminée. En recommençant récursivement, on obtient un arbre de fusion de P.

Ainsi, il faut avoir davantage de précisions sur la position des réalisations de P_1 et P_2 dans (G, τ).

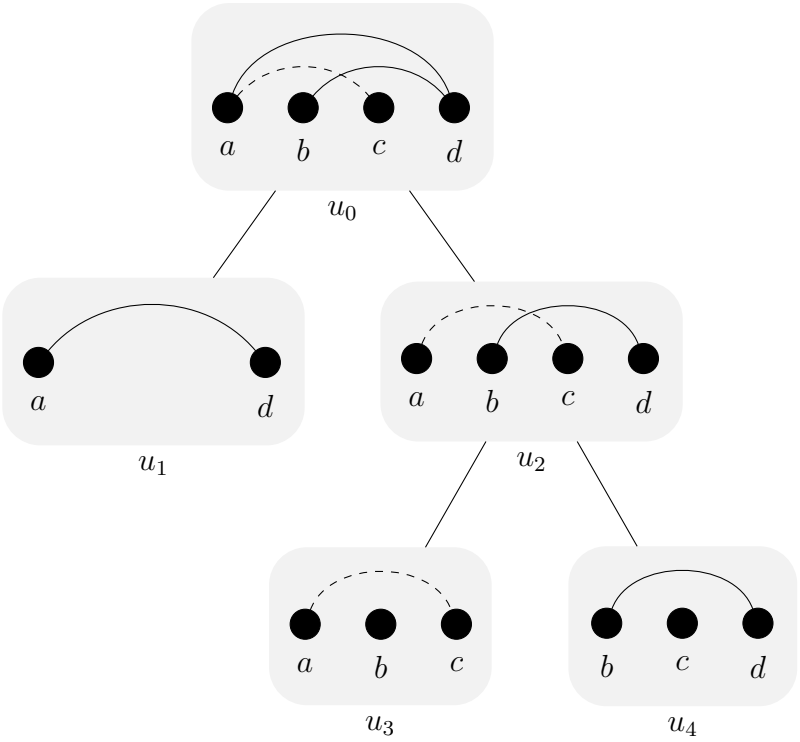


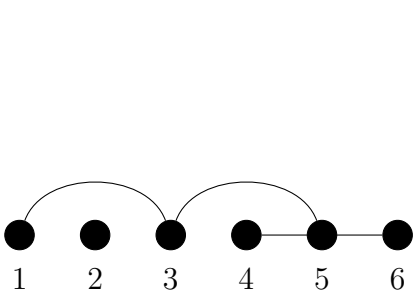
FIGURE 6.23 – Exemple d'arbre de fusion T. Chaque rectangle gris représente un nœud de l'arbre T. Le motif dessiné à l'intérieur d'un rectangle gris représente le motif qui étiquette le nœud représenté par le rectangle gris. La racine de T est le nœud u_0. Les feuilles de T sont les nœuds u_1, u_3 et u_4. Les deux fils d'un nœud u_i sont dessinés en dessous et sont reliés à u par un trait noir.

Lemme 6.23. *Soit P un motif qui se décompose en P_1 et P_2, c'est-à-dire que $P \to P_1, P_2$ (cf. définition 6.17). Soit $A = V(P_1) \cap V(P_2)$ l'ensemble des sommets communs aux deux motifs et $d = |A|$ le nombre de sommets en commun. Soit $\mathfrak{T}_1$ le tableau des réalisations partielles de (P_1, A) et $\mathfrak{T}_2$ le tableau des réalisations partielles de (P_2, A) (cf. définition 6.14). Alors (G, τ) contient P si, et seulement si, il existe d sommets de (G, τ) (que l'on nomme $u_1, \ldots, u_d$) tels que $\mathfrak{T}_1[u_1, \ldots, u_d] = \mathfrak{T}_2[u_1, \ldots, u_d] = 1$.*

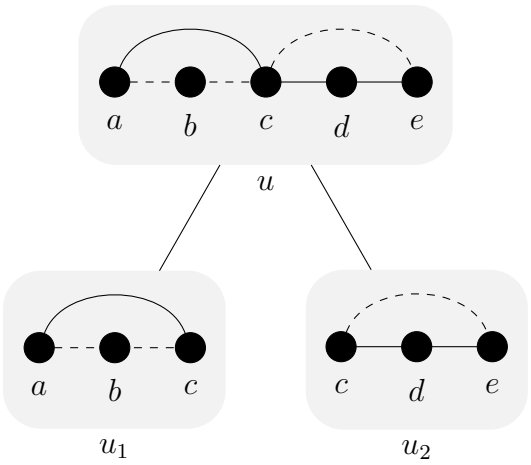
La preuve du sens réciproque de ce lemme est assez longue car elle s'attelle à être rigoureuse vis à vis des définitions des différents éléments de ce lemme. Cependant, l'idée est en réalité assez simple : si le graphe ordonné (G, τ) contient les motifs P_1 et P_2 de sorte à ce que les sommets en commun à ces deux motifs se trouvent aux mêmes endroits dans (G, τ), alors on peut trouver une fusion de P_1 et P_2 dans (G, τ). Or, comme P_1 et P_2 ont une unique fusion possible qui est P, on déduit que (G, τ) contient P.

Démonstration. On procède par double implication :

($\Leftarrow$) On suppose qu'il existe d sommets $u_1, \ldots, u_d$ de (G, τ) tels que $\mathfrak{T}_1[u_1, \ldots, u_d] = \mathfrak{T}_2[u_1, \ldots, u_d] = 1$. On pose $S = \{u_1, \ldots, u_d\}$ et $A = \{a_1, \ldots, a_d\}$, avec $u_1 <_\tau \ldots <_\tau u_d$ et $a_1 <_{\tau(P_1)} \ldots <_{\tau(P_1)} a_d$. Autrement dit, la numérotation des sommets de S suit l'ordre τ et la numérotation des sommets de A suit l'ordre τ_1. Comme $P \to P_1, P_2$, il existe une fusion de P_1 et P_2 en P, ce qui signifie d'après le lemme 6.19 que l'ordre des sommets de A est le même dans P_1 et P_2. On a donc aussi $a_1 <_{\tau(P_2)} \ldots <_{\tau(P_2)} a_d$. Par définition du tableau des réalisations partielles, il existe une réalisation partielle en S du motif ancré (P_1, A) dans le graphe ordonné (G, τ) et



(a) Graphe orienté (G, τ).



(b) Motif P qui se décompose en P_1 et P_2.

FIGURE 6.24 – Exemple de recherche improductive d'un motif P dans un graphe ordonné (G, τ). Il y a une réalisation du motif P_1 dans (G, τ) via la fonction f définie par $f(a) = 1$, $f(b) = 2$ et $f(c) = 3$. De plus, il y a une réalisation du motif P_2 dans (G, τ) via la fonction f définie par $f(c) = 4$, $f(d) = 5$ et $f(e) = 6$. Cependant, il n'y a pas de réalisation du motif P dans (G, τ). Pour pouvoir en déduire une réalisation de P, il aurait fallu que la valeur de $f(c)$ soit la même dans les deux réalisations de P_1 et P_2.

une réalisation partielle en S du motif ancré (P_2, A) dans le graphe ordonné (G, τ). Par définition d'une réalisation partielle (cf. définition 6.13), il existe une fonction f_1 qui réalise P_1 dans G, τ (cf. définition 6.9) telle que $f_1(A) = S$. De même, il existe une fonction f_2 qui réalise P_2 dans G, τ telle que $f_2(A) = S$. Comme $u_1 <_\tau \ldots <_\tau u_d$, $a_1 <_{\tau(P_1)} \ldots <_{\tau(P_1)} a_d$, $f_1(A) = S$ et f_1 conserve l'ordre, on a $f_1(a_i) = u_i$ pour tout $i \in [d]$. De même, on a $f_2(a_i) = u_i$ pour tout $i \in [d]$. Ainsi, pour tous les sommets dans $x \in A$, on a $f_1(x) = f_2(x)$. On construit alors la fonction $f : V(P) \to V(G)$ par :

$$f(x) = \begin{cases} f_1(x) & \text{si } x \in V(P_1) \\ f_2(x) & \text{si } x \in V(P_2) \end{cases}$$

Cette fonction est bien définie puisque les sommets en commun à P_1 et P_2 sont ceux dans $A = V(P_1) \cap V(P_2)$, pour lesquels ont a prouvé que les images par f_1 et f_2 étaient les mêmes.

Il reste à montrer que f réalise P dans (G, τ). Avant tout, on montre que f est injective. Soit $x, y \in V(P)$ tels que $f(x) = f(y)$. Si x et y sont des sommets du même P_i alors $f(x) = f_i(x)$ et $f(y) = f_i(y)$, d'où $f_i(x) = f_i(y)$ et $x = y$ par injectivité de f_i (cf. lemme 6.1). Sinon, sans perte de généralité, $x \in V(P_1) \setminus A$ et $y \in V(P_2) \setminus A$. Supposons que $x \neq y$. Sans perte de généralité, supposons que $x <_\tau y$. D'après le lemme 6.22 et en en reprenant les notations, on sait qu'il ne peut pas y avoir un j tel $x \in V_1^j$ et $y \in V_2^j$ car dans ce cas, la fusion de P_1 et P_2 n'est pas unique. Ainsi, $x \in V_1^{j_1}$ et $y \in V_2^{j_2}$ avec $j_1 < j_2$. Dès lors, il existe un sommet $a \in V(P)$ entre x et y : $x \leq_{\tau(P)} a \leq_{\tau(P)} y$. Comme x et a sont dans $V(P_1)$ et que l'ordre $\tau(P_1)$ est un sous-ordre de $\tau(P)$ par définition d'une fusion, on a $x \leq_{\tau(P_1)} a$. Puisque f_1 conserve l'ordre $\tau(P_1)$, on a $f_1(x) \leq_\tau f_1(a)$. De la même manière, on déduit que $f_2(a) \leq_\tau f_2(y)$. Par définition de f, ces égalités donnent $f(x) \leq_\tau f(a) \leq_\tau f(y)$. Or, puisque $f(x) = f(y)$, on obtient une contradiction. Ainsi, $x = y$ et f est injective.

On vérifie les différents points de la définition 6.9 :

— On veut montrer que f conserve l'ordre : f étant injective, on peut considérer pour tout sommet v de G le sommets $f^{-1}(v)$ de P. On construit alors l'ordre τ' en prenant l'ordre τ restreint aux sommets de $f(V(P_1) \cup V(P_2))$ et en appliquant f^{-1} à chacun de ces sommets. Cela signifie que si deux sommets de la forme $f(x)$ et $f(y)$ (qui sont donc des sommets de G) vérifient $f(x) <_\tau f(y)$, alors on a $x <_{\tau'} y$, et réciproquement.

Par construction, τ' est un ordre de $V(P_1) \cup V(P_2)$ qui a $\tau(P_1)$ et $\tau(P_2)$ comme sous-ordre. En effet, soit x et y deux sommets de P_1 tels que $x <_{\tau'} y$. On montre que $x <_{\tau(P_1)} y$: on a donc $f(x) <_\tau f(y)$ puis $f_1(x) <_\tau f_1(y)$ par définition de f puis $x <_{\tau(P_1)} y$ d'après le lemme 6.2. Ainsi, $\tau(P_1)$ est un sous-ordre de τ'. De même, $\tau(P_2)$ est un sous-ordre de τ'. D'après le lemme 6.5.2, $\tau' = \tau(P)$. Maintenant, on considère $x, y \in V(P)$ tels que $x <_{\tau(P)} y$ et on montre que $f(x) <_\tau f(y)$. Puisque $\tau' = \tau(P)$, on a $x <_{\tau'} y$. Puis, par définition de τ', on a $f(x) <_\tau f(y)$.

— La fonction f respecte les arêtes obligatoires : Soit $(x, y) \in M(P)$. Par définition d'une fusion, on a $M(P) = M(P_1) \cup M(P_2)$ et donc $x, y \in M(P_i)$ pour un certain $i \in \{1, 2\}$. Puisque f_i réalise P_i dans (G, τ), on a $(f_i(x), f_i(y)) = (f(x), f(y)) \in E(G)$.

— De la même manière, f respecte les arêtes interdites.

($\Rightarrow$) Si (G, τ) contient P alors il existe une fonction $f : V(P) \to V(G)$ qui vérifie les points de la définition 6.9. On pose $S = \{u_1, \ldots, u_d\}$ les d sommets obtenus en prenant l'image des d sommets de A par f (on sait qu'on en a bien d distincts puisque f est injective). La fonction f restreinte à $V(P_1)$ est une réalisation partielle en S de (P_1, A) (cf. définition 6.13) et donc $\mathfrak{T}_1[u_1, \ldots, u_d] = 1$. De même, $\mathfrak{T}_2[u_1, \ldots, u_d] = 1$.

□

Cette propriété permet de faire une étape de récurrence : pour répondre à la question « Est-ce que (G, τ) contient P ? », on répond aux questions « Quel est le tableau des réalisations partielles de (P_1, A) ? » et « Quel est le tableau des réalisations partielles de (P_2, A) ? ». Pour pouvoir continuer la récurrence, il faut que la question soit la même avant et après l'étape de récurrence. Cela motive donc à se poser maintenant la question « Quel est le tableau des réalisations partielles de (P, A) ? » pour chaque P de l'arbre de fusion, avec un ensemble A d'ancres à déterminer pour chaque P. La détermination de A est purement récursive : plus on a d'ancres à calculer, plus les fils devront apporter d'information pour les calculer et donc plus les fils auront d'ancres à calculer. Commençons par la racine. Dans ce cas, on n'a besoin d'aucune ancre. Pour les fils de la racine, comme dans le lemme 6.23, les ancres sont les sommets en commun à ces deux fils. Cependant, comme ces sommets sont ancrés, on se rendra compte qu'il faut qu'ils soient ancrés dans tous les nœuds qui descendent de ces fils. On décide de définir l'ensemble A_i de chaque motif P_i récursivement de la façon suivante :

Définition 6.19 (Ancres associées à un arbre de fusion)**.** *Soit u un nœud de l'arbre T et soit u_1 et u_2 les deux fils de u. Soit P le motif qui étiquette u et soit A les ancres associées à P. Soit P_1 et P_2 les motifs qui étiquettent u_1 et u_2. Soit $I = V(P_1) \cap V(P_2)$. On définit A_1 et A_2 les ancres associées à P_1 et P_2 par $A_1 = I \cup (A \cap V(P_1))$ et $A_2 = I \cup (A \cap V(P_2))$. Pour que la définition récursive soit complète, le cas de base concerne la racine u de l'arbre T, dont le motif qui l'étiquette doit avoir pour ensemble d'ancres $A = \emptyset$.*

Autrement dit, les ancres de A_1 sont l'union des sommets en commun à P_1 et P_2 (de façon à pouvoir appliquer le lemme 6.23) et des sommets qui sont des ancres dans le père de P_1 (c'est-à-dire les ancres de A, mais dont on a gardé que les sommets qui sont dans P_1). Ces derniers sommets servent à garantir l'aspect récursif de l'arbre de fusion. Notons que cette union n'est pas forcément disjointe.

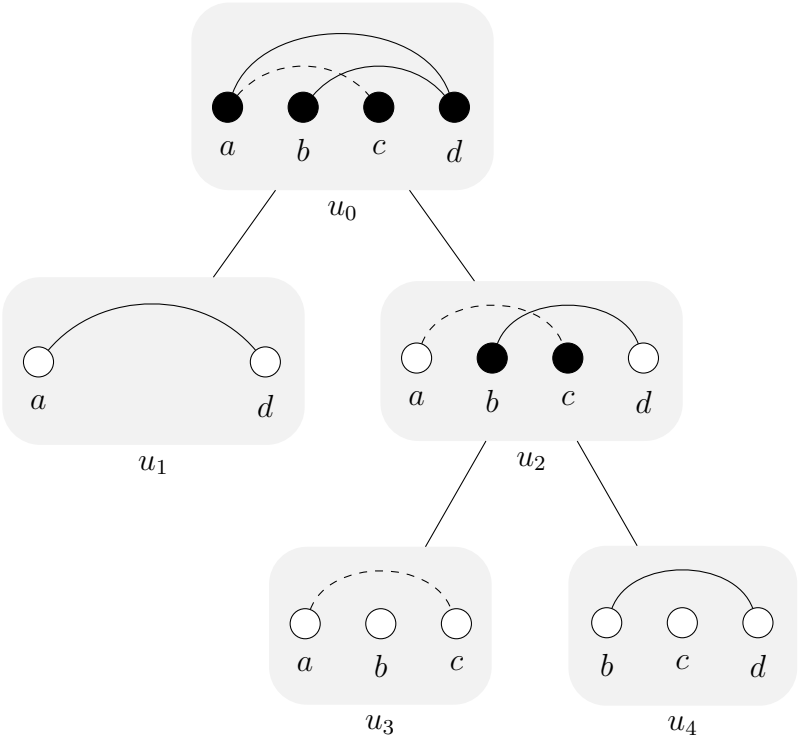


FIGURE 6.25 – Exemple d'arbre de fusion T avec les ancres représentés par des sommets blancs.

Exemple 6.12. Reprenons l'exemple de l'arbre de fusion de la figure 6.23 et appliquons la définition 6.19 afin de calculer les ancres de chacun des motifs de cet arbre. On obtient alors la figure 6.25. En notant P_i le motif qui étiquette u_i sur la figure 6.25 et A_i l'ensemble des ancres associé à P_i, on obtient :

— $A_0 = \emptyset$ car les ancres du motif qui étiquette la racine est toujours l'ensemble vide,

— $A_1 = A_2 = \{a, d\}$ car les motifs qui étiquettent les nœuds fils de u_0 ont pour ensemble d'ancres les sommets communs à P_1 et P_2 (c'est-à-dire a et d) plus les sommets qui sont ancres dans P_0 (à savoir aucun sommet),

— $A_3 = \{a, b, c\}$ car les motifs qui étiquettent les nœuds fils de u_2 ont pour ensemble d'ancres les sommets communs à P_3 et P_4 (c'est-à-dire b et c) plus les sommets qui sont ancres dans P_2 (c'est-à-dire a et d, mais on ne garde que les sommets qui sont dans P_3),

— de même, $A_4 = \{b, c, d\}$.

Un arbre de fusion vérifie certaines propriétés.

Tout d'abord, si on s'intéresse à un sommet i de P et aux nœuds de T dans lequel i est une ancre, on remarque que cela forme un ou deux sous-arbres de T et que, dans le cas où ces sous-arbres sont au nombre de deux, leur racine sont frères.

Lemme 6.24. *Soit P un motif et T un arbre de fusion de P. Soit a un sommet de P. Soit U l'ensemble des nœuds u de T dont l'étiquette possède a comme ancre. Soit $U_{\max}$ l'ensemble des nœuds u de T dont l'étiquette possède a comme ancre et tel que le père de u n'ait pas a comme ancre. Alors U est une union d'au plus deux sous-arbres de T et tous les nœuds de $U_{\max}$ sont frères.*

Démonstration. Déjà, on sait que si a est une ancre dans un motif P_i associé au nœud u_i, alors a est aussi une ancre des fils de u_i, si tant est que a appartiennent à l'ensemble des sommets des fils de P_i. Ainsi, par récurrence, si on considère un sous-arbre T' de T dont le motif de la racine de ce sous-arbre a a comme ancre, alors pour chaque motif de T', soit a est une ancre de ce motif, soit a n'est pas un sommet de ce motif. Or, dès que a n'est pas un sommet d'un motif P_i, il ne peut pas être sommet d'un fils de P_i, puisque les fils d'un motif P_i doivent former une fusion donnant P_i. Ainsi, $U \cap T'$ est connexe et forme donc un sous-arbre de T.

Pour montrer que U est l'union de deux sous-arbres, il suffit de montrer que $U_{\max}$ est de taille au plus 2. Il suffit pour cela de montrer que les nœuds de $U_{\max}$ sont frères, ce qui permet de conclure vu que T est un arbre binaire.

Soit u_1 et u_2 deux nœuds de $U_{\max}$. Soit u leur plus petit ascendant commun. Le sommet a est un sommet du motif étiqueté à u puisque a est un sommet d'un descendant de u. Ainsi, a est sommet des deux fils de u. Par définition des ancres, a est donc une ancre dans les deux fils de u. Puisque u_1 et u_2 sont dans $U_{\max}$, cela signifie que les fils de u sont u_1 et u_2 (dans le cas contraire, cela contredirait le fait que u_1 ou u_2 soit dans $u_{\max}$). □

Autrement dit, l'ensemble des nœuds de l'arbre de fusion dont le sommet a est ancre du motif étiqueté forme soit un sous-arbre, soit un sous-arbre dont on a enlevé la racine (ce qui fait donc deux sous-arbres dont les racines sont sœurs). On peut aussi s'intéresser à un sommet d'un motif qui n'est pas une ancre pour ce motif et voir où ce sommet peut apparaitre dans l'arbre de fusion et où il peut être une ancre.

Corollaire 6.25. *Soit T un arbre de fusion, u un nœud de T, P le motif qui étiquette u et soit b un sommet de P qui n'est pas une ancre. Alors :*

- *pour les motifs P' qui étiquettent un nœud ascendant de u, b est un sommet de P' mais il ne peut pas y être une ancre,*
- *pour les motifs P' qui étiquettent un nœud ni ascendant ni descendant de u, b ne peut pas être un sommet de P', et a fortiori, il ne peut pas y être une ancre.*

Démonstration. Avant tout, puisque b est un sommet de P, b est aussi un sommet de tous les ascendants de u. En effet, la fusion de motifs produit un motif ayant pour sommets l'union des sommets des motifs fusionnés et les nœuds ascendants de u dans T sont obtenus par fusions successives à partir de P. Le sommet b est donc un sommet dans tous les ascendants de u. On montre maintenant les deux points du lemme :

- Le sommet b est un sommet de P' via la remarque précédente. Cependant, si b était une ancre dans P', alors il le serait aussi dans P d'après le lemme 6.24.
- On considère le plus petit ascendant commun de u et u', que l'on nomme u''. On nomme P'' le motif qui étiquette u''. Avec le point précédent, on sait que le sommet b est un sommet de P'' mais n'est pas une ancre dans P''. Ainsi, b ne peut être présent que dans l'un des deux fils de u'', car s'il était dans les deux, il serait une ancre. Notons u_- le fils de u'' qui est dans la direction de u et u'_- le fils de u'' qui est dans la direction de u' (cf. figure 6.26). On sait que b est présent dans le motif associé à

u_- d'après le point précédent. Il n'est donc pas présent dans le motif associé à u'_-. Ainsi, en utilisant la remarque du début de cette preuve, b n'est pas présent dans u', car sinon il le serait dans son ascendant u'_-. □

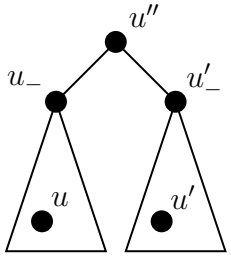


FIGURE 6.26 – Disposition relative des sommets u, u', u'', u_- et u'_- dans la preuve du corollaire 6.25.

Ainsi, le sommet b ne peut être éventuellement une ancre que dans les motifs qui étiquettent un nœud qui est descendant de u. On a alors le lemme suivant :

Lemme 6.26. *Soit un arbre de fusion T ayant au moins deux feuilles. Soit e une arête (obligatoire ou interdite) d'un motif P. On sait qu'il y a une feuille u de T dont le motif étiqueté contient e comme unique arête. Alors, au moins une des deux extrémités a de e dans u est ancrée. En outre, si l'autre extrémité b n'est pas une ancre, alors b n'est une ancre dans aucun motif de T.*

Démonstration. Supposons qu'aucune des deux extrémités de e ne soit ancrée. Comme T a au moins deux feuilles, on sait que la feuille u a une sœur u'. Notons (P', A') le motif ancré associé à u'. Par définition d'un arbre du fusion, il faut que la fusion des motifs P et P' soit unique. Or, comme P n'a pas d'ancre, cela signifie qu'aucun des sommets de P n'est dans P' (puisque tout sommet en commun doit être une ancre d'après la définition 6.19). Ainsi, on peut construire deux ordres τ différents tels que $\tau(P) \prec \tau$ et $\tau(P') \prec \tau$: dans le premier ordre τ, on met d'abord les sommets de P puis ensuite ceux de P', et on fait l'inverse pour le second. La fusion de P et P' n'est pas uniquement déterminée, ce qui contredit la définition d'arbre de fusion.

Dans ce cas, l'autre extrémité b n'est pas une ancre et d'après le corollaire 6.25, b ne peut pas être une ancre dans les motifs qui étiquettent un nœud qui est descendant de u. Comme u est une feuille, il n'y a pas de tel nœud et b n'est une ancre dans aucun nœud de T. □

On obtient finalement la proposition suivante. Les différents termes et notations impliqués sont rappelés juste après l'énoncé.

Proposition 6.27. *Soit $P \to P_1, P_2$. Soit A, A_1 et A_2 l'ensemble des ancres de P, P_1 et P_2 définis dans la définition 6.19. Soit I l'ensemble des sommets en commun à P_1 et P_2 : $I = V(P_1) \cap V(P_2)$. Soit $\mathfrak{T}$, $\mathfrak{T}_1$ et $\mathfrak{T}_2$ les tableaux des réalisations partielles de (P, A), (P_1, A_1) et (P_2, A_2). Alors $\mathfrak{T} = \mathfrak{T}_1 \odot_{I\setminus A} \circledast_{I\cap A} \mathfrak{T}_2$, où le produit tensoriel mixte (cf. définition 2.28) est effectué dans l'algèbre de Boole $\mathbb{B}$.*

Cette proposition justifie la définition des ancres faite dans la définition 6.19, car les ancres sont précisément définies de sorte à ce que $\mathfrak{T}$ puisse être calculable à partir de $\mathfrak{T}_1$ et $\mathfrak{T}_2$. Pour démontrer cette proposition, on commence par décrire l'ensemble des objets impliqués.

On sait que A est un ensemble de sommets de $V(P) = V(P_1) \cup V(P_2)$ potentiellement quelconque. En revanche, I est fixé et vaut $I = V(P_1) \cap V(P_2)$. De même, A_1 et A_2 sont fixés en fonction de A et valent $A_1 = I \cup (A \cap V(P_1)) = ((V(P_1) \cap V(P_2)) \cup (A \cap V(P_1))) = V(P_1) \cap (A \cup V(P_2))$ et $A_2 = I \cup (A \cap V(P_2)) = ((V(P_1) \cap V(P_2)) \cup (A \cap V(P_2))) = V(P_2) \cap (A \cup V(P_1))$. On rappelle que le produit tensoriel mixte s'exprime dans notre cas par la formule suivante, dans laquelle les calculs sont effectués dans l'algèbre de Boole $\mathbb{B}$, c'est-à-dire l'ensemble $\{0, 1\}$ dans lequel la somme exprime en réalité un maximum :

$$(\mathfrak{T}_1 \odot_{I \setminus A} \circledast_{I \cap A} \mathfrak{T}_2)[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] = \sum_{I \setminus A : U} \mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, I \setminus A : U] \times \mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, I \setminus A : U],$$

où la somme (ou plutôt le maximum) s'effectue sur tous les dictionnaires $I \setminus A : U$, c'est-à-dire les dictionnaires qui ont pour ensemble d'étiquettes $I \setminus A$ (qui sont donc des sommets de P) et où U est une liste des valeurs associées à chaque étiquette, c'est-à-dire que U est une liste de sommets de G. De plus, $\mathcal{D}_1$ est la liste des étiquettes qui sont dans $\mathfrak{T}_1$ mais pas dans $\mathfrak{T}_2$, c'est-à-dire l'ensemble des ancres de $V(P_1) \setminus V(P_2)$, qui vaut donc

$$\begin{aligned}
&A_1 \cap (V(P_1) \setminus V(P_2)) \\
&= V(P_1) \cap (A \cup V(P_2)) \cap (V(P_1) \setminus V(P_2)) \\
&= (A \cup V(P_2)) \cap (V(P_1) \setminus V(P_2)) \\
&= A \cap (V(P_1) \setminus V(P_2)).
\end{aligned}$$

De même, $\mathcal{D}_2$ est la liste des étiquettes qui sont dans $\mathfrak{T}_2$ mais pas dans $\mathfrak{T}_1$ et vaut $A_2 \cap (V(P_2) \setminus V(P_1)) = A \cap (V(P_2) \setminus V(P_1))$. Enfin, $\mathcal{D}_\pi$ est la liste des étiquettes de $I \cap A$ par définition.

Ainsi, on vérifie bien que les dictionnaires $\mathcal{D}_1$, $\mathcal{D}_2$ et $\mathcal{D}_\pi$ sont disjoints et que leurs étiquettes forment une partition de A. De même pour $\mathcal{D}_1$, $\mathcal{D}_\pi$ et $I \setminus A : U$ vis-à-vis de A_1 et $\mathcal{D}_2$, $\mathcal{D}_\pi$ et $I \setminus A : U$ vis-à-vis de A_2.

Dans un premier temps, je présente la preuve de la proposition 6.27 dans le cas où $A = \emptyset$.

Démonstration (dans le cas $A = \emptyset$). Dans ce cas, $I \setminus A = I$, $I \cup A = \emptyset$, $A \cap (V(P_1) \setminus V(P_2)) = \emptyset$ et $A \cap (V(P_2) \setminus V(P_1)) = \emptyset$. Ainsi, les dictionnaires $\mathcal{D}_1$, $\mathcal{D}_2$ et $\mathcal{D}_\pi$ sont vides et le produit à calculer est

$$\sum_{I:U} \mathfrak{T}_1[I : U] \times \mathfrak{T}_2[I : U].$$

De plus, $\mathfrak{T}_1 \odot_{I \setminus A} \circledast_{I \cap A} \mathfrak{T}_2 = \mathfrak{T}_1 \odot_I \mathfrak{T}_2$ est un tenseur à zéro dimension, c'est-à-dire un nombre. On note t ce nombre qui correspond au contenu du tenseur $\mathfrak{T}$, c'est-à-dire $t = \mathfrak{T}[\emptyset]$. Par définition de $\mathfrak{T}$, $t = 1$ si et seulement si (G, τ) contient P (et $t = 0$ sinon). D'après le lemme 6.23, (G, τ) contient P si, et seulement si, il existe d sommets de (G, τ) (que l'on nomme $U = u_1, \ldots, u_d$) tels que $\mathfrak{T}_1[u_1, \ldots, u_d] = \mathfrak{T}_2[u_1, \ldots, u_d] = 1$, qui s'écrit alors $\mathfrak{T}_1[I : U] = \mathfrak{T}_2[I : U] = 1$. Pour ce U particulier, on a donc $\mathfrak{T}_1[I : U] \times \mathfrak{T}_2[I : U] = 1$ et donc $\sum_{I:U} \mathfrak{T}_1[I : U] \times \mathfrak{T}_2[I : U] = 1$ (puisque la somme est un maximum sur des valeurs valant 0 ou 1). Ainsi, $t = 1$ si et seulement si $\sum_{I:U} \mathfrak{T}_1[I : U] \times \mathfrak{T}_2[I : U] = 1$ (et ces deux valeurs valent donc 0 dans l'autre cas puisqu'on est dans l'algèbre de Boole $\mathbb{B}$). On a donc $\mathfrak{T}[\emptyset] = \sum_{I:U} \mathfrak{T}_1[I : U] \times \mathfrak{T}_2[I : U]$, c'est-à-dire $\mathfrak{T} = \mathfrak{T}_1 \odot_I \mathfrak{T}_2$. □

Avant de passer au cas général, j'introduis quelques notations : Je note $\mathcal{D}_\pi = I \cap A : U'$, $U'' = U \cup U'$. Je note U_1 les valeurs de $\mathcal{D}_1$ et U_2 les valeurs de $\mathcal{D}_2$. Ainsi, les ensembles U, U', U'', U_1 et U_2 sont des ensembles de sommets de G.

Démonstration (dans le cas général). On veut montrer $\mathfrak{T} = \mathfrak{T}_1 \odot_{I\setminus A} \circledast_{I\cap A} \mathfrak{T}_2$. Pour ce faire, on fixe des dictionnaires $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi$ et on montre :

$$\mathfrak{T}[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] = \sum_{I\setminus A:U} \mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, I \setminus A : U] \times \mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, I \setminus A : U].$$

Si $\sum_{I\setminus A:U} \mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, I \setminus A : U] \times \mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, I \setminus A : U] = 1$, alors en particulier on a un dictionnaire $I \setminus A : U$ tel que $\mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, I \setminus A : U] = 1$ et $\mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, I \setminus A : U] = 1$ (puisqu'on rappelle que les calculs se font dans l'algèbre de Boole $\mathbb{B}$). On a $d = |U''|$ sommets de (G, τ) (qui sont ceux de U'') tels qu'il y ait une réalisation partielle en U'' des motifs ancrés (P_1, I) et (P_2, I) dans (G, τ), où I est toujours égal à $V(P_1) \cap V(P_2)$. Ainsi, d'après le lemme 6.23, on déduit qu'il existe une réalisation partielle en U'' du motif ancré (P, I) dans le graphe ordonné (G, τ). Cependant, si on suit la preuve du lemme 6.23 et notamment la construction de la fonction f, on note que la réalisation partielle en U'' du motif ancré (P, I) dans le graphe ordonné (G, τ) envoie les sommets qui sont dans $V(P_1) \setminus V(P_2)$ au même endroit que f_1 et les sommets qui sont dans $V(P_2) \setminus V(P_1)$ au même endroit que f_2. Cela signifie que les sommets dans $A \cap (V(P_1) \setminus V(P_2))$ sont envoyés sur les mêmes sommets de G que l'on considère le motif P ou le motif P_1. De même pour P et P_2 vis-à-vis des sommets dans $A \cap (V(P_2) \setminus V(P_1))$. Cela donne une réalisation partielle en $U \cup U' \cup U_1 \cup U_2$ du motif ancré $(P, A \cup U)$ dans le graphe ordonné (G, τ). À fortiori, il s'agit une réalisation partielle en $U' \cup U_1 \cup U_2$ du motif ancré (P, A) dans le graphe ordonné (G, τ) et donc $\mathfrak{T}[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] = 1$.

Dans l'autre sens, si $\mathfrak{T}[\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_\pi] = 1$ alors il existe une réalisation partielle en $U_1 \cup U_2 \cup U'$ du motif ancré (P, A) dans le graphe ordonné (G, τ). A fortiori, il existe une réalisation du motif P dans le graphe ordonné (G, τ). D'après le lemme 6.23, on déduit qu'il existe d sommets de (G, τ) (que l'on nomme $U = u_1, \ldots, u_d$) tels qu'il y a une réalisation partielle en U des motifs ancrés (P_1, I) et (P_2, I) dans (G, τ), où I est toujours égal à $V(P_1) \cap V(P_2)$. Cependant, si on suit la preuve du lemme 6.23, on note que les réalisations partielles en U des motifs ancrés (P_1, I) et (P_2, I) dans (G, τ) sont des restrictions de la réalisation partielle en S du motif ancré (P, A). Cela signifie que les sommets de $V(P_1)$ sont envoyés sur les mêmes sommets de G dans la réalisation partielle de (P, A) et dans celle de (P_1, I). Ainsi, on a une réalisation partielle en $U_1 \cup U''$ du motif $(P_1, (A \cup I) \cap V(P_1)) = (P_1, A_1)$, c'est-à-dire $\mathfrak{T}_1[\mathcal{D}_1, \mathcal{D}_\pi, I \setminus A : U] = 1$. De même, $\mathfrak{T}_2[\mathcal{D}_2, \mathcal{D}_\pi, I \setminus A : U] = 1$. □

Remarque 6.5. Si on effectue le produit tensoriel mixte dans le corps $(\mathbb{R}, +, \times)$ classique, alors le tenseur $\mathfrak{T}$ ne calcule plus simplement si une réalisation partielle du motif P est présente ou non, mais permet plus précisément de dénombrer le nombre de réalisations partielles du motif P.

6.5.4 Largeur de fusion

Étant donné un arbre de fusion T d'un motif P, on peut calculer la complexité nécessaire pour déterminer si P est contenu dans un graphe ordonné (G, τ). On commence par ajouter des ancres à chaque motif de l'arbre suivant la définition 6.19. Soit k le nombre de sommets de P. On sait alors que l'arbre de fusion a une taille d'au plus k^2. Si (P_i, A_i) est un motif ancré qui étiquette un nœud de l'arbre de fusion, on note $\mathfrak{T}_i$ le tableau des réalisations partielles de (P_i, A_i) dans (G, τ) et d_i le nombre de dimensions de $\mathfrak{T}_i$. On note que $d_i = |A_i|$ par définition du tableau des réalisations partielles. Soit $d_{\max}$ le plus

grand des d_i. Le nombre $d_{\max}$ correspond donc au plus grand nombre d'ancres d'un motif. Soit n le nombre de sommets de G. Alors, d'après la proportion 2.20, on sait que l'on peut calculer un $\mathfrak{T}_i$ de l'arbre à partir des tableaux des réalisations partielles de ses fils en temps $\mathcal{O}(\sqrt{n^{\omega d_{\max}}})$. De plus, une feuille étant constituée d'un motif ayant une arête dont l'une des extrémités est une ancre (cf. lemme 6.26), d'après le lemme 6.18, on peut calculer son tableaux des réalisations partielles directement en temps $\mathcal{O}(n^2+n^{d_{\max}})$. Ainsi, on peut construire le tableau des réalisations partielles associé au motif P récursivement en temps $\mathcal{O}(k^2(n^2+\sqrt{n^{\omega d_{\max}}}))$.

Exemple 6.13. Reprenons le cas de l'arbre de fusion T de la figure 6.25. La valeur de $d_{\max}$ pour cet arbre de fusion T est 3 car c'est le nombre maximum d'ancres d'un motif qui étiquette un nœud de T.

Ainsi, étant donné un motif P, on veut choisir un arbre de fusion qui minimise la quantité $d_{\max}$. Cela motive la définition suivante :

Définition 6.20 (Largeur de fusion)**.** *Soit P un motif. Si T est un arbre de fusion de P, la largeur de cet arbre de fusion est le nombre maximum d'ancres d'un motif de T. La* largeur de fusion *de P, noté $mw(P)$ (pour* merge-width*), est le plus petit entier p tel que P admette un arbre de fusion de largeur inférieure ou égale à p. Autrement dit :*

$$mw(P) = \min_{T} \max_{u_i \in T} |A_i|.$$

Il est important de noter que la largeur de fusion d'un motif P dépend de l'ordre $\tau(P)$, bien que cela ne soit pas explicite. Cela vient du fait que l'arbre du fusion doit contenir des fusions uniquement déterminées et que cette notion dépend de l'ordre des sommets dans les deux motifs à fusionner. En revanche, si deux motifs P_1 et P_2 vérifient $M(P_1)\cup F(P_1) = M(P_2)\cup F(P_2)$, alors $\mathrm{mw}(P_1) = \mathrm{mw}(P_2)$. Cela vient du fait que la notion de fusion uniquement déterminée ne dépend pas du type des arêtes (cf. lemme 6.20).

Remarque 6.6. Pour tout motif P, on a $\mathrm{mw}(P) \leq |V(P)|$. En effet, P possède un arbre de fusion T via la remarque 6.4. Comme l'ensemble des ancres est incluse dans l'ensemble des sommets d'un motif et que toutes les étiquettes de T sont des motifs ayant $V(P)$ comme ensemble de sommets, chaque ensemble d'ancres A de chaque motif qui étiquette un nœud de T vérifie $|A| \leq |V(P)|$, d'où l'inégalité voulue.

On déduit enfin le théorème principal de cette section :

Théorème 6.7. *Soit P un motif et soit $p = mw(P)$ sa largeur de fusion. On peut décider si un graphe ordonné (G, τ) à n sommets contient ou évite P en temps polynomial $\mathcal{O}(n^2 + n^{\omega p/2})$.*

Dès que la largeur de fusion p est supérieure ou égale à deux, cette complexité se simplifie en $\mathcal{O}(n^{\omega p/2})$. On rappelle que l'algorithme de détection est créé pour un motif P fixé. En particulier, le temps nécessaire pour calculer un arbre de fusion optimal pour le motif P (ce qui revient à considérer le temps nécessaire pour trouver l'algorithme de détection) n'est pas pris en compte dans la complexité temporelle $\mathcal{O}(n^2 + n^{\omega p/2})$.

6.5.5 Application

Le théorème 6.7 peut s'appliquer à la classe des motifs planaires extérieurs. Si P est un motif planaire extérieur, on peut montrer que $\mathrm{mw}(P) \leq 2$. Cela se fait d'une façon

très similaire à la preuve présentée dans la section 6.4.2 qui montrait que P pouvait être détecté en temps $\mathcal{O}(n^\omega)$. En appliquant le théorème 6.7, on obtient également la borne $\mathcal{O}(n^\omega)$. Cela montre que ce théorème général donne d'aussi bonnes bornes qu'une étude fine.

Lemme 6.28. *Soit P un motif planaire extérieur. Alors $mw(P) \leq 2$.*

Démonstration. D'après le lemme 6.16, on peut décomposer récursivement un motif planaire extérieur jusqu'à n'avoir que des motifs composés d'une seule arête. On construit l'arbre de fusion T en suivant cette décomposition :

— Si P contient une arête couvrante, il se décompose en $P \setminus (1,k)$ et $(1,k)$. L'arbre de fusion T est constitué de la racine u étiquetée par P, qui possède deux sous-arbre fils. Le premier est obtenu récursivement en décomposant $P \setminus (1,k)$. Le deuxième est un nœud étiqueté avec la motif $(1,k)$.

— Si P contient un sommet séparateur t, il se décompose en $P[1,t]$ et $P[t,k]$. L'arbre de fusion T est constitué de la racine u étiquetée par P, qui possède deux sous-arbre fils. Le premier est obtenu récursivement en décomposant $P[1,t]$ et le second en décomposant $P[t,k]$.

On vérifie que T est un arbre de fusion. Pour ce faire, si u est un nœud interne ayant exactement deux fils u_1, u_2, il faut que le motif P qui étiquette u se décompose en P_1 qui étiquette u_1 et P_2 qui étiquette P_2. On vérifie cela dans les deux cas :

— Soit $P_1 = P \setminus (1,k)$ et $P_2 = (1,k)$. Dans ce cas, $V(P_1) = V(P)$ et le seul ordre $\tau(P)$ vérifiant $\tau(P_1) \prec \tau(P)$ est $\tau(P_1)$. Ainsi, la fusion de P_1 et P_2 est uniquement déterminée.

— Soit $P_1 = P[1,t]$ et $P_2 = P[t,k]$. Dans ce cas, il n'y a qu'un seul ordre $\tau(P)$ vérifiant $\tau(P_1) \prec \tau(P)$ et $\tau(P_2) \prec \tau(P)$ à cause du sommet t qui est présent dans les deux ordres $\tau(P_1)$ et $\tau(P_2)$. En effet, le sommet t est le plus grand sommet de l'ordre $\tau(P_1)$ et le plus petit sommet de l'ordre $\tau(P_2)$. Le seul ordre possible $\tau(P)$ est celui tel que si $u \in V(P_1) \setminus \{t\}$ et $v \in V(P_2) \setminus \{t\}$ alors $u <_{\tau(P)} v$.

Il reste maintenant à calculer la largeur de fusion de T. On montre récursivement que l'ensemble des ancres A du motif P qui étiquette un nœud u de T est inclus dans $\{\min V(P), \max V(P)\}$. Autrement dit, les seules ancres possibles de P sont les sommets qui sont aux extrémités de P. On note A_1 et A_2 les ancres de P_1 et P_2. D'après la définition 6.19, on a $A_1 = I \cup (A \cap V(P_1))$ et $A_2 = I \cup (A \cap V(P_2))$. On vérifie que $A_1 \subseteq \{\min V(P_1), \max V(P_1)\}$ et que $A_2 \subseteq \{\min V(P_2), \max V(P_2)\}$ dans les deux cas :

— Soit $P_1 = P \setminus (1,k)$ et $P_2 = (1,k)$. Dans ce cas, $I = \{1,k\}$ et $A \subseteq \{1,k\}$ par récurrence. Ainsi, $A_1 \subseteq \{1,k\}$. De plus, comme $V(P_2) \subseteq \{1,k\}$, on déduit $A_2 \subseteq \{1,k\}$.

— Soit $P_1 = P[1,t]$ et $P_2 = P[t,k]$. Dans ce cas, $I = \{t\}$ et $A \subseteq \{1,k\}$ par récurrence. Ainsi, $A_1 \subseteq I \cup (A \cap V(P_1)) \subseteq \{t\} \cup \{1\}$. De même, $A_2 \subseteq \{t,k\}$.

Finalement, pour tout ensemble d'ancres A de T, $|A| \leq 2$, ce qui conclut. □

6.5.6 Bornes

On cherche maintenant à comparer la largeur de fusion avec d'autres paramètres de graphe classiques. La largeur arborescente d'un graphe (ou *treewidth* en anglais) est définie dans [RS86]. Elle suit une logique très similaire à la définition de la largeur de fusion.

Définition 6.21 (Arbre de décomposition arborescente [RS86])**.** *Soit G un graphe. Un arbre de* décomposition arborescente *de G est un arbre T tel que :*

- *chaque nœud de T est étiqueté par un ensemble de sommets B,*
- *chaque sommet de G est dans l'étiquette B d'un nœud de T,*
- *pour chaque arête (x, y) de G, il existe un nœud dont l'étiquette B contient les sommets x et y,*
- *pour chaque sommet x de G, l'ensemble des nœuds de T dont l'étiquette contient x est connexe.*

Définition 6.22 (Largeur arborescente [RS86])**.** *Soit G un graphe. Si T est un arbre de décomposition arborescente de G, la largeur de T est égal au cardinal maximum d'une étiquette de T moins 1. La* largeur arborescente *de G, notée $tw(G)$, est égal à la largeur minimum d'un arbre de décomposition arborescente de G :*

$$tw(G) = \min_{T} \max_{u_i \in T} |B_i| - 1.$$

On définit la largeur arborescente d'un motif P comme étant la largeur arborescente du graphe sous-jacent, c'est-à-dire du graphe G tel que $V(G) = V(P)$ et $E(G) = E(P) = M(P) \cup F(P)$. On peut comparer la largeur de fusion avec la largeur arborescente :

Théorème 6.8. *Pour tout motif P, $tw(P) \leq \frac{3}{2} mw(P) - 1$, si $mw(P) \geq 2$.*

Démonstration. L'arbre de fusion T est presque un arbre vérifiant les conditions de la largeur arborescente. Soit $p = \text{mw}(P)$. Tout d'abord, chaque étiquette est élaguée pour ne contenir plus que A_i. Ensuite, il faut insérer un nœud entre chaque nœud u et ses fils u_1 et u_2. Le nouveau nœud u' est étiqueté par $A_1 \cup A_2$. On a donc u qui a pour fils u' qui a pour fils u_1 et u_2, avec les étiquettes suivantes :

- le nœud u a pour étiquette l'ensemble des ancres A associé au motif P qui étiquetait le nœud u dans l'arbre de fusion,
- le nœud u_1 a pour étiquette l'ensemble d'ancres A_1 défini dans la définition 6.19, à savoir $A_1 = I \cup (A \cap V(P_1))$, où P_1 est le motif qui étiquetait u_1 dans l'arbre de fusion, P_2 celui qui étiquetait u_2 et $I = V(P_1) \cup V(P_2)$,
- le nœud u_2 a pour étiquette l'ensemble d'ancres $A_2 = I \cup (A \cap V(P_2))$,
- le nœud u' a pour étiquette l'ensemble d'ancres $A_1 \cup A_2 = I \cup (A \cap V(P_1)) \cup (A \cap V(P_2)) = I \cup A$.

On note ce nouvel arbre T'. La taille de $A_1 \cup A_2$ est trivialement bornée par $2p$ vu que A_1 et A_2 sont tous deux de taille au plus p. Cependant, on peut obtenir une meilleur borne de $\frac{3}{2}p$ en introduisant $x = |A_1 \cap A_2|$. Pour ce faire, on partitionne $A_1 \cup A_2$ de façon à obtenir $|A_1 \cup A_2| = |A_1 \setminus A_2| + |A_2 \setminus A_1| + |A_1 \cap A_2| \leq (p - x) + (p - x) + x = 2p - x$. De plus, $A_1 \cup A_2 = A \cup I$ et $I \subseteq A_1 \cap A_2$ vu que $I \subseteq A_1$ et $I \subseteq A_2$ par définition de A_1 et A_2. Ainsi, $|A_1 \cup A_2| \leq |A| + |A_1 \cap A_2| \leq p + x$. finalement, $|A_1 \cup A_2| \leq \min(2p - x, p + x) \leq \frac{3}{2}p$, car le maximum est atteint quand $x = p/2$.

Il reste à montrer que ce nouvel arbre T' est bien un arbre de décomposition arborescente de P :

- Tout d'abord, on montre que l'ensemble des nœuds dont l'ensemble associé contient le sommet a forme un sous-arbre de T'. On sait grâce au lemme 6.24 que dans T, cela forme soit un sous-arbre, soit une union de deux sous-arbres dont les racines

sont sœurs. Grâce aux nœuds intermédiaires que l'on a rajoutés, on a pu connecter dans T' les deux sous-arbres dont les racines sont sœurs. Ainsi, l'ensemble des nœuds dont l'ensemble associé contient le sommet a forme un sous-arbre de T'.

— Ensuite, on montre que pour toute arête de P, il existe un nœud de T' qui contient les deux extrémités de l'arête. D'après le lemme 6.26, chaque arête e est présente dans un nœud u de T et il y a deux cas possibles : soit les deux extrémités a et b sont des ancres, auquel cas les deux extrémités sont dans le nœud correspondant dans T' ; soit seule l'extrémité a est une ancre et b ne l'est pas. Dans ce cas là, on va ajouter un nouveau nœud u' à l'arbre T', qui sera fils de u et qui contiendra les deux extrémités de e. Cela ne change pas la condition de connexité précédente car le lemme 6.26 nous assure que dans ce cas, l'extrémité manquante b n'était ancre dans aucun motif de T. L'ensemble des nœuds dont l'ensemble associé contient le sommet b forme donc un sous-arbre de T' constitué uniquement de nœud u'. De plus, cela ne change pas la taille maximale d'un nœud de T' car on a supposé que $\mathrm{mw}(P) \geq 2$, ce qui signifie qu'il y avait déjà au moins un nœud constitué de deux ancres.

Ainsi, vu que chaque nœud a une taille bornée par $\frac{3}{2}p$, la largeur de cet arbre est $\frac{3}{2}p - 1$. On a donc $\mathrm{tw}(P) \leq \frac{3}{2}p - 1$, d'où le résultat. □

Exemple 6.14. Le cas des graphes planaires extérieurs permet d'illustrer le cas où cette borne est atteinte. On a montré dans la section 6.5.5 que les motifs planaires extérieurs ont une largeur de fusion d'au plus 2. Or, il est connu que les graphes planaires extérieurs ont une largeur arborescente maximale valant 2. L'inégalité du théorème 6.8 s'écrit donc avec $\mathrm{mw}(P) = 2$ et $\mathrm{tw}(P) = 2$. Dans ce cas, $\frac{2}{3}(\mathrm{tw}(P) + 1) = 2 \times 3/3 = 2$, ce qui est égal à $\mathrm{mw}(P)$.

Exemple 6.15. La condition $\mathrm{mw}(P) \geq 2$ n'est utile que pour le cas du motif Bipartite (défini par $V(G) = \{1, 2, 3\}$, $\tau = [1, 2, 3]$ et $E(G) = \{(1, 2), (2, 3)\}$). Ce motif a une largeur arborescente de 1 et on montre facilement qu'il a une largeur de fusion de 1. Dans ce cas, $\frac{2}{3}(\mathrm{tw}(P)+1) = 2 \times 2/3 = \frac{4}{3}$, ce qui est supérieur à $\mathrm{mw}(P)$ et cela contredirait le théorème 6.8 si on n'imposait pas la condition $\mathrm{mw}(P) \geq 2$. Ainsi, le cas $\mathrm{mw}(P) < 2$ peut arriver, mais uniquement pour un seul motif.

Un autre paramètre de graphe classique est la largeur de chemin [RS83]. Il s'agit de la même définition que la largeur arborescente en ajoutant la contrainte selon laquelle l'arbre T doit être un chemin. Ce paramètre est équivalent au nombre sommet-séparant [EST87], dont la définition est la suivante :

Définition 6.23 (Nombre séparant, sommet séparant, nombre sommet-séparant [EST87])**.** *Étant donné un graphe ordonné (G, τ), le* nombre séparant *d'un sommet v de (G, τ) est le nombre de sommets $u <_\tau v$ tel que u a un voisin w vérifiant $v \leq_\tau w$. Les sommets u vérifiant cette propriété par rapport à v sont appelés des sommets* séparant *v. Le* nombre sommet-séparant *d'un graphe ordonné est le plus grand nombre séparant de l'un de ces sommets. Le nombre sommet-séparant d'un graphe G est le plus petit nombre sommet-séparant du graphe ordonné (G, τ) parmi tous les ordres τ possibles.*

On remarque que la définition du nombre sommet-séparant d'un graphe fait intervenir la notion de nombre sommet-séparant pour un graphe ordonné. C'est cette version qui nous intéresse, étant donné que l'on travaille sur des graphes ordonnés. Afin de continuer dans la tradition des paramètres de largeur, on appelle *largeur de chemin ordonnée* d'un

graphe ordonné (G, τ) (noté opw$((G, \tau))$ pour *ordered pathwidth* en anglais) le nombre sommet-séparant de (G, τ). La largeur de chemin ordonnée d'un motif P est la largeur de chemin ordonnée du graphe ordonné (G, τ) sous-jacent, c'est-à-dire tel que $V(G) = V(P)$, $E(G) = E(P) = M(P) \cup F(P)$ et $\tau = \tau(P)$. On peut comparer la largeur de fusion avec la largeur de chemin ordonnée :

Théorème 6.9. *Soit P un motif. On a $mw(P) \leq opw(P) + 2$.*

Pour prouver ce théorème, on introduit un arbre de fusion trivial d'un motif, appelé *l'arbre de fusion linéaire*. Étant donné un motif P ayant pour sommets $V(P) = \{1, \dots k\}$, on définit deux familles de sous-motifs de P qui vont être les étiquettes des nœuds de l'arbre de fusion linéaire de P :

— pour $1 \leq i \leq k$, on définit P_i comme étant la restriction de P à ces i premiers sommets, avec toutes les arêtes de P dont les deux extrémités sont dans $\{1, \dots, i\}$,

— pour $2 \leq i \leq k$, on définit P_i^{N-} comme étant la restriction de P aux sommets $i-1$, i et aux voisins de i qui sont à sa gauche (c'est-à-dire les voisins de i qui sont dans $\{1, \dots, i-1\}$), avec uniquement les arêtes qui ont i comme extrémité droite.

Dans les deux cas, l'ordre des sommets est l'ordre croissant. Maintenant, on montre que le motif P_i se décompose en P_{i-1} et P_i^{N-}. Comme $V(P_{i-1}) = \{1, \dots, i-1\}$ et $\{i-1, i\} \subseteq V(P_i^{N-}) \subseteq \{1, \dots, i\}$, le seul ordre qui étend à la fois $\tau(P_{i-1})$ et $\tau(P_i^{N-})$ est l'ordre croissant $[1, \dots, i]$.

Démonstration du théorème 6.9. On construit l'arbre de fusion T. Il s'agit de l'arbre ayant des nœuds u_i pour $1 \leq i \leq k$, des nœuds u'_i pour $2 \leq i \leq k$ et d'autres nœuds permettant de décomposer les nœuds u'_i. Chaque nœud u_i a pour étiquette P_i et pour fils les nœuds u_{i-1} et u'_i. Chaque nœud u'_i a pour étiquette P_i^{N-} et pour sous-arbre un arbre de fusion quelconque de P_i^{N-}, qui existe d'après la remarque 6.4.

Enfin, on calcule la largeur de fusion de T. Pour ce faire, on regarde quel est l'ensemble d'ancre A_i du motif P_i pour tout i. Étant donné un sommet i, on note S_i l'ensemble des sommets séparant i. On remarque que $S_{i+1} \subseteq S_i \cup \{i\}$. En effet, si on prend un sommet u de S_{i+1} qui n'est pas i, alors $u < i + 1$ et u a un voisin w vérifiant $i + 1 \leq w$. comme $u \neq i$, on a $u < i$. De plus, $i + 1 \leq w$ implique $i \leq w$. Ainsi, u est un sommet séparant i.

Tout d'abord, les ancres d'un motif P_i^{N-} sont inclus dans l'ensemble de ces sommets, qui est inclus dans $S_i \cup \{i-1, i\}$.

On montre maintenant par récurrence décroissante que A_i est inclus dans $S_i \cup \{i\}$. Le cas de base est trivial car $A_n = \emptyset \subseteq \{n\} \cup S_n$. On montre l'hérédité. Par définition des ancres, $A_i = I \cup (A_{i+1} \cap \{1, \dots, i\})$. Or, I est l'ensemble des sommets en commun à P_i et P_{i+1}^{N-}, c'est-à-dire que $I \subseteq S_i \cup \{i\}$. De plus, $A_{i+1} \cap \{1, \dots, i\} \subseteq (S_{i+1} \cup \{i+1\}) \cap \{1, \dots, i\} = S_{i+1} \subseteq \{i\} \cup S_i$. Ainsi, $A_i \subseteq \{i\} \cup S_i$, ce qui conclut la récurrence.

Finalement, les motifs P_i ont un ensemble d'ancres inclus dans $S_i \cup \{i\}$ et les motifs P_i^{N-} ont un ensemble d'ancres inclus dans $S_i \cup \{i-1, i\}$. La taille de l'ensemble des ancres d'un motif P est donc systématiquement inférieure ou égale à 2 plus la taille d'un ensemble de sommets séparant un sommet i. Par définition du nombre sommet-séparant d'un motif (qui est le maximum de sommets séparant un sommet i), on a bien l'inégalité $\text{mw}(P) \leq \text{opw}(P) + 2$. □

On remarque que le « +2 » est lié à la nécessité d'avoir le sommet $i-1$ dans P_i^{N-} de sorte à ce que la fusion soit uniquement déterminée. Si le sommet $i-1$ est présent dans les voisinages des sommets qui ont un nombre séparant égal au nombre sommet-séparant du

motif, ce « +2 » devient un « +1 » et on peut avoir la borne $\mathrm{mw}(P) \leq \mathrm{opw}(P) + 1$ pour ces motifs-là. Autrement dit, l'arbre de fusion linéaire d'un motif peut donner la borne $\mathrm{mw}(P) \leq \mathrm{opw}(P) + 1$ pour certains motifs P particuliers. C'est par exemple le cas des chemins (ordonnés naturellement) pour lesquels le nombre séparant de chaque sommet (excepté le premier sommet) est 1 à cause de son voisin juste à gauche. Ainsi, le « +2 » qui était nécessaire pour inclure le sommet juste à gauche *en plus* des voisins n'est pas utile ; un « +1 » suffit. On conclut que la largeur de fusion d'un chemin ordonné naturellement est au plus 2. En revanche, nous pensons que les chemins ordonnés différemment permettent d'illustrer le cas où cette borne est atteinte.

Exemple 6.16. Dans cet exemple, on va construire un motif ordonné dont le graphe sous-jacent est une union de chemin mais dont l'ordre n'est pas l'ordre naturel. On commence par définir le motif « arc-en-ciel » de taille k comme étant le motif R_k tel que $V(R_k) = \{1, \ldots, 2k\}$, $M(R_k) = \{(i, j) \mid i + j = k + 1\}$, $F(R_k) = \emptyset$ et $\tau(R_k) = [1, \ldots, 2k]$. La figure 6.27 montre un exemple de motif arc-en-ciel. Le motif qui nous intéresse dans cet exemple est obtenu en combinant plusieurs motifs arc-en-ciel. Pour ce faire, étant donné deux motifs P_1 et P_2 ainsi qu'un entier t, on appelle *fusion t-jointe* le motif obtenu en identifiant les t derniers sommets de P_1 avec les t premiers sommets P_2 tout en conservant l'ordre des sommets. Par exemple, la figure 6.28 montre la fusion 4-jointe de deux motifs R_4, que l'on nomme $2R_4$. On peut alors faire une fusions 4-jointe de R_4 et $2R_4$ pour obtenir $3R_4$, et ainsi de suite. La figure 6.29 montre le motif $4R_4$. D'une manière générale, le motif kR_k a une largeur de chemin ordonné de k et nous pensons que sa largeur de fusion est également égale à k. Nous n'avons aucune preuve de ce dernier point mais il s'agit de l'une de nos perspectives principales.

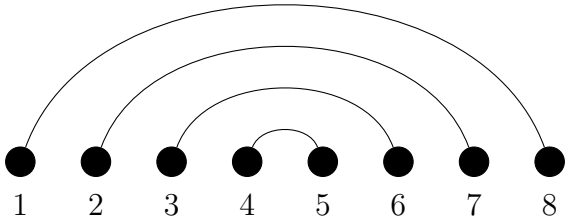


FIGURE 6.27 – Motif arc-en-ciel R_4 de taille 4.

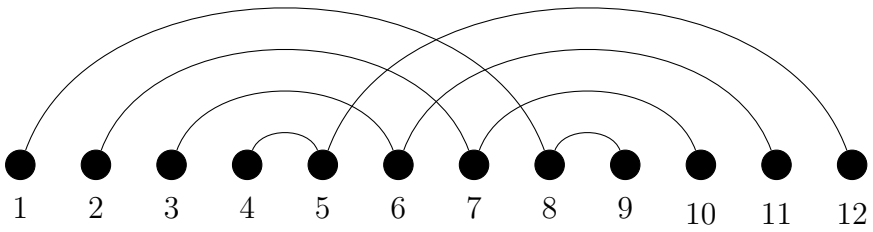


FIGURE 6.28 – Motif $2R_4$ obtenu en faisant le fusion 4-jointe de deux motifs arc-en-ciel de taille 4.

Conjecture 6.1. *Le motif kR_k a pour largeur de fusion $mw(kR_k) = k$.*

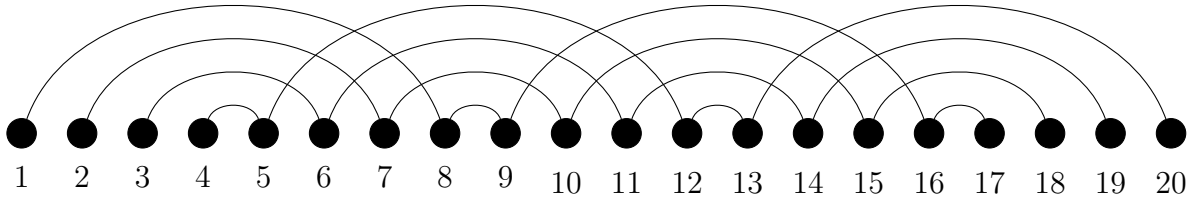


FIGURE 6.29 – Motif $4R_4$ obtenu en faisant le fusion 4-jointe de quatre motifs arc-en-ciel de taille 4.

Après avoir montré les cas où les bornes des théorèmes 6.8 et 6.9 étaient atteintes, il reste à discuter des cas où l'écart est le plus grand. Concernant la comparaison entre largeur arborescente et largeur de fusion, le cas du motif kR_k exhibe un écart aussi grand que l'on veut si l'on admet la conjecture 6.1, puisque kR_k est une union de chemins et a donc une largeur arborescente de 1 pour tout k. Concernant la comparaison entre largeur de fusion et largeur de chemin ordonné, les cas des arbres binaires donne un écart aussi grand que l'on veut puisque la largeur de chemin d'un arbre binaire complet à n sommets est de l'ordre de $\log(n)$ [Sch90] (ce qui signifie que n'importe quel ordre de cet arbre a une largeur de chemin ordonné de l'ordre de $\log(n)$) alors que la largeur de fusion d'un tel arbre (ordonné de façon à être planaire extérieur) est au plus 2 grâce au lemme 6.28. Ces écarts arbitrairement grands montrent la nécessité d'introduire ce nouveau paramètre de graphe pour notre problème car il est plus fin que ceux existants.

6.6 Perspectives

Petits motifs. Le théorème 6.1 indique que la plupart des motifs à trois sommets peuvent être détectés en temps linéaire, excepté TRIANGLE et COMPARABILITY (ainsi que leurs complémentaires). Il est généralement admis qu'il est impossible de détecter un triangle en temps sous-quadratique [WW10]. Cependant, le cas de COMPARABILITY est nouveau. Il n'y a aucune certitude qu'il ne puisse pas être détecté en temps linéaire, étant donné que le motif CHORDAL qui lui est très similaire le peut et étant donné que la classe des graphes évitant COMPARABILITY (qui est la classe des graphes de comparabilité) peut être détectée en temps linéaire en construisant un ordre qui évite COMPARABILITY dès que cela est possible [MS97]. Il serait donc intéressant d'approfondir ce cas et soit de trouver un algorithme linéaire, soit de trouver d'autres motifs similaires (à quatre sommets par exemple) pour lesquels il est impossible de trouver un algorithme linéaire et ainsi dégager une obstruction à la linéarité.

Largeur de fusion. Un point sur lequel il reste encore beaucoup de choses à découvrir est ce nouveau paramètre qu'est la largeur de fusion. Tout d'abord, nous aimerions prouver la conjecture 6.1, ce qui montrerait qu'un motif aussi simple qu'un chemin peut être compliqué dès que l'ordre induit plein de croisements d'arêtes. Ensuite, nous voudrions nous concentrer sur les obstructions de la largeur de fusion, c'est-à-dire trouver des sous-motifs permettant d'affirmer qu'un motif les possédant a une grande largeur de fusion. Le sous-motif kR_k est un bon candidat si la conjecture est vraie. Cependant, ce sous-motif étant assez grand, il pourrait être intéressant d'avoir des obstructions plus petites ou plus malléables. Enfin, il serait intéressant de trouver d'autres classes « naturelles » de

largeur de fusion bornée. Nos premiers candidats étaient les classes de motifs de largeur arborescente bornée ou d'épaisseur de livre borné [BK79], mais si la conjecture 6.1 est vraie, elle fournit un contre-exemple à ces classes puisque kR_k a une largeur arborescente de 1 et une épaisseur de livre de 2 pour tout k. Cela exclurait aussi la classe des motifs planaires (pas nécessairement planaires extérieurs) puisque kR_k est planaire et que, dans tous les cas, un motif planaire a une épaisseur de livre d'au plus 4 [Yan89].

Chapitre 7

Conclusion

Au cours de cette thèse, j'ai pu mener un travail de recherche aussi bien théorique qu'expérimental et ainsi balayer de nombreux pans de l'informatique moderne. Les travaux de compression ont mené au développement d'un programme informatique permettant de compresser des graphes issus du réel d'une façon efficace et nouvelle, en cherchant diverses structures et en prenant en compte les chevauchements entre elles. Ce programme fonctionne en utilisant des algorithmes existants de recherche de structures tels que Slashburn, BigClam, Louvain et Metis, en utilisant une heuristique gloutonne pour les sélectionner ainsi que du raffinement de partition pour obtenir l'objet final que nous compressons en utilisant le système de numération combinatoire ainsi que les représentations d'Élias. Nous avons espoir que cet algorithme puisse être véritablement utilisé tel quel ou dans une version améliorée afin de compresser des données réelles. Cette idée de vouloir chercher les structures les plus adaptées à la compression m'a conduit à la volonté de créer mes propres structures dont la théorie sous-jacente pourrait être mise au service de la compression.

Je suis alors parti de structures connues telles que les modules et les splits et je les ai étudiées afin d'en comprendre les moindres détails. Cela m'a amené à redémontrer la stabilité des splits par union disjointe. J'ai ensuite pu étendre les splits à une forme plus souple, que j'ai appelé les r-splits. Le but était alors d'étudier ces nouvelles structures de la même façon que les splits avaient été étudiés auparavant et de se demander si tout ce qui valait pour les splits restait vrai pour les r-splits. Certains théorèmes ont ainsi pu être adaptés, comme la représentation et la recherche polynomiale de ces structures. J'ai également obtenu une borne inférieure sur le nombre d'arêtes d'un hypergraphe de r-splits, ce qui a permis de déduire une borne inférieure sur le nombre de r-splits essentiels d'un graphe r-rang connecté. D'autres théorèmes sont encore en questionnement, comme la possibilité d'écrire cette représentation polynomiale sous une forme arborescente ou d'effectuer cette recherche polynomiale au moyen de relations d'orthogonalité.

Pendant ces années de thèse, j'ai eu l'occasion de discuter avec Michel Habib et Laurent Feuilloley qui avait conjointement travaillé sur la recherche de motifs dans les graphes ordonnés, sujet qui m'intéressait beaucoup de par sa nouveauté dans le domaine de l'algorithmique et de par son lien fort avec ce que j'avais déjà fait. J'ai pu rejoindre leur projet et apporter de nouvelles idées, notamment concernant la détection de motifs de taille arbitraire, en proposant des algorithmes efficaces pour trouver des motifs ayant des contraintes plus ou moins fortes. J'ai ainsi pu élaborer une suite d'algorithmes permettant de détecter un motif planaire extérieur positif sans cycle en temps linéaire, de détecter un motif planaire extérieur sans cycle en temps quadratique et de détecter un motif planaire

extérieur en temps égal à celui de la multiplication de matrice. Cela m'a permis d'introduire la largeur de fusion en m'inspirant des autres paramètres existants – notamment la largeur arborescente – et ainsi introduire un paramètre fin adapté au problème. J'ai ainsi obtenu un algorithme de détection de motif quelconque paramétré par la largeur de fusion. Cela m'a aussi permis de manipuler des tenseurs, objets abstraits que j'affectionne particulièrement et qui se sont révélés indispensables à l'étude de la complexité temporelle de la recherche de motifs de largeur de fusion bornée.

Bien que j'ai pu répondre à un certain nombre de questions pendant ma thèse, chaque problème résolu a fait naitre un ensemble de nouvelles questions, parmi lesquelles :

— Concernant l'algorithme de compression de graphe, quelle heuristique peut-on choisir afin d'accélérer les temps de calcul tout en maintenant un résultat satisfaisant ? Une solution peut être d'attribuer un score à chaque structure et de les sélectionner dans cet ordre, afin d'avoir une boucle linéaire en le nombre de structures, et non quadratique.

— Concernant les splits, existe-t-il une preuve purement algébrique démontrant leur stabilité par union disjointe ? Il s'agit à priori d'un problème d'algèbre linéaire assez classique, mais je n'ai pu trouver de réponse. L'avantage serait alors de pouvoir étendre facilement cette preuve aux r-splits, comme cela a été fait pour les autres preuves de stabilité prouvées algébriquement.

— Les r-splits ont-ils une structure arborescente comme c'est le cas pour les splits ? Pour l'instant, on sait seulement qu'une famille de r-splits tous r-orthogonaux entre eux a une telle structure et seulement si leur taille est comprise entre $2r$ et $n - 2r$. Peut-on étendre cette structure à toute famille de r-splits ?

— À propos de la détection de motifs, peut-on trouver des obstructions pour la largeur de fusion ainsi que des bornes supérieures ? La fusion d'arcs-en-ciel semble être un bon candidat pour une obstruction par sous-graphe interdit. Pour la borne supérieure, pouvons-nous trouver une mesure plus fine que la largeur de chemin ?

Finalement, tous ces résultats permettent de mieux comprendre la recherche de régularités dans les graphes, notamment les r-splits nouvellement créés ou les motifs appartenant à certaines classes, en proposant des algorithmes efficaces pour les chercher, et de mieux comprendre la représentation succincte de graphes en proposant un algorithme de compression basé sur la recherche de régularités.

Bibliographie

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[AYZ97] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3) :209–223, 1997.

[BC08] Gregory Buehrer and Kumar Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *Proceedings of the 2008 international conference on web search and data mining*, pages 95–106, 2008.

[BCHP08] Anna Bretscher, Derek Corneil, Michel Habib, and Christophe Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22(4) :1277–1296, 2008.

[BDG+07] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2) :172–188, 2007.

[BGdMT22] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, and Stéphan Thomassé. Twin-width V: linear minors, modular counting, and matrix multiplication. *arXiv preprint, arXiv:2209.12023*, 2022.

[BGK+20] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. *arXiv preprint, arXiv:2007.14161*, 2020.

[BGLL08] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10) :P10008, 2008.

[BGOdM+22] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: ordered graphs and matrices. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 924–937, 2022.

[BH74] James R Bunch and John E Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125) :231–236, 1974.

[BH18] Maciej Besta and Torsten Hoefler. Survey and taxonomy of lossless graph compression and space-efficient graph representations. *arXiv preprint, arXiv:1806.01799*, 2018.

[BK79] Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3) :320–331, 1979.

[BLN09] Nieves R Brisaboa, Susana Ladra, and Gonzalo Navarro. k^2-trees for compact web graph representation. In *International symposium on string processing and information retrieval*, pages 18–30. Springer, 2009.

[BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P Spinrad. *Graph classes: a survey*. SIAM, 1999.

[BV04] Paolo Boldi and Sebastiano Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595–602, 2004.

[BX08] Binh-Minh Bui-Xuan. *Tree-representation of set families in graph decompositions and efficient algorithms*. PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc, 2008.

[BXHR12] Binh-Minh Bui-Xuan, Michel Habib, and Michaël Rao. Tree-representation of set families and applications to combinatorial decompositions. *European Journal of Combinatorics*, 33(5) :688–711, 2012.

[CDMR12] Pierre Charbit, Fabien De Montgolfier, and Mathieu Raffinot. Linear time split decomposition revisited. *SIAM Journal on Discrete Mathematics*, 26(2) :499–514, 2012.

[CE80] William H Cunningham and Jack Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3) :734–765, 1980.

[CHM81] Michel Chein, Michel Habib, and Marie-Catherine Maurer. Partitive hypergraphs. *Discrete mathematics*, 37(1) :35–50, 1981.

[CLL+16] A Chatterjee, M Levan, C Lanham, M Zerrudo, M Nelson, and S Radhakrishnan. Exploiting topological structures for graph compression based on quadtrees. In *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 192–197. IEEE, 2016.

[CN10] Francisco Claude and Gonzalo Navarro. Fast and compact web graph representations. *ACM Transactions on the Web (TWEB)*, 4(4) :1–31, 2010.

[Cov99] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[Cun82] William H Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2) :214–228, 1982.

[Cun83] William H Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1) :53–68, 1983.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.

[Dam90] P. Damaschke. Forbidden ordered subgraphs. *Topics in Combinatorics and Graph Theory, R. Bodendiek and R. Hennm Eds*, pages 219–229, 1990.

[DFHP23] Guillaume Ducoffe, Laurent Feuilloley, Michel Habib, and François Pitois. Pattern detection in ordered graphs. *arXiv preprint, arXiv:2302.11619*, 2023.

[Dor09] Frederic Dorn. Planar subgraph isomorphism revisited. *arXiv preprint, arXiv:0909.4692*, 2009.

[DPS02] Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3) :313–356, 2002.

[EG77] Jack Edmonds and Rick Giles. A min-max relation for submodular functions on graphs. In *Annals of Discrete Mathematics*, volume 1, pages 185–204. Elsevier, 1977.

[EG04] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3) :57–67, 2004.

[Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE transactions on information theory*, 21(2) :194–203, 1975.

[ES46] Paul Erdös and Arthur H Stone. On the structure of linear graphs. 1946.

[EST87] John Arthur Ellis, I Hal Sudborough, and Jonathan S Turner. *Graph separation and search number*. University of Victoria. Department of Computer Science, 1987.

[FG65] Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3) :835–855, 1965.

[FH21a] Laurent Feuilloley and Michel Habib. Classifying grounded intersection graphs via ordered forbidden patterns. *arXiv preprint, arXiv:2112.00629*, 2021.

[FH21b] Laurent Feuilloley and Michel Habib. Graph classes and forbidden patterns on three vertices. *SIAM Journal on Discrete Mathematics (SIDMA)*, 35(1) :55–90, 2021.

[FJ99] Tamás Fleiner and Tibor Jordán. Coverings and structure of crossing families. *Mathematical programming*, 84 :505–518, 1999.

[FKLL15] Peter Floderus, Mirosław Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Detecting and counting small pattern graphs. *SIAM Journal on Discrete Mathematics*, 29(3) :1322–1339, 2015.

[Gal67] Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1-2) :25–66, 1967.

[GB11] Szymon Grabowski and Wojciech Bieniecki. Merging adjacency lists for efficient web graph compression. In *Man-Machine Interactions 2*, pages 385–392. Springer, 2011.

[GM14] Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 82–101. SIAM, 2014.

[GP12] Emeric Gioan and Christophe Paul. Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics*, 160(6) :708–733, 2012.

[GU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.

[HKSS13] Chinh T Hoang, Marcin Kamiński, Joe Sawada, and R Sritharan. Finding and listing induced paths and cycles. *Discrete applied mathematics*, 161(4-5) :633–641, 2013.

[HMPV00] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2) :59–84, 2000.

[HP98] Xiaohan Huang and Victor Y Pan. Fast rectangular matrix multiplication and applications. *Journal of complexity*, 14(2) :257–299, 1998.

[HP10] Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1) :41–59, 2010.

[HP23] Ishay Haviv and Michal Parnas. On the binary and boolean rank of regular matrices. *Journal of Computer and System Sciences*, 134 :73–86, 2023.

[IMH82] Oscar H Ibarra, Shlomo Moran, and Roger Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of algorithms*, 3(1) :45–56, 1982.

[IO09] Satoru Iwata and James B Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1230–1237. SIAM, 2009.

[Kar97] George Karypis. METIS: Unstructured graph partitioning and sparse matrix ordering system. *Technical report*, 1997.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3) :455–500, 2009.

[KF11] U. Kang and Christos Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 300–309, 12 2011.

[KKVF15] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(3) :183–202, 2015.

[KLL13] Mirosław Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and detecting small subgraphs via equations. *SIAM Journal on Discrete Mathematics*, 27(2) :892–909, 2013.

[KNR88] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 334–343, 1988.

[Kol06] Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA ..., 2006.

[KOSU12] Shuji Kijima, Yota Otachi, Toshiki Saitoh, and Takeaki Uno. Subgraph isomorphism in graph classes. *Discrete Mathematics*, 312(21) :3164–3173, 2012.

[Kra10] Andreas Krause. SFO: A toolbox for submodular function optimization. *Journal of Machine Learning Research*, 11 :1141–1144, 2010.

[LG12] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *2012 IEEE 53rd annual symposium on foundations of computer science*, pages 514–523. IEEE, 2012.

[LL12] Xiaonan Li and Sanyang Liu. Matroidal approaches to rough sets via closure operators. *International Journal of Approximate Reasoning*, 53(4) :513–527, 2012.

[LM00] N Jesper Larsson and Alistair Moffat. Off-line dictionary-based compression. *Proceedings of the IEEE*, 88(11) :1722–1732, 2000.

[LR83] Grazia Lotti and Francesco Romani. On the asymptotic complexity of rectangular matrix multiplication. *Theoretical Computer Science*, 23(2) :171–185, 1983.

[LSS+19] Sebastian Lamm, Christian Schulz, Darren Strash, Robert Williger, and Huashuo Zhang. Exactly solving the maximum weight independent set problem on large real-world graphs. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 144–158. SIAM, 2019.

[McC05] S Thomas McCormick. Submodular function minimization. *Handbooks in operations research and management science*, 12 :321–391, 2005.

[McC14] Albert Joseph McConnell. *Applications of tensor analysis*. Courier Corporation, 2014.

[MP16] Sebastian Maneth and Fabian Peternek. Compressing graphs by grammars. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 109–120. IEEE, 2016.

[MS97] Ross M McConnell and Jeremy P Spinrad. Linear-time transitive orientation. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 19–25. Citeseer, 1997.

[Mur99] Kazuo Murota. *Matrices and matroids for systems analysis*, volume 20. Springer Science & Business Media, 1999.

[OS06] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4) :514–528, 2006.

[Oum23] Sang-il Oum. Rank connectivity and pivot-minors of graphs. *European Journal of Combinatorics*, 108 :103634, 2023.

[Oxl06] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

[PHST24] François Pitois, Mohammed Haddad, Hamida Seba, and Olivier Togni. Hypergraphs with polynomial representation: Introducing r-splits. *Discrete Mathematics & Theoretical Computer Science*, 25(Special issues), 2024.

[Pro73] Robert L Probert. *On the complexity of matrix multiplication*. PhD thesis, University of Waterloo Dept. of Applied Analysis and Computer Science, 1973.

[PS98] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[PSH23] François Pitois, Hamida Seba, and Mohammed Haddad. A fine-grained structural partitioning approach to graph compression. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 392–397. Springer, 2023.

[PT06] János Pach and Gábor Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel Journal of Mathematics*, 155(1) :359–380, 2006.

[RA15] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[RAS19] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding, 2019.

[Rig48] Jacques Riguet. Relations binaires, fermetures, correspondances de galois. *Bulletin de la société mathématique de France*, 76 :114–155, 1948.

[RS83] Neil Robertson and Paul D Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1) :39–61, 1983.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of algorithms*, 7(3) :309–322, 1986.

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2) :266–283, 1976.

[RZ18] Ryan A Rossi and Rong Zhou. Graphzip: a clique-based sparse graph compression method. *Journal of Big Data*, 5(1) :1–14, 2018.

[Sau72] Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1) :145–147, 1972.

[Sch90] Petra Scheffler. A linear algorithm for the pathwidth of trees. In *Topics in Combinatorics and Graph Theory: Essays in Honour of Gerhard Ringel*, pages 613–620. Springer, 1990.

[She72] Saharon Shelah. A combinatorial problem ; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1) :247–261, 1972.

[SXB+12] Quan Shi, Yanghua Xiao, Nik Bessis, Yiqi Lu, Yaoliang Chen, and Richard Hill. Optimizing K^2 trees: A case for validating the maturity of network of practices. *Computers & Mathematics with Applications*, 63(2) :427–436, 2012.

[Tar18] Gábor Tardos. Extremal theory of ordered graphs. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3235–3243. World Scientific, 2018.

[TBR18] Abraham Traoré, Maxime Berar, and Alain Rakotomamonjy. Décomposition en ligne de tenseurs. In *Conference sur l'apprentissage statistique*, 2018.

[Tru92] Klaus Truemper. *Matroid decomposition.* Academic Press, 1992.

[VC71] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of Probability and Its Applications*, volume 16, pages 264–280. 1971.

[Vil21] Simon Vilmin. *Algorithms on closure systems and their representations.* PhD thesis, Université Clermont Auvergne, 2021.

[WNC87] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6) :520–540, jun 1987.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual*

Symposium on Foundations of Computer Science, pages 645–654. IEEE, 2010.

[WWWY14] Virginia Vassilevska Williams, Joshua R Wang, Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms*, pages 1671–1680. SIAM, 2014.

[Yan89] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1) :36–67, 1989.

[YL13] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3) :337–343, 1977.