N° d'ordre : 438
N° attribué par la bibliothèque : 07ENSL0438

ÉCOLE NORMALE SUPÉRIEURE DE LYON
Laboratoire de l'Informatique du Parallélisme

THÈSE

présentée et soutenue publiquement le 17 décembre 2007 par

Laurent LYAUDET

pour l'obtention du grade de

Docteur de l'École Normale Supérieure de Lyon

spécialité : Informatique

au titre de l'École doctorale de mathématiques et d'informatique fondamentale de Lyon

Graphes et hypergraphes : complexités algorithmique et algébrique

Directeurs de thèse : Jacques MAZOYER
Ioan TODINCA

Après avis de : Pierre FRAIGNIAUD
Stéphan THOMASSÉ

Devant la commission d'examen formée de :

Alain BRETTO	Membre
Arnaud DURAND	Membre
Pierre FRAIGNIAUD	Membre/Rapporteur
Jacques MAZOYER	Membre
Stéphan THOMASSÉ	Membre/Rapporteur
Ioan TODINCA	Membre

À Arnaud et Vincent.

Remerciements

Je tiens tout d'abord à remercier Vincent Bouchitté pour la qualité des enseignements qu'il m'a donné en cryptologie et en théorie des graphes, pour m'avoir accepté comme stagiaire de DEA puis comme doctorant. Ce mémoire est dédié à la sienne ainsi qu'à celle de mon frère Arnaud.

Je veux aussi remercier mes deux directeurs de thèse : Jacques Mazoyer pour m'avoir garanti dès le décés de Vincent que je pourrai finir ma thèse et Ioan Todinca pour avoir suivi de son mieux depuis Orléans mes avancées scientifiques ainsi que pour ses relectures précises de mes articles et de ce tapuscrit. Votre aide à tous deux m'a été précieuse.

Je remercie Stéphan Thomassé et Pierre Fraigniaud d'avoir accepté d'être les rapporteurs de cette thèse, ainsi qu'Alain Bretto et Arnaud Durand qui ont accepté d'être membres de mon jury de thèse.

Jeg takker Uffe Flarup, forbi han havde tilladet mig hen til hjælp mine graf teori sikkerhed i algebraisk "complexity". Jeg er tilfreds jeg mødte Uffe og vi arbejder sammen. Mange tak Uffe ! Je remercie aussi Pascal Koiran de m'avoir permis de découvrir la complexité algébrique et d'avoir partagé sa problématique avec moi.

Je remercie Guillaume Malod pour sa relecture et ses nombreux commentaires sur la partie complexité algébrique de ce mémoire ainsi que pour nos discussions sur ce sujet.

Il est temps d'attaquer la longue liste des remerciements de mes amis thésards. Je commence par Sylvain Périfel qui m'a autorisé à plagier honteusement son chapitre de définitions et de rappels des classes de bases de complexité pour le transférer de sa thèse à la mienne. Je poursuis avec Florent Becker qui a tenté vainement de me ravir le titre de mec le plus classe de l'équipe à l'aide de ressources internet telles tubgirl (google image – même moi j'aurai pas osé... enfin seulement après le repas), Damien Regnault pour ses pains au chocolat, Jean-Baptiste Rouquier pour son harmonographe, Sylvain Sené pour sa bonne humeur, Stéphane Leroux pour avoir réussi à faire des blagues plus mauvaises que les miennes et m'avoir enseigné un peu de QiGong, Victor Poupet pour avoir sauvé la langue anglaise de mes phrases les plus moches en me corrigeant parfois, Vincent Nesme pour les séances de musique classique et l'incitation à la concentration (conscience ou « awareness » comme dirait Van Damme : il a une paire de ciseaux à la main !), ...

Je remercie aussi les autres membres de l'équipe MC2 et notamment Eric

Thierry pour des discussions scientifiques qui m'ont permis de partager certaines de mes interrogations.

Je remercie Maurice Tchuente pour m'avoir invité au Cameroun (pour le travail, si, si) et son accueil. Je remercie Paulin Melatagia, René Ndoundam et les autres membres du laboratoire d'informatique de Yaoundé pour leur chaleureux accueil.

Je remercie les membres de ma famille et les amis qui sont venus assister à ma soutenance de thèse. Je remercie ma mère pour avoir fait une relecture ortho-grammatico-typographique de ce document et m'avoir aidé à préparer le pot (l'épreuve la plus dure d'une thèse).

Je remercie les membres du Taekwondo Club du parc. Votre companie et les entraînements ont souvent été mon rocher. Je remercie Frédéric Bendahmane d'avoir vaincu sa leucémie ; je n'aurai pas aimé que ma thèse finisse comme elle avait commencée.

Toutes les personnes que j'ai injustement oubliées peuvent m'envoyer un courrier (électronique ou non) de protestation :).

Table des matières

Introduction

L'ubiquité des graphes ou hypergraphes en algorithmique et en complexité est notoirement connue depuis l'origine de ces problématiques. On peut citer les nombreuses structures de données ayant recours à des graphes – et plus souvent à des arbres –, les algorithmes de flots, les nombreux problèmes d'optimisation sur des graphes dont la variante décisionnelle est NP-complète (Coloration propre, clique maximum, ...), la preuve de polynomialité de 2−SAT, les matroïdes vus comme hypergraphes héréditaires munis d'un axiome d'échange, ...

Cette ubiquité est ambivalente dans la complexité des problèmes qu'elle engendre et des solutions qu'elle apporte. Ainsi, l'exubérance de structures nouvelles pouvant apparaître dans les graphes aléatoires permet de montrer souvent très simplement l'existence d'une structure, solution à un problème, alors même qu'une tentative de construction effective d'une telle structure aurait découragé bien des chercheurs. Pour dompter cette exubérance qui est fréquemment la source de la NP-complétude (ou pire) de nombre de problèmes de graphes, il a fallu bien souvent se résoudre à se restreindre à des classes de graphes dont les contraintes supplémentaires permettaient de donner un terreau où pouvait germer l'intuition. Cela a favorisé l'étude de classes topologiques comme les graphes planaires ou les classes de graphes de genre borné, de classes algébriques comme les graphes de Cayley, de classes de graphes d'intersection comme les graphes d'intervalles, de cordes ou de permutation.

Le sujet de mon étude porte sur la nature des liens qui peuvent exister entre certaines classes de graphes et certaines classes de complexité.

Décompositions hiérarchiques de graphes

Afin de restreindre la complexité des graphes et des problèmes qui y sont liés, une autre approche consistant à chercher à décomposer un graphe en éléments simples tout en préservant sa connectivité fut amorcée. Le prototype de cette démarche est la décomposition modulaire qui construit ou représente le graphe à l'aide de modules et d'opérations de composition de ces modules, un module étant un ensemble de sommets ayant le même voisinage hors du module. Chaque graphe peut donc être vu comme une formule construite avec ces opérations dont les constantes sont des modules premiers (indécomposables). Par commodité, on préfère voir la décomposition modulaire comme étant l'arbre

syntaxique de cette formule.

Depuis environ un quart de siècle, cette approche consistant à décomposer un graphe pour le plonger dans une structure d'arbre a été affinée. Dans un premier temps sont apparues les notions de décompositions linéaire et arborescente, toutes deux décomposant un graphe en « sacs » de sommets, chaque sac de sommets étant ensuite associé bijectivement avec le noeud d'un chemin ou d'un arbre. On impose bien entendu que chaque sommet et chaque arête soit présent dans au moins un sac ce qui pour l'arête se traduit par l'existence d'un sac contenant simultanément ses deux extrémités. Pour respecter la connectivité du graphe, il faut que tous les sacs contenant un noeud donné induisent un sous-graphe connexe de la décomposition (sous-chemin ou sous-arbre).

On associe à ces décompositions les notions de largeur linéaire et arborescente. Celles-ci reposent sur l'idée intuitive qu'un graphe décomposé avec des petits sacs a une structure plus proche d'un chemin ou d'un arbre qu'un graphe qui nécessite l'emploi de gros sacs. Ainsi un graphe est de largeur linéaire ou arborescente au plus k s'il possède une décomposition éponyme n'utilisant que des sacs de taille au plus $k+1$. Un point important ici est que, contrairement à la décomposition modulaire qui est unique, un graphe possède une multitude (formellement, une infinité ou un nombre exponentiel si on se restreint aux décompositions sans redondance inutile) de décompositions linéaires ou arborescentes et pour définir sa largeur associée on prend le minimum des largeurs de toutes ses décompositions ; les autres types de décomposition hiérarchique dont nous parlerons sont aussi dans cette situation. Dans le cas de la largeur arborescente, cette définition colle parfaitement avec le concept de k-arbre partiel qui découle des travaux de Dirac et Rose [60]. Celui-ci est basé sur l'idée que tout arbre est construit inductivement à partir d'un sommet isolé puis en ajoutant des feuilles ; de même un k-arbre est obtenu inductivement à partir d'une clique de taille k puis en ajoutant des k-feuilles (des sommets reliés à une clique de taille k dans le graphe déjà construit).

Ces deux types de décompositions sont des outils fondamentaux (ré)introduits par Robertson et Seymour dans leur étude des mineurs de graphe [56, 57] (Halin avait déjà introduit sous un autre nom les décompositions arborescentes en 1976 [33]). Rappelons qu'un graphe H est mineur d'un graphe G, si H peut être obtenu en enlevant des sommets, des arêtes ou en contractant des arêtes de G. Le premier résultat important sur les mineurs de graphe est le théorème de Kuratowski [47] caractérisant les graphes planaires comme étant la famille des graphes n'ayant ni K_5, ni $K_{3,3}$ comme mineur. Robertson et Seymour ont généralisé ce résultat en montrant que toute classe de graphes close par minoration est caractérisée par un nombre fini de mineurs exclus, ceci étant une conséquence directe du théorème des mineurs de graphes. Celui-ci fut appelé conjecture de Wagner par Robertson et Seymour qui en ont fourni une preuve en 2004 [59], bien qu'il semble qu'en fait Wagner ne l'ait jamais conjecturé mais en ait discuté vers 1960 avec ses étudiants de l'époque, Halin et Mader. Ce théorème énonce que les graphes finis sont bel-ordonnés par minoration, ou de manière équivalente que toute famille infinie de graphes finis contient deux graphes dont l'un est mineur de l'autre. Ce théorème généralise le théorème de Kruskal pour les arbres étiquetés [46] qui lui-même généralise le lemme de Higman pour les mots [35].

Un autre résultat fort joli de leur étude des mineurs de graphes est la ca-

ractérisation suivante : « Une famille de graphes est de largeur arborescente bornée si et seulement si elle possède un mineur exclu planaire ». Puisqu'une famille de graphes de largeur arborescente bornée est close par minoration et que tout graphe est représentable sans croisements dans un espace à trois dimensions, l'intérêt de ce résultat réside bien entendu dans l'aspect planaire du mineur exclu. C'est là toute la beauté de ce résultat, il montre que la part de richesse des graphes qui leur permet de s'éloigner d'une structure d'arbre est une richesse intrinsèquement planaire. Si l'espace à trois dimensions ou les surfaces de genre supérieur apportent une richesse supplémentaire, celle-ci est d'une autre nature.

Le caractère central des décompositions arborescentes est encore accentué par l'équivalence entre la largeur arborescente d'un graphe et le nombre d'étiquettes distinctes nécessaires pour le construire au moyen d'un type particulier d'algèbres (universelles) de graphes, dites algèbres HR (Hyperedge Replacement). Ces algèbres permettent de construire des graphes, dont les sommets sont étiquetés, à partir de graphes de base (un sommet isolé ou une arête), d'opérations de gestion des étiquettes (renommage, oubli) et d'une opération de composition parallèle de deux graphes qui fusionne les sommets de même étiquette.

Après avoir été introduites dans un but propre à la théorie des graphes, les décompositions linéaire et arborescente ont su trouver un grand nombre d'applications en algorithmique des graphes. En effet, nombreux sont les problèmes NP-difficiles sur les graphes que l'on sait résoudre en temps polynomial sur les arbres et, dans un grand nombre de cas, un algorithme polynomial sur un arbre peut être étendu en un algorithme de programmation dynamique qui fonctionne en temps polynomial sur une classe de graphes de largeur linéaire ou arborescente bornée. Un tel algorithme fonctionne généralement sur le modèle suivant : étant donnée une décomposition arborescente enracinée, on calcule de bas en haut, pour chaque noeud, un ensemble de solutions pour le sous-graphe induit par les sacs présents en dessous du noeud, à partir des solutions pour les fils de ce noeud. L'idée essentielle ici est que l'ensemble de solutions construit en chaque noeud est d'une taille indépendante de la taille du graphe ; plus précisément, on construit un ensemble de solutions qui couvre l'ensemble des possibilités de recollement avec le reste du graphe. Ce recollement s'effectuant uniquement via les sommets du sac du noeud considéré, la taille de l'ensemble des solutions à considérer est bornée par une fonction de la largeur linéaire ou arborescente.

Dans cette optique, un résultat majeur est le théorème de Courcelle démontré à la fin des années 1980 [20]. Celui-ci énonce que tout problème de décision exprimable dans la logique monadique du second ordre (sur le vocabulaire τ_1 ou τ_2) est décidable en temps linéaire sur une classe de graphes de largeur arborescente bornée. Le vocabulaire τ_1 est uniquement constitué d'une relation binaire traduisant l'adjacence entre deux sommets, alors que le vocabulaire τ_2 contient une relation binaire traduisant l'incidence entre un sommet et une arête et deux prédicats unaires permettant de distinguer les sommets des arêtes. Il est bien connu que l'existence d'un circuit hamiltonien peut être formalisée dans la logique monadique du second ordre sur le vocabulaire τ_2 mais pas sur le vocabulaire τ_1. Rappelons que la logique monadique du second ordre ne permet que de quantifier des relations unaires, c'est-à-dire des ensembles

d'objets. Cela se traduit par la possibilité de quantifier des ensembles de sommets quand on utilise le vocabulaire τ_1 et celle de quantifier des ensembles de sommets ou d'arêtes quand on utilise le vocabulaire τ_2.

Ces travaux ont par la suite été étendus de deux manières : la première fut de démontrer que ce résultat pouvait s'appliquer à des problèmes d'optimisation ou de comptage en étendant la logique monadique du second ordre avec des opérateurs lui permettant de « compter » ; la seconde fut l'émergence d'un nouveau type de décomposition hiérarchique de graphes, les clique-décompositions introduites par Courcelle, Engelfriet et Rozenberg [23]. Ces décompositions sont en fait les arbres syntaxiques de termes d'un autre type d'algèbre de graphe que j'appellerai algèbre de clique. Cette algèbre construit des graphes étiquetés mais, contrairement à l'algèbre HR, elle n'a comme constructeur de base qu'un sommet isolé et utilise donc les étiquettes non plus pour fusionner des sommets mais pour ajouter des arêtes. Cet ajout se fait par « blocs de bipartis complets » puisqu'il consiste, par exemple, à lier tout couple de sommets dont l'un est d'étiquette a et l'autre d'étiquette b. On définit à nouveau la largeur de clique d'un graphe comme étant le nombre d'étiquettes distinctes nécessaires pour construire ce graphe à l'aide de l'algèbre de clique. Cette notion de clique-décomposition est plus puissante que la décomposition arborescente ou modulaire car, si l'on définit la largeur modulaire d'un graphe comme la taille du plus grand module premier de sa décomposition modulaire, alors toute famille de graphes de largeur arborescente ou modulaire bornée est de largeur de clique bornée. Notons que les largeurs arborescente et modulaire ne sont pas comparables. Cette puissance a toutefois un revers, on sait seulement montrer que tout problème de décision exprimable dans la logique monadique du second ordre sur le vocabulaire τ_1 est décidable en temps linéaire sur une classe de graphes de largeur de clique bornée. On sait même que si $\mathsf{P}_1 \neq \mathsf{NP}_1$ (l'indice 1 indique qu'il s'agit des sous-classes de P et NP restreints aux langages sur un alphabet unaire), alors il existe un problème de la logique monadique du second ordre sur le vocabulaire τ_2 qui n'est pas décidable en temps polynomial.

Les décompositions arborescente et de clique ont chacune un équivalent approché en terme de décomposition de graphes sur un arbre ternaire. Dans le cas de la décomposition arborescente, il s'agit de la décomposition en branches introduite par Robertson et Seymour [58]. Celle-ci consiste en un arbre ternaire dont les feuilles sont en bijection avec les arêtes du graphe. Ainsi chaque arête de l'arbre induit une séparation des arêtes du graphe en deux parties. La frontière de cette séparation est l'ensemble des sommets du graphe qui sont incidents à au moins une arête dans chaque partie de la séparation. La largeur de la décomposition en branches est alors le maximum des tailles des frontières des séparations induites par les arêtes de l'arbre ternaire. Les largeurs arborescente et de branche d'un graphe diffèrent d'au plus un facteur $\frac{3}{2}$. Dans le cas de la clique-décomposition, il s'agit de la décomposition de rang introduite par Oum et Seymour [37]. Cette fois, les feuilles de l'arbre ternaire sont en bijection avec les sommets du graphe et toute arête de l'arbre définit une partition (I, J) des sommets. La largeur d'une arête est alors le rang sur $\mathbb{Z}_2$ de la sous-matrice d'adjacence de lignes I et de colonnes J. Là aussi, la largeur d'une décomposition de rang est le maximum des rangs des sous-matrices associées aux arêtes de l'arbre. Le lien entre largeurs de rang et de clique est un peu plus relaché

car si la largeur de rang est inférieure à celle de clique, la largeur de clique est réciproquement inférieure à une fonction exponentielle, dans le pire des cas, de la largeur de rang.

L'apport de toutes ces décompositions hiérarchiques de graphes est lui-même ambivalent car si elles ont permis de fournir nombre de solutions aussi bien à des problèmes théoriques que pratiques, elles ont aussi engendré bon nombre de problématiques nouvelles. Exceptées la décomposition modulaire et sa largeur associée qui peuvent être calculées en temps linéaire, toutes ces autres largeurs sont NP-difficiles à calculer. Néanmoins, il existe, pour k fixé, des algorithmes de reconnaissance polynomiaux des graphes de largeur linéaire ou arborescente au plus k. On ne sait pas s'il existe de tels algorithmes pour la largeur de clique et il est probable qu'on ne connaisse la réponse à cette question qu'après d'importants progrès théoriques sur les obstructions à cette largeur. Toutefois, il existe, toujours pour k fixé, un algorithme de reconnaissance polynomial des graphes de largeur de rang au plus k qui fournit un algorithme d'approximation de facteur exponentiel de la largeur de clique. La NP-difficulté du calcul de ces largeurs a aussi motivé la conception d'algorithmes pour calculer ou approcher ces largeurs sur différentes classes de graphes : approximation de la largeur arborescente des graphes planaires (via le calcul exact de la largeur de branche), calcul de la largeur arborescente de graphes d'intersections tels que les graphes de cordes, calcul de la largeur linéaire pour des graphes de largeur arborescente bornée, ...

Graphes et complexités

Depuis le théorème de Fagin [29] démontré en 1974, on sait l'importance des graphes pour la théorie de la complexité descriptive. En effet, celui-ci énonce qu'une famille de structures (son langage associé) est dans NP si et seulement si elle est axiomatisable dans la logique existentielle du second ordre (sur les vocabulaires munis d'un ordre) ; une famille de telles structures peut toujours être interprétée comme une famille de graphes.

Avant de poursuivre plus avant, il me faut dire deux mots de complexité algébrique et plus précisément du modèle de Valiant. Alors qu'en complexité booléenne on s'attache à classifier des langages qui sont des ensembles de mots sur un alphabet, dans le modèle de Valiant, l'alphabet est remplacé par un corps $\mathbb{K}$, les mots sont remplacés par des polynômes sur $\mathbb{K}$ et les langages deviennent des suites de polynômes. La complexité de telles suites de polynômes n'est pas mesurée au moyen de machines de Turing mais au moyen de circuits arithmétiques (graphes orientés acycliques avec des portes d'addition ou de multiplication). Ainsi la classe VP (une analogue algébrique de P) correspond aux suites de polynômes de degré polynomial et calculables par une suite de circuits arithmétiques de taille polynomiale. La classe VNP est une analogue algébrique de NP dont la définition est trop complexe pour figurer dans cette introduction. Enfin, une autre classe importante est celle des formules (ou expressions ou termes) arithmétiques, notée VP_e(e comme expression), qui correspond aux suites de polynômes calculables par une suite de circuits arithmétiques de taille polynomiale dont les graphes sous-jacents sont des arbres.

Cette importance, disais-je, s'illustre aussi par les nombreux problèmes de graphes NP-complets ainsi que par les caractérisations combinatoires des pro-

blèmes #P-complets[1] (ou VNP-complets dans le cadre de la complexité algébrique) les plus importants tels que le permanent[2] et l'hamiltonien [67, 68]. Ainsi, il est bien connu que le permanent d'une matrice entière dont les entrées sont 0 ou 1 correspond au nombre de couvertures par circuits du graphe défini par cette matrice ou, de manière équivalente, au nombre de couplages parfaits de son biparti d'adjacence. De même, l'hamiltonien de cette matrice correspond au nombre de circuits hamiltoniens du graphe associé. Ces caractérisations combinatoires sont souvent le pont amenant à définir des familles d'instances « faciles » pour ces problèmes. Un résultat célèbre de ce type est celui de Fisher, Kasteleyn et Temperley en 1961 [30, 42, 64] montrant que le nombre de couplages parfaits d'un graphe planaire est calculable en temps polynomial à l'aide du Pfaffien. De même, les résultats de Courcelle, Makowsky et Rotics [24] permettent de calculer efficacement le permanent et l'hamiltonien des graphes de largeur arborescente bornée.

L'objet de ce mémoire est de défendre la thèse selon laquelle les classes de graphes les plus significatives, telles que la classe des graphes planaires ou celles définies par le caractère borné d'une de leurs largeurs, sont non seulement de bons outils pour trouver des algorithmes efficaces mais surtout qu'elles capturent intrinsèquement une part de la complexité des graphes et par extension des complexités algorithmique et algébrique, ce qui devrait se traduire par le fait que la restriction à ces classes de problèmes de graphes vérifiant certains critères fournit des problèmes complets[3] pour des classes naturelles de complexité.

[1] Grossièrement, un problème de comptage est #P-complet si son calcul effectif permet de connaître le nombre de solutions de tout problème dans NP. La notion de VNP-complétude sera quant à elle définie ultérieurement.

[2] Rappelons que le permanent et l'hamiltonien d'une matrice $M = (m_{i,j})$ de dimension $n \times n$ sont définis par la somme $\sum_{\sigma \in X} \prod_{i=1}^{n} m_{i,\sigma(i)}$ qui porte sur $X = S_n$, l'ensemble des permutations sur n éléments, dans le cas du permanent et $X = C_n$, l'ensemble des n-cycles de S_n, dans le cas de l'hamiltonien.

[3] complet ne signifie pas nécessairement difficile (au sens humain ; au sens théorie de la complexité, si), cela signifie « le plus grand parmi ». Ainsi « le plus grand parmi les nains » reste un « nain » et un problème P-complet n'est pas bien difficile mais il capture toute la complexité de P.

Contenu du mémoire

Ce tapuscrit est divisé en deux parties : la première porte sur des résultats de complexité algébrique et la seconde présente des résultats de complexité algorithmique.

La première partie de ce mémoire est dédiée à la complexité algébrique des interprétations combinatoires du permanent et de l'hamiltonien quand on se restreint aux graphes planaires ou aux graphes de largeur linéaire, arborescente ou de clique bornée. Elle comporte 6 chapitres.

Les deux premiers chapitres sont des chapitres introductifs. Le premier rappelle les définitions d'un grand nombre de classes de complexité booléenne, ainsi que les définitions des classes de complexité algébrique liées au modèle de Valiant. Le second rappelle les définitions des principales décompositions et largeurs de graphe, ainsi que celles du permanent et de l'hamiltonien et de leurs caractérisations combinatoires. Il présente aussi de nouvelles notions de largeurs de clique pondérée plus adaptées à nos travaux.

Le chapitre 3 s'attache à démontrer que le permanent et l'hamiltonien sont complets pour la classe des formules arithmétiques quand on se restreint aux graphes de largeur arborescente bornée. On y montre donc que le permanent ou l'hamiltonien de graphes de largeur arborescente bornée peut être calculé dans $\mathsf{VP_e}$ et réciproquement, que toute formule arithmétique est égale au permanent d'un graphe de largeur arborescente 2 ou à l'hamiltonien d'un graphe de largeur arborescente 6.

Le chapitre 4 comprend la démonstration d'un résultat plus puissant puisque le permanent et l'hamiltonien sont complets pour la classe des formules arithmétiques quand on se restreint aux graphes de largeur linéaire bornée. Nous y montrons aussi au passage que les formules arithmétiques sont équivalentes aux circuits (faiblement) asymétriques de largeur bornée.

Le chapitre 5 fournit la preuve d'un encadrement de la complexité du permanent et de l'hamiltonien sur les graphes de largeur de clique pondérée bornée entre la classe des formules arithmétiques et la classe VP des circuits multiplicativement disjoints.

Le chapitre 6 étudie la complexité algébrique du permanent et de l'hamiltonien des graphes planaires. S'il était déjà connu que les couvertures par circuits ou les circuits hamiltoniens restent VNP-complets, nous montrons que les couplages parfaits sont eux VDET-complets, VDET étant la classe de complexité du déterminant.

Les résultats de cette première partie ont été obtenus en collaboration avec Uffe Flarup et Pascal Koiran. Plus précisément, les résultats des chapitres 3 et 6 ont été obtenus avec Uffe Flarup et Pascal Koiran. Ils ont d'ailleurs fait l'objet d'une publication à ISAAC 2007[31]. Les résultats des chapitres 4 et 5 ont été obtenus avec Uffe Flarup et sont rédigés dans un article qui sera soumis très prochainement.

La seconde partie de ce mémoire dédiée à la complexité algorithmique ne comprend que le chapitre 7. Ce chapitre illustre les effets de seuil et les variations de complexité qui peuvent apparaître lors de l'étude d'un problème de partitionnement d'hypergraphes, lié au calcul de la largeur de branche, mais qui possède aussi des interprétations naturelles liées notamment à l'ordonnancement et au problème de transport. Ce problème consiste à partitionner les sommets de l'hypergraphe en k classes de sorte que chaque hyperarête ait des

sommets dans au plus l classes distinctes. Nous montrons que la complexité de ce problème peut varier de manière radicale selon les valeurs des paramètres k et l et la classe d'hypergraphes considérée. Les résultats de ce chapitre font l'objet d'un article soumis au journal Theoretical Computer Science [50].

Enfin, ce mémoire comprend un chapitre de conclusions et perspectives.

Première partie

Complexité algébrique

Chapitre 1

Classes de complexité

Ce chapitre est le premier chapitre préliminaire à la partie 1 de ce document. Il comporte bien entendu les définitions des classes de complexité algébrique dont nos travaux donnent une caractérisation en termes de couvertures de différentes familles de graphes. Néanmoins, afin que le lecteur puisse situer ces classes parmi les nombreuses classes de complexité, je présente un grand nombre d'autres classes notamment de complexité booléenne classique, en m'inspirant de la présentation faite par Sylvain Perifel dans sa thèse. La question de l'uniformité des circuits est également abordée ; c'est certainement l'une des parties les plus intéressantes du chapitre. Le lecteur trouvera donc ci-après les définitions des classes de complexité usuelles ainsi que du modèle de Valiant. Pour comprendre nos résultats, le lecteur pourra se contenter de piocher les définitions qui lui manquent parmi celles d'un circuit arithmétique, d'une formule arithmétique, d'un circuit asymétrique ou faiblement asymétrique, des classes de complexité $\mathsf{VP_e}$, VDET, VNC, VP et VNP, présentées dans le cadre du modèle de Valiant.

Il sera aussi profitable de connaître les définitions des classes NC, #P, GapL et $\oplus$P pour suivre mes commentaires en dehors des preuves. Je signale au lecteur l'existence d'un index lui permettant de trouver rapidement, au cours de sa lecture des chapitres suivants, les numéros de pages des définitions dont il aurait besoin.

1.1 Classes booléennes, partie 1

Nous donnons ici un aperçu des classes que nous rencontrerons dans ce tapuscrit. On pourra se référer au livre de Papadimitriou [55] ou à celui de Balcázar, Díaz et Gabarró [2] pour les plus connues d'entre elles, au panorama plus complet du livre d'Hemaspaandra et Ogiwara [34] pour les autres, voire au *Complexity Zoo* [17] pour les plus curieux.

Langages

Nous sommes ici dans le cas booléen : un langage est simplement un ensemble de mots sur l'alphabet $\{0,1\}$. Ainsi, un langage A est une partie de $\{0,1\}^*$. Si n

est un entier, on notera $A^{=n}$ l'ensemble des mots de A de taille n.

Les classiques

Pour une présentation des machines de Turing, on choisira son livre favori de calculabilité ou de complexité. Si $f : \mathbb{N} \to \mathbb{N}$ est une fonction, on note $\mathsf{DTIME}(f(n))$ (respectivement $\mathsf{DSPACE}(f(n))$) l'ensemble des langages A reconnus par une machine de Turing fonctionnant en temps $O(f(n))$ (resp. travaillant en espace $O(f(n))$). On note $\mathsf{NTIME}(f(n))$ l'ensemble des langages A reconnus par une machine de Turing non-déterministe en temps $O(f(n))$. On définit alors les classes suivantes :

L	$= \mathsf{DSPACE}(\log n)$	(espace logarithmique),
P	$= \cup_{k\geqslant 0}\mathsf{DTIME}(n^k)$	(temps polynomial),
NP	$= \cup_{k\geqslant 0}\mathsf{NTIME}(n^k)$	(temps non-déterministe polynomial),
PSPACE	$= \cup_{k\geqslant 0}\mathsf{DSPACE}(n^k)$	(espace polynomial) et
EXP	$= \cup_{k\geqslant 0}\mathsf{DTIME}(2^{n^k})$	(temps exponentiel).

Remarque. L'espace requis par une machine de Turing dans un calcul peut aussi être vu comme le temps mis par un algorithme parallèle. Il s'agit de la *parallel computation thesis* (voir par exemple Papadimitriou [55, p. 398]) : le temps parallèle $f(n)$ est inclus dans l'espace $f(n)$ qui est lui-même inclus dans le temps parallèle $f(n)^2$. On verra par la suite le lien avec la profondeur des circuits, facilement interprétée en termes de temps parallèle.
Remarque. On peut aussi définir NP de la manière suivante. Un langage A est dans NP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$\begin{array}{rcl} x \in A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant 1 \text{ et} \\ x \notin A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| = 0. \end{array}$$

La notation (x, y) représente le codage d'un couple de mots. Elle est ici définie par $(x, y) = f(x, y)$ où f est une bijection calculable en temps polynomial de $\{0,1\}^* \times \{0,1\}^*$ dans $\{0,1\}^*$.

Oracles

Nous rappelons ici l'idée intuitive d'oracles. Un oracle A est un langage. Une machine $\mathcal{M}$ avec oracle A est simplement une machine de Turing munie d'un état spécial permettant d'interroger l'oracle. Lorsque la machine entre dans cet état, à l'étape suivante elle entre dans un nouvel état q_{oui} ou q_{non}, selon que le mot sur le ruban de la machine est respectivement dans A ou non. Ainsi, en une seule étape de calcul, on sait si un mot est dans le langage A ou non, et ce quelle que soit la complexité de A.

On peut adjoindre un oracle à une classe de complexité : il suffit de considérer des machines de Turing avec cet oracle. On notera alors l'oracle en exposant de la classe. Par exemple, P^A est l'ensemble des langages reconnus en temps polynomial par des machines de Turing munies de l'oracle A.

Enfin, si l'oracle peut être choisi arbitrairement dans toute une classe $\mathcal{C}$, on notera cette classe en exposant : par exemple, P^{NP} est la classe des langages reconnus en temps polynomial par des machines de Turing munies d'un oracle NP.

Hiérarchie polynomiale

La hiérarchie polynomiale est constituée de niveaux notés Σ_i^{p} pour $i \geqslant 0$ (ici, p signifie polynomial pour différencier d'autres hiérarchies, notamment la hiérarchie arithmétique). Ces niveaux sont définis comme des classes à oracles :

$$\Sigma_0^{\mathsf{p}} = \mathsf{P} \text{ et } \Sigma_{i+1}^{\mathsf{p}} = \mathsf{NP}^{\Sigma_i^{\mathsf{p}}}.$$

On peut aussi définir les classes Σ_i^{p} grâce à une alternance de i quantificateurs ; on pourra voir à ce sujet Papadimitriou [55] par exemple. La réunion de toutes ces classes forme la hiérarchie polynomiale :

$$\mathsf{PH} = \cup_{i \geqslant 0} \Sigma_i^{\mathsf{p}}.$$

Classes probabilistes

Avant d'attaquer les classes de comptage, nous rappelons la définition des deux classes courantes d'algorithmes probabilistes : BPP et RP. La première est l'ensemble des problèmes résolus par des machines probabilistes pouvant faire des erreurs « de chaque côté » ; pour la seconde en revanche, seul un côté est autorisé. En pratique, on fixe une constante entre $\frac{1}{2}$ et 1 comme $\frac{2}{3}$; on demande alors que tout mot du langage soit reconnu avec probabilité au moins $\frac{2}{3}$ et que tout mot hors du langage soit rejeté avec probabilité au moins $\frac{2}{3}$ pour BPP. Pour RP on demande de reconnaître les mots du langage avec probabilité $\frac{1}{2}$ et de toujours rejeter les mots hors du langage. Formellement cela donne quelque chose comme ce qui suit. Un langage A est dans BPP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$\begin{array}{rcl} x \in A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (2/3)2^{p(|x|)} \text{ et} \\ x \notin A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \leqslant (1/3)2^{p(|x|)}. \end{array}$$

De la même manière, A est dans RP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$\begin{array}{rcl} x \in A & \Longrightarrow & |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (1/2)2^{p(|x|)} \text{ et} \\ x \notin A & \Longrightarrow & \{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\} = \emptyset. \end{array}$$

Classes de comptage

Lorsque le seuil entre acceptation et rejet est nul dans l'algorithme probabiliste, on obtient la classe PP. On ne parle plus vraiment d'une classe probabiliste, il s'agit plutôt d'une classe de comptage. En voici la définition formelle. Un langage A est dans PP s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \geqslant (1/2)2^{p(|x|)}.$$

Sous ses airs anodins, cette classe est assez volumineuse : Toda [65] a montré qu'utilisée en oracle d'une machine polynomiale, elle contient toute la hiérarchie polynomiale. En d'autres termes, le théorème de Toda s'écrit ainsi :

$$\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}.$$

Remarquons que nous avons les inclusions suivantes entre les classes ci-dessus :

$$\mathsf{P} \subseteq \mathsf{RP} \subseteq \mathsf{BPP} \subseteq \mathsf{PP}.$$

Une conjecture officieuse mais répandue veut que $\mathsf{P} = \mathsf{RP} = \mathsf{BPP}$; cette opinion étant liée aux progrès enregistrés dans les travaux théoriques sur les générateurs pseudo-aléatoires. Il y a 20 ans, la conjecture $\mathsf{P} \neq \mathsf{BPP}$ était bien plus populaire.

De la même façon, on peut compter le nombre de mots y modulo un entier. On obtient ainsi les classes $\mathsf{Mod}_k\mathsf{P}$, dont voici la définition : si $k \geqslant 2$ est un entier, un langage A est dans $\mathsf{Mod}_k\mathsf{P}$ s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}| \not\equiv 0 \mod k.$$

Lorsque k est un nombre premier, Beigel et Gill [5] ont montré que $\mathsf{Mod}_k\mathsf{P}$ est close par complément : on peut donc prendre $\equiv 0$ plutôt que $\not\equiv 0$ dans la définition. Comme le veut l'usage, on notera $\oplus\mathsf{P}$ plutôt que $\mathsf{Mod}_2\mathsf{P}$.

Hiérarchie de comptage

La hiérarchie de comptage introduite par Wagner [72] est définie à partir de la classe PP d'une manière similaire à la hiérarchie polynomiale à partir de NP. Ainsi, elle est constituée des niveaux C_i pour $i \geqslant 0$ définis comme suit :

$$\mathsf{C}_0 = \mathsf{P} \text{ et } \mathsf{C}_{i+1} = \mathsf{PP}^{\mathsf{C}_i}.$$

La hiérarchie de comptage est alors

$$\mathsf{CH} = \cup_{i \geqslant 0} \mathsf{C}_i.$$

Classes de fonctions

Des classes de *fonctions* sont également définies : il s'agit de fonctions $f : \{0,1\}^* \to \mathbb{N}$ qui prennent un mot booléen et renvoient un entier. Ainsi, une fonction f est dans $\#\mathsf{P}$ s'il existe un langage $B \in \mathsf{P}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}|.$$

De même, une fonction f est dans $\#\mathsf{L}$ s'il existe un langage $B \in \mathsf{L}$ et un polynôme $p(n)$ tel que pour tout mot x,

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in B\}|.$$

Les classes GapP et GapL sont alors les clôtures de $\#\mathsf{P}$ et $\#\mathsf{L}$ par soustraction, c'est-à-dire que l'on considère maintenant des fonctions $g : \{0,1\}^* \to \mathbb{Z}$.

Notons qu'il est facile de voir que $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$. Autrement dit, la classe de fonctions $\#\mathsf{P}$ (ou un problème $\#\mathsf{P}$-complet) est extrèmement puissante puisqu'en oracle d'une machine polynomiale, elle permet de décider tout problème de la hiérarchie polynomiale.

Mentionnons que, pour toutes ces classes de comptage, on utilise le plus souvent des réductions parcimonieuses entre problèmes, de telles réductions préservant le nombre de solutions.

Complémentaire

Pour une classe de complexité $\mathcal{C}$, on notera co$\mathcal{C}$ l'ensemble des langages ${}^cA$ où $A \in \mathcal{C}$, c'est-à-dire l'ensemble des complémentaires des langages de $\mathcal{C}$. Des exemples bien connus sont notamment coNP et coRP.

Conseils

On peut également aider le calcul d'une machine par un conseil. Ce conseil est donné sous la forme d'un mot identique pour toutes les entrées de même taille. On parle alors de classe non-uniforme car ce conseil peut, par exemple, coder un circuit et le calcul effectué sur les entrées de taille n peut alors être très différent de celui effectué sur les entrées de taille $n+1$. Ce conseil peut être donné à une machine de Turing sous la forme d'un ruban additionnel sur lequel est écrit le mot-conseil correspondant à la taille du mot sur le ruban d'entrée de la machine. Voici la définition formelle d'une classe avec conseil.

Définition 1.1.1
Soit $\mathcal{C}$ une classe de complexité et $a : \mathbb{N} \to \mathbb{N}$ une fonction. La classe $\mathcal{C}/a(n)$ est l'ensemble des langages A tels qu'il existe une fonction $c : \mathbb{N} \to \{0,1\}^*$ et un langage $B \in \mathcal{C}$ satisfaisant :

- *pour tout n, $|c(n)| \leqslant a(n)$;*
- *pour tout mot x, $x \in A \iff (x, c(|x|)) \in B$.*

Si, plutôt qu'une seule fonction a, on a une famille F de fonctions, alors $\mathcal{C}/F = \cup_{a \in F} \mathcal{C}/a$.

Dans ce cadre, on peut définir l'ensemble poly des fonctions polynomialement bornées. Ainsi, on obtient par exemple P/poly (temps et conseil polynomiaux), NP/poly (temps non-déterministe et conseil polynomiaux), PSPACE/poly (espace et conseil polynomiaux), EXP/poly (temps exponentiel et conseil polynomial), etc.

Notons enfin que l'on définira une dernière classe booléenne à la section 1.3, mais nous aurons besoin pour cela de circuits. Il s'agira de la classe NC et de ses variantes selon l'uniformité des circuits.

On trouvera dans la figure 1.1 un résumé graphique illustrant les inclusions connues de quelques classes importantes (tout bon livre de complexité contient au moins un dessin de la sorte, éventuellement plus fourni). Bien sûr, pour la plupart des inclusions on ne sait pas si elles sont strictes.

1.2 Circuits

Les classes de complexité algébriques vont nécessiter l'introduction de circuits. Nous présentons deux types de circuits : booléen et arithmétique. Nous verrons dans la partie suivante que les classes booléennes peuvent également être définies en termes de circuits.

Définition 1.2.1
Un circuit est un graphe orienté sans cycle, dont les sommets, appelés portes, ont un degré entrant 0, 1 ou 2 et un degré sortant arbitraire. Une seule porte,

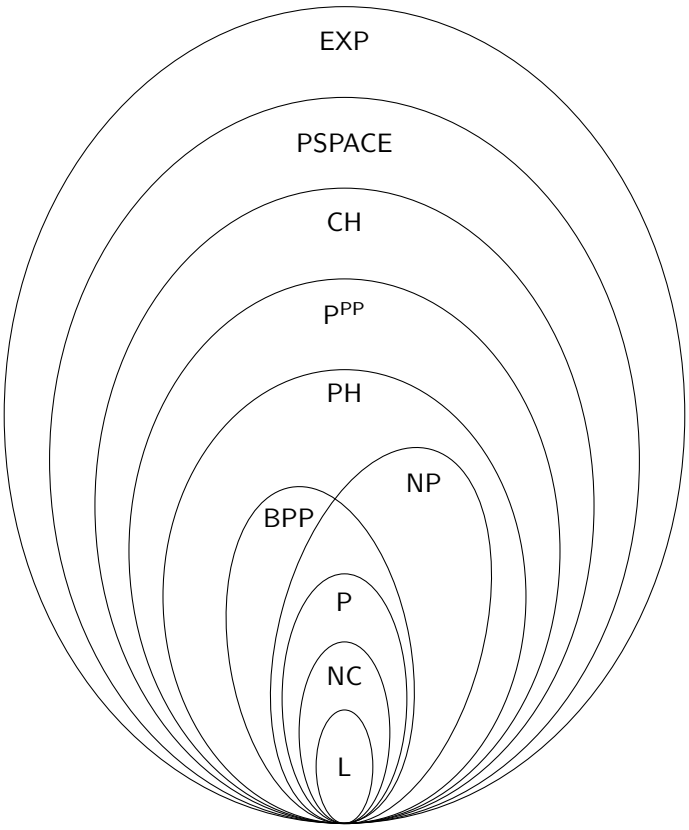


FIGURE 1.1 – Morphologie des classes de complexité : du rachitisme à l'obésité.

appelée sortie, a un degré sortant nul. Les portes de degré entrant nul sont appelées entrées. Les sommets du graphe sont étiquetés. Le choix pour les étiquettes dépend du type de circuit et du degré entrant de la porte.

Pour une assignation des variables d'entrée du circuit, on définira par induction une valeur calculée en chaque porte du circuit, selon le type de circuit. On notera $C(\bar{x})$ la valeur calculée par la sortie de C sur l'assignation $\bar{x}$. On trouvera quelques illustrations dans les figures des paragraphes qui suivent.

Quelques caractéristiques des circuits

On définit deux caractéristiques essentielles des circuits : leur taille et leur profondeur.

Définition 1.2.2
Soit C un circuit. La taille de C, notée $|C|$, est le nombre de sommets du graphe. La profondeur de C est la taille du plus long chemin d'une entrée à la sortie.

En particulier, puisque chaque porte a un degré entrant $\leqslant 2$, la taille d'un circuit est majorée par 2^{p+1} où p est la profondeur du circuit.

Un circuit est simplement un objet booléen. Pour le décrire, il suffit de donner la liste des portes, leur type et leurs fils. On remarquera que cette description est de taille polynomiale en la taille du circuit.

Circuits booléens

Dans un circuit booléen, une porte est d'un des types suivants :

- une entrée, étiquetée par une variable x_i ;
- une porte NON, de degré entrant 1 et étiquetée par $\neg$;
- une porte ET (respectivement OU), de degré entrant 2 et étiquetée par $\wedge$ (resp. $\vee$).

Les entrées x_i prennent des valeurs booléennes. La valeur des autres portes est définie récursivement : c'est la négation de la valeur de son entrée pour une porte $\neg$ et la conjonction (respectivement disjonction) des valeurs de ses entrées pour une porte $\wedge$ (resp. $\vee$). La valeur du circuit est alors la valeur de la porte de sortie.

Après avoir fixé la valeur des entrées, la valeur d'un circuit booléen est donc un élément de $\{0, 1\}$.

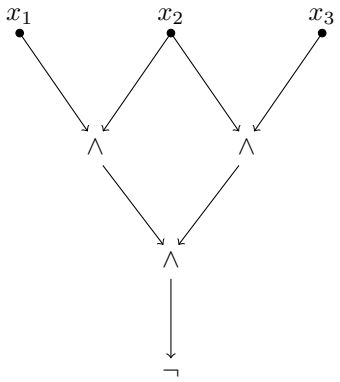


FIGURE 1.2 – Un circuit booléen de taille 7 et de profondeur 3 valant 1 si le mot $x_1x_2x_3$ en entrée contient un zéro et 0 sinon.

Circuits arithmétiques

Dans un circuit arithmétique, on se place sur un corps $\mathbb{K}$ quelconque. Une porte est alors d'un des types suivants :

- une constante, de degré entrant 0 et étiquetée par une constante $c \in \mathbb{K}$;
- une entrée, étiquetée par une variable x_i ;
- une porte d'addition (respectivement de multiplication), de degré entrant 2 et étiquetée par $+$ (resp. par $\times$).

Un tel circuit calcule un polynôme à coefficients dans $\mathbb{K}$ défini comme suit. Les portes constantes étiquetées par $c \in \mathbb{K}$ calculent le polynôme constant c. Les entrées x_i calculent le polynôme x_i. Le polynôme calculé par les autres portes est défini récursivement : c'est la somme des polynômes entrant pour une porte $+$ et leur produit pour une porte $\times$. Le polynôme calculé par le circuit est alors celui calculé par la porte de sortie.

Un circuit arithmétique sur $\mathbb{K}$ calcule donc un polynôme à plusieurs variables $x_1, \ldots, x_n$ et à coefficients dans $\mathbb{K}$. Si la seule constante autorisée est -1, on parlera de circuit sans constante : le polynôme calculé aura alors des coefficients entiers. Afin de simplifier les preuves, nous avons fait le choix de considérer des circuits sans porte de soustraction, c'est pourquoi nous devons utiliser -1 comme constante ; nous aurions pu définir des circuits avec soustraction et dans ce cas on aurait plus naturellement choisi d'autoriser la seule constante 1 dans la version sans constante.

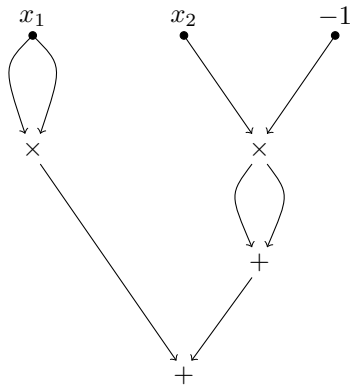


FIGURE 1.3 – Un circuit arithmétique de taille 7 et de profondeur 3 calculant le polynôme $x_1^2 - 2x_2$.

Les circuits arithmétiques sont au moins aussi puissants que les circuits booléens au sens suivant.

Lemme 1.2.3
Soit C un circuit booléen de taille t. Il existe un circuit arithmétique C' de taille $\leqslant 4t$ qui simule C dans le sens suivant : sur toute entrée booléenne $\bar{x} \in \{0,1\}^n$, on a $C'(\bar{x}) = C(\bar{x})$.
De plus, la construction de C' à partir de C se fait en espace logarithmique.

Preuve :

Il suffit de remplacer $x \wedge y$ par xy, $x \vee y$ par $x + y - xy$ et $\neg x$ par $1 - x$. Ces changements étant purement locaux, ils se font aisément en espace logarithmique. ∎

Uniformité

Les circuits sont des modèles de calcul intrinsèquement non-uniformes, c'est-à-dire qu'il y a un dispositif différent pour chaque longueur de mots d'entrée. Comme on va le voir, on peut toutefois imposer à une famille de circuits d'être uniforme, en passant par les machines de Turing (qui, elles, sont intrinsèquement uniformes). Cela donne lieu à plusieurs notions d'uniformité selon la puissance de la machine de Turing que l'on considère. Nous définissons tout cela ci-dessous et verrons plus loin que ces différentes notions sont parfois équivalentes.

Définition 1.2.4
Soient (C_n) une famille de circuits et $\mathcal{C}$ une classe de complexité. On considère le langage suivant (la description de (C_n))

$$\mathrm{DES}_C = \{(1^n, i, b) \mid \textit{le } i\textit{-ème bit de l'encodage de } C_n \textit{ est } b\}.$$

On dit que la famille (C_n) est $\mathcal{C}$-uniforme si $\mathrm{DES}_C \in \mathcal{C}$.

Les classes usuelles pour $\mathcal{C}$ dans la définition ci-dessus sont L, P et PSPACE. Sauf mention contraire, lorsqu'on parlera d'uniformité dans la suite, il s'agira d'uniformité polynomiale (c'est-à-dire de P-uniformité). On remarquera que, dans certains cas (et notamment pour PSPACE et ses versions algébriques), cela n'a guère d'importance puisque ces trois notions sont équivalentes comme nous le montrerons plus loin. Nous utiliserons brièvement aussi la P/poly-uniformité (ce qui n'est pertinent que lorsque le circuit a une taille superpolynomiale).
Remarque. Lorsque le circuit est de taille polynomiale, il est équivalent de reconnaître le langage DES_C et de construire la description du circuit. En revanche, pour un circuit de taille exponentielle, le temps requis pour la construction du circuit sera toujours exponentiel alors qu'on pourrait reconnaître DES_C en temps polynomial.

1.3 Classes booléennes, partie 2

Les classes de complexité booléenne peuvent aussi être définies à l'aide de circuits booléens. En effet, il est bien connu, depuis Karp et Lipton [39] que P/poly est l'ensemble des langages reconnus par des circuits de taille polynomiale. Nous approfondissons ces liens dans cette partie : une condition d'uniformité sur les familles de circuits booléens permet de retrouver les classes de complexité uniformes.

Langage reconnu par des circuits

Soient C un circuit booléen à n entrées et x un mot. On dit que x est accepté par C si $|x| = n$ et la valeur de C sur x, notée $C(x)$, est 1. Sinon, c'est-à-dire si $C(x) = 0$ ou $|x| \neq n$, on dit que x est rejeté par C.

Soit (C_n) une famille de circuits booléens, le circuit C_n ayant n entrées (c'est-à-dire qu'il accepte en entrée des mots de taille n). Le langage reconnu par la famille (C_n) est l'ensemble des mots reconnus par un des circuits C_n.

Temps, taille

On peut grossièrement assimiler la taille d'un circuit au temps de calcul d'un algorithme séquentiel. C'est la signification du lemme suivant. Pour la preuve, on introduit tout d'abord un langage P-complet.

Définition 1.3.1
Le langage PVC *(problème de la valeur d'un circuit) est l'ensemble des circuits booléens dont les entrées sont instanciées par 0 ou 1 et dont la valeur est 1. Plus précisément, un mot du langage* PVC *est le codage d'un circuit booléen C et d'une assignation $\bar{x}$ des variables d'entrée de C tel que $C(\bar{x}) = 1$.*

Ce problème est P-complet pour les réductions en espace logarithmique, voir Papadimitriou [55, théorème 8.1]

Lemme 1.3.2
Les trois assertions suivantes sont équivalentes.

1. *Le langage A est dans* P.
2. *Le langage A est reconnu par une famille* L*-uniforme de circuits booléens de taille polynomiale.*
3. *Le langage A est reconnu par une famille* P*-uniforme de circuits booléens de taille polynomiale.*

Preuve :

La partie la moins facile, (1) implique (2), est montrée dans Papadimitriou [55, théorème 11.5]. La preuve repose sur la P-complétude du problème PVC pour les réductions logspace : il suffit de montrer que ce problème est décidable par des circuits L-uniformes.

L'implication $(2) \Rightarrow (3)$ est triviale. Pour montrer $(3) \Rightarrow (1)$, il suffit de voir qu'en temps polynomial on peut construire le circuit P-uniforme reconnaissant A et le simuler. ■

Espace, profondeur

On peut grossièrement assimiler la profondeur d'un circuit à l'espace utilisé par un calcul (ou encore au temps de calcul d'un algorithme parallèle). Le lemme suivant précise cette idée. Par ailleurs, il est intéressant de constater que les trois notions d'uniformité L, P et PSPACE coïncident pour les circuits de profondeur polynomiale : la raison en est que l'on a des problèmes très simples (donc décidés par des circuits simples) qui sont PSPACE-complets pour les réductions logspace.

Pour la preuve, nous aurons besoin du langage suivant.

Définition 1.3.3
Le langage QBF *(quantified boolean formula) est l'ensemble des formules vraies de la forme $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$, où ϕ est une formule booléenne du premier ordre sans quantificateur et Q est un quantificateur dépendant de la parité de n.*

L'intérêt de ce langage est qu'il est PSPACE-complet pour les réductions log-space. On pourra voir pour cela le livre de Papadimitriou [55].

Lemme 1.3.4
Les quatre assertions suivantes sont équivalentes.

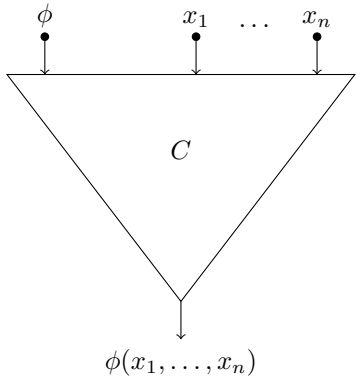
1. *Le langage A est dans* PSPACE.
2. *Le langage A est reconnu par une famille* L*-uniforme de circuits booléens de profondeur polynomiale.*
3. *Le langage A est reconnu par une famille* P*-uniforme de circuits booléens de profondeur polynomiale.*
4. *Le langage A est reconnu par une famille* PSPACE*-uniforme de circuits booléens de profondeur polynomiale.*

Preuve :

Voici le plan de la preuve : nous montrons qu'un langage $A \in$ PSPACE est reconnu par une famille logspace-uniforme de circuits booléens de profondeur polynomiale, donc *a fortiori* par une famille P-uniforme ou PSPACE-uniforme ; puis nous montrons qu'une famille PSPACE-uniforme peut être simulée par un algorithme fonctionnant en espace polynomial.

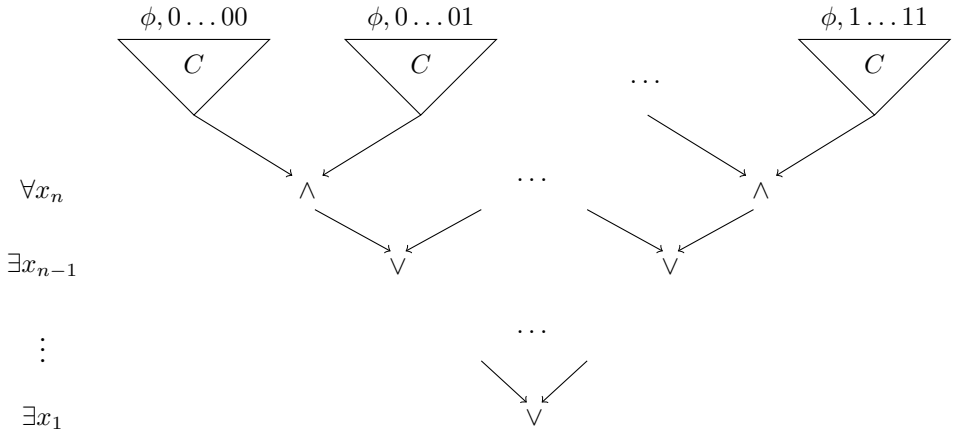
Pour la première partie, il suffit de fournir une famille L-uniforme de circuits pour le problème QBF qui est PSPACE-complet pour les réductions logspace. Quitte à ajouter des variables inutiles, on peut supposer que toute entrée de taille $2n$ a n variables. Ainsi, une entrée de taille $2n$ est de la forme $\exists x_1 \forall x_2 \exists x_3 \dots Q x_n \phi(x_1, \dots, x_n)$, où x_i prend ses valeurs dans $\{0, 1\}$. Enfin, on peut supposer que ϕ est donnée sous forme d'arbre (dont les sommets sont étiquetés par $\wedge$, $\vee$ et $\neg$).

Si les variables x_i sont fixées, alors on peut évaluer $\phi(x_1, \dots, x_n)$ simplement en parcourant l'arbre. Cela se fait par un circuit C logspace-uniforme comme sur la figure ci-dessous.



Maintenant, on peut construire le circuit entier, en partant de la sortie : on met d'abord un arbre binaire complet, de profondeur n, le i-ème niveau correspondant à la variable x_i. Les sommets du niveau i sont donc étiquetés par $\vee$ si le quantificateur correspondant est existentiel et par $\wedge$ s'il est

universel. Puis on met au-dessus le circuit C pour évaluer $\phi(x_1, \ldots, x_n)$, en remplaçant $x_1, \ldots, x_n$ par les valeurs correspondant à leur emplacement. Tout cela est illustré sur la figure ci-dessous. On se convainc aisément que cette construction se fait en espace logarithmique.



Il reste maintenant le second point du lemme, à savoir, montrer qu'en espace polynomial on peut évaluer une famille PSPACE-uniforme de circuits booléens de profondeur polynomiale. Encore une fois, on part de la sortie : il suffit d'évaluer tour à tour les deux sous-circuits de cette porte (ou l'unique sous-circuit s'il s'agit d'une porte $\neg$). On connaît cette porte ainsi que ses fils par la description PSPACE du circuit. Pour évaluer la valeur de la porte, on le fait par induction : on évalue d'abord le fils gauche, on garde le résultat en effaçant l'espace de calcul, puis le fils droit. On efface l'espace de calcul une nouvelle fois et on effectue l'opération sur les deux résultats. L'espace utilisé est donc une constante plus le maximum de l'espace utilisé pour le fils droit et pour le fils gauche. En tout, on a utilisé un espace polynomial car la profondeur est polynomiale, ce qui prouve le lemme. ■

Il est facile de modifier la preuve précédente pour obtenir le résultat suivant.

Lemme 1.3.5
Un langage A est dans PSPACE/poly *si et seulement s'il est reconnu par une famille* P/poly*-uniforme de circuits booléens de profondeur polynomiale.*

Une dernière classe

Nous définissons enfin une dernière classe booléenne, NC. C'est la réunion des différentes classes NC^i pour $i \geqslant 1$. Nous présentons d'abord la version non-uniforme ; bien qu'il s'agisse d'une classe définie par des circuits et non par des conseils, pour souligner sa non-uniformité nous utiliserons la notation NC/poly.

Pour $i \geqslant 1$, la classe NC^i/poly est l'ensemble des langages reconnus par des circuits booléens de taille polynomiale et de profondeur $O(\log^i n)$. La classe NC/poly est alors $\cup_{i \geqslant 1} \mathsf{NC}^i$/poly, ou en d'autres termes, l'ensemble des langages

reconnus par des circuits booléens de taille polynomiale et de profondeur polylogarithmique.

On définit également des versions uniformes : NC est la version L-uniforme de NC/poly et P-uniform NC est la version P-uniforme, comme on l'aura deviné. On ne sait pas si ces deux versions coïncident. Bien entendu les classes NC^i pour $i \geqslant 1$ sont les versions L-uniformes des $\mathsf{NC}^i/\mathsf{poly}$.

1.4 Modèle de Valiant

Dans le modèle de Valiant, on calcule des polynômes. Les éléments des classes de complexité sont des familles de polynômes (qui remplacent en quelque sorte les langages des classes booléennes). Nous donnons ici la définition des classes principales que nous utiliserons par la suite ; on pourra se référer au livre de Bürgisser [15] ou à la thèse de Malod [52] pour approfondir le sujet.

Calcul de polynômes

On s'intéresse à des suites de polynômes, calculées par des suites de circuits arithmétiques.

Définition 1.4.1
Soient $\mathbb{K}$ un corps quelconque, (f_n) une famille de polynômes à plusieurs variables, $f_n \in \mathbb{K}[x_1, \ldots, x_{u(n)}]$, et (C_n) une famille de circuits arithmétiques dont les constantes éventuelles sont dans $\mathbb{K}$. On dit que la famille de polynômes (f_n) est calculée par la famille de circuits (C_n) si pour tout n, le circuit C_n calcule le polynôme f_n.

Classes de Valiant

Les deux classes originales, VP et VNP, sont définies dans Valiant [67]. Ce sont des classes de familles non-uniformes de polynômes, de degré polynomialement borné et utilisant des constantes arbitraires.

En essence, la classe VP est l'ensemble des familles de polynômes calculables par des circuits de taille polynomiale (ce qui correspond aux familles de polynômes « facilement calculables »), et VNP consiste en une somme exponentielle de familles VP (ce qui correspond aux familles de polynômes « facilement définissables »). Une définition plus précise est donnée ci-dessous. La plupart du temps, nous abrégerons $x_1, \ldots, x_{u(n)}$ en $\bar{x}$.

Définition 1.4.2
Soit $\mathbb{K}$ un corps quelconque.

- *La classe* VP *est l'ensemble des familles de polynômes (f_n) de degré polynomialement borné, $f_n \in \mathbb{K}[x_1, \ldots, x_{u(n)}]$, calculées par une famille de circuits de taille polynomiale.*
- *Une famille $(f_n(\bar{x}))$ est dans la classe* VNP *s'il existe une famille $(g_n(\bar{x}, \bar{y})) \in$ VP telle que*

$$f_n(\bar{x}) = \sum_{\bar{\varepsilon} \in \{0,1\}^{|\bar{y}|}} g_n(\bar{x}, \bar{\varepsilon}).$$

La restriction sur le degré des polynômes implique par exemple que la famille (X^{2^n}) n'est pas dans VP alors que chaque polynôme X^{2^n} peut être calculé à l'aide de n portes de multiplication.

Il est difficile de justifier *a priori* la définition de la classe VNP et notamment la présence de cette somme exponentielle. On constate plutôt, par les analogies avec les résultats de complexité booléenne, que cette somme exponentielle joue, en quelque sorte, le rôle d'un quantificateur existentiel. Avant de voir les variantes annoncées de ces classes, nous introduisons la notion de VNP-complétude de Valiant [67]. Nous avons d'abord besoin d'une notion de réduction.

Définition 1.4.3
Soit $\mathbb{K}$ un corps quelconque.

- *Un polynôme f est une* projection *d'un polynôme g si $f(x_1, \ldots, x_n) = g(a_1, \ldots, a_m)$, où les a_i sont des éléments de $\mathbb{K}$ ou des variables parmi $x_1, \ldots, x_n$.*
- *Soient (f_n) et (g_n) deux familles de polynômes. On dit que (g_n) est une p-projection de (f_n) s'il existe une fonction $t(n)$ polynomialement bornée telle que pour tout n, g_n soit une projection de $f_{t(n)}$.*
- *Soit (f_n) une famille de polynômes. On dit que (f_n) est* VNP*-complète si $(f_n) \in$ VNP *et si toute famille $(g_n) \in$ VNP *est une p-projection de (f_n).*

Un des résultats célèbres de Valiant est la VNP-complétude du permanent sur tout corps de caractéristique différente de 2. Le permanent est la famille (per_n) définie comme suit :

$$\mathrm{per}_n(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} x_{i,\sigma(i)},$$

où la somme est prise sur toutes les permutations σ de $\{1, \ldots, n\}$. Malgré la similarité avec le déterminant (la différence étant seulement que la signature de la permutation est prise en compte dans ce dernier), ce résultat de complétude, avec celui de #P-complétude dans un cadre booléen, semble indiquer que le permanent est difficile à calculer. Une autre famille VNP-complète, sur tout corps $\mathbb{K}$ cette fois, est le hamiltonien, où la somme est prise seulement sur les n-cycles σ. Nous retrouverons ces polynômes et leur lien avec les couvertures de graphes dans les chapitres suivants.

Classes de Valiant-Malod

Dans ce paragraphe et le suivant, nous présentons plusieurs variantes des classes de Valiant, selon l'uniformité, le degré et l'utilisation de constantes. Dans un cadre non-uniforme, ces variantes sont issues de Malod [52].

Malod a introduit une version de VP et VNP où le degré des polynômes n'est plus restreint : il peut être exponentiel, la seule contrainte étant la taille polynomiale du circuit. Il s'agit donc exactement de la définition 1.4.2 sans la contrainte sur le degré. On notera ces classes $\mathsf{VP}_{\mathsf{nb}}$ et $\mathsf{VNP}_{\mathsf{nb}}$, où nb signifie *non borné*.

Des deux classes en degré non borné, on définit aisément des versions sans constante, en imposant que l'unique constante des circuits soit -1. On obtient ainsi les classes $\mathsf{VP}^0_{\mathsf{nb}}$ et $\mathsf{VNP}^0_{\mathsf{nb}}$. En revanche, les versions sans constante des classes de degré borné sont un peu plus délicates à définir si l'on veut éviter que des constantes de taille exponentielle ne puissent être calculées. Elles reposent sur la notion de degré d'un circuit, également appelé degré formel complet par Malod [52] (aussi dans [54]), que l'on définit maintenant.

Définition 1.4.4
Soit C un circuit arithmétique. Le degré *d'une porte de C est défini par induction : le degré d'une constante ou d'une variable est 1 ; celui d'une porte $+$ est le maximum des degrés de ses entrées ; celui d'une porte $\times$ est la somme des degrés de ses entrées.*
Le degré de C, noté $\deg_{\mathrm{f}}(C)$, est alors le degré de la porte de sortie de C.

Voici donc la définition des versions sans constante de VP et VNP.

Définition 1.4.5
- *Une famille de polynômes (f_n) appartient à la classe VP^0 si (f_n) est calculée par une famille de circuits arithmétiques sans constante de taille et de degré polynomialement bornés.*

- *Une famille $(f_n(\bar{x}))$ est dans la classe VNP^0 s'il existe une famille $(g_n(\bar{x},\bar{y})) \in \mathsf{VP}^0$ telle que*

$$f_n(\bar{x}) = \sum_{\bar{\varepsilon}\in\{0,1\}^{|\bar{y}|}} g_n(\bar{x},\bar{\varepsilon}).$$

Uniformité

Des quatre classes sans constante, on définit aisément des versions uniformes, en imposant une condition d'uniformité polynomiale sur les circuits. On obtient ainsi les classes Uniform VP^0, Uniform VNP^0, Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ et Uniform $\mathsf{VNP}^0_{\mathsf{nb}}$.
Remarque. Dans les classes avec constantes, une notion d'uniformité a moins de sens car les constantes peuvent être arbitraires, c'est pourquoi nous ne définirons pas de variantes uniformes des classes avec constantes.

Le lemme suivant montre que considérer l'uniformité L ou polynomiale ne change rien pour les classes en degré non borné. La preuve repose sur la P-complétude pour la réduction logspace d'un problème très uniforme.

Lemme 1.4.6
Les classes Uniform $\mathsf{VP}^0_{\mathsf{nb}}$ *et* Uniform $\mathsf{VNP}^0_{\mathsf{nb}}$ *sont définies de manière équivalente en considérant l'uniformité* P *ou l'uniformité* L.

Preuve :

Par définition de $\mathsf{VNP}^0_{\mathsf{nb}}$, il suffit de montrer le résultat pour $\mathsf{VP}^0_{\mathsf{nb}}$.

Soit (C_n) une famille P-uniforme de circuits de taille polynomiale, calculant une famille de polynômes (f_n). On veut construire une famille L-uniforme de circuits (D_n) de taille polynomiale calculant les mêmes polynômes. L'idée est la suivante : D_n construit C_n au fur et à mesure en utilisant sa description, tout en le simulant. Afin de rester L-uniforme, on utilisera le lemme 1.3.2.

Puisque la description de (C_n) est dans P, on peut appliquer le lemme 1.3.2 : soit (B_n) la famille L-uniforme de circuits booléens de taille polynomiale qui décrit (C_n). Ainsi, le circuit B_n nous dit le type t de la porte i de C_n (étiquetée par $\circ$) : par exemple $t = 0$ si $\circ = +$ et $t = 1$ si $\circ = \times$. Pour calculer $x \circ y$, il suffit alors de calculer $txy + (1-t)(x+y)$. Or par le lemme 1.2.3, on peut simuler le circuit booléen B_n par un circuit arithmétique B'_n en gardant l'uniformité logspace.

Ainsi, on remplace chaque porte de C_n par une copie de B'_n avec pour entrée le code binaire correspondant au numéro de la porte ; à la sortie de B'_n on a le type de la porte et on applique ce qui précède pour effectuer le bon calcul de C_n. ∎

Remarque. En revanche, la preuve ne fonctionne plus pour les versions en degré borné, car on ne peut alors pas simuler n'importe quel circuit booléen de taille polynomiale à cause de la contrainte du degré. Il serait intéressant de savoir si l'uniformité polynomiale est strictement plus puissante que l'uniformité logspace pour VP^0 et VNP^0.

Sous-classes de $\mathsf{VP_{nb}}$

Comme pour P dans le cadre de la complexité booléenne, on peut définir des sous-classes de $\mathsf{VP_{nb}}$ dans le modèle de Valiant en imposant des restrictions sur les circuits employés.

Le premier type de restrictions porte sur la profondeur des circuits. Nous définissons ainsi VNC un analogue à la classe booléenne NC. C'est la réunion des différentes classes VNC^i pour $i \geqslant 1$.

Pour $i \geqslant 1$, la classe VNC^i est l'ensemble des familles de polynômes de degré polynomial, calculées par des circuits arithmétiques de taille polynomiale et de profondeur $O(\log^i n)$. La classe VNC est alors $\cup_{i \geqslant 1} \mathsf{VNC}^i$, ou en d'autres termes, l'ensemble des familles de polynômes de degré polynomial, calculées par des circuits arithmétiques de taille polynomiale et de profondeur polylogarithmique.

Le second type de restrictions porte sur la structure des circuits. On peut ainsi imposer que le graphe sous-jacent au circuit soit un arbre. Ou de manière équivalente que toutes les portes, sauf la sortie, soient de degré sortant 1. On obtient alors la classe $\mathsf{VP_e}$ constituée des familles de polynômes calculées par des formules (identiquement termes ou expressions) arithmétiques de taille polynomiale.

On peut exprimer différemment cette restriction à l'aide des notions suivantes :

Définition 1.4.7
Soient C un circuit et g une porte de C. On appelle sous-circuit *de C associé à g l'ensemble des portes g' de C telles qu'il existe un chemin orienté de g' à g dans*

C. Intuitivement, c'est la partie du circuit qui intervient dans la valeur calculée par g. Si g est de degré entrant 2 et g', g'' sont ses deux voisins intérieurs, alors on appelle sous-circuit gauche de g le sous-circuit de C associé à g' et sous-circuit droit de g le sous-circuit de C associé à g''.

La classe $\mathsf{VP_e}$ peut alors s'exprimer comme l'ensemble des familles de polynômes calculées par des circuits arithmétiques de taille polynomiale tels que pour toute porte g de degré entrant 2 :

- les sous-circuits gauche et droit de g sont disjoints ;
- les seuls arcs sortant de ces sous-circuits ont g comme extrémité.

On peut représenter ces contraintes par les 3 « murs » sur le schéma suivant (un mur ne peut être traversé par un arc du circuit) :

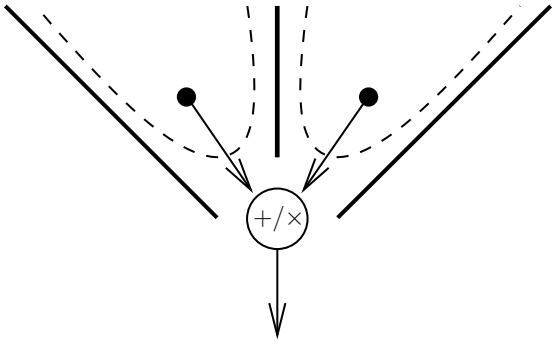


FIGURE 1.4 – Formule arithmétique.

Bien entendu, si on retire tous les « murs », on obtient la classe $\mathsf{VP_{nb}}$. Si l'on retire tous les murs sur les portes d'addition et que l'on garde uniquement le mur du milieu sur les portes de multiplication, on obtient ce qu'on appelle un circuit multiplicativement disjoint.

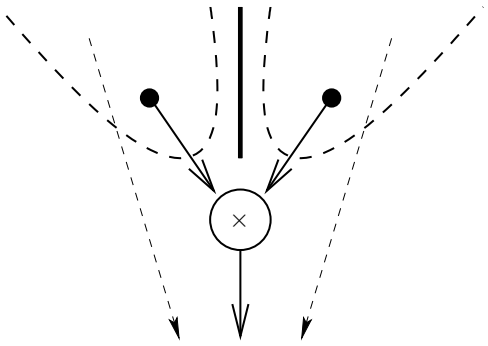


FIGURE 1.5 – Circuit multiplicativement disjoint.

En voici la définition formelle :

Définition 1.4.8 (Circuit multiplicativement disjoint)
Une porte g est dite disjointe *si les sous-circuits gauche et droit de g sont disjoints. Un circuit C est dit* multiplicativement disjoint *si toute porte de multiplication est disjointe.*

Cette notion de circuit multiplicativement disjoint est introduite par Malod dans [52]. Il y montre que la classe des familles de polynômes calculées par des familles de circuits multiplicativement disjoints de taille polynomiale est exactement VP. Ce résultat montre que l'on peut remplacer la contrainte de polynomialité du degré par une contrainte sur la structure du circuit.

Il reste à présent deux dernières restrictions sur les circuits à considérer.

Définition 1.4.9 (Circuit asymétrique et faiblement asymétrique)
Une porte g est dite asymétrique *si l'un de ses sous-circuits est réduit à une entrée ou une constante. Elle est dite* faiblement asymétrique *si l'un de ses sous-circuits est déconnecté du reste du circuit. Un circuit C est dit* asymétrique *si toute porte de multiplication est asymétrique. Il est dit* faiblement asymétrique *si toute porte de multiplication est faiblement asymétrique.*

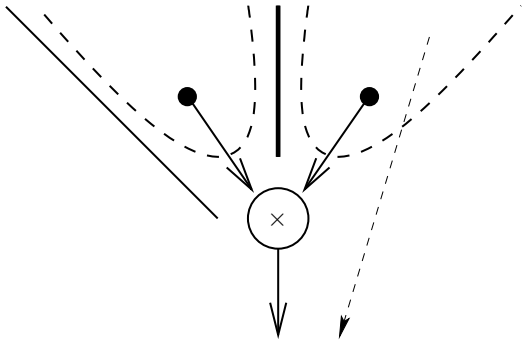


FIGURE 1.6 – Circuit faiblement asymétrique.

Ces deux classes de circuits ont été introduites par Toda dans [66]. Dans cet article, il montre que les classes des familles de polynômes calculées par des circuits asymétriques ou faiblement asymétriques de taille polynomiale coïncident en une même classe nommée VDET. Cette appellation découle du fait que cette classe capture exactement la complexité du déterminant puisque celui-ci est complet pour VDET.

Toutes ces classes satisfont les inclusions et égalités suivantes :

$$\mathsf{VP_e} = \mathsf{VNC}^1 \subseteq \mathsf{VDET} \subseteq \mathsf{VP} = \mathsf{VNC}^2 \subset \mathsf{VP_{nb}}.$$

On peut définir la classe $\mathsf{VNP_e}$ de manière analogue à VNP.

Définition 1.4.10
Soit $\mathbb{K}$ un corps quelconque. Une famille $(f_n(\bar{x}))$ est dans la classe $\mathsf{VNP_e}$ s'il existe une famille $(g_n(\bar{x},\bar{y})) \in \mathsf{VP_e}$ telle que

$$f_n(\bar{x}) = \sum_{\bar{\varepsilon}\in\{0,1\}^{|\bar{y}|}} g_n(\bar{x},\bar{\varepsilon}).$$

L'une des deux étapes de la preuve de VNP-complétude du permanent dans [68] consiste à montrer que $\mathsf{VNP}_e = \mathsf{VNP}$ (l'autre étape étant de montrer que le permanent est VNP_e-complet). Une preuve plus simple basée sur la notion de circuits multiplicativement disjoints est donnée dans [52].

Terminons ce chapitre sur un dessin des inclusions des classes de complexité algébrique les plus significatives.

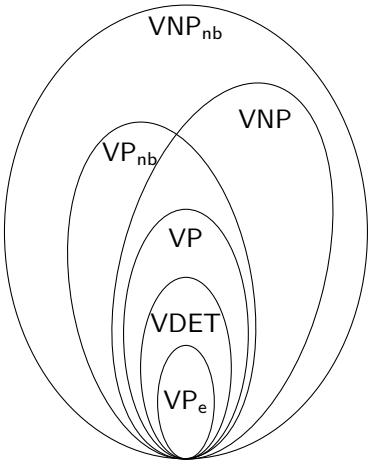


FIGURE 1.7 – Quelques classes de complexité du modèle de Valiant.

Chapitre 2

Largeurs et couvertures de graphes

Ce chapitre est le second chapitre préliminaire à la première partie de ce document. Nous y rappellerons les définitions standard de largeur arborescente, largeur linéaire et largeur de clique, ainsi que des équivalents à la largeur de clique. Nous introduirons aussi des variantes pondérées de la largeur de clique et de ses équivalents. Enfin nous présenterons les trois types de couvertures de graphes étudiés par la suite, ainsi que leur lien avec les classes de complexité introduites au chapitre précédent. J'emploierai ultérieurement le terme d'arête quand il s'agira de parler à la fois de graphes non-orientés et orientés.

2.1 Largeurs linéaire et arborescente

Nous rappelons à présent différentes définitions des largeur linéaire et arborescente. Ces deux largeurs sont le plus souvent définies comme suit :

Définition 2.1.1 (décomposition arborescente, largeur arborescente)

Une décomposition arborescente *d'un graphe $G = (V, E)$ est un couple $(T = (I, E_T), \{X_i, i \in I\})$ où T est un arbre et les X_i sont des ensembles de sommets de G qui étiquettent les sommets de T tels que :*

- $\bigcup_{i \in I} X_i = V$;
- $\forall xy \in E, \exists i \in I$ *tel que* $x, y \in X_i$;
- $\forall x \in V$, *le sous-graphe induit par l'ensemble T_x des sommets $i \in I$ tels que $x \in X_i$ est un sous-arbre de T.*

La largeur d'une décomposition est le $\max_{i \in I} |X_i| - 1$. *La* largeur arborescente *d'un graphe, notée $tw(G)$, est le minimum des largeurs de ses décompositions arborescentes.*

Remarque. On appellera sacs de la décomposition les ensembles X_i de sommets de G.

Définition 2.1.2 (décomposition linéaire, largeur linéaire)

Une décomposition linéaire *est une décomposition arborescente où l'arbre T est une chaîne. La* largeur linéaire *d'un graphe, notée $pw(G)$, est le minimum des largeurs de ses décompositions linéaires.*

Le -1 dans la formule de la largeur a été introduit pour que la largeur arborescente d'un arbre ou d'une forêt et la largeur linéaire d'un chemin soient égales à 1.
Exemple. La figure ci-dessous montre un graphe G_{ex} de largeurs arborescente et linéaire 2 ainsi qu'une décomposition arborescente optimale et une décomposition linéaire de ce graphe.

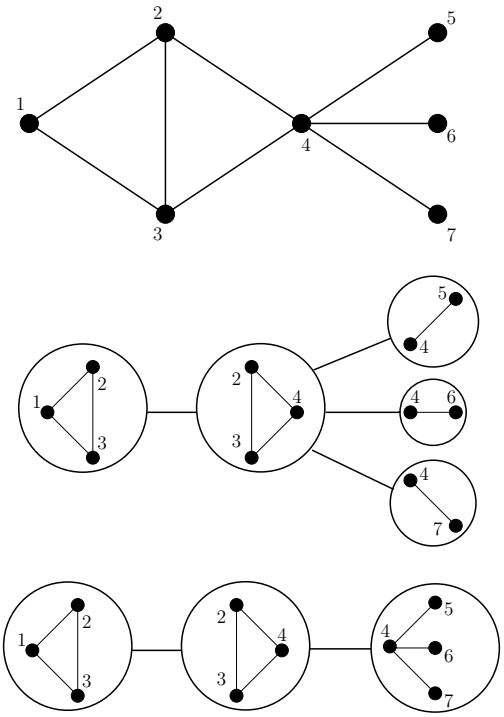


FIGURE 2.1 – Un graphe et ses décompositions arborescente et linéaire.

Exemple. Montrons que les cycles sont de largeur linéaire au plus 2 en construisant une décomposition linéaire de C où chaque sac X_t est de taille au plus 3. Soient $v_1, v_2, \ldots, v_n$ les sommets de C. Les arêtes de C sont $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1)$. On place le sommet v_1 dans tous les sacs de la décomposition linéaire. Les sommets v_2 et v_3 sont contenus dans X_1, v_3 et v_4 dans X_2 et ainsi de suite. Finalement, v_{n-1} et v_n sont contenus dans X_{n-2}. On obtient ainsi une décomposition linéaire de largeur 2.

Nous allons à présent donner une définition équivalente à l'aide d'expressions algébriques ou termes, en nous inspirant de la présentation faite par Courcelle dans [22]. Pour cela, rappelons succinctement quelques définitons d'algèbre universelle. Une *signature fonctionnelle* est un couple formé

d'un ensemble fini ou dénombrable F de *symboles de fonctions* et d'une fonction $\alpha : F \to \mathbb{N}$ qui associe à chacun d'eux une *arité*. Un symbole d'arité 0 est une *constante*. Une *F-algèbre* est un objet $\mathbf{M} = (M, (f_M)_{f \in F})$ dont M est le domaine et dont, pour chaque $f \in F$, la composante f_M est une fonction totale $M^{\alpha(f)} \to M$. Chaque constante désigne un élément de M. Par la suite, on considérera différentes algèbres mais le domaine M sera invariablement un sous-ensemble des *graphes avec sources*. Un tel graphe comporte des sommets distingués qu'on appelle sources. Chacune de ces sources est munie d'une étiquette. On notera $\mathcal{T}(F)$ l'ensemble des termes finis écrits avec ces symboles et bien formés relativement à leurs arités. Tout terme $t \in \mathcal{T}(F)$ a une valeur dans le domaine M correspondant à son évaluation dans la F-algèbre $\mathbf{M}$. À chaque terme, on peut associer de manière bijective son *arbre syntaxique*.

Les graphes de largeur arborescente au plus k peuvent être définis à l'aide d'algèbres HR (Hyperedge Replacement) ([21]) :

Définition 2.1.3
Un graphe G est de largeur arborescente au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité $k+1$ tel que G peut être construit au moyen d'un nombre fini des opérations suivantes :

(i) ver_a, loop_a, edge_{ab}, $a, b \in \mathcal{S}$ *(Constructeurs de base : création d'un sommet source isolé étiqueté par a, création d'un sommet source isolé muni d'une boucle et étiqueté par a, création de deux sommets sources étiquetés par a et b et reliés par une arête) ;*

(ii) $\mathrm{ren}_{a \leftrightarrow b}(H)$, $a, b \in \mathcal{S}$ *(assigne l'étiquette b aux sources étiquetées par a et inversement) ;*

(iii) $\mathrm{forg}_a(H)$, $a \in \mathcal{S}$ *(oublie les sources étiquetées par a qui deviennent des sommets normaux non-étiquetés) ;*

(iv) $H \,//\, H'$ *(composition parallèle de deux graphes : union des graphes H et H' où deux sources avec la même étiquette sont fusionnées en un unique sommet).*

L'opération de composition parallèle $//$ est associative ; on peut donc écrire $H_1 \,//\, H_2 \,//\, \ldots \,//\, H_k$ sans parenthèse.

Les graphes de largeur linéaire au plus k peuvent être définis au moyen de la même algèbre, en se restreignant aux termes t vérifiant la propriété suivante : dans tout sous-terme de t de la forme $t_1 // \ldots // t_q$ au plus un des termes $t_1, \ldots, t_q$ n'est pas réduit à une constante (identiquement un constructeur de base). Le lecteur en trouvera une preuve dans [22].

Exemple. Le terme

$$\begin{array}{l}\mathrm{forg}_c[\mathrm{forg}_b[\\ \quad \mathrm{forg}_a[\mathrm{edge}_{ab} \,//\, \mathrm{edge}_{ac} \,//\, \mathrm{edge}_{bc}] \,//\, \mathrm{edge}_{ab} \,//\, \mathrm{edge}_{ac}]]\\ \quad //\\ \mathrm{forg}_b[\mathrm{edge}_{ab} \,//\, \mathrm{edge}_{ac}] \,//\, \mathrm{edge}_{ab}\end{array}$$

permet de construire le graphe G_{ex} avec 3 étiquettes. Son arbre syntaxique et les sous-graphes construits sont représentés dans la figure 2.2. Pour des raisons de clarté, on ne représentera pas les noeuds forg ; on se contentera de mettre entre parenthèses les étiquettes des sommets qui ne seront plus des sources lors de la prochaine composition parallèle.

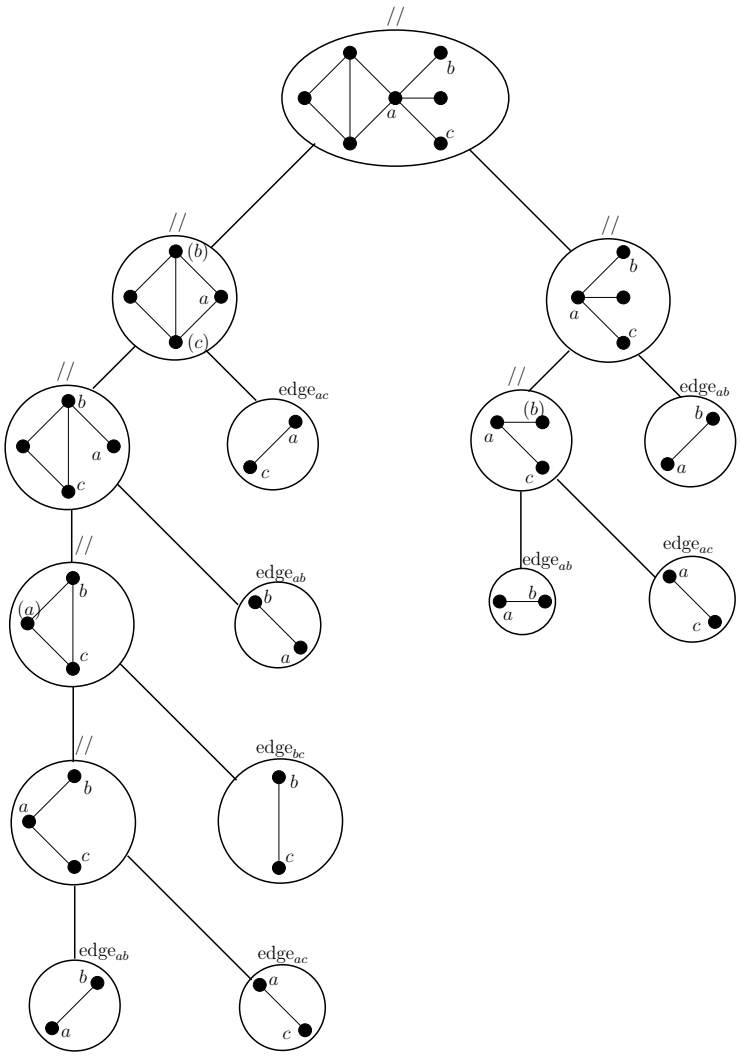


FIGURE 2.2 – Un arbre syntaxique pour construire G_{ex}.

On peut aisément étendre cette algèbre aux graphes orientés pondérés en remplaçant les constructeurs de base loop_a et edge_{ab} par loop_a^w et arc_{ab}^w. loop_a^w crée un sommet isolé muni d'une boucle de poids w et d'une étiquette a. arc_{ab}^w crée deux sommets étiquetés a et b reliés par un arc de poids w de a vers b. On définit la largeur arborescente pondérée d'un graphe orienté pondéré G, notée $Wtw(G)$, comme le nombre minimum d'étiquettes moins un tel que G peut être construit à l'aide de ces opérations. On définit de même la largeur linéaire pondérée, notée $Wpw(G)$, en appliquant à nouveau la restriction sur les termes considérés. Il est facile alors de démontrer le résultat suivant.

Proposition 2.1.4
Soit G un graphe orienté et pondéré. Soit G' le graphe non-orienté et non-pondéré sous-jacent. Alors $Wtw(G) = tw(G')$ et $Wpw(G) = pw(G')$.

Pour cette raison, on négligera les notions de largeur linéaire et arborescente pondérées. La largeur linéaire (resp. arborescente) d'un graphe orienté pondéré sera définie naturellement comme la largeur linéaire (resp. arborescente) du graphe non-orienté et non-pondéré sous-jacent.

2.2 Largeurs de clique, NLC et MC

Nous rappelons à présent les définitions des largeurs de clique, NLC (Node Label Controlled) et MC (M-Clique) d'un graphe non-pondéré introduites dans [23, 25], [73] et [26] respectivement. Il m'arrivera de parler des largeurs de clique ou des algèbres de clique pour désigner ces trois largeurs et algèbres simultanément. Nous introduirons par la suite des variantes pondérées de ces trois largeurs.

L'algèbre de clique permet de construire des graphes orientés mais nous omettons cette possibilité pour le moment.

Définition 2.2.1
Un graphe G est de largeur de clique, notée $cw(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées clique-opérations) :

(i) ver_a, $a \in \mathcal{S}$ *(Constructeur de base : crée un sommet source isolé d'étiquette a) ;*

(ii) $\rho_{a\to b}(H)$, $a, b \in \mathcal{S}$ *(renomme les sources d'étiquette a en sources d'étiquette b) ;*

(iii) $\eta_{a,b}(H)$, $a, b \in \mathcal{S}$, $a \neq b$ *(ajoute une arête entre tout couple de sommets dont l'un est étiqueté a et l'autre est étiqueté b) ;*

(iv) $H \oplus H'$ *(union disjointe des graphes H et H').*

L'algèbre NLC permet de construire des graphes non-orientés de manière très similaire à l'algèbre de clique ; la différence principale est qu'une arête entre deux sommets ne peut être ajoutée qu'au niveau de l'opération qui regroupe ses deux extrémités dans l'arbre syntaxique.

Définition 2.2.2
Un graphe G est de largeur NLC *, notée $w_{\mathrm{NLC}}(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées opérations* NLC*) :*

(i) ver_a, $a \in \mathcal{S}$ *(Constructeur de base : crée un sommet source isolé d'étiquette a) ;*

(ii) $\circ_R(H)$ *pour toute application R de $\mathcal{S}$ dans $\mathcal{S}$ (pour toute étiquette $a \in \mathcal{S}$, ré-étiqueter les sommets d'étiquette a avec l'étiquette $R(a)$) ;*

(iii) $H \times_S H'$ *pour toute relation binaire $S \subseteq \mathcal{S}^2$ (union disjointe des graphes H et H' à laquelle sont ajoutées les arêtes entre tous les couples de sommets $x \in H$ (d'étiquette l_x), $y \in H'$ (d'étiquette l_y) tels que $(l_x, l_y) \in S$).*

Une différence importante entre les algèbres de clique et NLC, d'une part, et l'algèbre MC , d'autre part, est que les deux premières construisent des graphes où les sommets ont exactement une étiquette, alors que la dernière construit des graphes où chaque sommet se voit assigné un ensemble (potentiellement vide) d'étiquettes.

Définition 2.2.3
Un graphe G est de largeur MC *, notée $w_{\mathrm{MC}}(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées opérations* MC*) :*

(i) ver_A *(Constructeur de base : crée un sommet source isolé ayant un ensemble d'étiquettes A, $A \subseteq \mathcal{S}$) ;*

(ii) $H \otimes_{S,h,h'} H'$ *pour toute relation binaire $S \subseteq \mathcal{S}^2$ et toutes applications $h, h' : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ (union disjointe des graphes H et H' à laquelle sont ajoutées les arêtes entre tous les couples de sommets $x \in H$ (d'ensemble d'étiquettes L_x), $y \in H'$ (d'ensemble d'étiquettes L_y) tels que $l_x \in L_x$, $l_y \in L_y$ et $(l_x, l_y) \in S$; les étiquettes des sommets de H sont ensuite modifiées via h et celles des sommets de H' sont modifiées via h').*

Les inégalités qui suivent entre ces différentes largeurs sont énoncées dans [26] (L'une d'elles est démontrée dans [78]) :

$$w_{\mathrm{MC}}(G) \leqslant w_{\mathrm{NLC}}(G) \leqslant cw(G) \leqslant 2^{w_{\mathrm{MC}}(G)+1} - 1.$$

En conséquence, les largeurs de clique, NLC et MC sont équivalentes lorsqu'il s'agit de caractériser des classes de graphes de largeur bornée.

Nous avons vu qu'il n'était pas nécessaire d'étendre l'algèbre HR pour donner une définition raisonnable de la largeur arborescente des graphes pondérés, la largeur arborescente du graphe non-pondéré sous-jacent étant une définition équivalente. Néanmoins, ce résultat est fortement dépendant du fait que, dans l'algèbre HR, les arêtes sont ajoutées une à une. C'est une différence majeure avec les algèbres de clique puisque celles-ci ajoutent des arêtes par « blocs de bipartis complets » . On peut aussi remarquer que si l'on considère les non-arêtes comme des arêtes de poids 0 alors tout graphe pondéré a pour graphe non-pondéré sous-jacent une clique (qui est de largeur de clique 2). Pour ces raisons et d'autres qui apparaîtront dans l'usage fait ultérieurement de ces algèbres, il n'est pas raisonnable de définir les largeurs de clique pondérées d'un tel graphe comme les largeurs de clique du graphe non-pondéré sous-jacent. En conséquence, nous ajoutons une condition d'uniformité sur la manière dont les arêtes sont ajoutées : les arêtes créées lors d'une même opération entre tous les couples de sommets ayant un même couple d'étiquettes auront le même poids.

Nous introduisons à présent les définitions des algèbres et largeurs de clique pondérée, W-NLC et W-MC. Ces algèbres permettent de construire des graphes simples pondérés mixtes (contenant à la fois des arcs et des arêtes) où les poids sont à valeur dans un ensemble W. Néanmoins, nous n'utiliserons pas la possibilité d'avoir en même temps des arcs et des arêtes. Dans les trois algèbres suivantes, une arête entre le sommet x et le sommet y n'est ajoutée que

s'il n'en existe pas déjà une entre les sommets x et y. De même, on ajoute un arc du sommet x au sommet y seulement s'il n'y en a pas déjà un dans ce sens. On pourrait donc avoir deux sommets x, y reliés par une arête de poids w_1, un arc de x à y de poids w_2 et un arc de y à x de poids w_3, si l'on ne se limitait pas dans la suite à construire des graphes orientés ou non-orientés mais jamais mixtes.

Définition 2.2.4
Un graphe G est de largeur de clique pondérée, notée $Wcw(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées W-clique-opérations) :

(i) ver_a, $a \in \mathcal{S}$ *(Constructeur de base : crée un sommet source isolé d'étiquette a) ;*

(ii) $\rho_{a\to b}(H)$, $a, b \in \mathcal{S}$ *(renomme les sources d'étiquette a en sources d'étiquette b) ;*

(iii) $\eta^w_{a,b}(H)$, $a, b \in \mathcal{S}$, $a \neq b$, $w \in W$ *(ajoute une arête de poids w entre tout couple de sommets dont l'un est étiqueté a et l'autre est étiqueté b) ;*

(iv) $\alpha^w_{a,b}(H)$, $a, b \in \mathcal{S}$, $a \neq b$, $w \in W$ *(ajoute un arc de poids w entre tout couple de sommets dont l'un est étiqueté a et l'autre est étiqueté b) ;*

(v) $H \oplus H'$ *(union disjointe des graphes H et H').*

Définition 2.2.5
*Un graphe G est de largeur W-*NLC *, notée W-$w_{\mathrm{NLC}}(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées opérations W-*NLC*) :*

(i) ver_a, $a \in \mathcal{S}$ *(Constructeur de base : crée un sommet source isolé d'étiquette a) ;*

(ii) $\circ_R(H)$ *pour toute application R de $\mathcal{S}$ dans $\mathcal{S}$ (pour toute étiquette $a \in \mathcal{S}$, ré-étiqueter les sommets d'étiquette a avec l'étiquette $R(a)$) ;*

(iii) $H \times_S H'$ *pour toute fonction partielle $S : \mathcal{S}^2 \times \{-1, 0, 1\} \to W$ (union disjointe des graphes H et H' à laquelle est ajoutée une arête (un arc) de poids w entre tous les couples de sommets $x \in H$ (d'étiquette l_x), $y \in H'$ (d'étiquette l_y) tels que $S(l_x, l_y, 0) = w$ ($S(l_x, l_y, s) = w$, $s \neq 0$) ; l'arc va de x à y si $s = 1$ et de y à x si $s = -1$).*

Définition 2.2.6
*Un graphe G est de largeur W-*MC *, notée W-$w_{\mathrm{MC}}(G)$, au plus k si et seulement s'il existe un ensemble $\mathcal{S}$ d'étiquettes de sources de cardinalité k tel que G peut être construit au moyen d'un nombre fini des opérations suivantes (nommées opérations W-*MC*) :*

(i) ver_A *(Constructeur de base : crée un sommet source isolé ayant un ensemble d'étiquettes A, $A \subseteq \mathcal{S}$) ;*

(ii) $H \otimes_{S,h,h'} H'$ *pour toute fonction partielle* $S : \mathcal{S}^2 \times \{-1, 0, 1\} \to W$ *et toutes applications* $h, h' : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ *(union disjointe des graphes* H *et* H' *à laquelle est ajoutée une arête (un arc) de poids* w *entre tous les couples de sommets* $x \in H$ *(d'ensemble d'étiquettes* L_x*),* $y \in H'$ *(d'ensemble d'étiquettes* L_y*) tels que* $l_x \in L_x$*,* $l_y \in L_y$ *et* $S(l_x, l_y, 0) = w$ *(*$S(l_x, l_y, s) = w$*,* $s \neq 0$*) ; l'arc va de* x *à* y *si* $s = 1$ *et de* y *à* x *si* $s = -1$ *; les étiquettes des sommets de* H *sont ensuite modifiées via* h *et celles des sommets de* H' *sont modifiées via* h'*).*

Dans la dernière opération de l'algèbre W-MC, il existe une possibilité que deux arêtes ou deux arcs ayant les mêmes extrémités soient ajoutés simultanément. Dans ce cas, le graphe obtenu n'est pas défini. Pour cette raison, on considérera comme bien formés les termes (ou arbres syntaxiques) où cela ne se produit pas.

Nous avons en quelque sorte triché dans les 3 définitions précédentes. En effet, il peut exister un nombre indénombrable d'opérations créant les arêtes puisque l'ensemble W n'est pas nécessairement dénombrable. Néanmoins, si W_G dénote l'ensemble des poids effectivement présents sur les arêtes de G, on obtient une définition équivalente en remplaçant W par W_G et cette fois on définit bien une algèbre puisque W_G est fini. Nous avons choisi cette notation afin de pouvoir parler simplement de la largeur de clique pondérée d'une famille de graphes pondérés dont les poids sont à valeur dans un ensemble non-nécessairement dénombrable. Pour en finir avec les remarques sur la cardinalité de W, il est évident que les largeurs de clique d'un graphe G non-pondéré sont égales aux largeurs de clique pondérées du graphe G muni d'une pondération à valeur dans W où $|W| = 1$.

Dans la suite, on appellera parfois décompositions de clique pondérée, W-NLC ou W-MC d'un graphe G, l'arbre syntaxique d'un terme construisant G sur l'algèbre adéquate.

Les trois algèbres précédentes peuvent être étendues aux graphes pondérés avec des boucles en ajoutant le constructeur de base $\mathrm{verloop}_a^w$ ou $\mathrm{verloop}_A^w$ qui crée un sommet source muni d'une boucle de poids w et d'une étiquette a ou d'un ensemble d'étiquettes A. On peut alors aisément montrer le résultat suivant.

Proposition 2.2.7
Soit G *un graphe mixte pondéré avec des boucles. Soit* $Unloop(G)$ *le graphe mixte pondéré obtenu en enlevant les boucles de* G*. Alors :*

- $Wcw(G) = Wcw(Unloop(G))$ *;*
- $W\text{-}w_{\mathrm{NLC}}(G) = W\text{-}w_{\mathrm{NLC}}(Unloop(G))$ *;*
- $W\text{-}w_{\mathrm{MC}}(G) = W\text{-}w_{\mathrm{MC}}(Unloop(G))$*.*

Ce fait nous amènera à négliger les détails techniques relatifs aux boucles dans la preuve du théorème suivant. Celui-ci montre que les inégalités entre les trois largeurs de clique sont toujours valides dans le cas pondéré. La preuve en est d'ailleurs quasi-identique à celle du cas non-pondéré, à la nuance près que cette dernière n'existe pas, à ma connaissance, en version intégrale et détaillée.

Théorème 2.2.8
Pour tout graphe pondéré G,

$$W\text{-}w_{\mathrm{MC}}(G) \leqslant W\text{-}w_{\mathrm{NLC}}(G) \leqslant Wcw(G) \leqslant 2^{W\text{-}w_{\mathrm{MC}}(G)+1} - 1.$$

Preuve :

Première inégalité :

Soient G un graphe pondéré de largeur W-NLC k et T une décomposition W-NLC de G de largeur k. Soit $\mathcal{S}$ l'ensemble d'étiquettes utilisées par T. On peut considérer sans perte de généralité que dans T :

- il n'y a pas deux opérations $\circ_R(H)$ consécutives, sinon on remplace T par T' où deux noeuds consécutifs de T munis des opérations $\circ_R(H)$ et $\circ_{R'}(H)$ sont remplacés par un noeud $\circ_{R''}(H)$ ($R'' = R' \circ R$).
- aucune opération ver_a n'est suivie d'une opération $\circ_R(H)$, sinon on peut remplacer T par T' où ces deux opérations sont remplacées par ver_b où $b = R(a)$.
- Chaque opération $H \times_S H'$ est suivie d'exactement une opération $\circ_R(H)$, sinon on peut ajouter l'opération $\circ_{Id}(H)$ (Id est la fonction identité de $\mathcal{S}$).

On peut remplacer l'opération W-NLC ver_a par l'opération W-MC $\mathrm{ver}_{\{a\}}$; tandis que les opérations W-NLC consécutives $H \times_S H'$ et $\circ_R(H)$ peuvent être remplacées par l'opération W-MC $H \otimes_{S,h,h} H'$ où $h(\{a\}) = \{R(a)\}, \forall a \in \mathcal{S}$. Il est clair que l'on obtient par ces modifications de T une décomposition W-MC construisant G avec le même ensemble $\mathcal{S}$ d'étiquettes de cardinalité k. En conséquence, $W\text{-}w_{\mathrm{MC}}(G) \leqslant W\text{-}w_{\mathrm{NLC}}(G)$.

Seconde inégalité :

Soient G un graphe pondéré de largeur de clique pondérée k et T une décomposition de clique pondérée de largeur k. Soit $\mathcal{S}$ l'ensemble d'étiquettes utilisées par T. On peut considérer sans perte de généralité que dans T :

- après une opération d'union disjointe $H \oplus H'$ toutes les arêtes de G entre $x \in H$ et $y \in H'$ sont ajoutées entre cette opération $H \oplus H'$ et la première opération suivante O d'union disjointe ou de renommage. Sinon considérons la première opération O' de type $\eta^w_{a,b}(H)$ ou $\alpha^w_{a,b}(H)$ après O qui ajoute une arête entre un sommet x' de H et un sommet y' de H'. On peut ajouter une opération $\eta^w_{a',b'}(H)$ ou $\alpha^w_{a',b'}(H)$ avant O où a' et b' sont les étiquettes dans $H \oplus H'$ des extrémités de l'arête ajoutée par l'opération O'.
- chaque opération $\eta^w_{a,b}(H)$ ou $\alpha^w_{a,b}(H)$ ajoute au moins une arête.
- toutes les opérations $\eta^w_{a,b}(H)$ ou $\alpha^w_{a,b}(H)$ sont situées entre une opération d'union disjointe $H \oplus H'$ et la première opération suivante O d'union disjointe ou de renommage.

On peut remplacer la W-clique-opération ver_a par l'opération W-NLC ver_a et la W-clique-opération $\rho_{a\to b}(H)$ par l'opération W-NLC $\circ_R(H)$ où $R(a) = b$ et $R(c) = c, \forall c \in \mathcal{S}, c \neq a$. Finalement, chaque groupe

formé d'une W-clique-opération $H \oplus H'$ et des opérations $\eta^w_{a,b}(H)$ ou $\alpha^w_{a,b}(H)$ suivantes peut être remplacée par l'opération W-NLC $H \times_S H'$, où $S(a,b,0) = S(b,a,0) = w$ s'il existe une opération $\eta^w_{a,b}(H)$ dans le groupe et $S(a,b,1) = S(a,b,-1) = w$ s'il existe une opération $\alpha^w_{a,b}(H)$. Il est clair que l'on obtient par ces modifications de T une décomposition W-NLC construisant G avec le même ensemble $\mathcal{S}$ d'étiquettes de cardinalité k. En conséquence, $W\text{-}w_{\mathrm{NLC}}(G) \leqslant Wcw(G)$.

Dernière inégalité :

Soient G un graphe pondéré de largeur W-MC k et T une décomposition W-MC de largeur k. Soit $\mathcal{S}$ l'ensemble d'étiquettes utilisées par T. Soit $\mathcal{S}'$ un ensemble d'étiquettes de cardinalité $2^{k+1}-1$, $\mathcal{S}' = \mathcal{S}_l \sqcup \mathcal{S}_r \sqcup \{empty\}$ où $|\mathcal{S}_l| = |\mathcal{S}_r| = 2^k - 1$. On définit trois bijections $l : \mathcal{P}(\mathcal{S})\backslash\emptyset \to \mathcal{S}_l$, $r : \mathcal{P}(\mathcal{S})\backslash\emptyset \to \mathcal{S}_r$ et $u : \mathcal{S}_l \to \mathcal{S}_r$ tel que $u(l(A)) = r(A), \forall A \in \mathcal{P}(\mathcal{S})$. On notera ρ_f une suite de W-clique-opérations $\rho_{a\to b}$ qui réalise une fonction f de $\mathcal{S}'$ dans $\mathcal{S}'$. On associe à chaque fonction $S : \mathcal{S}^2 \times \{-1,0,1\} \to W$ deux suites η_S et α_S constituées des W-clique-opérations $\eta^w_{l(A),r(B)}$ et $\alpha^w_{l(A),r(B)}$ (resp. $\alpha^w_{r(B),l(A)}$) pour tous les couples $(a,b) \in \mathcal{S}^2, (A,B) \in (\mathcal{P}(\mathcal{S})\backslash\emptyset)^2$ tels que $S(a,b,0) = w$ et $S(a,b,1) = w$ (resp. $S(a,b,-1) = w$), $a \in A$ et $b \in B$.

On peut remplacer l'opération W-MC ver_A par la W-clique-opération $\mathrm{ver}_{l(A)}$ si $A \neq \emptyset$ et ver_{empty} sinon. Chaque opération W-MC $H \otimes_{S,h,h'} H'$ sera remplacée par les W-clique-opérations suivantes :

- on applique ρ_u au sous-arbre construisant H' ;
- on joint les deux sous-arbres avec $H \oplus H'$;
- on applique η_S ;
- on applique α_S ;
- on applique $\rho_{l\circ h\circ l^{-1}}$;
- on applique $\rho_{l\circ h'\circ r^{-1}}$.

Il est clair que l'on obtient par ces modifications de T une décomposition de clique pondérée construisant G avec comme ensemble d'étiquettes $\mathcal{S}'$ de cardinalité $2^{k+1}-1$. On a donc bien $Wcw(G) \leqslant 2^{W\text{-}w_{\mathrm{MC}}(G)+1} - 1$. ∎

2.3 Couvertures de graphe

Dans cette seconde partie, nous nous attacherons à caractériser l'expressivité de trois types de couvertures de graphes en terme de classes de complexité algébrique. Il s'agit ici de couvrir les sommets du graphe à l'aide d'un ensemble d'arêtes (ou d'arcs dans le cas orienté).

Définition 2.3.1 (couverture)
Une couverture *d'un graphe est un sous-ensemble de ses arêtes tel que tout sommet est l'extrémité d'au moins une des arêtes de la couverture. Le* poids *d'une couverture est le* produit *des poids des arêtes de la couverture.*

Le premier type de couverture est celui des couvertures par circuits disjoints d'un graphe orienté : chaque sommet doit être incident à exactement un arc entrant et un arc sortant de la couverture. Toute boucle éventuelle compte bien entendue à la fois comme un arc entrant et un arc sortant. Le second est une légère variante du premier puisqu'il s'agit des couvertures par circuit hamiltonien. Le troisième type est celui des couplages parfaits : chaque sommet doit être incident à exactement une arête de la couverture.

Ces couvertures ont une formulation algébrique bien connue puisqu'elles sont liées aux opérateurs d'algèbre linéaire que sont le permanent et l'hamiltonien d'une matrice.

Définition 2.3.2 (permanent, hamiltonien)
Le permanent *d'une matrice $M = (m_{i,j})$ de dimension $(n \times n)$ est égal à*

$$\mathrm{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}.$$

*L'*hamiltonien *d'une matrice $M = (m_{i,j})$ de dimension $(n \times n)$ est égal à*

$$\mathrm{ham}(M) = \sum_{\sigma \in C_n} \prod_{i=1}^{n} m_{i,\sigma(i)},$$

où C_n est le sous-ensemble de S_n constitué des permutations ayant un unique cycle de longueur n.

En effet, il est aisé de constater que si l'on associe au graphe G sa matrice d'adjacence pondérée M_G, alors la somme des poids des couvertures par circuits de G est égale au permanent de M_G. Ou réciproquement, que le permanent de M est égal à la somme des poids des couvertures par circuits de G_M, où G_M est le graphe orienté dont M est la matrice d'adjacence. Cette équivalence avec la définition 2.3.2 est claire puisque toute permutation peut s'écrire comme un produit de cycles disjoints et cette décomposition est unique. Quant à la somme des poids des couvertures par circuit hamiltonien de G, elle est égale à l'hamiltonien de M_G.

Le lien avec la somme des poids des couplages parfaits est un peu moins direct et pourtant c'est l'interprétation combinatoire du permanent la plus fréquemment donnée. Elle nécessite de considérer le graphe suivant.

Définition 2.3.3
Soit G un graphe orienté (pondéré ou non). Le biparti d'adjacence *de G, noté $B(G)$, est un graphe biparti, non-orienté (pondéré ou non) obtenu comme suit :*

- *Chaque sommet $u \in V(G)$ est partagé en deux sommets u^+ et u^- ;*
- *chaque arc uv (de poids w) est remplacé par une arête entre u^+ et v^- (de poids w). Une boucle sur u (de poids w) est remplacée par une arête entre u^+ et u^- (de poids w). Ainsi les arcs de G sont en bijection avec les arêtes de $B(G)$.*

Il est clair que les couvertures par circuits de G sont en bijection avec les couplages parfaits de $B(G)$ au moyen de la fonction de $\mathcal{P}(E(G))$ dans $\mathcal{P}(E(B(G)))$

qui à toute partie des arcs de de G associe les arêtes correspondantes. De plus cette bijection conserve les poids des couvertures. On peut donc définir le permanent d'une matrice M comme la somme des poids des couplages parfaits de $B(G_M)$. Soient G non-orienté et G' le graphe orienté construit à partir de G en remplaçant chaque arête $\{x, y\}$ de poids w par deux arcs xy et yx de poids w. La matrice d'adjacence de G est définie habituellement comme étant la matrice d'adjacence de G'. Cette matrice est symétrique. Réciproquement, une matrice symétrique peut être interprétée comme la matrice d'adjacence d'un graphe orienté ou non-orienté. On peut remarquer que la matrice d'adjacence de $B(G_M)$ est $\begin{pmatrix} 0 & M \\ M^t & 0 \end{pmatrix}$.

Maintenant que nous avons rappelé la définition du biparti d'adjacence, profitons-en pour prouver deux résultats simples, utilisés ultérieurement, sur les liens entre les largeurs d'un graphe orienté et celles de son biparti d'adjacence.

Proposition 2.3.4
Si G est de largeur arborescente (resp. linéaire) k alors $B(G)$ est de largeur arborescente (resp. linéaire) au plus $2k + 1$.

Preuve :

Soit $\langle T, (X_t)_{t \in V(T)} \rangle$ une k-décomposition arborescente (linéaire) de G. Il est évident que $\langle T, (X'_t)_{t \in V(T)} \rangle$, où $X'_t = \{u^+, u^- | u \in X_t\}$, est une décomposition arborescente (linéaire) de $B(G)$ de largeur au plus $2k + 1$. ∎

Proposition 2.3.5
Si G est de largeur de clique pondérée k alors $B(G)$ est de largeur de clique au plus $2k$.

Preuve :

Soit T une décomposition de clique pondérée de G utilisant l'ensemble d'étiquettes $\mathcal{S}$ de cardinalité k. On peut remplacer la W-clique-opération ver_a par les trois opérations $(\mathrm{ver}_{a^+}) \oplus (\mathrm{ver}_{a^-})$, l'opération $\rho_{a \to b}(H)$ par les opérations $\rho_{a^+ \to b^+}(H)$ et $\rho_{a^- \to b^-}(H)$. Finalement chaque opération $\alpha^w_{a,b}(H)$ peut être remplacée par $\eta^w_{a^+,b^-}(H)$. Il est clair qu'on obtient ainsi une décomposition de clique pondérée de $B(G)$ utilisant l'ensemble d'étiquettes $\{a^+, a^- | a \in \mathcal{S}\}$ de cardinalité $2k$. ∎

Revenons à nos couvertures et rappelons que dans le cadre de la complexité usuelle, celle des machines de Turing, il est #P-complet de calculer le permanent ou l'hamiltonien d'une matrice dont les entrées sont 0 ou 1. En fait, la classe #P a même été introduite par Valiant dans l'article [68] où il montre la #P-complétude du permanent. Ces résultats ont pour conséquence que compter le nombre de couvertures par circuits ou par circuit hamiltonien d'un graphe orienté ainsi que le nombre de couplages parfaits d'un graphe biparti non-orienté sont des problèmes #P-complet. L'attention s'est alors focalisée sur le fait qu'un problème P, comme décider si un graphe biparti admet un

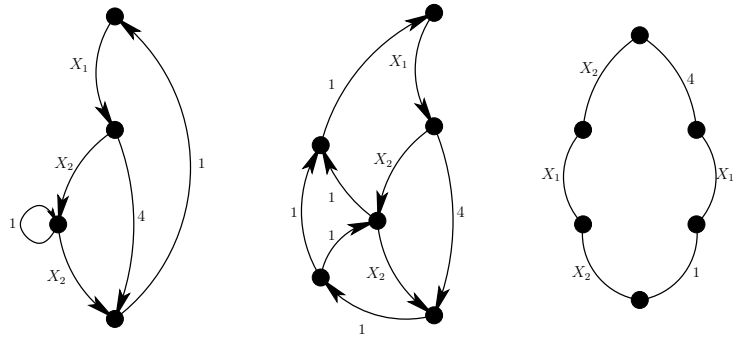


FIGURE 2.3 – Expression du polynôme $X_1X_2^2 + 4X_1$ au moyen de trois couvertures de graphes.(a) Couverture par circuits. (b) Couverture par circuit hamiltonien. (c) Couplage parfait.

couplage parfait, puisse être #P-complet dans sa version problème de comptage. Depuis, d'autres résultats plus spectaculaires encore ont montré que certains problèmes de comptage sont #P-complet alors même que le problème de décision associé est toujours vrai ; c'est le cas par exemple du problème du comptage du nombre d'extensions linéaires d'un ordre partiel ([14]).

Afin de généraliser ces problèmes de comptage, on peut considérer des graphes munis d'une pondération des arêtes sur un corps $\mathbb{K}$. En se plaçant sur le corps des rationnels et en supposant qu'on assigne un poids égal à 1 à toutes les arêtes du graphe, il est clair que l'on retrouve notre problème initial de comptage des couvertures.

Au delà de ces problèmes de complexité uniforme, on peut utiliser cette somme de poids de couverture, dans le cadre non uniforme, pour exprimer des polynômes. Prenons, par exemple, le polynôme $X_1X_2^2 + 4X_1$ sur $\mathbb{Q}$. Pour chacun des trois types de couverture, on peut aisément construire un graphe tel que la somme des poids des couvertures soit ce polynôme, ainsi que le montre la figure 2.3.

De manière équivalente, on peut représenter des polynômes à l'aide de permanents ou d'hamiltoniens. Il suffit que les entrées de la matrice M soient des variables ou des constantes du corps $\mathbb{K}$; $f = \mathrm{per}(M)$ et $f' = \mathrm{ham}(M)$ sont alors des polynômes à coefficients dans K (dans la terminologie de Valiant, f (resp. f') est une projection du polynôme permanent (resp. hamiltonien)).

On peut facilement montrer l'universalité pour chacune de ces couvertures, *i.e.* tout polynôme peut être exprimé ainsi. Mais on peut faire nettement mieux puisque Valiant a montré dans l'article [67] que les familles de polynômes permanent et hamiltonien sont VNP-complètes. Il faut ici distinguer les opérateurs permanent et hamiltonien, dont la définition est donnée plus haut, des polynômes formels définis comme suit :

Définition 2.3.6 (polynômes permanent et hamiltonien)
Le polynôme permanent *de dimension n est un polynôme à n^2 variables défini*

par

$$\mathrm{per}_n = \sum_{\sigma \in S_n} \prod_{i=1}^{n} X_{i,\sigma(i)}.$$

Le polynôme hamiltonien *de dimension n est un polynôme à n^2 variables défini par*

$$\mathrm{ham}_n = \sum_{\sigma \in C_n} \prod_{i=1}^{n} X_{i,\sigma(i)},$$

où C_n est le sous-ensemble de S_n constitué des permutations ayant un unique cycle de longueur n.

Bien entendu, si on pose $M_n^X = (X_{i,j})_{1 \leqslant i,j \leqslant n}$, on a $\mathrm{per}_n = \mathrm{per}(M_n^X)$ et $\mathrm{ham}_n = \mathrm{ham}(M_n^X)$.

Formellement, le résultat de Valiant s'énonce ainsi :

Théorème 2.3.7

- *La famille de polynômes (per_n) est* VNP*-complète sur tout corps de caractéristique différente de 2 ;*
- *La famille de polynômes (ham_n) est* VNP*-complète sur tout corps.*

Ce résultat peut être traduit sous la forme : « Les sommes des poids des trois types considérés de couvertures prises sur des familles de graphes dont la taille croît polynomialement permettent d'exprimer toutes les familles de polynômes de VNP ». À la nuance près que ce n'est pas le cas en caractéristique 2 pour les couvertures par circuits et les couplages parfaits. Cette « anomalie » a deux raisons : la première, syntaxique, est que la preuve de VNP-complétude du permanent utilise des constantes $\frac{1}{2}$; la seconde, sémantique, est qu'en caractéristique 2 $x = -x$ et donc permanent et déterminant sont égaux. Comme le déterminant est dans VP, il est peu probable que le permanent soit VNP-complet en caractéristique 2.

Dans la suite, nous considérerons des matrices dont les entrées sont prises dans l'ensemble $\overline{\mathbb{K}}$ des variables et des constantes d'un corps $\mathbb{K}$. Un 0 du corps $\mathbb{K}$ aux ligne et colonne i, j sera interprété comme un non-arc de i vers j. Nous allons allègrement confondre un graphe et sa matrice d'adjacence. Pour cette raison, une largeur (arborescente, linéaire, de clique pondérée, $\overline{\mathbb{K}}$-NLC, $\overline{\mathbb{K}}$-MC) d'une matrice sera définie comme la largeur correspondante du graphe pondéré dont elle est la matrice d'adjacence. De même, on parlera de matrice planaire lorsque le graphe associé est planaire.

Les chapitres suivants de cette thèse sont dévolus à l'étude de l'expressivité du permanent, de l'hamiltonien et de la somme des poids des couplages parfaits sur des classes de graphes de largeur (arborescente, linéaire, de clique pondérée, $\overline{\mathbb{K}}$-NLC, $\overline{\mathbb{K}}$-MC) bornée ainsi que sur la classe des graphes planaires. Le but étant de capturer des sous-classes « faciles » de VNP. Avant d'attaquer ces chapitres, deux remarques finales sur celui-ci.

Remarque. Si G est le graphe pondéré ayant pour matrice d'adjacence M_n^X et G' est le graphe G sans pondération, alors G' est une clique avec boucles. G' est donc de largeur de clique 1. Si l'on définit la largeur de clique d'un graphe pondéré comme celle de son graphe non-pondéré sous-jacent, on voit dès lors que le permanent et l'hamiltonien des classes de graphes de largeur de clique

bornée permettent de capturer des familles VNP-complètes. C'est un argument supplémentaire en faveur de la définition de la largeur de clique pondérée.
Remarque. Oum et Seymour [37] ont donné un algorithme d'approximation pour la largeur de rang. Cet algorithme fournit aussi une approximation de la largeur de clique car largeur de rang et de clique peuvent toutes deux être majorées par une fonction de l'autre largeur. La largeur de rang fait intervenir le calcul du rang sur $\mathbb{Z}_2$ de sous-matrices de la matrice d'adjacence. On peut facilement voir que la largeur de clique d'un graphe G non pondéré est égale à la largeur de clique pondérée sur $\mathbb{Z}_2$ de la clique dont les arcs sont de poids 1 s'ils appartiennent à G et 0 sinon (Cette interprétation donne d'ailleurs un peu plus de sens au nom de largeur de clique). Il semble donc naturel de se demander si cette « équivalence » entre ces deux largeurs est encore valable dans le cas pondéré. Cette question se pose sous deux formes légèrement différentes :

- La largeur de clique pondérée sur $\mathbb{K}$ est-elle équivalente à la largeur de rang sur $\mathbb{K}$?
- La largeur de clique pondérée sur $\overline{\mathbb{K}}$ est-elle équivalente à la largeur de rang sur le corps des fractions rationnelles sur $\mathbb{K}$?

La réponse à ces deux questions dépend-elle de la caractéristique de $\mathbb{K}$?

Chapitre 3

Expressivité des couvertures de graphes de largeur arborescente bornée

Dans [24], Courcelle, Makowsky et Rotics montrent que le permanent, le polynôme hamiltonien et la somme des couplages parfaits sont dans la classe de complexité VP pour les matrices de largeur arborescente bornée. Nous allons à présent montrer que les poids de ces couvertures sur des matrices de largeur arborescente bornée sont équivalents aux formules (termes ou expressions) arithmétiques. C'est une amélioration du résultat de [24] de deux manières. La première est que nous montrons que toute famille de formules peut s'exprimer comme poids des couvertures d'une famille de graphes de largeur arborescente bornée. La seconde est que nous précisons dans quelle sous-classe de VP ces couvertures se situent, l'ensemble des familles de polynômes représentables par des formules arithmétiques de taille polynomiale étant probablement un sous-ensemble strict de VP.

Nous allons donc nous attacher à démontrer le théorème suivant.

Théorème 3.0.1
Soit (f_n) une famille de polynômes à coefficients dans un corps $\mathbb{K}$. Les quatre propriétés suivantes sont équivalentes :

- *(f_n) peut être représentée par une famille de formules arithmétiques de taille polynomiale.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur arborescente bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{per}(M_n)$.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur arborescente bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{ham}(M_n)$.*
- *Il existe une famille (M_n) de matrices symétriques de taille polynomiale et de largeur arborescente bornée telle que les entrées de M_n sont des*

constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \sum_{P \in \mathcal{P}(M_n)} W(P)$, où $\mathcal{P}(M_n)$ est l'ensemble des couplages parfaits de G_{M_n}.

Rappelons que, de par la VNP-complétude de l'hamiltonien, si on supprime l'hypothèse de largeur arborescente bornée sur M_n on représente l'ensemble des familles de VNP au lieu des familles représentées par des formules arithmétiques de taille polynomiale. Le permanent satisfait aussi cette propriété si la caractéristique de $\mathbb{K}$ est différente de 2.

3.1 Des formules vers les couvertures des graphes de largeur arborescente bornée

Le théorème 3.0.1 découle des propositions 3.1.1, 3.1.2, 3.1.3, 3.2.4, 3.2.5 et 3.2.6. Nous commençons par montrer que toute formule peut être exprimée à l'aide de couvertures de graphes de taille linéaire et de largeur arborescente bornée.

Proposition 3.1.1
Toute formule arithmétique peut être exprimée comme le permanent d'une matrice de largeur arborescente au plus 2 et de taille au plus $n \times n$ où n est la taille de la formule. Toutes les entrées de la matrice sont 0, 1, des constantes ou variables de la formule.

Preuve :

La première étape consiste à construire un graphe orienté sans circuits *série-parallèle* (SP), dans lequel il y a une connexion entre les poids des chemins et la valeur calculée par la formule. L'idée globale qui sous-tend la construction est relativement standard, le lecteur intéressé peut consulter [53]. Les graphes SP en général peuvent avoir un nombre arbitraire d'arcs entre deux sommets. Mais ici on construit un graphe SP tel qu'il y a au plus un arc d'un sommet u à un sommet v. Cette propriété sera nécessaire dans la seconde étape où nous ferons le lien entre les couvertures par circuit et le permanent de la matrice d'adjacence pondérée.

Les graphes SP ont deux sommets distingués, une *source* et un *puits*, qu'on notera s et t. On notera $SC(G)$ la somme des poids de tous les chemins orientés de s à t, où le poids d'un chemin est le *produit* des poids de ses arcs.

Soit φ une formule de taille p. Nous montrons en premier lieu par induction sur p que l'on peut construire un graphe SP G orienté et pondéré tel que $val(\varphi) = SC(G)$ et $|V(G)| \leqslant p + 1$. Pour le cas de base $\varphi = w$, on crée deux sommets s et t connectés par un arc de s à t de poids w. Le nombre de sommets utilisés est $2 = taille(\varphi) + 1$.

Supposons que $\varphi = \varphi_1 + \varphi_2$. Soit G_i le graphe associé à φ_i par l'hypothèse d'induction. On définit G comme l'union d'un nouveau sommet s, de G_1 et de G_2 dans laquelle on fusionne t_1 avec t_2 (on nomme t le sommet obtenu), on ajoute un arc de poids 1 de s à s_1, ainsi que de s à s_2. Par construction et hypothèse d'induction, le graphe G obtenu satisfait $SC(G) = 1 \cdot SC(G_1) + 1 \cdot SC(G_2) = val(\varphi_1) + val(\varphi_2)$. G ne comporte pas de circuits et il y a bien au plus un arc de u à v, pour tous sommets

u et v de G (Le nouveau sommet s existe pour cette raison). On a ajouté un nouveau sommet, mais puisque t_1 a été fusionné avec t_2, on obtient $|V(G)| = |V(G_1)|+|V(G_2)| \leqslant taille(\varphi_1)+1+taille(\varphi_2)+1 = taille(\varphi)+1$.

Supposons à présent que $\varphi = \varphi_1 \times \varphi_2$. On construit G une nouvelle fois en prenant l'union de G_1 et G_2 dans laquelle on fusionne t_1 avec s_2. s_1 devient la source s de G et t_2 devient son puits t. Pour tous chemins orientés Ch_i de s_i à t_i dans G_i, il existe dans G un chemin orienté de s à t, concaténation de Ch_1 et Ch_2, dont le poids est égal au produit des poids des chemins Ch_1 et Ch_2. Réciproquement, il est évident que tout chemin de s à t dans G est de ce type. On obtient donc $SC(G) = SC(G_1) \cdot SC(G_2)$. On vérifie aisément l'absence de circuits, d'arcs multiples ainsi que $|V(G)| = |V(G_1)| + |V(G_2)| - 1 \leqslant taille(\varphi_1) + taille(\varphi_2) + 1 < taille(\varphi) + 1$.

La seconde étape de la preuve consiste dès lors à construire un graphe G' tel qu'il existe une bijection entre ses couvertures par circuits et les chemins orientés de s à t dans G. Il suffit pour construire un tel G' d'ajouter un arc « retour » de poids 1 de t à s, ainsi que des boucles, elles aussi de poids 1, sur tous les sommets distincts de s et t. Il est clair à présent qu'à tout s,t-chemin de G on peut associer le circuit de G' obtenu par concaténation avec l'arc « retour ». Les sommets hors de ce cycle peuvent alors être couverts par les boucles pour obtenir une couverture par circuits dont le poids est identique à celui du chemin. On constate aussitôt que toute couverture par circuits de G' comprend nécessairement un tel circuit puisque s et t n'ont pas de boucles. De plus les autres sommets n'ont dès lors plus d'autre choix de couverture que leur boucle puisque G n'a pas de circuit. On a donc bien la bijection souhaitée entre chemins et couvertures par circuits. Puisque G' ne comporte aucun arc multiple, on peut le représenter par une matrice M de taille au plus $(n+1) \times (n+1)$ telle que $G_M = G'$ et $per(M) = val(\varphi)$.

Il reste à montrer que G' est de largeur arborescente au plus 2. Pour cela, soit on se rappelle que les graphes série-parallèle sont de largeur arborescente au plus 2, soit on fournit une décomposition arborescente, soit on fournit une algèbre HR permettant de construire G' en utilisant moins de 3 étiquettes. Comme nous réutiliserons la troisième méthode un peu plus loin et qu'il est probable que le lecteur ne soit que peu, voire pas, habitué à celle-ci, nous allons nous exercer sur ce cas simple. Voici donc la troisième solution avec les étiquettes s,t et r une étiquette temporaire. Pour le cas de base, l'opération edge_{st} est suffisante. Les opérations suivantes nous fournissent la construction désirée pour la simulation de l'addition de deux formules :

$$\mathrm{ren}_{s\leftrightarrow r}(\mathrm{forg}_s[\mathrm{edge}_{sr} \mathbin{//} \mathrm{loop}_s \mathbin{//} G_1] \mathbin{//} \mathrm{forg}_s[\mathrm{edge}_{sr} \mathbin{//} \mathrm{loop}_s \mathbin{//} G_2]).$$

Pour simuler la multiplication de formules, on utilise ces opérations :

$$\mathrm{forg}_r[\mathrm{ren}_{t\leftrightarrow r}(G_1) \mathbin{//} \mathrm{ren}_{s\leftrightarrow r}(\mathrm{loop}_s \mathbin{//} G_2)].$$

Enfin, on n'oubliera pas de faire une composition parallèle finale avec edge_{st} pour l'arc « retour ».

Pour se ramener à une taille n au lieu de $n+1$, il suffit de remarquer que l'on peut contracter l'arc « retour » sans changer la somme des poids des

couvertures par circuits, ni créer d'arcs multiples, ni augmenter la largeur arborescente.
■

Nous montrons à présent un résultat similaire pour l'hamiltonien. La preuve réutilise en partie la démonstration précédente en la complétant spécifiquement afin d'obtenir des circuits hamiltoniens.

Proposition 3.1.2
Toute formule arithmétique de taille n peut s'exprimer comme l'hamiltonien d'une matrice de largeur arborescente au plus 6 et de taille au plus $(2n+1) \times (2n+1)$. Toutes les entrées de la matrice sont 0, 1, des constantes ou variables de la formule.

Preuve :

La première étape consiste à construire le graphe G d'une manière similaire à celle utilisée dans la preuve de la proposition 3.1.1. On obtient donc un graphe série-parallèle tel que la somme des poids des chemins de s à t est égale à la formule. L'étape suivante est d'effectuer la preuve d'universalité du polynôme hamiltonien de [52] avec une largeur arborescente au plus 6. La construction montrant l'universalité consiste à ajouter $|V(G)|-1$ nouveaux sommets t_i à G et une floppée d'arcs de poids 1 afin de créer G', comme indiqué sur le dessin 3.1.

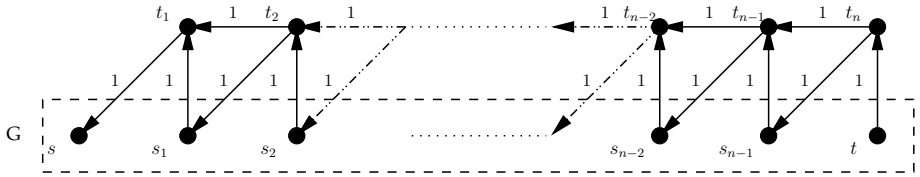


FIGURE 3.1 – Universalité du polynôme hamiltonien.

Plus formellement, on numérote arbitrairement les sommets $s_1, \ldots, s_{n-1}$ de G distincts de s et t. Pour i compris entre 1 et $n-1$, on ajoute les arcs de poids 1 (t_{i+1}, s_i), (t_{i+1}, t_i), (s_i, t_i) ainsi que (t, t_n) et (t_1, s). On appelle G' le graphe ainsi obtenu. Les sommets et arcs supplémentaires permettent de visiter n'importe quel sous-ensemble des sommets de G le long d'un chemin orienté de poids 1 de t à s utilisant tous les t_i. En conséquence, tout chemin de s à t dans G peut être suivi d'un unique chemin de t à s couvrant tous les sommets restants afin d'obtenir un circuit hamiltonien de même poids. Réciproquement, tout circuit hamiltonien est de ce type. En effet, si l'on fixe son origine au sommet s, il ne peut pas passer par t_1 avant d'être passé par t puique le seul arc sortant de t_1 va en s. Supposons que l'on a montré qu'il ne pouvait passer par t_i avant t et montrons qu'il ne peut dès lors pas passer par t_{i+1} avant t. S'il passe en t_{i+1} avant t, il ne peut donc prendre l'arc sortant de t_{i+1} vers t_i. Il doit donc prendre l'arc sortant de t_{i+1} vers s_i et continuer vers t. On a alors compromis toute possibilité de rentrer ultérieurement en t_i puisque seuls t_{i+1} et s_i sont les voisins intérieurs de t_i.

Le circuit ne peut donc être hamiltonien. On a bien une bijection préservant les poids entre les chemins de s à t de G et les circuits hamiltoniens de G'.

Pour effectuer cette construction en gardant une largeur arborescente bornée, il faut prendre une numérotation s_i raisonnable des sommets de G ; raisonnable signifiant conforme à la construction initiale de G. À cette fin, nous donnons à nouveau une algèbre HR permettant de construire le graphe G' en utilisant au plus 7 étiquettes. Les étiquettes a, b, c et d dans les figures 3.2 et 3.3 jouent le rôle de s, t, t_1 et t_n respectivement.

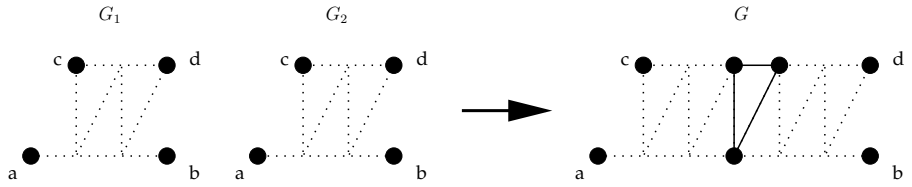


FIGURE 3.2 – Composition série (simulation de la multiplication).

La composition série est réalisée au moyen des opérations suivantes :

$\text{forg}_e[\text{forg}_f[\text{forg}_g[$
$\quad \text{ren}_{d\leftrightarrow f}(\text{ren}_{b\leftrightarrow e}(G_1))$ //
$\quad \text{ren}_{c\leftrightarrow g}(\text{ren}_{a\leftrightarrow e}(G_2))$ //
$\quad \text{edge}_{ef}$ // edge_{eg} // edge_{fg}
$]]]$

La construction ci-dessus ne prend pas en compte le cas où G_1 et/ou G_2 sont des graphes créés à partir du cas de base. Dans le cas de base, c'est un unique sommet qui joue le rôle de t_1 et t_n (il devrait avoir les étiquettes c et d). Néanmoins, il est clair qu'on peut modifier cette construction pour traiter aussi ces cas plus simples.

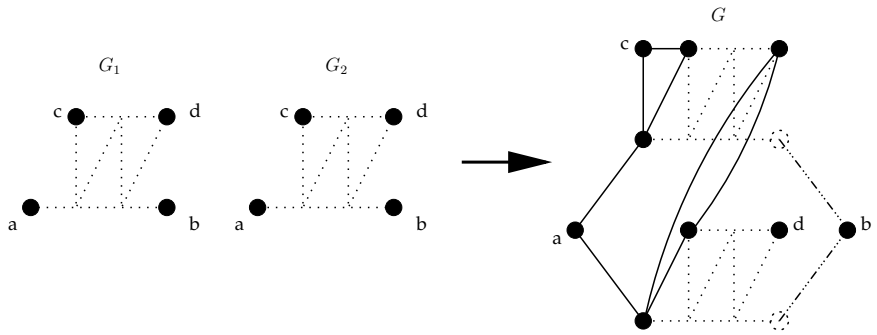


FIGURE 3.3 – Composition parallèle (simulation de l'addition).

Pour la composition parallèle, on crée deux nouveaux sommets (le nouveau a et le nouveau c) tandis qu'on fusionne les deux b. On peut le faire au moyen des opérations suivantes :

$$
\begin{array}{l}
\mathrm{forg}_f[\\
\quad \mathrm{ren}_{a\leftrightarrow e}(\mathrm{ren}_{c\leftrightarrow g}(\\
\qquad \mathrm{forg}_a[\mathrm{forg}_c[\mathrm{edge}_{ag} \;//\; \mathrm{edge}_{cg} \;//\; \mathrm{ren}_{d\leftrightarrow f}(\mathrm{edge}_{ae} \;//\; \mathrm{edge}_{ac} \;//\; G_1)]] \;// \\
\qquad \mathrm{forg}_a[\mathrm{forg}_c[\mathrm{edge}_{af} \;//\; \mathrm{edge}_{cf} \;//\; \mathrm{edge}_{ae} \;//\; \mathrm{edge}_{ac} \;//\; G_2]] \\
\quad)) \\
]
\end{array}
$$

L'étape finale de la construction consiste à relier les sommets a et c ainsi que les sommets b et d.

■

De la proposition 3.1.1, on peut déduire la proposition suivante.

Proposition 3.1.3
Toute formule arithmétique de taille n peut s'exprimer comme la somme des poids des couplages parfaits d'une matrice symétrique de largeur arborescente au plus 5 et de taille au plus $(2n) \times (2n)$. Toutes les entrées de la matrice sont 0, 1, des constantes ou variables de la formule.

Preuve :

Soit G le graphe orienté de largeur arborescente 2 et de taille n construit dans la preuve de la proposition 3.1.1. $B(G)$ est de largeur arborescente au plus 5 d'après la proposition 2.3.4 et nous avons vu au chapitre précédent que la somme des poids des couplages parfaits de $B(G)$ est égale au permanent de G.

■

On peut aussi par une preuve directe montrer la proposition suivante qui est légèrement plus puissante.

Proposition 3.1.4
Toute formule arithmétique de taille n peut s'exprimer comme la somme des poids des couplages parfaits d'une matrice symétrique de taille au plus $(2n) \times (2n)$ étant la matrice d'adjacence d'un graphe biparti de largeur arborescente au plus 2. Toutes les entrées de la matrice sont 0, 1, des constantes ou variables de la formule.

Preuve :

La preuve est grandement similaire à celle de la proposition 3.1.1. La première étape consiste à construire un graphe SP biparti non-orienté, dans lequel il y a une connexion entre les poids des chemins et la valeur calculée par la formule.

On notera $SC(G)$ la somme des poids de tous les chemins de s à t. On construit G de sorte que tous les chemins de s à t ont une longueur impaire et toutes les arêtes qui sont à distance impaire de s sont de poids 1. Pour vérifier que le graphe construit est bien biparti, le lecteur pourra vérifier que jamais deux sommets qui sont à la même distance de s ne sont adjacents.

Soit φ une formule de taille p. Nous montrons en premier lieu par induction sur p que l'on peut construire un graphe SP G biparti non-orienté et

pondéré tel que $val(\varphi) = SC(G)$ et $|V(G)| \leqslant 2p$. Pour le cas de base $\varphi = w$, on crée deux sommets s et t connectés par une arête entre s et t de poids w. Le nombre de sommets utilisés est $2 = 2 \cdot taille(\varphi)$.

Supposons que $\varphi = \varphi_1 + \varphi_2$. Soit G_i le graphe associé à φ_i par l'hypothèse d'induction. On ajoute deux chemins (u_1, v_1, w_1) et (u_2, v_2, w_2) de longueur 2 avec un poids 1 sur les deux arêtes. On obtient G'_1 (resp. G'_2) en fusionnant t_1 (resp. t_2) et u_1 (resp. u_2). w_1 (resp. w_2) devient le nouveau puits de G'_1 (resp. G'_2). On définit G comme l'union de G'_1 et de G'_2 dans laquelle on fusionne s_1 (resp. w_1) avec s_2 (resp. w_2) en appelant s (resp. t) le nouveau sommet obtenu. Par construction et hypothèse d'induction, le graphe G obtenu satisfait $SC(G) = 1 \cdot 1 \cdot SC(G_1) + 1 \cdot 1 \cdot SC(G_2) = val(\varphi_1) + val(\varphi_2)$. G est biparti et il y a bien au plus une arête entre u et v, pour tous sommets u et v de G (Les nouveaux sommets existent pour cette raison). On a ajouté 4 nouveaux sommets, mais puisque s_1 (resp. w_1) a été fusionné avec s_2 (resp. w_2), on obtient $|V(G)| = |V(G_1)| + |V(G_2)| + 2 \leqslant 2 \cdot taille(\varphi_1) + 2 \cdot taille(\varphi_2) + 2 = 2 \cdot taille(\varphi)$.

Supposons à présent que $\varphi = \varphi_1 \times \varphi_2$. On construit G en prenant l'union de G_1, G_2 et de l'arête de poids 1 $t'_1 s'_2$ dans laquelle on fusionne t_1 avec t'_1, s_2 avec s'_2. s_1 devient la source s de G et t_2 devient son puits t. Pour tous chemins Ch_i de s_i à t_i dans G_i, il existe dans G un chemin de s à t, concaténation de Ch_1, $t'_1 s'_2$ et Ch_2, dont le poids est égal au produit des poids des chemins Ch_1 et Ch_2. Réciproquement, il est évident que tout chemin de s à t dans G est de ce type. On obtient donc $SC(G) = SC(G_1) \cdot SC(G_2)$. On vérifie aisément l'ensemble des conditions sur G ainsi que $|V(G)| = |V_1| + |V_2| \leqslant 2 \cdot taille(\varphi_1) + 2 \cdot taille(\varphi_2) < 2 \cdot taille(\varphi)$.

La seconde étape consiste à remarquer que tout couplage parfait de G doit contenir les arêtes à distance paire de s le long d'un des chemins de s à t et toutes les arêtes à distance impaire de s hors de ce chemin (La condition d'avoir uniquement des chemins de longueur impaire de s à t à toutes les étapes de la construction est cruciale ici). Réciproquement, à tout (s,t)-chemin on peut associer un tel couplage parfait. Ces deux preuves peuvent être faites par induction en suivant la construction de G. Puisque toute arête à distance impaire de s est de poids 1, il est clair que le poids d'un couplage parfait est le poids du (s,t)-chemin correspondant.

Puisqu'il existe au plus une arête entre deux sommets u et v dans G, il existe une matrice M de taille au plus $(2n) \times (2n)$ telle que $G_M = G$ et $\sum_{P \in \mathcal{P}} W(P) = val(\varphi)$.

Le graphe G peut être construit au moyen d'une algèbre HR avec seulement 3 étiquettes ; il est donc de largeur arborescente au plus 2. Pour le cas de base, l'opération $edge_{st}$ est suffisante. Les opérations suivantes nous fournissent la construction désirée pour la simulation de l'addition de deux formules :

$$\begin{array}{l} \mathrm{ren}_{r \leftrightarrow t}(\mathrm{forg}_t[\mathrm{forg}_s[\mathrm{edge}_{ts} // \mathrm{edge}_{sr}] // G_1]) \\ // \\ \mathrm{ren}_{r \leftrightarrow t}(\mathrm{forg}_t[\mathrm{forg}_s[\mathrm{edge}_{ts} // \mathrm{edge}_{sr}] // G_2]). \end{array}$$

Pour simuler la multiplication de formules on utilise ces opérations :

$$\mathrm{forg}_r[\mathrm{forg}_t[G_1 \ // \ \mathrm{edge}_{tr}] \ // \ \mathrm{ren}_{s \leftrightarrow r}(G_2)].$$

■

Nous venons de montrer pour chacun des trois types de couvertures que toute formule arithmétique pouvait être exprimée comme la somme des poids de ces couvertures pour un graphe de taille linéaire en la taille de la formule et de largeur arborescente au plus 6. Dans le cadre de la complexité non-uniforme, on ne prend pas en compte la calculabilité ou la complexité d'une réduction entre problèmes ou d'une transformation d'un type de circuits vers un autre, seule l'existence importe. On peut néanmoins remarquer que les trois réductions des propositions précédentes peuvent être réalisées en temps linéaire par un simple parcours de l'arbre syntaxique de la formule. Avec un peu plus d'attention, on peut aussi constater que ces constructions sont faisables en temps parallèle $O(\log n)$ sur un nombre de processeurs $O(n)$ sur une machine EREW PRAM.

Il reste maintenant à démontrer que toute somme des poids des couvertures d'un graphe de largeur arborescente bornée peut s'exprimer comme une formule de taille polynomiale. C'est l'objet de la prochaine section.

3.2 Des couvertures des graphes de largeur arborescente bornée vers les formules

Courcelle, Makowsky et Rotics montrent que le permanent, le polynôme hamiltonien et la somme des couplages parfaits sont dans la classe de complexité VP pour les matrices de largeur arborescente bornée. En réalité ils montrent ce résultat pour une classe bien plus large de problèmes puisqu'ils montrent que tout problème d'évaluation définissable en logique MS_2 est dans VP pour les matrices de largeur arborescente bornée (La logique MS_2 est la logique monadique du second ordre sur la structure relationnelle ayant comme univers l'ensemble des sommets et des arêtes). Les preuves que nous donnons à présent sont plus directes et fournissent une caractérisation plus précise. Elles reposent sur l'utilisation d'un algorithme de programmation dynamique qui, étant donné une décomposition arborescente de largeur k et de profondeur d d'un graphe pondéré, construit un circuit de profondeur $O(k^2 \cdot d)$ calculant la somme des poids des couvertures.

Dans les constructions qui suivent, il est nécessaire de considérer des couvertures partielles qui généralisent les couvertures par circuit.

Définition 3.2.1
Une couverture partielle d'un graphe orienté est l'union de chemins orientés et de circuits telle que tout sommet du graphe appartient à au plus l'un des chemins ou circuits. Le poids d'une couverture partielle est le produit de ses arcs.

Le théorème suivant extrait de [8] est un outil standard dans la conception d'algorithmes parallèles pour les graphes de largeur arborescente bornée (Le lecteur intéressé peut aussi consulter [10] et [75]).

Théorème 3.2.2
Soit $G = (V, E)$ un graphe de largeur arborescente au plus k avec n sommets. Il existe une décomposition arborescente $\langle T, (X_t)_{t \in V_T} \rangle$ de G de largeur $3k + 2$ telle que $T = (V_T, E_T)$ est un arbre binaire de profondeur au plus $2\lceil log_{\frac{5}{4}}(2n) \rceil$.

On utilisera aussi le lemme standard suivant :

Lemme 3.2.3
Soit φ un circuit de profondeur d. Il existe une formule de profondeur d et de taille $O(2^d)$ représentant le même polynôme.

Preuve :

On construit la formule à partir du circuit en dupliquant tout un sous-circuit chaque fois que le résultat d'une porte est réutilisé. La formule construite de cette manière est aussi de hauteur d. Dans la formule produite, le nombre de portes à distance j de la racine est au plus le double du nombre de portes à distance $j - 1$ de la racine ; ainsi la formule a au plus $\sum_{i=0}^{d} 2^i = 2 \cdot 2^d - 1$ portes. ∎

Proposition 3.2.4
Le permanent d'une matrice M de dimension $n \times n$ et de largeur arborescente bornée par k peut s'exprimer comme une formule de taille $O(n^{O(1)})$.

Preuve :

La preuve consiste à construire un circuit de profondeur $O(log(n))$, qu'on peut alors exprimer comme une formule de taille $O(n^{O(1)})$ à l'aide du lemme 3.2.3. Considérons le graphe $G = G_M$ et appliquons le théorème 3.2.2 afin d'obtenir une décomposition arborescente binaire équilibrée T de largeur k'. À tout noeud t de T, on associe T_t le sous-arbre de T enraciné en t ; on note $X(T_t)$ l'ensemble des sommets de G qui appartiennent à X_u pour au moins un noeud u de T_t. On note G_t le sous-graphe de G induit par $X(T_t)$.

Considérons une couverture partielle C de G_t. Tout arc $(u, v) \in X_t^2$ est utilisé ou non par C. De même, tout sommet de X_t est de degré entrant et sortant 0 ou 1 dans C. On note $\lambda_t = I_t(C)$ la liste de ces informations pour tout arc $(u, v) \in X_t^2$ et tout sommet de X_t. Par abus de langage, on dira qu'un arc dans X_t^2 est utilisé par λ_t s'il est utilisé par une couverture partielle satisfaisant $I_t(C) = \lambda_t$ (ou de manière équivalente, par toute couverture partielle satisfaisant $I_t(C) = \lambda_t$).

Nous allons calculer pour chaque liste possible λ_t un poids $w(\lambda_t)$, défini comme la somme des poids de toutes les couvertures partielles C de G_t satisfaisant les trois propriétés suivantes :

(i) les deux extrémités de tout chemin de C appartiennent à X_t ;

(ii) tous les sommets non couverts appartiennent à X_t ;

(iii) $I_t(C) = \lambda_t$.

Notons que le nombre de poids à calculer en chaque noeud de T est borné par une constante (qui dépend de k'). Quand t est la racine de T, on peut aisément calculer le permanent de M à partir des poids $w(\lambda_t)$: il est égal à la somme des poids $w(\lambda_t)$ sur toutes les λ_t qui assignent un degré entrant et sortant 1 à tous les sommets de X_t. Quand t est une feuille de T, on peut calculer les poids en un nombre constant d'opérations puisque G_t a au plus k' sommets. Il reste donc à expliquer comment calculer les poids $w(\lambda_t)$ quand t n'est pas une feuille.

Notre algorithme procède ici de bas en haut : nous calculons les poids pour t à partir des poids déjà calculés pour son fils gauche (noté l) et son fils droit (noté r). L'idée étant que l'on peut obtenir une couverture partielle de G_t en prenant l'union d'une couverture partielle de G_l, d'une couverture partielle de G_r et en ajoutant des arcs supplémentaires. Réciproquement, une couverture partielle de G_t induit une couverture partielle de G_l ainsi qu'une couverture partielle de G_r. Afin d'éviter de compter plusieurs fois la même couverture partielle, nous devons définir ces couvertures partielles de G_l et G_r de sorte que toute couverture partielle de G_t induise une unique couverture partielle de G_l et une unique couverture partielle de G_r. Pour cela, on dira que (λ_l, λ_r) est compatible avec λ_t ssi les propriétés suivantes sont vérifiées :

- aucun arc de X_t^2 n'est utilisé dans λ_l ou λ_r ;
- pour tout sommet $x \in X_t$ au plus une des listes λ_l, λ_r lui assigne un degré entrant 1 ;
- pour tout sommet $x \in X_t$ au plus une des listes λ_l, λ_r lui assigne un degré sortant 1 ;
- pour tout sommet $x \in X_t$ si λ_l ou λ_r lui assigne un degré entrant 1 alors λ_t lui assigne un degré entrant 1 et aucun arc utilisé par λ_t n'est un arc entrant de x ;
- pour tout sommet $x \in X_t$ si λ_l ou λ_r lui assigne un degré sortant 1 alors λ_t lui assigne un degré sortant 1 et aucun arc utilisé par λ_t n'est un arc sortant de x ;
- tout sommet $x \in X_l \backslash X_t$ est de degré entrant et sortant 1 dans λ_l ;
- tout sommet $x \in X_r \backslash X_t$ est de degré entrant et sortant 1 dans λ_r.

Il nous faut à présent montrer deux choses. S'il existe une couverture partielle C de G_t satisfaisant les propriétés (i) et (ii) telle que $I_t(C) = \lambda_t$, alors elle induit une couverture partielle C_l de G_l ainsi qu'une couverture partielle C_r de G_r telles que C_l et C_r satisfont (i), (ii), $I_l(C_l) = \lambda_l$, $I_r(C_r) = \lambda_r$ et (λ_l, λ_r) est compatible avec λ_t. Réciproquement, si (λ_l, λ_r) est compatible avec λ_t et C_l, C_r sont des couvertures partielles de G_l, G_r satisfaisant (i), (ii), $I_l(C_l) = \lambda_l$ et $I_r(C_r) = \lambda_r$, alors il existe une unique couverture partielle C de G_t contenant C_l et C_r telle que $I_t(C) = \lambda_t$.

Considérons une couverture partielle C de G_t qui satisfait les propriétés (i) et (ii) définies ci-dessus. On peut assigner à C un unique triplet (C_l, C_r, S) défini comme suit. D'abord, on définit S comme l'ensemble des arcs de $C \cap X_t^2$. Puis on définit C_l comme l'ensemble des arcs de C qui ont leurs deux extrémités dans $X(T_l)$ et au moins l'une d'entre elles hors

de X_t. Enfin, on définit C_r comme l'ensemble des arcs de C qui ont leurs deux extrémités dans $X(T_r)$ et au moins l'une d'entre elles hors de X_t. Notons que $w(C) = w(C_l)w(C_r)w(S)$ puisque (C_l, C_r, S) est une partition des arcs de C. De plus, C_l est une couverture partielle de G_l et les propriétés (i) et (ii) sont satisfaites : les extrémités des chemins de C_l et les sommets non-couverts de $X(T_l)$ appartiennent tous à $X_l \cap X_t$. De même, C_r est une couverture partielle de $X(T_r)$ et les propriétés (i) et (ii) sont satisfaites. Si $I_l(C_l) = \lambda_l$ et $I_r(C_r) = \lambda_r$, il est clair que (λ_l, λ_r) est compatible avec λ_t. Toute autre partition de C en trois parties dont une couverture partielle de G_l, une couverture partielle de G_r et un sous-ensemble d'arcs dans X_t^2 aurait un arc de X_t^2 utilisé par C_l ou C_r. En conséquence (λ_l, λ_r) ne serait pas compatible avec λ_t.

Supposons à présent que (λ_l, λ_r) est compatible avec λ_t et C_l, C_r sont des couvertures partielles de G_l, G_r satisfaisant (i), (ii), $I_l(C_l) = \lambda_l$ et $I_r(C_r) = \lambda_r$. On définit S_{λ_t} comme l'ensemble des arcs de X_t^2 qui sont utilisés par λ_t. Il est clair que S_{λ_t}, C_l et C_r sont disjoints. Considérons $C = S_{\lambda_t} \cup C_l \cup C_r$. Puisque (λ_l, λ_r) est compatible avec λ_t, C est une couverture partielle satisfaisant (i) et (ii). Il est évident que C est la seule couverture partielle contenant C_l et C_r telle que $I_t(C) = \lambda_t$.

Ces considérations nous conduisent à l'égalité

$$w(\lambda_t) = \sum_{(\lambda_l, \lambda_r)} w(\lambda_l)w(\lambda_r)w(S_{\lambda_t}).$$

La somme est effectuée sur tous les couples (λ_l, λ_r) qui sont compatibles avec λ_t. Le poids $w(\lambda_t)$ peut donc être calculé par un nombre constant d'opérations arithmétiques. Puisque la hauteur de T est $O(log(n))$, l'algorithme ci-dessus produit un circuit lui aussi de hauteur $O(log(n))$. ■

Proposition 3.2.5
L'hamiltonien d'une matrice M de dimension $n \times n$ et de largeur arborescente bornée par k peut s'exprimer comme une formule de taille $O(n^{O(1)})$.

Preuve :

La preuve est extrêmement similaire à celle de la proposition précédente. On construit un circuit de profondeur $O(log(n))$ qui peut alors être exprimé par une formule de taille $O(n^{O(1)})$ à l'aide du lemme 3.2.3. La seule différence étant que l'on considère pour tout noeud de la décomposition arborescente, excepté la racine, des couvertures partielles ne contenant que des chemins. Au niveau de la racine de T, on combine les couvertures partielles des deux fils avec un ensemble d'arcs inclus dans le sac de la racine afin de former un cycle hamiltonien. Pour cela, on considère des listes d'informations λ_t qui spécifient pour chaque arc de X_t^2 son utilisation ou non et pour chaque sommet de X_t ses degrés entrant et sortant, ainsi que l'identifiant du chemin dont il est extrémité, si nécessaire.

On calcule alors pour chaque liste possible λ_t un poids $w(\lambda_t)$, défini comme la somme des poids de toutes les couvertures partielles C de G_t satisfaisant les quatre propriétés suivantes :

(i) les deux extrémités de tout chemin de C appartiennent à X_t et ont le même identifiant dans λ_t ;

(ii) tous les sommets non couverts appartiennent à X_t ;

(iii) C ne contient pas de circuits ;

(iv) $I_t(C) = \lambda_t$.

Cette fois, on dira que (λ_l, λ_r) est compatible avec λ_t ssi les propriétés suivantes sont vérifiées :

- aucun arc de X_t^2 n'est utilisé dans λ_l ou λ_r ;
- pour tout sommet $x \in X_t$ au plus une des listes λ_l, λ_r lui assigne un degré entrant 1 ;
- pour tout sommet $x \in X_t$ au plus une des listes λ_l, λ_r lui assigne un degré sortant 1 ;
- pour tout sommet $x \in X_t$ si λ_l ou λ_r lui assigne un degré entrant 1 alors λ_t lui assigne un degré entrant 1 et aucun arc utilisé par λ_t n'est un arc entrant de x ;
- pour tout sommet $x \in X_t$ si λ_l ou λ_r lui assigne un degré sortant 1 alors λ_t lui assigne un degré sortant 1 et aucun arc utilisé par λ_t n'est un arc sortant de x ;
- tout sommet $x \in X_l \backslash X_t$ est de degré entrant et sortant 1 dans λ_l ;
- tout sommet $x \in X_r \backslash X_t$ est de degré entrant et sortant 1 dans λ_r ;
- aucuns chemins de λ_l et λ_r n'ont les deux mêmes extrémités ;
- S_{λ_t} ne crée pas de circuits ;
- chaque fois que des chemins de λ_l et λ_r sont reliés sans former de circuit, les deux extrémités du chemin créé ont le même identifiant dans λ_t.

On obtient à nouveau l'égalité

$$w(\lambda_t) = \sum_{(\lambda_l, \lambda_r)} w(\lambda_l) w(\lambda_r) w(S_{\lambda_t}).$$

La somme est effectuée sur tous les couples (λ_l, λ_r) qui sont compatibles avec λ_t. Le poids $w(\lambda_t)$ peut donc être calculé par un nombre constant d'opérations arithmétiques. Puisque la hauteur de T est $O(log(n))$, l'algorithme ci-dessus produit un circuit lui aussi de hauteur $O(log(n))$. ∎

Proposition 3.2.6
La somme des poids des couplages parfaits d'une matrice symétrique M de dimension $n \times n$ et de largeur arborescente bornée par k peut s'exprimer comme une formule de taille $O(n^{O(1)})$.

Preuve :

La preuve est à nouveau très similaire à celles des deux propositions précédentes. On construit un circuit de profondeur $O(log(n))$ qui peut alors être exprimé par une formule de taille $O(n^{O(1)})$ à l'aide du lemme 3.2.3. Cette fois, on considère des couplages des sous-graphes induits au niveau des noeuds de la décomposition arborescente équilibrée.

Soit C un couplage de G_t. Toute arête $(u, v) \in X_t^2$ est utilisée ou non par C. De même, tout sommet de X_t est de degré 0 ou 1 dans C. On note $\lambda_t = I_t(C)$ la liste de ces informations pour toute arête $(u, v) \in X_t^2$ et tout sommet de X_t.

On calcule pour chaque liste possible λ_t un poids $w(\lambda_t)$, défini comme la somme des poids de tous les couplages C de G_t satisfaisant les deux propriétés suivantes :

(i) tous les sommets non couverts appartiennent à X_t ;

(ii) $I_t(C) = \lambda_t$.

Quand t est la racine de T, on peut facilement calculer la somme des poids des couplages parfaits de M à partir des poids $w(\lambda_t)$: elle est égale à la somme des $w(\lambda_t)$ sur tous les λ_t qui assignent un degré 1 à tous les sommets de X_t.

On dit que (λ_l, λ_r) est compatible avec λ_t ssi les propriétés suivantes sont vérifiées :

- aucune arête de X_t^2 n'est utilisée dans λ_l ou λ_r ;
- pour tout sommet $x \in X_t$ au plus une des listes λ_l, λ_r lui assigne un degré 1 ;
- pour tout sommet $x \in X_t$ si λ_l ou λ_r lui assigne un degré 1 alors λ_t lui assigne un degré 1 et aucune arête utilisée par λ_t n'est incidente à x ;
- tout sommet $x \in X_l \backslash X_t$ est de degré 1 dans λ_l ;
- tout sommet $x \in X_r \backslash X_t$ est de degré 1 dans λ_r.

Une fois de plus, on obtient l'égalité

$$w(\lambda_t) = \sum_{(\lambda_l, \lambda_r)} w(\lambda_l) w(\lambda_r) w(S_{\lambda_t}).$$

La somme est effectuée sur tous les couples (λ_l, λ_r) qui sont compatibles avec λ_t. Le poids $w(\lambda_t)$ peut donc être calculé par un nombre constant d'opérations arithmétiques. Puisque la hauteur de T est $O(log(n))$, l'algorithme ci-dessus produit un circuit lui aussi de hauteur $O(log(n))$. ∎

On aurait aussi pu commencer par démontrer la proposition 3.2.6 pour en déduire la proposition 3.2.4 en considérant le biparti d'adjacence comme dans la preuve de la proposition 3.1.3.

On peut à nouveau remarquer que l'algorithme de programmation dynamique donné dans les trois preuves peut s'exécuter en temps séquentiel $O(n)$ ou en temps parallèle $O(d)$ sur un nombre de processeurs $O(n)$ sur une machine EREW PRAM. De plus, si G est de largeur arborescente k, on peut trouver

une décomposition arborescente binaire équilibrée de largeur $O(k)$ en temps séquentiel linéaire ou en temps parallèle $O(\log^2 n)$ sur un nombre de processeurs $O(n)$ sur une machine EREW PRAM (voir [10]). En utilisant ces résultats et la proposition 3.1.1 par exemple, on obtient le corollaire suivant.

Corollaire 3.2.7
Toute expression arithmétique de taille n peut être transformée en un circuit arithmétique de profondeur $O(\log n)$ en temps séquentiel linéaire ou en temps parallèle $O(\log^2 n)$ sur un nombre de processeurs $O(n)$ sur une machine EREW PRAM.

La parallélisation des expressions arithmétiques est un résultat bien connu démontré pour la première fois par Brent en 1974 dans [13]. D'ailleurs le résultat optimal de Brent est bien meilleur que celui que nous obtenons « à l'aide d'un marteau-pilon pour écraser une mouche » puisqu'il montre que toute expression arithmétique est parallélisable en temps $O(\log n)$ sur un nombre de processeurs $O(\frac{n}{\log n})$. Remarquons qu'on peut s'en rapprocher un peu plus en se ramenant à un temps parallèle $O(\log n \log^* n)$ en utilisant l'algorithme parallèle de calcul d'une décomposition arborescente pour les graphes séries-parallèles ou de largeur arborescente 2 donné dans [11].

Dans un cadre non-uniforme, $\mathsf{VNC}^1 \subseteq \mathsf{VP_e}$ découle directement du lemme 3.2.3 et nous venons de redémontrer que $\mathsf{VP_e} \subseteq \mathsf{VNC}^1$. Le lecteur vient donc d'avoir la preuve à son insu que $\mathsf{VP_e} = \mathsf{VNC}^1$.

Dans ce chapitre, nous avons démontré que les trois types de couvertures de graphe considérés ont la même expressivité algébrique quand on se restreint aux graphes de largeur arborescente bornée ; ils permettent d'exprimer les formules arithmétiques. Ce comportement uniforme est trompeur puisque les classes de complexité exprimées par ces couvertures peuvent être radicalement différentes (sauf effondrement généralisé des classes de complexité usuelles uniformes et algébriques non-uniformes) ; nous en verrons un exemple quand il s'agira d'étudier leur expressivité sur les graphes planaires.

Chapitre 4

Expressivité des couvertures de graphes de largeur linéaire bornée

Nous nous intéressons à présent au pouvoir expressif du permanent, de l'hamiltonien et des couplages parfaits pour les matrices de largeur linéaire bornée. Nous montrons que dans chaque cas nous capturons exactement les familles de polynômes calculées par des circuits asymétriques de taille polynomiale et de largeur bornée. Un sous-produit de ces preuves est la démonstration de l'équivalence entre les circuits asymétriques de taille polynomiale et de largeur bornée et les circuits faiblement asymétriques de taille polynomiale et de largeur bornée, cette équivalence ne pouvant être immédiatement déduite de l'équivalence déjà connue entre circuits asymétriques de taille polynomiale et circuits faiblement asymétriques de taille polynomiale, dans le cas où la largeur n'est pas bornée.

Nous terminons ce chapitre en démontrant que les circuits asymétriques de taille polynomiale et de largeur bornée sont équivalents aux formules arithmétiques de taille polynomiale. Ceci nous montre qu'il y a un effondrement relatif de la complexité entre les classes de largeur linéaire bornée et celles de largeur arborescente bornée. Bien entendu, cet effondrement n'est obtenu que relativement aux opérateurs permanent et hamiltonien.

Si certaines des preuves de ce chapitre ressemblent parfois grandement à celles du chapitre précédent, elles nécessitent un peu plus de soin car la marge de manœuvre est plus faible qu'avec la largeur arborescente bornée.

Nous démontrons le théorème suivant quasi-identique au théorème 3.0.1.

Théorème 4.0.1
Soit (f_n) une famille de polynômes à coefficients dans un corps $\mathbb{K}$. Les six propriétés suivantes sont équivalentes :

- *(f_n) peut être représentée par une famille de formules arithmétiques de taille polynomiale.*

- *(f_n) peut être représentée par une famille de circuits asymétriques de taille polynomiale et de largeur bornée.*
- *(f_n) peut être représentée par une famille de circuits faiblement asymétriques de taille polynomiale et de largeur bornée.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur linéaire bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{per}(M_n)$.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur linéaire bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{ham}(M_n)$.*
- *Il existe une famille (M_n) de matrices symétriques de taille polynomiale et de largeur linéaire bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \sum_{P \in \mathcal{P}(M_n)} W(P)$, où $\mathcal{P}(M_n)$ est l'ensemble des couplages parfaits de G_{M_n}.*

4.1 Des circuits de largeur bornée vers les couvertures des graphes de largeur linéaire bornée

Bien que l'on imagine aisément ce qu'est un circuit de largeur bornée, il nous faut tout de même donner sa définition formelle.

Définition 4.1.1
Un circuit arithmétique φ est de largeur bornée $k \geqslant 1$ *s'il existe un ensemble fini de niveaux totalement ordonnés tel que :*

- *Chaque porte de φ est contenue dans exactement 1 niveau.*
- *Chaque niveau contient au plus k portes.*
- *Pour toute porte qui n'est pas une entrée de φ, si elle est dans le niveau i, alors ses deux entrées sont dans le niveau $i+1$.*

Nous pouvons à présent démontrer que tout circuit faiblement asymétrique de largeur bornée peut être exprimé au moyen d'un des trois types de couvertures sur les classes de graphes de largeur linéaire bornée.

Proposition 4.1.2
Le polynôme calculé par un circuit faiblement asymétrique de largeur bornée peut être exprimé comme le permanent d'une matrice de largeur linéaire bornée. La taille de cette matrice est polynomiale en la taille du circuit. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables du polynôme.

Preuve :

Soient φ un circuit faiblement asymétrique de largeur bornée $k \geqslant 1$ et $l > 1$ le nombre de niveaux de φ. Le graphe orienté G que nous allons construire est de largeur linéaire au plus $\lfloor \frac{7 \cdot k}{2} \rfloor - 1$ (tout sac de la décomposition linéaire contiendra au plus $\lfloor \frac{7 \cdot k}{2} \rfloor$ sommets) et sa décomposition

linéaire comprendra $l-1$ sacs. G aura deux sommets distingués s et t et la somme des poids de tous les chemins orientés de s à t sera égale à la valeur calculée par φ. Le sommet s sera présent dans tous les sacs de la décomposition linéaire de G.

Puisque φ est un circuit faiblement asymétrique, nous considérons une décomposition de φ en sous-circuits disjoints définis récursivement comme suit : la porte de sortie de φ appartient au *sous-circuit principal*. Si une porte du sous-circuit principal est une porte d'addition, alors ses deux entrées sont dans le sous-circuit principal. Si une porte g du sous-circuit principal est une porte de multiplication, alors l'une au moins de ses entrées est la porte de sortie d'un sous-circuit disjoint du reste de φ excepté au niveau de sa connexion avec g. Ce sous-circuit forme un *sous-circuit multiplicatif disjoint*. L'autre entrée de g appartient au sous-circuit principal. Si l'un des sous-circuits multiplicatifs disjoints φ' contient au moins une porte de multiplication, alors on décompose récursivement φ'. Notons qu'une telle décomposition n'est pas nécessairement unique (ce qui n'est d'ailleurs pas requis), car les deux entrées d'une porte de multiplication peuvent être disjointes du reste du circuit simultanément ; donc, dans ce cas de figure, l'une ou l'autre peut être choisie comme celle qui appartient au sous-circuit principal.

Soient $\varphi_0, \varphi_1, \ldots, \varphi_d$ les sous-circuits disjoints obtenus dans la décomposition (φ_0 est le sous-circuit principal). Le graphe G possèdera un sommet v_g pour toute porte g de φ et $d+1$ sommets additionnels $s = s_0, s_1, \ldots, s_d$ (t correspondra à v_g où g est la porte de sortie de φ). Pour toute porte g dans le sous-circuit φ_i, la construction suivante sera telle que la somme des poids des chemins orientés de s_i à v_g est égale à la valeur calculée en g dans φ.

Pour construire G, nous procédons de bas en haut sur la *décomposition* de φ en sous-circuits. Soit φ_i un sous-circuit, feuille de cette décomposition de φ (ainsi φ_i ne contient que des portes d'addition ou d'entrée). Supposons que φ_i est présent dans les niveaux top_i à bot_i ($1 \geqslant top_i \geqslant bot_i \geqslant l$) de φ. Nous ajoutons, en premier lieu, un sommet s_i à G dans le sac $bot_i - 1$ et, pour chaque porte d'entrée de valeur w dans le niveau plancher bot_i de φ_i, nous ajoutons un sommet à G, à nouveau dans le sac $bot_i - 1$, avec un arc de poids w de s_i à ce sommet. Soit n allant de $bot_i - 1$ à top_i : il faut ajouter le sommet existant s_i au sac $n-1$ et traiter les portes d'entrée de φ_i dans le niveau n comme décrit précédemment. Pour chaque porte d'addition de φ_i dans le niveau n, nous ajoutons un nouveau sommet à G (qui est ajouté aux sacs n et $n-1$ de la décomposition linéaire de G). Dans le sac n, nous avons déjà deux sommets qui représentent les entrées de cette porte d'addition ; nous ajoutons donc des arcs de poids 1 depuis ceux-ci vers le sommet nouvellement créé. Le sommet représentant la porte de sortie du circuit φ_i est notée t_i. La somme des poids des chemins orientés de s_i à t_i égale la valeur calculée par le sous-circuit φ_i.

Soit φ_i un sous-circuit de la décomposition de φ qui contient des portes de multiplication. On suppose que tous les sous-circuits plus bas dans le sous-arbre de la décomposition enraciné en φ_i ont déjà été traités. Les portes d'addition et d'entrée de φ_i sont traitées comme avant. Soit g une

porte de multiplication de φ_i dans le niveau n et φ_j le sous-circuit multiplicatif disjoint qui est l'une des entrées de g. Nous savons que les sommets s_j et t_j sont déjà présents dans le sac n, donc nous ajoutons un arc de poids 1 depuis le sommet représentant l'autre entrée de g vers le sommet s_j, ainsi qu'un arc de poids 1 de t_j vers un nouveau sommet v_g qui représente la porte g ; v_g est ajouté aux sacs n et $n-1$.

Pour tout b ($1 \geqslant b \geqslant l-1$), nous devons montrer qu'un nombre borné par une constante de sommets est ajouté au sac b durant toute la construction. Toute porte dans le niveau b de φ est représentée par un sommet et tous ces sommets doivent être ajoutés au sac b. Toute porte du niveau $b+1$ est aussi représentée par un sommet qui est ajouté au sac b (car ils sont utilisés ici comme entrée). Jusqu'à présent nous avons au plus $2 \cdot k$ sommets de porte dans chaque sac. À ceux-ci s'ajoutent un certain nombre de sommets s_i dans le sac b. Pour chaque sous-circuit φ_j qui possède une porte dans le niveau b ou $b+1$, nous avons le sommet s_j correspondant dans le sac b, ; il reste donc à montrer qu'au plus $\lfloor \frac{3 \cdot k}{2} \rfloor$ sous-circuits disjoints ont une porte dans le niveau b ou $b+1$. Chacun de ces sous-circuits est dans exactement un des 3 ensembles suivants :

C_1 : Sous-circuits qui ont une porte dans le niveau b, mais AUCUNE n'est une porte de multiplication.

C_2 : Sous-circuits qui ont une porte de multiplication dans le niveau b.

C_3 : Sous-circuits dont la racine est dans le niveau $b+1$.

Il y a au plus $\lfloor \frac{k}{2} \rfloor$ sous-circuits dans l'ensemble C_2. Sinon, puisque deux entrées d'une porte de multiplication sont dans des sous-circuits différents et puisque les sous-circuits de C_2 sont disjoints, le niveau $b+1$ devrait contenir au moins $2 \cdot (\lfloor \frac{k}{2} \rfloor + 1)$ portes et donc être de largeur supérieure à k. De par la construction des sous-circuits, tous les sous-circuits de C_3 sont considérés comme les sous-circuits multiplicatifs disjoints de portes de multiplication distinctes dans le niveau b, ainsi il y a au plus $|C_3|$ portes de multiplication dans le niveau b. Puisque les sous-circuits dans C_1 n'ont pas de porte de multiplication dans le niveau b, on obtient que $|C_1| + |C_3| \leqslant k$. Ainsi, au plus $|C_1| + |C_2| + |C_3| \leqslant \lfloor \frac{3 \cdot k}{2} \rfloor$ sous-circuits distincts ont leur sommet s_i ajouté au sac b. Au final, jusqu'à $\lfloor \frac{7 \cdot k}{2} \rfloor$ sommets sont ajoutés au sac b. (Cette borne est optimale pour la construction du graphe proposée. En effet, il suffit de mettre $\lfloor \frac{k}{2} \rfloor$ portes de multiplication – toutes dans des sous-circuits différents – et $\lceil \frac{k}{2} \rceil$ entrées dans le niveau b, ainsi que les entrées des portes de multiplication au niveau $b+1$.)

Remarquons que dans le niveau 1 de φ on a seulement la porte de sortie. Cette porte est représentée par le sommet t de G (qui est dans le sac 1 de la décomposition linéaire).

On peut montrer par induction que la somme des poids de tous les chemins orientés de s à t dans G est égale à la valeur calculée par le circuit φ. Le dernier détail de la réduction vers le permanent est d'ajouter un arc retour de poids 1 de t vers s et des boucles de poids 1 sur tous les noeuds distincts de s et t.

■

La preuve de la proposition 4.1.2 peut être modifiée pour démontrer la proposition suivante.

Proposition 4.1.3
Le polynôme calculé par un circuit faiblement asymétrique de largeur bornée peut être exprimé comme l'hamiltonien d'une matrice de largeur linéaire bornée. La taille de cette matrice est polynomiale en la taille du circuit. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables du polynôme.

En effet, il suffit d'adapter à nouveau la preuve d'universalité de l'hamiltonien de [52]. Dans le cas du permanent, chaque sac de la décomposition linéaire contenait au plus $\lfloor \frac{7 \cdot k}{2} \rfloor$ sommets ; pour chacun de ces sommets il nous faut introduire un sommet supplémentaire dans le même sac. De plus chaque sac doit contenir 2 sommets additionnels afin de connecter les sacs adjacents de la décomposition linéaire. Au total chaque sac contient à présent au plus $7 \cdot k + 2$ sommets.

Proposition 4.1.4
Le polynôme calculé par un circuit faiblement asymétrique de largeur bornée peut être exprimé comme la somme des poids des couplages parfaits d'une matrice symétrique de largeur linéaire bornée. La taille de cette matrice est polynomiale en la taille du circuit. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables du polynôme.

Preuve :

C'est une conséquence directe des propositions 4.1.2 et 2.3.4, si l'on considère le biparti d'adjacence du graphe G construit ci-dessus. ∎

4.2 Des couvertures des graphes de largeur linéaire bornée vers les circuits de largeur bornée

Les preuves de cette sous-section ressemblent beaucoup à celles de la sous-section 3.2 du chapitre précédent.

Définition 4.2.1 (couverture par chemins)
Une couverture par chemins *d'un graphe orienté G est un sous-ensemble de ses arcs tel que ces arcs forment des chemins orientés disjoints non-cycliques de G. Nous avons besoin que chaque sommet de G soit dans exactement un chemin. Nous autorisons donc les « chemins » de longueur 0 commençant et terminant sur le même sommet. Ces chemins n'ont* pas *la même interprétation qu'une boucle. Le* poids *d'une couverture par chemins est le produit du poids de ses arcs (Dans le cas spécial où la couverture par chemins ne comporte pas d'arcs, le poids est défini comme valant 1).*

Proposition 4.2.2
L'hamiltonien d'une matrice de largeur linéaire bornée peut être calculé par un circuit asymétrique de largeur bornée dont la taille est polynomiale en celle de la matrice.

Preuve :

Soient M une matrice de largeur linéaire bornée k et G_M le graphe orienté ayant M comme matrice d'adjacence. Chaque sac de la décomposition linéaire T de G_M contient au plus $k+1$ sommets. On choisit l'une des extrémités de T que l'on désigne par la *feuille* de T tandis que l'autre extrémité est appelée la *racine*. À tout noeud t de T, on associe T_t le sous-chemin de T enraciné en t ; on note $X(T_t)$ l'ensemble des sommets de G qui appartiennent à X_u pour au moins un noeud u de T_t. On note G_t le sous-graphe de G induit par $X(T_t)$.

Considérons une couverture par chemins C de G_t. Tout arc $(u, v) \in X_t^2$ est utilisé ou non par C. De même, tout sommet de X_t est de degré entrant et sortant 0 ou 1 dans C. De plus, on peut donner un même identifiant de chemin de C à ses deux extrémités. On note $\lambda_t = I_t(C)$ la liste de ces informations pour tout arc $(u, v) \in X_t^2$ et tout sommet de X_t. Par abus de langage, on dira qu'un arc dans X_t^2 est utilisé par λ_t s'il est utilisé par une couverture par chemins satisfaisant $I_t(C) = \lambda_t$ (ou de manière équivalente, par toute couverture par chemins satisfaisant $I_t(C) = \lambda_t$). On définit S_{λ_t} comme l'ensemble des arcs de X_t^2 qui sont utilisés par λ_t.

Nous allons calculer pour chaque liste possible λ_t un poids $w(\lambda_t)$, défini comme la somme des poids de toutes les couvertures par chemins C de G_t satisfaisant les trois propriétés suivantes :

(i) les deux extrémités de tout chemin de C appartiennent à X_t et ont le même identifiant dans λ_t ;

(ii) tous les sommets couverts par des chemins de longueur 0 appartiennent à X_t ;

(iii) $I_t(C) = \lambda_t$.

Remarquons qu'on aurait pu se passer de rappeler la propriété (ii) puisqu'elle est une conséquence de la propriété (i).

Notons que le nombre de poids à calculer en chaque noeud de T est borné par une constante (qui dépend de k). Quand t est la racine de T, on peut aisément calculer l'hamiltonien de M à partir des poids $w(\lambda_t)$: il suffit, pour tout ensemble d'arcs de X_t^2 qui complète en un circuit hamiltonien les couvertures par chemins de description λ_t, de multiplier la valeur $w(\lambda_t)$ par les poids de ces arcs au moyen d'un arbre (chemin) de multiplications asymétriques, puis de faire la somme sur toutes les nouvelles valeurs calculées. Quand t est une feuille de T, on peut calculer les poids en un nombre constant d'opérations puisque G_t a au plus k sommets. Il reste donc à expliquer comment calculer les poids $w(\lambda_t)$ quand t n'est pas une feuille.

Notre algorithme procède ici de bas en haut : nous calculons les poids pour t à partir des poids déjà calculés pour son fils noté t'. L'idée étant que l'on peut obtenir une couverture par chemins de G_t en prenant une couverture par chemins de $G_{t'}$ et en ajoutant des arcs de X_t^2. Réciproquement, une couverture par chemins de G_t induit une couverture par chemins de $G_{t'}$. Afin d'éviter de compter plusieurs fois la même couverture par chemins, nous devons définir cette couverture par chemins de $G_{t'}$ de sorte que toute

couverture par chemins de G_t induise une unique couverture par chemins de $G_{t'}$. Pour cela, on dira que λ_t et $\lambda_{t'}$ sont compatibles si et seulement si les propriétés suivantes sont vérifiées :

- aucun arc de X_t^2 n'est utilisé dans $\lambda_{t'}$;
- pour tout sommet $x \in X_t$, si $\lambda_{t'}$ lui assigne un degré entrant 1, alors λ_t lui assigne un degré entrant 1 et aucun arc utilisé par λ_t n'est un arc entrant de x ;
- pour tout sommet $x \in X_t$, si $\lambda_{t'}$ lui assigne un degré sortant 1, alors λ_t lui assigne un degré sortant 1 et aucun arc utilisé par λ_t n'est un arc sortant de x ;
- tout sommet $x \in X_{t'} \backslash X_t$ est de degré entrant et sortant 1 dans $\lambda_{t'}$;
- S_{λ_t} ne crée pas de circuits ;
- les identifiants de chemin des extrémités d'un chemin de λ_t sont en accord avec les identifiants de chemin de $\lambda_{t'}$ et les arcs ajoutés S_{λ_t}.

Il nous faut à présent montrer deux choses. S'il existe une couverture par chemins C de G_t satisfaisant les propriétés (i) et (ii) telle que $I_t(C) = \lambda_t$, alors elle induit une couverture par chemins C' de $G_{t'}$ qui satisfait (i), (ii), $I_{t'}(C') = \lambda_{t'}$ et $\lambda_{t'}$ est compatible avec λ_t. Réciproquement, si $\lambda_{t'}$ est compatible avec λ_t et C' est une couverture par chemins de $G_{t'}$ satisfaisant (i), (ii) et $I_{t'}(C') = \lambda_{t'}$, alors il existe une unique couverture par chemins C de G_t contenant C' telle que $I_t(C) = \lambda_t$.

Considérons une couverture par chemins C de G_t qui satisfait les propriétés (i) et (ii) définies ci-dessus. On peut assigner à C un unique couple (C', S) défini comme suit. D'abord, on définit S comme l'ensemble des arcs de $C \cap X_t^2$. Puis on définit C' comme l'ensemble des arcs de C qui ont leurs deux extrémités dans $X(T_{t'})$ et au moins l'une d'entre elles hors de X_t. Notons que $w(C) = w(C')w(S)$ puisque (C', S) est une partition des arcs de C. De plus, C' est une couverture par chemins de $G_{t'}$ et les propriétés (i) et (ii) sont satisfaites : les extrémités des chemins de C' appartiennent toutes à $X_{t'} \cap X_t$. Si $I_{t'}(C') = \lambda_{t'}$, il est clair que $\lambda_{t'}$ est compatible avec λ_t. Toute autre partition de C en deux parties dont une couverture par chemins de $G_{t'}$ et un sous-ensemble d'arcs dans X_t^2 aurait un arc de X_t^2 utilisé par C'. En conséquence $\lambda_{t'}$ ne serait pas compatible avec λ_t.

Supposons à présent que $\lambda_{t'}$ est compatible avec λ_t et C' est une couverture par chemins de $G_{t'}$ satisfaisant (i), (ii), $I_{t'}(C') = \lambda_{t'}$. Il est clair que S_{λ_t} et C' sont disjoints. Considérons $C = S_{\lambda_t} \cup C'$. Puisque $\lambda_{t'}$ est compatible avec λ_t, C est une couverture par chemins satisfaisant (i) et (ii). Il est évident que C est la seule couverture par chemins contenant C' telle que $I_t(C) = \lambda_t$.

Ces considérations nous conduisent à l'égalité

$$w(\lambda_t) = \sum_{\lambda_{t'}} w(\lambda_{t'}) w(S_{\lambda_t}).$$

La somme est effectuée sur toutes les descriptions $\lambda_{t'}$ qui sont compatibles avec λ_t. Le poids $w(\lambda_t)$ peut donc être calculé par un nombre constant d'opérations arithmétiques.

Il est clair que chaque produit $w(\lambda_{t'})w(S_{\lambda_t})$ peut être calculé au moyen d'un arbre de multiplications asymétriques qui multiplie successivement $w(\lambda_{t'})$ aux poids de tous les arcs de S_{λ_t}.

Enfin, on se convainc aisément que le circuit construit en suivant la décomposition linéaire est de largeur bornée. ■

Proposition 4.2.3
La somme des poids des couplages parfaits d'une matrice symétrique de largeur linéaire bornée peut être calculée par un circuit asymétrique de largeur bornée dont la taille est polynomiale en celle de la matrice.

Preuve :

La preuve est quasi-identique à celle de la proposition précédente. Cette fois, on considère des couplages des sous-graphes induits au niveau des noeuds de la décomposition linéaire.

Soit C un couplage de G_t. Toute arête $(u, v) \in X_t^2$ est utilisée ou non par C. De même, tout sommet de X_t est de degré 0 ou 1 dans C. On note $\lambda_t = I_t(C)$ la liste de ces informations pour toute arête $(u, v) \in X_t^2$ et tout sommet de X_t.

On calcule pour chaque liste possible λ_t un poids $w(\lambda_t)$, défini comme la somme des poids de tous les couplages C de G_t satisfaisant les deux propriétés suivantes :

(i) tous les sommets non couverts appartiennent à X_t ;
(ii) $I_t(C) = \lambda_t$.

Quand t est la racine de T, on peut facilement calculer la somme des poids des couplages parfaits de M à partir des poids $w(\lambda_t)$: elle est égale à la somme des $w(\lambda_t)$ sur tous les λ_t qui assignent un degré 1 à tous les sommets de X_t.

On dit que $\lambda_{t'}$ est compatible avec λ_t si et seulement si les propriétés suivantes sont vérifiées :

- aucune arête de X_t^2 n'est utilisée dans $\lambda_{t'}$;
- pour tout sommet $x \in X_t$, si $\lambda_{t'}$ lui assigne un degré 1, alors λ_t lui assigne un degré 1 et aucune arête utilisée par λ_t n'est incidente à x ;
- tout sommet $x \in X_{t'} \backslash X_t$ est de degré 1 dans $\lambda_{t'}$.

On obtient à nouveau l'égalité

$$w(\lambda_t) = \sum_{\lambda_{t'}} w(\lambda_{t'})w(S_{\lambda_t}).$$

La somme étant toujours effectuée sur toutes les descriptions $\lambda_{t'}$ qui sont compatibles avec λ_t.

Il est clair ici aussi que chaque produit $w(\lambda_{t'})w(S_{\lambda_t})$ peut être calculé au moyen d'un arbre de multiplications asymétriques qui multiplie successivement $w(\lambda_{t'})$ aux poids de tous les arcs de S_{λ_t}. Et l'on se convainc aussi aisément que le circuit construit en suivant la décomposition linéaire est de largeur bornée. ■

Proposition 4.2.4
Le permanent d'une matrice de largeur linéaire bornée peut être calculé par un circuit asymétrique de largeur bornée dont la taille est polynomiale en celle de la matrice.

Preuve :

Si M est la matrice de largeur linéaire bornée dont on veut calculer le permanent, alors en considérant le biparti d'adjacence de G_M, il est évident que cette proposition est une conséquence directe des propositions 4.2.3 et 2.3.4. ■

4.3 Relations entre les différents circuits de largeur bornée et les formules

Commençons par constater l'équivalence entre les circuits asymétriques et faiblement asymétriques qui découle des preuves précédentes de ce chapitre.

Corollaire 4.3.1
Une famille de polynômes est calculée par des circuits asymétriques de taille polynomiale et de largeur bornée si et seulement si elle est calculée par des circuits faiblement asymétriques de taille polynomiale et de largeur bornée.

Preuve :

Il est trivial de voir qu'une famille de polynômes calculée par des circuits asymétriques de taille polynomiale et de largeur bornée est calculée par des circuits faiblement asymétriques de taille polynomiale et de largeur bornée. Réciproquement, si une famille de polynômes est calculée par des circuits faiblement asymétriques de taille polynomiale et de largeur bornée alors, d'après la proposition 4.1.2, elle peut s'exprimer comme les permanents de graphes de taille polynomiale et de largeur linéaire bornée qui peuvent à leur tour être calculés par des circuits asymétriques de taille polynomiale et de largeur bornée, grâce à la proposition 4.2.4. ■

À présent, nous avons besoin du théorème suivant dû à Ben-Or et Cleve dans [6] (voir aussi [7]) afin de prouver l'équivalence entre circuits asymétriques de taille polynomiale et de largeur bornée et formules de taille polynomiale.

Théorème 4.3.2
Toute formule arithmétique de taille polynomiale peut être calculée par un « linear bijection straight-line program » de taille polynomiale qui utilise trois registres.

Soit $R_1, \ldots, R_m$ un ensemble de m registres, un « linear bijection straight-line (LBS) program » ou programme LBS est un vecteur de m valeurs initiales attribuées aux registres plus une suite d'instructions de la forme

(i) $R_j \leftarrow R_j + (R_i \times c)$ ou

(ii) $R_j \leftarrow R_j - (R_i \times c)$ ou

(iii) $R_j \leftarrow R_j + (R_i \times x_u)$ ou

(iv) $R_j \leftarrow R_j - (R_i \times x_u)$,

où $1 \leqslant i, j \leqslant m$, $i \neq j$, $1 \leqslant u \leqslant n$, c est une constante et $x_1, \ldots, x_n$ sont des variables (n est le nombre de variables). La taille d'un programme LBS est définie comme étant son nombre d'instructions. On suppose sans perte de généralité que la valeur calculée par un programme LBS est la valeur présente dans le premier registre après l'exécution de toutes les instructions.

Théorème 4.3.3
Une famille de polynômes est calculée par des circuits asymétriques de taille polynomiale et de largeur bornée si et seulement si c'est une famille de formules arithmétiques de taille polynomiale.

Preuve :

Soit (f_n) une famille de polynômes calculée par des circuits asymétriques de taille polynomiale et de largeur bornée, alors d'après la proposition 4.1.2 elle peut s'exprimer comme les permanents de graphes de taille polynomiale et de largeur linéaire bornée. Puisque les graphes de largeur linéaire bornée sont de largeur arborescente bornée, on sait par la proposition 3.2.4 qu'ils peuvent être calculés par une famille de formules arithmétiques de taille polynomiale.

Réciproquement, soit (f_n) une famille de formules arithmétiques de taille polynomiale. D'après le théorème 4.3.2, elle est calculée par une famille de programmes LBS de taille polynomiale qui utilisent trois registres. Nous allons modifier ces programmes afin d'obtenir des circuits asymétriques de largeur 6 équivalents. À chaque étape, l'ensemble des indices $\{i, j, k\}$ sera égal à $\{1, 2, 3\}$.

Supposons que les valeurs initiales des trois registres sont r_1, r_2, r_3, alors le premier niveau de notre circuit asymétrique contient trois portes d'entrée ayant les trois valeurs r_1, r_2, r_3 ainsi que deux autres entrées qui seront définies selon la prochaine instruction.

Si l'instruction suivante est $R_j \leftarrow R_j + (R_i \times U)$ où U est une variable ou une constante, alors nous donnons les valeurs 0 et U aux deux entrées non définies dans le niveau courant l et nous créons un nouveau niveau $l-1$ avec trois portes d'addition correspondant à R_i, R_j, R_k dont les entrées sont la porte correspondant à R_i (resp. R_j, R_k) dans le niveau l et l'entrée de valeur 0 dans le niveau l. On met aussi une porte de multiplication dont les entrées sont la porte correspondant à R_i et l'entrée de valeur U du niveau l. Et nous rajoutons une entrée de valeur 0. Puis nous créons un nouveau niveau $l-2$ avec trois portes d'addition correspondant à R_i, R_j, R_k dont les entrées sont la porte correspondant à R_i (resp. R_j, R_k) et l'entrée de valeur 0 pour i, k ou la porte calculant $(R_i \times U)$ pour j dans le niveau $l-1$. On met aussi deux autres entrées qui seront définies selon la prochaine instruction.

Si l'instruction suivante est $R_j \leftarrow R_j - (R_i \times U)$, il nous faut créer un niveau de plus que dans le premier cas. Nous attribuons d'abord les valeurs 0 et U aux deux entrées non définies dans le niveau courant l et nous créons un nouveau niveau $l-1$ avec trois portes d'addition correspondant à R_i, R_j, R_k dont les entrées sont la porte correspondant à R_i (resp. R_j, R_k) dans le niveau l et l'entrée de valeur 0 dans le niveau l. On met aussi une porte de multiplication dont les entrées sont la porte correspondant à R_i et l'entrée de valeur U du niveau l. Et nous rajoutons à nouveau une entrée de valeur 0 ainsi qu'une autre de valeur -1. Puis nous créons un nouveau niveau intermédiaire $l-2$ avec trois portes d'addition correspondant à R_i, R_j, R_k dont les entrées sont la porte correspondant à R_i (resp. R_j, R_k) et la porte de valeur 0. On met aussi une porte de multiplication dont les entrées sont la porte calculant $(R_i \times U)$ et l'entrée de valeur -1 du niveau $l-1$. Et nous rajoutons encore une entrée de valeur 0. Finalement on crée un nouveau niveau $l-3$ avec trois portes d'addition correspondant à R_i, R_j, R_k dont les entrées sont la porte correspondant à R_i (resp. R_j, R_k) et l'entrée de valeur 0 pour i, k ou la porte calculant $-(R_i \times U)$ pour j dans le niveau $l-2$. On met aussi deux autres entrées qui seront définies selon la prochaine instruction.

Dans les deux cas, il est clair par induction que les trois portes du niveau courant correspondant à R_i, R_j, R_k calculent les valeurs de ces registres si l'on exécute les instructions traitées jusqu'à présent. On en déduit le résultat.

■

Les preuves de ce chapitre nous ont permis de démontrer l'égalité de deux classes de complexité, en l'occurrence les formules arithmétiques de taille polynomiale et les circuits (faiblement) asymétriques de largeur bornée et de taille polynomiale.

Ces résultats montrent aussi, combinés avec ceux du chapitre précédent, que le permanent et l'hamiltonien sont des opérateurs trop grossiers pour distinguer les graphes de largeur linéaire bornée des graphes de largeur arborescente bornée. Il serait intéressant de trouver un opérateur algébrique, s'il en existe un, pour lequel ces deux classes de graphes n'engendrent pas les mêmes familles de polynômes.

Il semble également naturel de se poser la question de la complexité algébrique des formules de largeur bornée et de taille polynomiale, des circuits multiplicativement disjoints de largeur bornée et de taille polynomiale, ainsi que des circuits de largeur bornée et de taille polynomiale. Peut-être certaines de ces classes sont-elles à nouveau égales aux formules ou bien la restriction sur la largeur fait-elle reculer d'un cran la complexité : les circuits multiplicativement disjoints de largeur bornée et de taille polynomiale seraient égaux aux circuits asymétriques de taille polynomiale. Remarquons que les circuits de largeur bornée et de taille polynomiale ne sont pas dans VP puisqu'ils contiennent la famille de polynômes (X^{2^n}).

Enfin, il faut bien entendu signaler l'existence de résultats similaires en complexité booléenne puisque Barrington a montré en 1986 ([3, 4]) l'équivalence entre les langages de NC_1 et les langages reconnus par des programmes à branchement de taille polynomiale et de largeur bornée (Bounded-width polynomial-size branching programs).

Chapitre 5

Expressivité des couvertures de graphes de largeur de clique pondérée bornée

Dans ce chapitre, nous étudions l'expressivité du permanent, de l'hamiltonien et des couplages parfaits des matrices de largeur de clique pondérée bornée. Rappelons qu'il est sans intérêt de regarder cette expressivité sur les matrices de largeur de clique non-pondérée bornée, puisque la somme des poids des couvertures considérées n'est pas modifiée si l'on remplace une non-arête par une arête de poids 0. On peut donc toujours considérer que le graphe pondéré défini par la matrice est une clique pondérée de largeur de clique 1.

Les résultats que nous obtenons dans ce chapitre sont moins précis que ceux des autres chapitres de cette partie puisque nous ne montrons pas d'équivalence entre les familles de polynômes exprimables au moyen de ces couvertures sur les matrices de largeur de clique pondérée bornée et une classe de familles de polynômes déjà connue. Nous montrons seulement que, pour ces trois couvertures, les familles exprimables contiennent les formules arithmétiques et sont contenues dans VP.

Le théorème suivant découle des propositions 5.1.1, 5.1.2 et 5.1.3.

Théorème 5.0.1
Si (f_n) est une famille de polynômes à coefficients dans un corps $\mathbb{K}$, représentée par une famille de formules arithmétiques de taille polynomiale, alors :

- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{per}(M_n)$.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{ham}(M_n)$.*
- *Il existe une famille (M_n) de matrices symétriques de taille polynomiale*

et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \sum_{P \in \mathcal{P}(M_n)} W(P)$.

Théorème 5.0.2
Soit (f_n) une famille de polynômes à coefficients dans un corps $\mathbb{K}$ telle que l'une des conditions suivantes est vérifiée :

- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{per}(M_n)$.*
- *Il existe une famille (M_n) de matrices de taille polynomiale et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \mathrm{ham}(M_n)$.*
- *Il existe une famille (M_n) de matrices symétriques de taille polynomiale et de largeur de clique pondérée bornée telle que les entrées de M_n sont des constantes de $\mathbb{K}$ ou des variables de f_n et $f_n = \sum_{P \in \mathcal{P}(M_n)} W(P)$.*

Alors (f_n) est dans VP.

On déduit ce théorème des propositions 5.2.1, 5.2.2 et 5.2.3.

5.1 Des formules vers les couvertures de graphes de largeur de clique pondérée bornée

Il peut sembler dans un premier temps que toute formule arithmétique peut s'exprimer comme la somme des poids des couvertures d'une matrice de largeurs de clique pondérées bornées, grâce aux résultats du chapitre 3 combinés avec le fait que, dans le cas non-pondéré, les graphes de largeur arborescente bornée sont aussi de largeurs de clique bornées. Néanmoins, à cause des restrictions sur la manière dont les poids sont assignés dans les définitions des largeurs de clique pondérées, il est faux de penser que les graphes pondérés de largeur arborescente bornée soient aussi de largeurs de clique pondérées bornées. En fait, si l'on essaye de suivre les preuves dans [25, 19] qui montrent que les graphes de largeur arborescente bornée ont une largeur de clique bornée, on obtient que tout graphe pondéré G de largeur arborescente k est de largeur de clique pondérée au plus $3 \cdot (|W_G| + 1)^{k-1}$ ou $3 \cdot (\Delta + 1)^{k-1}$. W_G est l'ensemble des poids sur les arêtes de G et Δ est le degré maximum de G. Les arbres pondérés sont toujours de largeur de clique pondérée bornée (la borne est 3). Mais nous montrons dans l'annexe B qu'il existe une famille de graphes pondérés planaires de largeur arborescente 2 et de largeur de clique pondérée non bornée (c'est un résultat obtenu en collaboration avec Ioan Todinca).

Nous devons donc fournir de nouvelles preuves ; mais celles-ci réutilisent les preuves du chapitre 3 puisque nous modifions les graphes de largeur arborescente bornée construits en des graphes de largeur de clique pondérée bornée ayant la même somme des poids des couvertures. Nous aurions aussi pu réutiliser les résultats du chapitre 4 avec le fait que tout graphe pondéré de largeur linéaire bornée est de largeur de clique pondéré bornée, si nous n'avions pas obtenu les résultats sur la largeur linéaire en dernier. Cette voie, qui est

beaucoup plus simple, fournit des constantes moins bonnes puisqu'elle permet d'obtenir des graphes de largeur de clique pondérée au plus 22/45/44 dont les permanents/hamiltoniens/sommes des poids des couplages parfaits sont égaux aux formules. Toutes les preuves relatives à cette voie sont fournies dans l'annexe B. Les modifications des constructions pour la largeur arborescente bornée que nous présentons dans ce chapitre nous permettent d'obtenir les constantes 13/34/26 qui restent bien supérieures aux constantes 2/6/2 obtenues dans le cas de la largeur arborescente bornée (nous n'avons pas investi énormément de temps dans l'obtention de constantes faibles, mais cet écart reste curieux et mériterait peut-être d'être réduit).

Proposition 5.1.1
Toute formule arithmétique de taille n peut s'exprimer comme le permanent d'une matrice de taille $O(n^2)$ et de largeur de clique pondérée au plus 13. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.

Preuve :

Soit φ une formule de taille n. On sait par les résultats du chapitre 3 que φ peut s'exprimer comme le permanent d'une matrice M de largeur arborescente au plus 2 et de taille au plus $(n+1) \times (n+1)$. Soient G le graphe orienté défini par M et $\langle T, (X_t)_{t \in V_T} \rangle$ une décomposition arborescente de G de largeur 2. Quitte à augmenter de manière linéaire la taille de T, on peut supposer que T est un arbre binaire enraciné.

À partir de cette décomposition arborescente T de G, nous construisons un graphe G' tel que $per(G) = per(G')$. Une différence majeure entre les algèbres pour les graphes de largeur arborescente bornée et les algèbres pour ceux de largeur de clique bornée est que, dans le dernier cas, on ne peut pas fusionner des sommets. En conséquence, les graphes G et G' ne seront pas isomorphes mais il y aura une bijection entre leurs couvertures par circuits.

Pour tout arc (u, v) de G, il peut exister plusieurs noeuds $t \in V_T$ tels que u et v sont tous deux dans le sac X_t. Nous dirons que l'arc (u, v) de G appartient à un noeud $t \in V_T$, si t est le noeud *le plus proche* de la racine de T où u et v sont dans X_t (par définition d'une décomposition arborescente, pour chaque arc de G, il existe bien un unique noeud de T auquel il appartient).

L'idée générale de la construction de G' est la suivante : on traite les noeuds de T de bas en haut. Pour un noeud $t \in T_V$, on construit les sous-graphes représentant les fils l et r de t, puis l'on ajoute les arcs appartenant à t à l'aide d'un schéma d'étiquetage sur les sommets. Nous n'utilisons pas une étiquette par sommet de G sous peine de ne pas obtenir un nombre constant d'étiquettes. À la place, puisque $|X_l| \leqslant 3$ et $|X_r| \leqslant 3$, nous utilisons les étiquettes pour représenter les sommets de X_l et X_r et nous les réutilisons durant le parcours de T.

Un sommet v de G est représenté à travers plusieurs sommets dans G', mais seulement deux sont « actifs » à un moment donné de la construction de G' : un sommet de degré entrant 0 s'occupe des arcs sortants de v dans

G (il est muni d'une étiquette de suffixe *out*) et un sommet de degré sortant 0 s'occupe des arcs entrants de v dans G (il est muni d'une étiquette de suffixe *in*). Puisque X_l et X_r sont tous deux de taille au plus 3, nous avons besoin des étiquettes suivantes : *left-a-in, left-a-out, left-b-in, left-b-out, left-c-in* et *left-c-out* (et 6 étiquettes similaires *right*). De plus, nous avons aussi besoin d'une étiquette « puits » ou « poubelle », pour étiquetter les sommets déjà traités. Ce qui nous donne un total de 13 étiquettes nécessaires pour exprimer G' au moyen d'une algèbre de largeur de clique pondérée.

Parcours de T pour construire G' :

- Pour une feuille t de T, on construit 6 sommets (ou 4 si $|X_t| = 2$) avec les étiquettes *left-a-in, left-a-out, left-b-in, left-b-out, left-c-in* et *left-c-out* (en supposant que t est le fils gauche de son père). Pour les arcs appartenant au noeud t, par exemple un arc de poids w du sommet représenté par les étiquettes *left-b-in/out* vers le sommet représenté par les étiquettes *left-a-in/out*, on ajoute des arcs de poids w des sommets étiquetés *left-b-out* vers les sommets étiquetés *left-a-in* (en réalité, un unique arc est ajouté car ces deux étiquettes sont associées à un seul sommet de G' à tout niveau de la construction). Ensuite, si l'un des sommets de X_t, par exemple celui représenté par *left-b-in/out*, n'est pas présent dans X_p (p étant le père de t dans T), alors on ajoute un arc de poids 1 de *left-b-in* à *left-b-out*. De plus, si ce sommet a une boucle de poids w, on ajoute un arc de poids w de *left-b-out* à *left-b-in*. Dans les deux cas, on renomme *left-b-out* et *left-b-in* en *puits*.
- Pour un noeud interne $t \in V_T$ (y compris la racine de T), on considère d'abord les sommets de G qui sont à la fois dans X_l et X_r, par exemple *left-a-in/out* et *right-b-in/out* représentent le même sommet de G. On suppose que t est le fils gauche de son père dans T (si t est la racine, on peut considérer qu'il est fils gauche ou droit sans que cela importe). On ajoute une boucle de poids 1 aux deux sommets d'étiquettes *right-b-in* et *right-b-out*. Puis on ajoute un arc de poids 1 de *right-b-in* à *left-a-in* et un arc de poids 1 de *left-a-out* à *right-b-out*. Enfin *right-b-in* et *right-b-out* sont renommés en *puits*.

  On s'intéresse ensuite aux sommets de X_t qui ne sont ni dans X_l ni dans X_r. On ajoute deux sommets à G' pour chacun de ces sommets. Il y aura forcément des étiquettes *in/out* disponibles pour ces deux sommets, puisque, dans ce cas, au moins deux autres sommets ont été renommés en *puits* lors du traitement de chacun des deux fils de t.

  On considère à présent tous les arcs de G appartenant à t. Supposons qu'il en existe un de poids w du sommet représenté par *right-c-in/out* au sommet représenté par *left-b-in/out*. On ajoute alors un arc de poids w de *right-c-out* à *left-b-in*.

  Enfin, si un sommet de X_t, par exemple celui représenté par *left-b-in/out*, n'est pas présent dans X_p (p étant le père de t dans T s'il existe, sinon on considère que $X_p = \emptyset$ car t est la racine), alors on ajoute un arc de poids 1 de *left-b-in* à *left-b-out*. De plus, si ce sommet a une boucle de poids w, on ajoute un arc de poids w de *left-b-out* à *left-b-in*. Dans les deux cas, on renomme *left-b-out* et *left-b-in* en *puits*.

Correction de la construction :

Un sommet v de G est représenté à travers deux ensembles disjoints de sommets dans G' : un ensemble de sommets s'occupant des arcs entrants de v dans G et un ensemble de sommets s'occupant des arcs sortants de v dans G. On note ces deux ensembles de sommets de G' v_{in} et v_{out}. Un sommet de G' appartient à v_{in} (resp. v_{out}) s'il a été créé avec une étiquette *in* (resp. *out*) afin de représenter v. Par construction, il est clair que tout sommet de G' appartient à v_{in} ou à v_{out} pour un unique sommet v de G ; de plus $G'[v_{in}]$ est un arbre orienté où tous les arcs sont dirigés vers la racine et sont de poids 1 ; tous les sommets de cet arbre excepté la racine ont une boucle de poids 1. $G'[v_{out}]$ a des propriétés équivalentes, à la différence près que les arcs sont dirigés vers les feuilles au lieu de la racine.

À présent considérons deux sommets u et v de G avec un arc de poids w de u à v ainsi que les arbres u_{out} et v_{in} dans G'. À un moment de la construction de G', un arc de poids w a été ajouté d'un sommet de u_{out} à un sommet de v_{in} dans G'. Il y a donc un chemin de poids w de la racine de u_{out} à celle de v_{in} et tous les sommets de u_{out} et v_{in} hors de ce chemin ont une boucle de poids 1. Donc à une couverture par circuits de G comportant l'arc de u à v, on peut faire correspondre une couverture par circuits qui va utiliser ce chemin dans G' ainsi que les boucles de poids 1 pour couvrir tous les sommets restants de u_{out} et v_{in}. Afin de continuer la construction du chemin dans G', on dispose d'un arc de poids 1 de la racine de v_{in} à la racine de v_{out}. Dans l'éventualité d'une boucle dans la couverture par circuits de G, nous avons ajouté un arc retour de la racine de v_{out} à celle de v_{in} de même poids que celle de G. Une boucle de G correspond donc à un circuit de longueur 2 dans G' et, dans ce cas, tous les autres noeuds de v_{in} et v_{out} sont couverts par des boucles de poids 1.

Il est alors aisé de vérifier que les couvertures par circuits de G' sont en bijection avec celles de G et que les paires correspondantes sont de même poids. On constate aussi que la taille de v_{in} ou v_{out} est en $O(n)$, pour tout sommet $v \in G$; donc le graphe G' est de taille au plus $O(n^2)$. Finalement, notons qu'entre deux sommets de G' il n'existe jamais plus d'un arc ; donc on peut trouver une matrice M' telle que le graphe correspondant est G' et donc $per(M') = per(M)$. ■

Proposition 5.1.2
Toute formule arithmétique de taille n peut s'exprimer comme l'hamiltonien d'une matrice de taille $O(n^2)$ et de largeur de clique pondérée au plus 34. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.

Preuve :

Soit φ une formule de taille n. On sait par les résultats du chapitre 3 que φ peut s'exprimer comme l'hamiltonien d'une matrice M de largeur arborescente au plus 6 et de taille au plus $(2n+1) \times (2n+1)$. Soient G le graphe orienté défini par M et $\langle T, (X_t)_{t \in V_T} \rangle$ une décomposition arborescente de G

de largeur 6. Quitte à augmenter de manière linéaire la taille de T, on peut supposer que T est un arbre binaire enraciné.

L'idée générale de la preuve est la même que celle de la preuve de la proposition 5.1.1, c.-à-d. de traiter de bas en haut les noeuds de la décomposition arborescente T de G. Puisque pour tout $t \in V_T$, $|X_t| \leqslant 7$ dans cette décomposition arborescente, il nous faut au moins $2 \cdot 14 + 1 = 29$ étiquettes durant le parcours de T pour construire G'.

Néanmoins, si l'on utilise exactement le même raisonnement que pour la proposition 5.1.1, alors pour toute couverture par circuits dans le graphe produit de nombreux sommets sont couverts par des boucles. Au lieu d'introduire de telles boucles, nous les remplaçons par la construction utilisée par Malod [52] afin de montrer l'universalité du polynôme hamiltonien. Pour cela, nous avons besoin de 5 étiquettes supplémentaires : *left-h1, left-h2, right-h1, right-h2* et *temp*, pour un total de 34 étiquettes. Les étiquettes *h1* et *h2* marquent respectivement le début et la fin du chemin annexe construit pour couvrir tous les sommets en un cycle hamiltonien.

Pour une feuille t de T, on commence par construire deux sommets étiquetés *left-h1* et *left-h2* (en supposant que t est le fils gauche de son père dans T) en ajoutant un arc de poids 1 *left-h1* à *left-h2*. Le reste de la construction relative à t est fait comme pour le permanent.

Pour un noeud interne t de T, on ajoute d'abord un arc de poids 1 de *left-h2* à *right-h1*, on renomme *left-h2* et *right-h1* en *puits* et on renomme *right-h2* en *left-h2* (en supposant que t est le fils gauche de son père dans T). Certains sommets, par exemple le sommet d'étiquette *right-c-in*, peuvent se voir ajouter une boucle durant le traitement de t dans le cas du permanent. Au lieu d'ajouter une telle boucle, on fait la construction suivante :

- ajouter un nouveau sommet d'étiquette *temp* ;
- ajouter un arc de poids 1 de *left-h2* à *right-c-in* ;
- ajouter un arc de poids 1 de *right-c-in* à *temp* ;
- ajouter un arc de poids 1 de *left-h2* à *temp* ;
- renommer *left-h2* en *puits* ;
- renommer *temp* en *left-h2*.

Le reste de la construction relative à t est fait comme pour le permanent.

Quand on atteint la racine r de T, on choisit un sommet de X_r, par exemple le sommet représenté par les étiquettes *left-a-in/out*. Pour ce sommet, au lieu d'ajouter un arc de poids 1 de *left-a-in* à *left-a-out*, on ajoute un arc de poids 1 de *left-a-in* à *left-h1* et un arc de poids 1 de *left-h2* à *left-a-out*. Ainsi, pour tout cycle hamiltonien de G, on déroute le cycle correspondant de G' pour qu'il visite tous les autres sommets de G' le long d'un chemin de poids 1. ∎

Proposition 5.1.3
Toute formule arithmétique de taille n peut s'exprimer comme la somme des poids des couplages parfaits d'une matrice symétrique de taille $O(n^2)$ et de

largeur de clique pondérée au plus 26*. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.*

Preuve :

Ce résultat est une conséquence directe des propositions 5.1.1 et 2.3.5 si l'on considère le biparti d'adjacence du graphe construit dans la preuve de la proposition 5.1.1. ∎

5.2 Des couvertures de graphes de largeur de clique pondérée bornée vers VP

Nous allons à présent montrer que la somme des poids des trois types de couvertures peut être calculée par un circuit de taille polynomiale quand la largeur de clique pondérée du graphe est bornée.

Le problème de décision de l'existence d'un cycle hamiltonien peut être résolu en temps polynomial sur les matrices de largeur de clique bornée par un résultat de Wanke [73] (voir aussi [28]). Nous étendons les idées de ces articles afin de calculer efficacement l'hamiltonien des matrices de largeur de clique pondérée bornée.

Proposition 5.2.1
L'hamiltonien d'une matrice $n \times n$ *de largeur* $\mathbb{K}$-MC *bornée peut s'exprimer comme un circuit de taille* $O(n^{O(1)})$ *et donc est dans* VP*.*

Preuve :

Soient M une matrice $n \times n$ de largeur $\mathbb{K}$-MC bornée et G le graphe orienté défini par M. On construit le circuit à l'aide d'un parcours de l'arbre syntaxique T du terme construisant G. On note T_t le sous-arbre enraciné en t de T pour $t \in V_T$. On note G_t le sous-graphe de G construit par le sous-terme correspondant à l'arbre syntaxique T_t.

Notre but étant de produire un circuit calculant la somme des poids de tous les cycles hamiltoniens de G, nous allons utiliser des portes internes au circuit qui vont calculer les poids de toutes les couvertures par chemins de tous les sous-graphes G_t, puis nous combinerons ces résultats intermédiaires. Bien entendu, le nombre total de couvertures par chemins peut croître exponentiellement en la taille de G_t, nous ne décrirons donc pas une couverture par chemins directement par les arcs participant à la couverture. À la place, nous décrirons une couverture par chemins de G_t par les ensembles d'étiquettes des extrémités de début et de fin de chaque chemin de la couverture. Ou plutôt nous donnerons, pour tout couple d'ensembles d'étiquettes, le nombre de chemins de la couverture ayant des extrémités étiquetées de cette manière. Une telle description n'est bien évidemment pas nécessairement associée à une couverture de manière unique, puisque deux couvertures par chemins distinctes peuvent avoir le même nombre de chemins avec les mêmes couples d'ensembles d'étiquettes associés. Néanmoins, nous n'avons pas besoin du poids de chaque couverture de manière

individuelle. Si plusieurs couvertures de G_t ont la même description, nous nous contentons de calculer la somme des poids de ces couvertures. Notons dès à présent que si le terme utilise k étiquettes, chaque description sera un vecteur de $\mathbb{N}^{2^{2k}}$ où la coordonnée $i \times 2^k + j$ est le nombre de chemins dont les extrémités ont les ensembles d'étiquettes i et j (i et j étant les codages entiers triviaux des ensembles d'étiquettes) et qu'il existe au plus $(n+1)^{2^{2k}}$ descriptions à considérer si n est le nombre de sommets de G.

Pour une feuille de l'arbre syntaxique T, on construit une porte d'entrée de poids 1 représentant la couverture par chemins composée d'un chemin de longueur 0 , commençant et finissant sur le sommet créé au niveau de la feuille. Bien entendu cette porte est associée au vecteur où toutes les coordonnées sont nulles, exceptée celle correspondant au couple (i, i) d'ensembles d'étiquettes qui est mise à 1, i étant l'ensemble d'étiquettes du sommet créé. Par définition cette couverture par chemins est de poids 1.

Pour un noeud interne $t \in T$, l'opération de l'algèbre décrit quels sont les arcs à ajouter et comment réétiqueter les sommets. On obtient les nouvelles couvertures par chemins en considérant une couverture du fils gauche de t et une couverture du fils droit de t : pour chaque couple de couvertures ainsi formé, on considère tous les sous-ensembles des arcs ajoutés au noeud t et l'on vérifie, pour chacun de ces sous-ensembles, si leur ajout au couple de couvertures va résulter en une couverture par chemins valide. Si c'est le cas, on ajoute une porte qui calcule le poids de cette couverture par chemins en multipliant les poids des couvertures gauche et droite ainsi que les poids des arcs ajoutés. Une fois que tous les couples de couvertures ont été traités, il faut regrouper les nouvelles couvertures par chemins ayant la même description. Si c'est le cas, on fait la somme des poids des couvertures équivalentes avec un arbre de portes d'additions.

Au niveau de la racine r de T, on combine les couvertures des fils de r afin de produire des cycles hamiltoniens au lieu des couvertures par chemins. Finalement, la sortie du circuit est la somme de toutes les portes calculant les poids de cycles hamiltoniens.

Correction de la construction :

La première étape de la preuve est par induction sur la hauteur de l'arbre syntaxique T. Nous allons montrer par induction que, pour tout noeud t de T et toute description de G_t, il existe une porte du circuit construit qui calcule la somme des poids des couvertures par chemins de G_t ayant cette description (à condition qu'il existe au moins une telle couverture). Pour le cas de base – les feuilles de T – c'est trivialement vrai.

Pour l'étape d'induction, on considère deux graphes disjoints G_1 et G_2 qui sont connectés avec des arcs au niveau du noeud t de l'arbre syntaxique T. Les arcs entre un sommet de G_1 et un sommet de G_2 ne peuvent être ajoutés qu'au niveau du noeud t ; donc toute couverture par chemins C de G_t peut être partitionnée en trois parties : une couverture par chemins C_1 de G_1, une couverture par chemins C_2 de G_2 et un nombre polynomial d'arcs ajoutés au niveau du noeud t. Comme par induction, nous avons une porte qui calcule la somme des poids des couvertures ayant la même description que C_1 (resp. C_2), il est clair que le poids de la couverture C est

pris en compte une unique fois au niveau de la porte calculant la somme des poids des couvertures de G_t ayant la même description que C. De la même manière, il est certain que le poids de toute couverture de G_t ayant cette description figure dans la valeur calculée par cette porte.

Afin de finir cette preuve de correction, nous devons vérifier le cas de la racine t de T. À la racine nous calculons le poids des circuits hamiltoniens au lieu des poids des couvertures par chemins. Tout circuit hamiltonien de G peut, tout comme une couverture par chemins, être partitionné en trois parties : une couverture par chemins C_1 de G_1, une couverture par chemins C_2 de G_2 et un nombre polynomial d'arcs ajoutés au niveau de la racine. Ces arcs refermant tous les chemins des deux couvertures en un unique circuit. Par hypothèse d'induction, tous les poids nécessaires sont déjà calculés.

La taille du circuit est polynomiale puisqu'à chaque étape (noeud de l'arbre syntaxique), le nombre de portes ajoutées est quadratiquement borné en le nombre de descriptions qui est lui-même polynomialement borné quand la largeur $\mathbb{K}$-MC est bornée. ■

Proposition 5.2.2
La somme des poids des couplages parfaits d'une matrice symétrique $n \times n$ de largeur $\mathbb{K}$-NLC bornée peut s'exprimer comme un circuit de taille $O(n^{O(1)})$ et donc est dans VP*.*

Preuve :

Soit M une matrice symétrique $n \times n$ de largeur $\mathbb{K}$-NLC bornée. On note G le graphe non-orienté défini par M. On construit le circuit à l'aide d'un parcours de l'arbre syntaxique T du terme construisant G. On note T_t le sous-arbre enraciné en t de T pour $t \in V_T$. On note G_t le sous-graphe de G construit par le sous-terme correspondant à l'arbre syntaxique T_t. Soit k la largeur $\mathbb{K}$-NLC de G. On suppose sans perte de généralité que T est l'arbre syntaxique d'un terme utilisant l'ensemble d'étiquettes $\{a_1, \ldots, a_k\}$.

L'idée générale va consister à produire des sous-circuits calculant la somme des poids des couplages des graphes G_t ayant une certaine description en parcourant T de bas en haut. On ne gardera à la racine que la somme correspondant aux poids des couplages parfaits. Pour cela, on décrit un couplage de G_t par les étiquettes associées aux sommets non-couverts par des arêtes du couplage. Plus précisément, pour chaque couplage de G_t et chaque étiquette a, on donne le nombre de a-sommets qui ne sont pas couverts par le couplage. Bien entendu, là encore deux couplages différents peuvent avoir la même description et ce n'est toujours pas un problème car nous nous contenterons de la somme des poids des couplages de G_t ayant cette description. Il est clair que le nombre de descriptions à considérer est au plus $(n+1)^k$.

Pour une feuille ver_{a_i} de l'arbre syntaxique T de G, on construit une porte d'entrée terminale de valeur 1, représentant un couplage vide. La description associée à cette porte est $((a_1, 0), \ldots, (a_i, 1), \ldots, (a_k, 0))$.

Pour un noeud interne $t \in T$ avec une opération $\circ_R(H)$, nous avons juste besoin de changer la description des portes terminales du circuit construit jusqu'à maintenant. Plus précisément, si la description de la porte était $((a_1, n_1), \ldots, (a_i, n_i), \ldots, (a_k, n_k))$, alors elle devient

$$((a_1, \sum_{a_j \in R^{-1}(a_1)} n_j), \ldots, (a_i, \sum_{a_j \in R^{-1}(a_i)} n_j), \ldots, (a_k, \sum_{a_j \in R^{-1}(a_k)} n_j)).$$

Pour un noeud interne $t \in T$ avec une opération $H \times_S H'$, nous créons d'abord une porte de multiplication utilisant les valeurs de chaque couple de portes terminales du fils gauche l de t et du fils droit r de t. Cela correspond au poids des unions disjointes des couplages de l et r. Il y a au plus $(n+1)^{2k}$ portes de ce type. À chacune de ces portes, nous associons des descriptions gauche et droite correspondant aux sommets de l et r. Ces portes deviennent les nouvelles portes terminales. On ordonne à présent les étiquettes de la manière suivante $a_1 < a_2 < \cdots < a_k$ ainsi que les couples (a_i, a_j) avec l'ordre lexicographique correspondant. Nous allons considérer que les arêtes sont ajoutées via S par blocs correspondant à un couple (a_i, a_j) (toutes les arêtes dans le même bloc sont ajoutées en même temps) et que tous les blocs sont ajoutés séquentiellement en ordre lexicographique. Ainsi nous avons au plus k^2 étapes d'ajout d'arêtes à considérer. Supposons que $S(a_i, a_j) = w_{ij}$. Pour l'étape correspondant à (a_i, a_j), on obtient de nouveaux couplages en considérant chaque porte terminale g_0. Soient $((a_1, n_1), \ldots, (a_i, n_i), \ldots, (a_k, n_k))$ et $((a_1, n'_1), \ldots, (a_j, n'_j), \ldots, (a_k, n'_k))$ les descriptions gauche et droite de g_0. Soit $n_{min} = min\{n_i, n'j\}$. Pour tout couplage correspondant à g_0 et tout entier p entre 0 et n_{min}, on peut obtenir $\binom{n_i}{p} \cdot \binom{n'_j}{p}$ couplages distincts en ajoutant p arêtes de poids w_{ij} entre p sommets parmi n_i de G_l et p sommets parmi n'_j de G_r. En conséquence, pour tout $p \neq 0$ on ajoute une porte de multiplication g_p dont les entrées sont g_0 et la valeur $\binom{n_i}{p} \cdot \binom{n'_j}{p} \cdot (w_{ij})^p$. Si w_{ij} est une constante de $\mathbb{K}$, c'est aussi le cas de $\binom{n_i}{p} \cdot \binom{n'_j}{p} \cdot (w_{ij})^p$. Si w_{ij} est une variable, il faut d'abord calculer $(w_{ij})^p$ avec $p-1$ portes de multiplications et le multiplier à la constante $\binom{n_i}{p} \cdot \binom{n'_j}{p}$. Cette nouvelle porte g_p a pour descriptions gauche et droite $((a_1, n_1), \ldots, (a_i, n_i - p), \ldots, (a_k, n_k))$ et $((a_1, n'_1), \ldots, (a_j, n'_j - p), \ldots, (a_k, n'_k))$. Il existe au plus $2 \cdot n^{2k+1}$ nouvelles portes puisque $p < n$. Finalement, on crée un arbre d'addition calculant la somme des portes g_p qui ont les mêmes descriptions gauche et droite. Chacun de ces arbres nécessite au plus $O((2k+2)\log(n))$ nouvelles portes et il y a au plus $(n+1)^{2k}$ arbres. Les sorties de ces arbres deviennent les nouvelles portes terminales. Quand toutes les k^2 étapes d'ajout d'arêtes sont finies, on calcule la description de chaque porte terminale en faisant la somme vectorielle de ses descriptions gauche et droite ; puis on crée un arbre d'addition calculant la somme des portes terminales ayant la même description globale. Les sorties de ces arbres sont les nouvelles portes terminales.

Finalement, on obtient la sortie du circuit au niveau de la racine de T. C'est la sortie de la porte terminale dont la description est $((a_1, 0), \ldots, (a_i, 0), \ldots, (a_k, 0))$.

Correction de la construction :

Le premier pas de la preuve se fait par induction sur la hauteur de l'arbre syntaxique T. Nous allons montrer que, pour tout noeud t de T, il existe, pour toute description de couplage de G_t, une porte du circuit qui calcule la somme des poids de tous les couplages de G_t ayant cette description. C'est trivialement vrai pour les feuilles de T.

Pour l'étape d'induction, on considère les deux graphes disjoints G_1 et G_2 qui sont reliés par des arêtes au niveau du noeud t. Les arêtes entre un sommet de G_1 et un sommet de G_2 ne peuvent être ajoutées qu'au niveau du noeud t ; donc tout couplage C de G_t peut être partitionné en trois parties : un couplage C_1 de G_1, un couplage C_2 de G_2 et un nombre polynomial d'arêtes ajoutées au noeud t. Considérons une description de couplage D ainsi que tous les couplages de G_t qui ont cette description. Tous les couplages peuvent être ainsi partitionnés en trois de manière unique et, par hypothèse d'induction, les sommes des poids des couplages de G_1 et G_2 sont déjà calculées pour chaque description. Il est donc clair que la somme des poids de tous les couplages de description D est bien calculée par l'une des nouvelles portes terminales créées.

Le nombre de nouvelles portes ajoutées pour chaque opération $H \times_S H'$ est au plus $O(k^2 \cdot n^{2k+1})$. Puisque le nombre de ces opérations est au plus n, on obtient un circuit de taille polynomiale. ■

Proposition 5.2.3
Le permanent d'une matrice $n \times n$ de largeur $\mathbb{K}$-NLC bornée peut s'exprimer comme un circuit de taille $O(n^{O(1)})$ et donc est dans VP.

Preuve :

C'est une conséquence directe des propositions 5.2.2 et 2.3.5 si l'on considère le biparti d'adjacence de G_M. ■

Les résultats de cette section sont un peu frustrants car ils ne permettent pas de caractériser la complexité des couvertures sur les matrices de largeurs de clique pondérées bornées. Nous devons nous contenter d'un encadrement de cette complexité entre VP_e et VP. Il semble raisonnable d'espérer montrer que l'encadrement peut être affiné du point de vue de la borne inférieure en le ramenant aux classes VDET et VP. Néanmoins, rien ne permet d'affirmer que la complexité de ces couvertures puisse ici encore coincider avec des classes de complexité algébrique existantes.

Chapitre 6

Expressivité des couvertures de graphes planaires

Intéressons-nous à présent aux familles de polynômes qui peuvent s'exprimer comme somme des poids des couvertures de graphes planaires. Cette question a déjà été bien étudiée puisqu'au moins les deux-tiers du travail ont déjà été réalisés.

6.1 Permanent et hamiltonien des graphes planaires

Bürgisser a montré en 1998 dans son habilitation ([16], voir aussi [15]) qu'il existe une famille (G_n) de graphes planaires pondérés telle que la famille de polynômes $(\mathrm{ham}(M_{G_n}))$ est VNP-complète sur tout corps de caractéristique différente de 2. Cette inconnue au niveau des corps de caractéristique 2 est liée à la preuve de Bürgisser qui se base sur la VNP-complétude du permanent. De plus la réduction utilise à nouveau des constantes $\frac{1}{2}$. Contrairement au permanent, on ne sait pas s'il est « facile » de calculer l'hamiltonien des graphes planaires sur un corps de caractéristique 2. Il est toutefois peu probable que ce problème soit dans VP en caractéristique 2 car il est fortement lié au problème de la parité du nombre de circuits hamiltoniens d'un graphe planaire, ce dernier étant $\oplus$P-complet par un résultat de Valiant en 2005 [70]. La preuve de Valiant, basée sur une réduction parcimonieuse à partir du problème $\#3-\mathsf{SAT}$, ne peut malheureusement pas être généralisée en une preuve de VNP-complétude en caractéristique 2. La complexité de l'hamiltonien des graphes planaires sur un corps de caractéristique 2 est donc toujours un problème ouvert à l'heure actuelle.

En ce qui concerne la complexité du permanent des graphes planaires, la réponse a été donnée très récemment par Datta, Kulkarni, Limaye et Mahajan [27]. Au moment où j'écris ces lignes, je devrais presque dire sera donnée puisque leur papier sera présenté début septembre à CSR 2007. Les auteurs y montrent la #P-complétude du permanent des graphes planaires. Bien qu'ils

ne le mentionnent pas, leur réduction peut aisément être modifiée afin de montrer la VNP-complétude du permanent des graphes planaires en toute caractéristique autre que 2. Voici comment : considérons tout d'abord une matrice M et G_M le graphe orienté associé. Il est aisé de voir qu'il existe une représentation planaire de G_M telle que deux arcs au plus s'intersectent en un point du plan. En chacun de ces croisements, on utilise alors le gadget suivant (voir figure 6.1) afin de planariser le graphe.

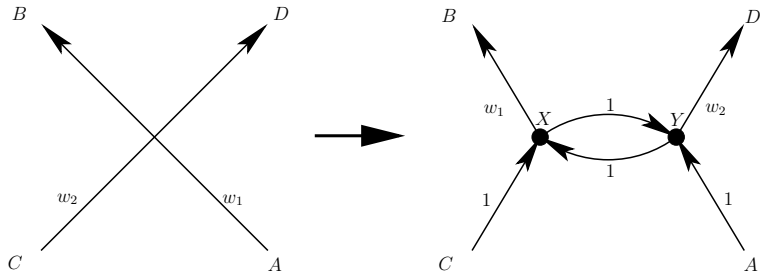


FIGURE 6.1 – Gadget de planarisation.

On constate que la somme des poids des couvertures par circuit du nouveau graphe n'est pas modifiée. En effet, une couverture par circuits n'utilisant aucun des deux arcs AB et CD sera complétée par le circuit XY de poids 1. Une couverture par circuits n'utilisant qu'un des deux arcs AB et CD utilisera le chemin de longueur 3 correspondant de même poids, $AYXB$ ou $CXYD$. Quant aux couvertures par circuits utilisant les deux arcs AB et CD, elles sont « épissées » , on utilise localement les chemins AYD et CXB. Cela revient à échanger les continuations vers B et D des circuits arrivant par A et C. Le produit des poids des arcs impliqués dans la couverture n'est pas modifié. Soient $M_n^X = (X_{i,j})_{1 \leqslant i,j \leqslant n}$ (déjà définie au chapitre 2) et G_n^X le graphe correspondant. Soient H_n^X le graphe planaire obtenu à partir de G_n^X au moyen de la planarisation précédente et N_n^X sa matrice d'adjacence. On a $\mathrm{per}_n = \mathrm{per}(M_n^X) = \mathrm{per}(N_n^X)$. De plus, le gadget de planarisation crée deux nouveaux sommets pour chaque croisement. Comme le nombre de croisements est polynomial en le nombre de sommets, la taille de N_n^X est polynomiale en n.

En caractéristique 2, cette même planarisation de graphe montre que le permanent des graphes planaires est VDET-complet.

6.2 Somme des poids des couplages parfaits des graphes planaires

Historiquement, compter le nombre de couplages parfaits dans un graphe planaire a été résolu par Fisher, Kasteleyn et Temperley en 1961 dans [30, 42, 64](voir aussi [43]). Leur résultat repose sur l'utilisation du Pfaffien dont voici la définition.

Définition 6.2.1 (Pfaffien)
Soit $A = (a_{i,j})$ une matrice antisymétrique de dimension $2n \times 2n$. Le Pfaffien

de A est défini par :

$$\mathrm{Pf}(A) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} a_{\sigma(2i-1),\sigma(2i)},$$

où $\mathrm{sgn}(\sigma)$ *est la signature de* σ*.*

Cette définition peut être simplifiée en utilisant l'antisymétrie de la matrice, ce qui évite d'additionner toutes les permutations possibles. Soit P l'ensemble de toutes les partitions de $\{1, 2, \ldots, 2n\}$ en paires, indépendamment de l'ordre ; P peut être vu comme l'ensemble des couplages parfaits du graphe complet ayant pour sommets les indices de la matrice. Un élément $p \in P$ peut s'écrire sous la forme $p = \{\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_n, j_n\}\}$. Soit $\pi = \begin{bmatrix} 1 & 2 & 3 & 4 & \cdots & 2n \\ i_1 & j_1 & i_2 & j_2 & \cdots & j_n \end{bmatrix}$ la permutation correspondante ; quand $i_k < j_k$ et $i_1 < i_2 < \cdots < i_n$, on dit que π est la permutation canonique associée à p. On constate qu'étant donnée une partition p, la quantitée $\mathrm{sgn}(\pi) a_{i_1,j_1} a_{i_2,j_2} \cdots a_{i_n,j_n}$ ne dépend que de p, indépendamment du choix de π. En effet, si l'on permute deux éléments d'une paire, l'inversion de la signature est compensée par l'antisymétrie de la matrice ; si l'on permute deux paires consécutives, on utilise un nombre pair de transpositions. Enfin, à chaque partition correspond $2^n n!$ permutations ; on en déduit donc la définition équivalente suivante :

Définition 6.2.2

$$\mathrm{Pf}(A) = \sum_{\pi \in \Pi} \mathrm{sgn}(\pi) \prod_{i=1}^{n} a_{\pi(2i-1),\pi(2i)},$$

où Π *est l'ensemble des permutations canoniques.*

Le Pfaffien est une généralisation du déterminant qui peut être vu comme un Pfaffien biparti grâce à l'identité suivante :

$$\det(M) = (-1)^{n(n-1)/2} \,\mathrm{Pf} \begin{pmatrix} 0 & M \\ -M^T & 0 \end{pmatrix}.$$

Étant donné un graphe non-orienté pondéré G de matrice d'adjacence $M = (m_{i,j})$, on assigne une orientation aux arêtes de G pour obtenir $\overrightarrow{G}$. La matrice de Tutte associée à $\overrightarrow{G}$ est la matrice d'adjacence antisymétrique $A = (a_{i,j})$, où $a_{i,j} = m_{i,j}$ si l'arête $\{i, j\}$ est orientée de i vers j dans $\overrightarrow{G}$ et $a_{i,j} = -m_{i,j}$ si l'arête $\{i, j\}$ est orientée de j vers i dans $\overrightarrow{G}$. Une telle orientation de G est dite pfaffienne si, dans la seconde somme qui définit le Pfaffien de A, tous les termes sont égaux au poids du couplage parfait correspondant ou si tous les termes sont égaux à l'opposé du poids du couplage parfait correspondant. Le résultat de Fisher, Kasteleyn et Temperley fut de démontrer que tout graphe planaire admet une orientation pfaffienne et que l'on peut la trouver en temps polynomial. Notons que $K_{3,3}$ est le plus petit graphe n'admettant pas d'orientation pfaffienne. Il suffit ensuite de calculer $|\mathrm{Pf}(A)|$, grâce à l'identité $(\mathrm{Pf}(A))^2 = \det(A)$, pour compter le nombre de couplages parfaits d'un graphe planaire. Depuis ce résultat, on a trouvé des algorithmes efficaces pour calculer

le Pfaffien ; à l'heure actuelle, on connaît précisément la complexité booléenne du déterminant et du Pfaffien puisqu'ils sont tous deux dans NC et même GapL-complets [51] si les entrées de la matrice sont des entiers. Ce même résultat implique clairement que la somme des poids des couplages parfaits des graphes planaires est dans VP.

Mentionnons les résultats récents de Valiant [69, 71] qui a montré que de nombreux problèmes pour lesquels on ne connaissait que des algorithmes exponentiels pouvaient être résolus en temps polynomial, au moyen de réductions holographiques de ces problèmes vers les couplages parfaits d'un graphe planaire.

Nous allons à présent préciser la complexité algébrique du calcul de la somme des poids des couplages parfaits des graphes planaires et démontrer qu'elle est exactement identique à celle du déterminant. Remarquons qu'on ne peut pas utiliser l'identité

$$\det(M) = (-1)^{n(n-1)/2} \operatorname{Pf} \begin{pmatrix} 0 & M \\ -M^T & 0 \end{pmatrix}$$

afin de montrer que la complexité algébrique du calcul de la somme des poids des couplages parfaits des graphes planaires est au moins celle du déterminant, puisqu'aucun résultat antérieur n'exprime la possibilité d'exprimer tout « Pfaffien biparti » comme un « Pfaffien planaire ».

Théorème 6.2.3
Soit (f_n) une famille de polynômes à coefficients dans un corps $\mathbb{K}$. Les trois propriétés suivantes sont équivalentes :

(i) (f_n) peut être calculée par une famille de circuits faiblement asymétriques de taille polynomiale.

(ii) (f_n) peut être calculée par une famille de circuits asymétriques de taille polynomiale.

(iii) Il existe une famille (G_n) de graphes planaires de taille polynomiale, dont les arêtes sont pondérées par des constantes de $\mathbb{K}$ ou des variables de f_n, telle que f_n est égal à la somme des poids des couplages parfaits de G_n.

L'équivalence entre (i) et (ii) est établie de deux manières différentes dans [66] et [53]. Rappelons que la classe de complexité VDET est définie comme la classe des familles de polynômes calculés par des circuits (faiblement) asymétriques de taille polynomiale ; le déterminant est VDET-complet.

L'équivalence de (iii) avec (i) et (ii) découle immédiatement des propositions 6.2.4 et 6.2.5. On démontre la proposition suivante en transformant un circuit asymétrique en un circuit planaire asymétrique équivalent, puis en construisant un graphe planaire dont la somme des poids des couplages parfaits est égale à la sortie du second circuit.

Proposition 6.2.4
La sortie de tout circuit asymétrique de taille n peut s'exprimer comme la somme des poids des couplages parfaits d'un graphe G, où G est un graphe pondéré, planaire, biparti avec $O(n^2)$ sommets. Le poids de chaque arête de G est égal à 1, -1 ou à une valeur d'entrée du circuit.

Preuve :

Soit φ un circuit asymétrique, i.e. pour chaque porte de multiplication *au moins* une de ses entrées est une entrée de φ (on peut supposer sans perte de généralité que c'est *exactement* une). De plus, on peut supposer moyennant une quantité linéaire de duplications que toutes les entrées de φ sont de degré sortant 1. Ainsi, toute entrée de φ est soit une entrée d'exactement une porte d'addition, soit une entrée d'exactement une porte de multiplication (on ignore le cas trivial où φ consiste en une seule porte) ; on fera la distinction tout au long de la preuve entre ces deux types de portes d'entrée.

Considérons une représentation planaire de φ dans laquelle toutes les entrées additives sont placées sur une ligne droite et toutes les autres portes sont dans le même demi-plan délimité par cette droite. Supposons que toutes les flèches du circuit sont dessinées par des traits droits. Il existe alors un nombre au plus quadratique de croisements de deux flèches dans le plan. En utilisant le gadget de planarisation de la figure 6.2, on remplace ces croisements par des sous-graphes planaires, en introduisant un nombre au plus quadratique de portes supplémentaires.

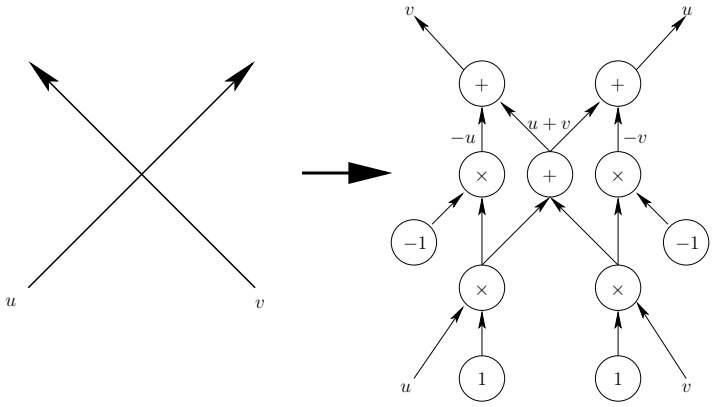


FIGURE 6.2 – Gadget de planarisation pour les circuits asymétriques.

On obtient ainsi un circuit asymétrique *planaire* φ' calculant la même valeur que φ. Nous allons à présent construire le graphe G dont la somme des poids des couplages parfaits est égale à la sortie de φ'. Pour cela, chaque porte du circuit correspondra à une étape de construction à l'exception des entrées multiplicatives qui seront traitées en même temps que leur porte de multiplication respective.

L'idée générale est que tout monôme du polynôme représenté par φ' sera encodé dans notre graphe par un chemin induit de l'entrée s à la sortie t. Chacun de ces chemins aura un couplage parfait de poids égal au monôme encodé.

Considérons un ordre topologique des portes de φ' dans lequel les entrées multiplicatives sont ordonnées avant les entrées additives que l'on ordonne de 1 à i (où i est le nombre d'entrées additives). Soit m le numéro

de la porte de sortie dans cet ordre topologique de φ'. Les étapes 1 à i de la construction de G sont montrées dans la figure 6.3. Elles constituent une phase d'initialisation.

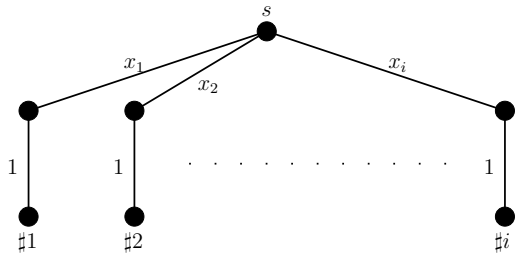


FIGURE 6.3 – Initialisation pour les entrées additives.

Le poids d'arête x_i correspond à la valeur de l'entrée de numéro i dans l'ordre topologique de φ'.

Pour chaque étape $i < m' \leqslant m$, une porte d'addition ou de multiplication est traitée comme le montre la figure 6.4.

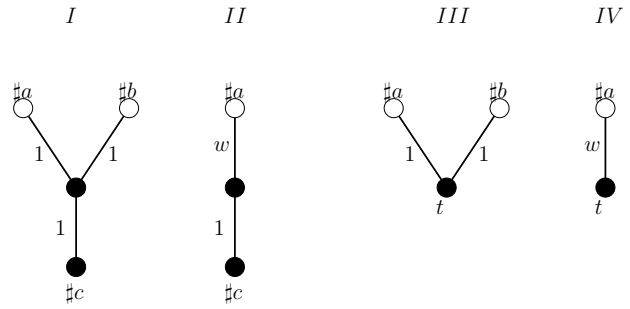


FIGURE 6.4 – I) Addition intermédiaire II) Multiplication intermédiaire III) Addition finale IV) Multiplication finale.

Les sommets blancs indiquent des sommets déjà présents dans le graphe intermédiaire construit ; tandis que les sommets noirs sont de nouveaux sommets introduits à cette étape. Pour simuler une porte d'addition, on ajoute 2 sommets et 3 arêtes, chacune de poids 1. Les arêtes sont utilisées pour connecter les 2 sommets qui représentent les entrées de l'addition. Pour simuler une porte de multiplication, on ajoute sous le sommet existant un chemin de longueur 2. Le poids d'arête w est la valeur de l'entrée multiplicative de φ' reliée à cette porte de multiplication. Finalement, la porte de sortie de φ' est gérée d'une manière spéciale.

On peut montrer par induction que la construction utilisée est correcte à l'aide des observations suivantes. À chaque étape $1 \leqslant m' < m$ dans la construction de G les propriétés suivantes seront vérifiées par le graphe construit jusque là : les étiquettes $\sharp 1, \sharp 2, \ldots, \sharp m'$ ont été assignées à m' sommets distincts. Pour tout $1 \leqslant j \leqslant m'$, si le sommet d'étiquette $\sharp j$ est

enlevé (ainsi que ses arêtes incidentes), alors la somme des poids des couplages parfaits du graphe restant est égale à la valeur calculée au niveau de la porte de numéro topologique j dans φ'. Il est clair que le graphe produit durant l'initialisation (Figure 6.3) a cette propriété. Pour les sommets restants dans l'ordre topologique, on doit simuler une porte d'addition ($\sharp c = \sharp a + \sharp b$) ou une porte de multiplication ($\sharp c = \sharp a \cdot w$). Pour chaque nouveau sommet étiqueté ajouté de cette manière, on peut voir qu'il simule la porte correspondante correctement, sans affecter les simulations faites par les autres sommets étiquetés du graphe.

On peut montrer que G est biparti en mettant simplement le sommet étiqueté s ainsi que les sommets étiquetés $\sharp i$, $1 \leqslant i \leqslant m$, d'un côté de la partition et tous les autres sommets de l'autre côté de la partition. ■

Remarque. La proposition précédente peut être démontrée pour les circuits *faiblement* asymétriques sans utiliser le résultat de [66] prouvant que les circuits faiblement asymétriques sont équivalents aux circuits asymétriques. Pour cela il suffit de modifier la simulation des portes de multiplication. Au lieu d'ajouter une arête de poids w, il faut ajouter un sous-graphe correspondant au sous-circuit d'entrée de la porte qui est disjoint du reste du circuit. Pour voir que les choses vont se recoller correctement, on peut remarquer que le graphe $G \setminus \{s, t\}$ a un unique couplage parfait et qu'il est de poids 1.

Nous démontrons à présent un peu plus que l'énoncé de la proposition suivante, puisque nous donnons la preuve que tout Pfaffien peut être calculé par un circuit asymétrique de taille polynomiale.

Proposition 6.2.5
Pour tout graphe planaire pondéré G avec n sommets, la somme des poids des couplages parfaits de G peut s'exprimer comme la valeur de sortie d'un circuit asymétrique de taille $O(n^{O(1)})$. Les entrées du circuit asymétrique sont des constantes ou des poids des arêtes de G.

Preuve :

La preuve qui suit combine des résultats de [43] et [51].

On sait par les résultats de Fisher, Kasteleyn et Temperley [43] que tout graphe planaire admet une *orientation pfaffienne* de ses arêtes. Une telle orientation ne dépend pas des poids des arêtes de G. Dans notre réduction vers un circuit asymétrique, nous supposerons donc qu'une orientation pfaffienne $\overrightarrow{G}$ nous est fournie avec G. Il nous faut donc montrer que $Pf(\overrightarrow{G})$ peut être calculé par un circuit asymétrique de taille polynomiale.

D'après le théorème 12 dans [51], on sait que $Pf(\overrightarrow{G})$ peut s'exprimer comme $SC(G')$ où G' est un graphe orienté, pondéré et acyclique avec deux sommets distingués, une source et un puits, notés s et t (rappelons que $SC(G')$, déjà défini dans la preuve de la proposition 3.1.1, est la somme des poids des chemins de s à t). De plus, la taille de G' est polynomiale en la taille de $\overrightarrow{G}$.

La dernière étape consiste donc à convertir G' en un circuit asymétrique de taille polynomiale dont la sortie est égale à $SC(G')$. Pour cela, considérons un ordre topologique des sommets de G' et traitons ces sommets selon

cet ordre. Le sommet s est remplacé par une entrée de valeur 1. Soient v un sommet de degré entrant 1, u son voisin intérieur et w le poids de l'arc de u à v. On remplace v par une porte de multiplication $c(v)$ dont les entrées sont la porte $c(u)$ correspondant à u ainsi qu'une nouvelle entrée du circuit de valeur w. Pour un sommet v de degré intérieur $d > 1$, on le remplace par d portes de multiplication similaires au cas de degré entrant 1. Ces d portes sont ensuite ajoutées au moyen de $d - 1$ portes d'addition de manière à sommer les poids des chemins venant des différents voisins internes. On note $c(v)$ la porte de sortie de cet arbre d'additions.

Il est clair que le circuit produit de cette façon vérifie que la valeur calculée en $c(v)$ est égale à la somme des poids des chemins de s à v. En particulier, c'est vrai pour t donc il calcule bien $SC(G')$. De plus, il est évident que ce circuit est asymétrique et de taille polynomiale. ■

La preuve de la proposition 6.2.5 montre que le Pfaffien est VDET-complet et pas seulement VDET-dur, ce qui montre que Pfaffien, « Pfaffien biparti » et « Pfaffien biparti planaire » ont tous trois exactement la même complexité algébrique. En ce qui concerne la complexité booléenne, comme je l'ai mentionné précédemment, Pfaffien et « Pfaffien biparti » sont équivalents ; tous deux sont GapL-complets. Néanmoins, il n'est pas évident que les constructions de la proposition 6.2.4 puissent être réalisées en espace logarithmique et que le « Pfaffien biparti planaire » soit lui aussi GapL-complet. Ces résultats nuancent le point de vue de Knuth [45] affirmant que le Pfaffien est plus fondamental que le déterminant car ce dernier se réduit au cas biparti du premier.

Il serait intéressant de trouver une construction combinatoire directe permettant de calculer la somme des poids des couplages parfaits d'un graphe planaire par un déterminant ou une identité algébrique exprimant le Pfaffien à l'aide du déterminant. Pour le moment, l'identité $(\mathrm{Pf}(A))^2 = \det(A)$ ne permet que de calculer la valeur absolue et de manière indirecte.

Deuxième partie

Complexité algorithmique

Chapitre 7

Variantes linéaires et NP-difficiles de partitionnement d'hypergraphes

Dans ce chapitre, je m'intéresse à une famille infinie de problèmes combinatoires paramétrés qui sont originaux, bien que naturels, et dont les complexités sont variées. Ces problèmes sont des variantes du problème de partitionnement d'hypergraphes. Celui-ci consiste à trouver une partition des sommets de l'hypergraphe telle qu'un certain nombre de contraintes sont satisfaites et/ou qu'une fonction objectif est minimisée/maximisée. En général, ce problème est NP-difficile. Une exception notable étant le problème MinCut Bipartition qui est polynomial (voir, par exemple, [38]). Rappelons que MinCut Bipartition est le problème de partitionner les sommets en deux ensembles non vides V_1 et V_2, afin de minimiser la somme des poids des hyperarêtes incidentes simultanément à des sommets de V_1 et de V_2. On a plutôt l'habitude d'en donner la formulation suivante : « Trouver un ensemble d'hyperarêtes de poids minimal qui déconnecte l'hypergraphe ».

On peut affirmer sans trop exagérer que presque tout problème algorithmique du type « comment bien diviser pour bien régner » peut trouver une formulation plus ou moins intuitive en termes de partitionnement d'hypergraphes. Des variantes du partitionnement d'hypergraphes sont donc utilisées dans de nombreux domaines comme la conception VLSI [1], le stockage efficace de bases de données sur des disques [63], les opérations sur les matrices creuses [76, 77], la recherche d'informations [74], le data mining [18, 40] et l'étude des processus semi-Markoviens [12]. Ces applications pratiques nécessitent souvent que les classes de la partition soient de tailles plus ou moins équilibrées. Ce détail fait que les variantes utilisées en pratique sont le plus souvent NP-difficiles car on peut alors réduire le problème MinCut Bisection dans la variante considérée. Rappelons que MinCut Bisection est, quant à lui, le problème de partitionner les sommets en deux ensembles non vides V_1 et

V_2 de même taille, afin de minimiser la somme des poids des hyperarêtes incidentes simultanément à des sommets de V_1 et de V_2. On peut aussi donner la formulation suivante : « Trouver un ensemble d'hyperarêtes de poids minimal qui déconnecte l'hypergraphe en deux ensembles de composantes connexes de même taille ». De nombreuses heuristiques ont été développées depuis trente ans (voir, par exemple, [41]) mais, à ma connaissance, aucun algorithme d'approximation (excepté [61]) ou PTAS n'est connu pour les variantes NP-difficiles classiques de ce problème, même sur des sous-classes d'hypergraphes.

Les variantes du partitionnement d'hypergraphes que nous allons étudier sont basées uniquement sur la satisfaction de contraintes.

Définition 7.0.1 (Problème P_k^l)
Étant donnés deux paramètres k et l, $1 \leqslant l < k$, le problème est défini comme suit : soit $\mathcal{H} = (V, \mathcal{E})$ un hypergraphe pour lequel $|V| = n$ et $|\mathcal{E}| = m$; soient $t_1, \ldots, t_k$ des entiers naturels tels que $n = \sum_{i=1}^{k} t_i$. Existe-t-il une coloration (partition) de V en k sous-ensembles de taille $t_1, \ldots, t_k$, telle que les sommets d'une hyperarête de $\mathcal{E}$ sont d'au plus l couleurs différentes ? On notera ce problème de décision P_k^l.

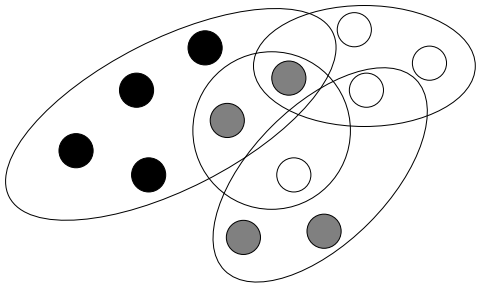


FIGURE 7.1 – Exemple de partitionnement d'hypergraphes : les sommets sont de 3 couleurs (blanc, gris, noir) et chaque hyperarête a des sommets d'au plus deux couleurs distinctes.

Le problème P_3^2 est fortement lié au calcul de la largeur de branche d'un graphe (voir [44], [49]). Dans [44], Kloks *et al.* montrent que P_3^2 est NP-complet, prouvant en particulier que le calcul de la largeur de branche est NP-complet sur les splitgraphes et les graphes bipartis.

Dans un premier temps, nous allons étudier la complexité de ces problèmes sur des hypergraphes arbitraires. La taille d'une instance est de l'ordre de $O(nm + k\log(n)) = O(nm)$. Nous allons alors montrer que P_k^1 est décidable en temps $O(nm + n^k)$ et P_k^l est NP-complet pour tout $l \geqslant 2$. On peut suggérer l'interprétation suivante de ce problème : étant donnés un ensemble de sommets représentant les tables d'une base de données, un ensemble d'hyperarêtes, chacune représentant les tables nécessaires à une requête, et les capacités de stockages des serveurs, peut-on stocker la base de données sur les serveurs de sorte que chaque requête a besoin de données provenant d'au plus l serveurs ? Nous faisons ici la supposition abusive que toutes les tables sont de même taille ; mais cela prouve que le problème de la « vie réelle » est NP-complet.

Dans un second temps, nous nous intéressons à la complexité de ces problèmes sur une sous-classe très restreinte des hypergraphes ; en effet, il s'agit des hypergraphes dont les hyperarêtes sont disjointes (*i.e.* de degré maximum 1). Bien que leur structure soit très simple, l'intérêt pour ce problème sur cette sous-classe est double : premièrement la taille de l'entrée est très compacte, de l'ordre de $O(m\log(n) + k\log(n)) = O(m\log(n))$, car il suffit de donner la taille de chaque hyperarête (voir les détails dans la section 7.3) ; deuxièmement cela correspond au problème naturel d'ordonnancement (ou d'empaquetage) suivant.
Considérons un ensemble de m programmes, le programme i étant composé de n_i tâches unitaires qui communiquent entre elles pour réaliser le programme. Soit n la somme de ces entiers. Considérons aussi un ensemble de k processeurs fixés et soient $t_1, \ldots, t_k$ la « capacité de calcul » de ces processeurs. On souhaite répartir les tâches sur les k processeurs et donc on suppose que la somme des t_i est supérieure ou égale à n. Peut-on exécuter les tâches sur les k processeurs de sorte que chaque programme soit dispersé sur au plus l processeurs ?
Nous obtiendrons sur cette sous-classe des résultats de complexité qui sont inversés avec ceux des hypergraphes quelconques. En effet, nous allons montrer que P_k^1 est NP-complet, pour tout k, et P_k^l, pour $l \geqslant 2$, est décidable en un temps majoré par $km + f(k,l)$ où $f(k,l)$ est exponentielle en k et l. Cette inversion de complexité s'explique quand $l = 1$ par le fait que le problème P_k^1 est faiblement NP-complet, il passe donc de polynomial à NP-complet quand il existe un encodage compact. Quand $l > 1$, elle s'explique par le fait que la structure des instances devient suffisamment simple pour être exploitée efficacement. Notons que ces résultats impliquent que le problème P_k^l est FPT (Fixed Parameter Tractable) pour les paramètres k et l quand $l \geqslant 2$ (rappelons qu'un problème paramétré par un entier k est FPT s'il existe un algorithme dont la complexité est majorée par $f(k) \times n^{O(1)}$, où n est la taille de l'instance, qui résout ce problème).

Enfin, nous donnerons quelques résultats partiels de complexité pour les hypergraphes d'intervalles de degré maximum 2, cette classe d'hypergraphes étant une extension très proche des hypergraphes dont les hyperarêtes sont disjointes.

7.1 Préliminaires

Rappelons qu'un hypergraphe $\mathcal{H}$ est un couple $(V, \mathcal{E})$ où V est l'ensemble des sommets et $\mathcal{E}$ est l'ensemble des hyperarêtes. Chaque hyperarête est un sous-ensemble de V. Par coloration d'un hypergraphe, on entend une coloration de ses sommets (contrairement à la notion de coloration propre, il n'est pas interdit qu'une hyperarête contienne plusieurs sommets de même couleur) Dans la suite, $t(e)$ sera la taille d'une hyperarête e, $t(c)$ celle de la couleur c et $c(e) = e(c) = |c \cap e|$ le nombre d'unités de la couleur c dans e. On dira que l'hyperarête e « voit » la couleur c ou réciproquement que la couleur c voit l'hyperarête e si $c(e) = e(c) > 0$. On notera C l'ensemble des couleurs, $C(e)$ l'ensemble des couleurs vues par e et $E(c)$ l'ensemble des hyperarêtes vues par c. On dira qu'une coloration des sommets est acceptable ou admissible si les couleurs ont les tailles spécifiées et chaque hyperarête voit au plus l couleurs.

On considérera par la suite certains types d'instances définis comme suit.

Définition 7.1.1 (Instance équilibrée, presque équilibrée)
Une instance $A = (\mathcal{H} = (V, \mathcal{E}), t_1, \ldots, t_k)$ *de* P_k^l *est dite* équilibrée *si* $t_1 = \cdots = t_k = q$ *et* $n = kq$. *Elle est dite* presque équilibrée *si* $t_i = \lfloor \frac{n}{k} \rfloor$ *ou* $t_i = \lceil \frac{n}{k} \rceil$, *pour tout* $1 \leqslant i \leqslant k$.

Faisons quelques remarques évidentes ; tout d'abord si $l = k$, alors le problème est vrai pour toute instance. Si $l = 0$, le problème est faux pour toute instance non-vide. Si une hyperarête est de taille inférieure ou égale à l, alors elle accepte toute coloration et on peut la retirer de l'instance. De même, si l'hyperarête e est incluse dans l'hyperarête e', on peut retirer e de l'instance sans modifier l'existence d'une coloration acceptable.

7.2 Complexité du problème sur des hypergraphes arbitraires

Dans cette section, nous allons d'abord montrer que P_k^1 est décidable en temps polynomial puis montrer que P_k^l est NP-complet pour tout $l \geqslant 2$.

7.2.1 Le cas polynomial

Théorème 7.2.1
Le problème P_k^1 *est décidable en temps* $O(nm + n^k)$, *pour tout* $k \geqslant 2$ *fixé.*

Preuve :

Comme toute hyperarête doit voir au plus une couleur, si deux hyperarêtes ont une intersection non-vide, elles doivent voir la même couleur. En conséquence, si $A = (\mathcal{H} = (V, \mathcal{E}), t_1, \ldots, t_k)$ est une instance de P_k^1 telle que $\exists e, e' \in \mathcal{E}$ qui s'intersectent, alors A est positive si et seulement si $A' = (\mathcal{H} = (V, \mathcal{E} \backslash \{e, e'\} \cup \{e \cup e'\}), t_1, \ldots, t_k)$ est positive. On peut donc répéter cette fusion d'hyperarêtes jusqu'à obtenir un hypergraphe dont les hyperarêtes sont disjointes. Étant donné un hypergraphe, il est facile de voir que l'on peut obtenir l'hypergraphe correspondant avec des hyperarêtes disjointes en temps $O(nm)$. À présent, sur un hypergraphe dont les hyperarêtes sont disjointes, le problème P_k^1 devient : peut-on trouver une partition $(E_1, \ldots, E_k)$ de $\mathcal{E}$ telle que $\sum_{e \in E_i} t(e) \leqslant t_i, i = 1..k$? Cette question peut être résolue en tant qu'instance du problème k-Subset-Sum en temps $O(n^k)$ (voir [36]). En conséquence, le problème P_k^1 peut être décidé en temps $O(nm + n^k)$.

■

7.2.2 Le cas NP-complet

Montrons que le problème P_k^l avec couleurs équilibrées est NP-complet pour $l \geqslant 2$ et k quelconque. Pour cela, nous utilisons le résultat de Kloks, Kratochvíl et Müller suivant.

Théorème 7.2.2 ([44], théorème 1)
P_3^2 est NP*-difficile sur les instances équilibrées.*

J'ai tout d'abord tenté de généraliser la preuve de ce résultat. De manière assez surprenante, une généralisation directe de la preuve de NP-difficulté de P_3^2 dans [44] montre la NP-difficulté de P_k^{k-1} uniquement si k est un nombre premier. J'ai donc dû procéder autrement et montrer la NP-difficulté des problèmes P_k^l pour $l \geqslant 2$ en me basant sur le résultat sans repartir de la preuve. Néanmoins, la première généralisation pour k premier est à mes yeux l'une des plus jolies preuves que j'ai écrites durant ma thèse. Elle a donc droit à une annexe à cette thèse bien qu'elle soit parfaitement inutile puisque les lemmes suivants montrent plus et plus facilement.

Lemme 7.2.3
Si P_k^2 est NP*-difficile sur les instances équilibrées, alors P_{k+1}^2 est aussi* NP*-difficile sur les instances équilibrées.*

Preuve :

Soit $A = (\mathcal{H} = (V, \mathcal{E}), t_1 = \cdots = t_k = q)$ une instance de P_k^2, $|V| = kq$. On construit l'instance $A' = (\mathcal{H}' = (V', \mathcal{E}'), t_1 = \cdots = t_k = t_{k+1} = q)$ de P_{k+1}^2 avec $V' = V \cup X$ et $\mathcal{E}' = \mathcal{E} \cup \{X\}$, où $V \cap X = \emptyset$ et $|X| = q$. Si A est une instance positive, alors on peut étendre une coloration acceptable de $\mathcal{H}$ en une coloration admissible de $\mathcal{H}'$, en colorant X avec la nouvelle couleur.

Si A' est une instance positive, on considère une coloration admissible de $\mathcal{H}'$. Si celle-ci n'utilise qu'une couleur pour colorer X, cette coloration restreinte à V est une coloration admissible de $\mathcal{H}$. Sinon, comme X est une hyperarête, elle voit au plus deux couleurs c_1 et c_2 et comme $|X| = q$, il y a autant de billes de c_2 (sommets de couleur c_2) dans X que de billes de c_1 dans V. Si on échange toutes les unités de c_2 dans X contre toutes les unités de c_1 dans V, alors X ne voit plus qu'une couleur et les autres hyperarêtes dans X ne voient pas plus de couleurs qu'avant. Cette nouvelle coloration restreinte à V est une coloration admissible de $\mathcal{H}$.

Il est facile de voir que A' peut être construite à partir de A en temps polynomial ; on en déduit donc le résultat. ■

Grâce au théorème 7.2.2, on a le corollaire suivant.

Corollaire 7.2.4
P_k^2 est NP*-difficile sur les instances équilibrées, $\forall k \geqslant 3$.*

Lemme 7.2.5
Si P_k^l est NP*-difficile sur les instances équilibrées, alors P_{k+1}^{l+1} l'est aussi.*

Preuve :

Soit $A = (\mathcal{H} = (V, \mathcal{E}), t_1 = \cdots = t_k = q)$ une instance de P_k^l, $|V| = kq$. On construit l'instance $A' = (\mathcal{H}' = (V', \mathcal{E}'), t_1 = \cdots = t_k = t_{k+1} = q)$ de P_{k+1}^{l+1} avec $V' = V \cup X$ et $\mathcal{E}' = \{e \cup X \mid e \in \mathcal{E}\}$, où $V \cap X = \emptyset$ et $|X| = q$. Si A est une instance positive, nous considérons une coloration admissible de $\mathcal{H}$. On peut étendre cette coloration en colorant X avec la nouvelle couleur et

obtenir une coloration admissible de $\mathcal{H}'$, car chacune des hyperarêtes de $\mathcal{E}'$ voit au plus l couleurs dans sa restriction à V et une couleur de plus dans sa restriction à X, donc au plus $l+1$ couleurs.

Si A' est une instance positive, on considère une coloration admissible de $\mathcal{H}'$. Si celle-ci n'utilise qu'une couleur pour colorer X, alors cette coloration restreinte à V est une coloration admissible de $\mathcal{H}$. Sinon, comme toutes les hyperarêtes de $\mathcal{E}'$ contiennent X, elles voient toutes les couleurs présentes dans X. Soit c_1 une de ces couleurs. On peut échanger toutes les billes des autres couleurs dans X contre les billes de c_1 dans V et les hyperarêtes ne voient pas plus de couleurs. Ainsi, on n'a plus qu'une couleur dans et uniquement dans X. Cette nouvelle coloration restreinte à V est une coloration admissible de $\mathcal{H}$.

Il est immédiat de voir que A' peut être construite à partir de A en temps polynomial ; on en déduit donc le résultat. ■

Théorème 7.2.6
P_k^l est NP-complet, $\forall 2 \leqslant l < k$.

Preuve :

Étant donné une coloration d'un hypergraphe, on peut vérifier en temps polynomial si c'est une coloration admissible, donc P_k^l est dans NP. D'après le corollaire 7.2.4, P^2_{k-l+2} est NP-difficile sur les instances équilibrées. En appliquant ensuite $l-2$ fois le lemme 7.2.5, on obtient la NP-difficulté de P_k^l sur les instances équilibrées. Ainsi P_k^l est NP-complet. ■

D'après les théorèmes 7.2.1 et 7.2.6, on voit que l'on a une distribution de complexité avec des problèmes polynomiaux sur la frontière gauche et des problèmes NP-complet partout ailleurs sur la figure 7.2.

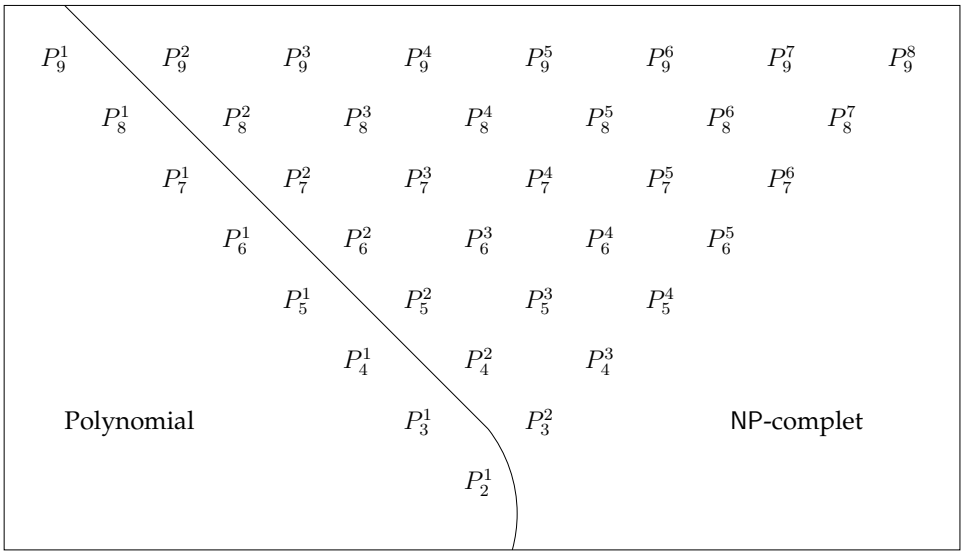


FIGURE 7.2 – Complexité sur les hypergraphes généraux.

7.3 Complexité du problème sur les hypergraphes ayant des hyperarêtes disjointes

Quand les hyperarêtes sont disjointes, il est pratique de voir l'hypergraphe comme un hypergraphe d'intervalles. En effet, on peut trouver un ordre total sur V tel que toute hyperarête est un intervalle de cet ordre. De plus, les hyperarêtes sont alors des sous-intervalles disjoints de V. Pour cette raison, nous allons renommer $\mathcal{E} = I$ et parler d'intervalles plutôt que d'hyperarêtes dans les sections qui suivent. Avec cette représentation, il est clair que l'on définit l'hypergraphe si l'on donne les extrémités « gauche » (plus petit élément de l'intervalle selon l'ordre) et « droite » (plus grand élément de l'intervalle selon l'ordre) de chaque intervalle. On a donc un encodage compact de l'instance en $O(m \log(n))$. Cet encodage compact existe pour tous les hypergraphes d'intervalles. Puisqu'ici nous considérons des hypergraphes de degré maximum 1, il est tentant de parler du problème en épaisseur 1. De même, nous parlerons du problème en épaisseur ep pour les hypergraphes d'intervalles de degré maximum ep.

On peut constater que le problème a une nature duale. En effet, on peut voir I et C comme deux familles d'hyperarêtes disjointes sur V. Le problème de décision consiste à demander si, étant données deux familles d'hyperarêtes I et C qui sont disjointes et étant donné la taille de chaque hyperarête, il existe un positionnement de ces hyperarêtes tel que chaque hyperarête de I soit adjacente à au plus l hyperarêtes de C.

Dans la suite, on considérera que les comparaisons, additions et soustractions sur les entiers sont réalisées en temps constant. En conséquence, les complexités polynomiales seront indépendantes de n.

7.3.1 Le cas NP-complet

Le théorème suivant est obtenu à l'aide d'une réduction polynomiale du problème 2-Partition.

Théorème 7.3.1
Le problème P_k^1 est NP-complet, $\forall k \geqslant 2$.

Preuve :

Le problème P_k^1 consiste à trouver une partition de V en k couleurs telle que chaque élément de I soit dans une seule couleur. Rappelons qu'une instance du problème 2-Partition consiste en un ensemble de m entiers naturels $e_1, \ldots, e_m$ dont la somme est n pair. Son codage est de taille $m\log(n)$. Le problème a une solution si et seulement si on peut trouver $m' \leqslant m$ entiers naturels de somme $n/2$ parmi $e_1, \ldots, e_m$. Nous allons transformer une instance du problème 2-Partition en une instance de P_k^1 en associant à chaque entier naturel e_i un sous-intervalle de taille e_i d'un intervalle V de taille $n + \frac{(k-2)n}{2}$, en ajoutant $k-2$ sous-intervalles de taille $n/2$ et en fixant $t_1 = t_2 = \cdots = t_k = n/2$. Cette instance est de taille $O((m+k-2)\log(n + \frac{(k-2)n}{2}) + k\log(n)) = O(m\log(n))$ car k est fixé et elle peut être construite en temps polynomial. Il est évident que l'instance construite est positive si et seulement si l'instance de 2-Partition l'est ; de plus, elle est équilibrée. En conséquence P_k^1 est NP-difficile sur les instances équilibrées et P_k^1 est NP-complet. ■

Remarquons dès à présent qu'il est évident que le problème P_k^l avec épaisseur $ep+1$ contient le problème P_k^l avec épaisseur ep, donc P_k^1 est NP-complet quelle que soit l'épaisseur. Mais le point intéressant est surtout que le problème P_k^1 en épaisseur supérieure est exactement identique au problème en épaisseur 1. En effet, si deux sous-intervalles ont une intersection non vide, comme chaque sous-intervalle doit voir une unique couleur, ils doivent être de la même couleur et on peut donc les remplacer par l'intervalle union des deux. On se ramène donc à un ensemble de sous-intervalles disjoints. En fait, quel que soit l'hypergraphe considéré, on se ramène à un hypergraphe dont les hyperarêtes sont disjointes comme dans la preuve du théorème 7.2.1. Ceci nous montre qu'il y a un effondrement de la complexité sur l'épaisseur au niveau du bord gauche de la famille de problèmes (cf. figure 7.3).

7.3.2 Le cas linéaire : premiers cas simples

Maintenant que nous savons que les problèmes du bord gauche sont NP-complets, nous allons nous intéresser aux membres de l'autre bord. Commençons par montrer que les problèmes de la moitié droite (k-2l<0) de la figure 7.3 peuvent être résolus en temps linéaire. Ce premier résultat sera obtenu en montrant que s'il existe une solution, alors l'une des solutions est linéaire ou continue pour l'ordre des sommets qui fait de l'hypergraphe un hypergraphe d'intervalle.

Définition 7.3.2 (t_p^{max}, t_p^{min})
Étant donnée une indexation des couleurs par ordre de taille croissante, $c_1, c_2, \ldots, c_k$ telle que $t(c_1) \leqslant t(c_2) \leqslant \cdots \leqslant t(c_k)$, on note t_r la taille de c_r, $t_p^{min} = \sum_{r=1}^{p} t_r$ la taille des p plus petites couleurs et $t_p^{max} = \sum_{r=k}^{k-p+1} t_r$ la taille des p plus grandes couleurs.

On définit aussi $L = t_l^{max}$ la taille limite. Cette quantité compte le nombre maximum de sommets pouvant être couverts par l couleurs. En conséquence, s'il existe un intervalle dont la taille excède L, il n'existe pas de coloration acceptable. Ceci nous fournit un premier type d'obstruction inconditionnelle.

Définition 7.3.3 (mégalomane)
Un mégalomane est un intervalle qui dépasse la taille limite $L = t_l^{max}$.

Pour raisonner, nous allons nous appuyer sur un type particulier de coloration :

Définition 7.3.4 (coloration continue)
Étant donné un ordre total sur V tel que $H = (V, I)$ est un hypergraphe d'intervalle pour cet ordre, on dira qu'une coloration de V est continue *relativement à cet ordre si toutes les couleurs sont des intervalles, sauf éventuellement une qui est l'union d'un intervalle contenant le plus petit élément de l'ordre et d'un intervalle contenant le plus grand élément de l'ordre. Une coloration est dite* continue *s'il existe un tel ordre pour lequel elle est continue.*

Supposons que l'on colore V de manière continue en commençant par la couleur 1, puis la 2 quand on a épuisé la couleur 1, ... On dira qu'un intervalle i épuise une couleur c si tous les éléments de V de couleur c sont contenus dans i. Pour qu'un sous-intervalle n'accepte pas cette coloration, il faut qu'il épuise au moins $l - 1$ couleurs tout en en voyant deux de plus (une à « gauche » et une à « droite ». Donnons-lui un nom :

Définition 7.3.5 (gêneur)
Un gêneur *est un sous-intervalle qui peut épuiser au moins $l - 1$ couleurs et voir deux couleurs de plus,* i.e. *il est de taille au moins $t_{l-1}^{min} + 2$.*

Remarque 1. C'est notre premier type d'obstruction conditionnelle. En effet, s'il n'existe pas de gêneur, une coloration continue fonctionne ; mais on peut aussi utiliser un éventuel gêneur à l'aide de la proposition suivante.

Proposition 7.3.6
Soit g un gêneur non-mégalomane. Il existe une coloration partielle de V restreinte aux sommets de g qui colore g avec l couleurs et épuise $l - 1$ de ces couleurs.

Preuve :

> Commençons par prendre les $l - 1$ plus petites couleurs et ajoutons la plus grosse couleur. Si ces l couleurs ne sont pas suffisantes pour couvrir g, on échange la plus petite couleur utilisée avec la plus grande couleur non-utilisée. On répète ces échanges jusqu'à ce que les l couleurs utilisées soient de taille supérieure à celle de g. Comme $t(g) \leqslant L$ la condition d'arrêt se produit forcément. Une fois que nous avons trouvé ces l couleurs, nous

savons que les $l-1$ plus petites couleurs utilisées sont plus petites que g, donc on peut les épuiser dans g. ■

On obtient donc le résultat suivant quand $k - 2l < 0$.

Théorème 7.3.7
Si $k - 2l < 0$, alors une instance de P_k^l est positive si et seulement si il n'existe pas de mégalomane, donc P_k^l est décidable en temps linéaire.

Preuve :

$\Rightarrow$: Par contraposée, s'il existe un mégalomane, alors il n'existe pas de coloration admissible.

$\Leftarrow$: S'il n'existe pas de gêneur, toute coloration continue est admissible. S'il en existe un, disons g, on commence une coloration continue depuis l'une de ses extrémités vers l'autre, en utilisant les l couleurs fournies par la proposition 7.3.6 en ordre de taille croissante. De cette manière, g est correctement coloré et au plus $k-(l-1) \leqslant l$ couleurs sont à l'extérieur de g. En conséquence, les intervalles restants sont correctement colorés.

Cela nous montre que pour répondre au problème de décision, il suffit de tester la condition sur la taille limite pour tous les intervalles. Si l'on cherche à fournir une coloration qui convienne, il suffit de faire une coloration continue en partant du bord d'un éventuel gêneur ou de n'importe où en cas d'absence de gêneurs. Pour cette coloration, on peut voir l'intervalle comme un cercle. Trouver l couleurs telles que les l couvrent et $l-1$ peuvent être épuisées dans un gêneur donné peut être réalisé en temps constant (l est fixé). On obtient donc bien la linéarité du problème. ■

On a donc la répartition suivante pour le moment :

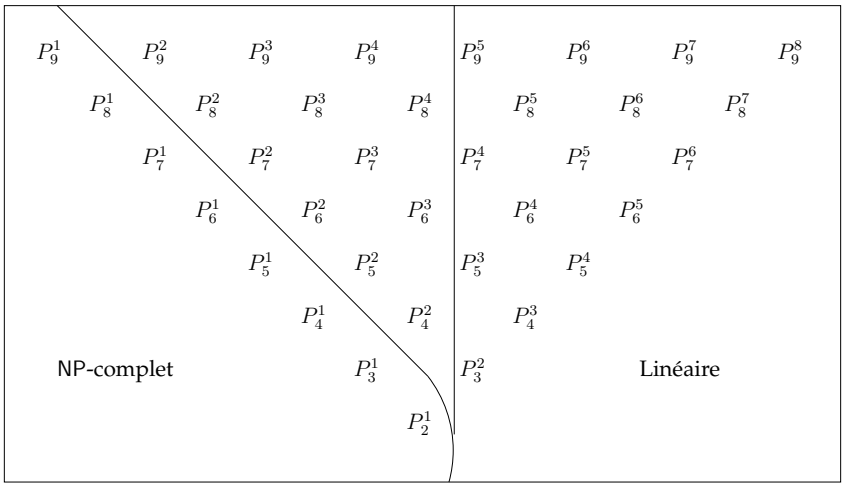


FIGURE 7.3 – Complexité partielle sur les hypergraphes d'épaisseur 1.

Pour obtenir ce premier résultat de linéarité, on s'est basé sur la notion de gêneur. Plus exactement, sur le fait que l'on peut épuiser suffisamment de couleurs dans un seul gêneur pour ne plus se soucier du reste. Considérons la famille de propriétés suivantes : l'intérieur de j gêneurs suffit à épuiser suffisamment de couleurs pour ne plus se soucier du reste. Ce qui donne de manière formalisée la propriété $\mathrm{Pr}_1(j) : k-j(l-1) \leqslant l$. Si $l \neq 1$, on peut toujours trouver j tel que la propriété $\mathrm{Pr}_1(j)$ soit vérifiée. On peut aussi toujours trouver j tel que la propriété $\mathrm{Pr}_1(j)$ soit fausse ($\mathrm{Pr}_1(0)$ est toujours fausse). On dira que le problème P_k^l a la propriété $\mathrm{Pr}_2(j)$ s'il vérifie $\mathrm{Pr}_1(j)$ et pas $\mathrm{Pr}_1(j-1)$. Le premier résultat de linéarité correspond donc au cas où le problème a la propriété $\mathrm{Pr}_2(1)$.

Cette approche brute va nous permettre de montrer que, pour les instances presque équilibrées, les deux tiers des problèmes sont linéaires ; toutefois, elle s'essouffle au delà. Nous allons à présent regarder les cas $\mathrm{Pr}_2(2)$ et $\mathrm{Pr}_2(3)$ avant d'introduire les derniers outils nous permettant de traiter le cas général.

Cas $\mathrm{Pr}_2(2)$: On dispose des deux inégalités suivantes : $k-2(l-1) \leqslant l$ et $k-(l-1) > l$. On suppose qu'il n'y a pas de mégalomane. S'il y a 0 ou 1 gêneur, une coloration continue suffit. S'il y en a deux, nous savons que leur intérieur suffit à épuiser suffisamment de couleurs donc il faut s'assurer qu'on peut les disposer convenablement.

Si l'instance est presque équilibrée (les couleurs sont de taille $\lceil \frac{n}{k} \rceil$ ou $\lfloor \frac{n}{k} \rfloor$), on vérifie qu'il y a suffisamment de «grosses» couleurs (de taille $\lceil \frac{n}{k} \rceil$) pour colorer les plus gros gêneurs. Il y a $r = n \mod k$ «grosses» couleurs. On exprime donc cette condition au moyen de l'inégalité suivante : $\sum_{i \in I} max(0, t(i) - l \lfloor \frac{n}{k} \rfloor) \leqslant r$. Or $k-(l-1) > l \Leftrightarrow k-2l+1 > 0 \Leftrightarrow k-2l \geqslant 0$. Ce qui veut dire que l'on peut séparer l'intervalle en deux blocs d'au moins l couleurs et caser chaque gêneur dans l'un des deux blocs. Chaque gêneur épuise alors $l-1$ couleurs de son bloc.

Si les couleurs sont de taille quelconque, cette mécanique simple se grippe. En effet, il se peut que l'on ne puisse pas mettre deux gêneurs dans deux blocs de l couleurs disjoints car il est nécessaire au deuxième gêneur d'utiliser les grosses miettes de l'autre plutôt que des petites couleurs. Voici un exemple d'une telle obligation : P_7^3, $n = 54 = 10+10+10+10+10+2+2$, $t(i_1) = t(i_2) = 25$, $t(i_3)$=4. C'est le premier problème, on doit considérer les interactions entre gêneurs.

Cas $\mathrm{Pr}_2(3)$: On dispose des deux inégalités suivantes : $k-3(l-1) \leqslant l$ et $k-2(l-1) > l$. On suppose qu'il n'y a pas de mégalomane. S'il y a 0 ou 1 gêneur, une coloration continue suffit.

S'il y en a deux, on les place dans deux blocs de l couleurs. Comme $k-2(l-1) > l \Leftrightarrow k-2l > l-2 \Leftrightarrow k-2l \geqslant l-1$, il nous reste au moins $l-1$ couleurs totalement libres ainsi qu'au plus deux partiellement entamées. Voilà un nouveau problème qui arrive. En éliminant les deux gêneurs, on a peut-être créé des petites couleurs qui viennent tout perturber en créant de nouveaux gêneurs.

S'il y en a trois, on aimerait pouvoir les caser dans trois blocs comme au cas précédent. Or $k-2(l-1) > l \Leftrightarrow k-3l+2 > 0 \Leftrightarrow k-3l+1 \geqslant 0$. Donc dans certains cas, comme P_5^2, il faut caser deux gêneurs dans un même bloc. C'est un nouvel exemple d'interaction entre gêneurs.

Afin de dépasser les problèmes soulevés et généraliser nos premiers résultats de linéarité, nous allons regarder en détail la structure des solutions. Dans ce but, nous allons considérer un graphe représentant une partition et observer les propriétés de ce graphe quand la partition est une solution.

7.3.3 Le cas linéaire : structure des solutions

Toute notre démarche se résume à épuiser des couleurs jusqu'à ce qu'il en reste au plus l. Si l'on montre que toute instance satisfaisable admet une solution dans laquelle un intervalle épuise au moins une couleur, alors le problème est faiblement polynomial.

« Pourquoi polynomial ? »

On peut en effet construire un algorithme qui épuise au moins une couleur à chaque étape ; il s'arrêtera donc au bout d'au plus $k - l$ étapes. À chaque étape, il suffit de tester pour tous les intervalles toutes les colorations partielles qui épuisent une couleur dans cet intervalle et tester si l'instance de $P^l_{k'}$ résultante est satisfaisable ($k' < k$). Ainsi chaque sous-problème à une étape donnée engendre $O(m)$ (choix de l'intervalle qui épuise une couleur) multiplié par $O(\sum_{r=1}^{l} C^r_k)$ (choix de l'ensemble de couleurs qui colore l'intervalle) sous-problèmes à l'étape suivante. On obtient un algorithme avec une complexité par rapport à m de l'ordre de $(\sum_{r=1}^{l} C^r_k)^{k-l} \times m^{k-l}$ donc en $O(m^{k-l})$ par rapport à m.

« Pourquoi faiblement ? »

Si l'on prend en compte la taille n et toutes les tailles des couleurs et des intervalles qui sont du même ordre de grandeur, on remarque qu'elles sont codées en espace logarithmique. Or, si $l \geqslant 3$, l'intervalle i qu'on a choisi peut épuiser une couleur et en voir deux autres sans les épuiser. Ce qui veut dire que l'excédent de ces deux couleurs par rapport à la taille de i peut être réparti de $O(n)$ manières entre elles. (Quand l augmente davantage, les choses empirent puisque l'excédent peut être réparti de $O(n^{l-2})$ manières différentes.) On se retrouve avec un nombre de sous-problèmes exponentiel en $\log(n)$.

« Pourquoi est-ce bizarre ? »

La remarque précédente pourrait nous faire craindre que les problèmes qui sont les plus proches des problèmes NP-complets soient polynomiaux alors que ceux qui sont un peu plus éloignés ne seraient que faiblement polynomiaux et peut-être même NP-complets. On aurait alors une famille de problèmes NP-complets sur le bord gauche ($l = 1$), une bande de problèmes polynomiaux juste à côté ($l = 2$) puis des problèmes faiblement polynomiaux ou NP-complets ($l \geqslant 3$ et $k - 2l \geqslant 0$) et enfin la moitié droite de problèmes linéaires ($k - 2l < 0$).

Afin d'obtenir un algorithme fortement polynomial et ainsi éviter pareille situation, nous devons vérifier une condition plus puissante. Nous devons prouver que toute instance satisfaisable admet une solution dans laquelle un intervalle épuise toutes les couleurs qu'il voit sauf au plus une. (Bien sûr, il faut aussi qu'il voie au moins deux couleurs afin d'être certain d'en épuiser au moins une.)

Les théorèmes suivants vont nous permettre d'aboutir à l'existence d'un intervalle qui épuise toutes les couleurs qu'il voit sauf au plus une et donc d'un

algorithme fortement polynomial. Ils vont nous servir à simplifier la structure du graphe suivant qui représente la structure d'une coloration.

Définition 7.3.8 (Graphe de coloration)
Étant donnée une coloration, on appelle graphe de la coloration *le graphe non orienté biparti $G = (I \cup C, E)$ où $(i, c) \in E$ pour $i \in I$ et $c \in C$ si et seulement si la couleur c voit l'intervalle i. On peut pondérer les arêtes par le nombre d'unités partagées par l'intervalle et la couleur concernés. On note cette pondération $p(i, c) = i(c) = c(i)$.*

Voici quelques propositions évidentes sur les propriétés d'un graphe de coloration.

Proposition 7.3.9
Soit G un graphe de coloration, alors $\sum_{c \in N(i)} p(i, c) = t(i), \forall i \in I$.

Proposition 7.3.10
Soit G un graphe de coloration. Si I couvre totalement V, alors $\sum_{i \in N(c)} p(i, c) = t(c), \forall c \in C$.

Cette représentation du problème avec un graphe de coloration est équivalente à une formulation du problème en termes d'un problème de transport avec contraintes uniformes sur les degrés. (Le nombre de fournisseurs est k et chaque client doit être de degré au plus l.) Il est probable (dès que j'aurai le temps) que les résultats qui suivent puissent être généralisés au problème de transport avec contraintes sur les degrés quand ces contraintes sont toutes supérieures à 2 (on peut tolérer un nombre logarithmique de clients qui demandent à n'avoir qu'un fournisseur). Je remercie Stéphan Thomassé qui m'a suggéré que mes graphes de coloration étaient un problème de transport et qu'il était probable que l'on puisse enlever l'uniformité sur les contraintes de degré. Remarquons que la formulation habituelle du problème de transport attribue un coût de transport d'une marchandise d'un fournisseur vers un client (potentiellement différent pour chaque couple fournisseur-client) et cherche à minimiser le coût global du transport (une subtilité que Stéphan avait oublié de me préciser :)). Ma variante peut être vue comme fixant un coût de transport uniforme et je doute par contre que les preuves qui suivent puissent être modifiées pour marcher dans le cas où les coûts ne sont plus uniformes (peut-être obtient-on un algorithme d'approximation pour minimiser le coût de transport tout en respectant les contraintes de degré).

Le premier théorème nous donne une propriété des « bonnes » solutions (celles qui en plus d'assurer que tout intervalle voit moins de l couleurs minimisent, sous cette condition, la somme globale du nombre de couleurs vues par les intervalles). Il est trivial si l'on considère que toute solution minimale d'un problème de transport est un arbre. Sa preuve peut donc être vue comme une redémonstration de ce fait.

Théorème 7.3.11
Soit $A = (H = (V, I), C)$ une instance de P_k^l. Si cette instance est satisfaisable, alors il existe une coloration acceptable telle que son graphe est une forêt.

Preuve :

Soit $G = (I \cup C, E)$ le graphe correspondant à une coloration acceptable quelconque. Supposons qu'il existe un cycle élémentaire dans G. Appelons-le $\mu = (i_1, c_1, \ldots, i_d, c_d)$. Soit $p = min_{j\in\{1,\ldots,d\}}\{p(i_j, c_j)\}$. Alors pour j allant de 1 à d, p billes de couleur c_j dans l'intervalle i_j sont remplacées par p billes de couleur c_{j-1} provenant de l'intervalle i_{j-1} (les indices sont pris modulo d). De manière plus formelle, les nouvelles valeurs sont $c_j(i_j) = c_j(i_j) - p \geqslant 0$ et $c_j(i_{j+1}) = c_j(i_{j+1}) + p$. Dans cet échange, un intervalle i_j tel que $p(i_j, c_j) = p$ a perdu toutes ses billes de couleur c_j, il voit une couleur de moins qu'avant et il n'est plus lié à son successeur (on a retiré l'arête $i_j c_j$). Tous les autres intervalles voient au plus autant de couleurs qu'avant. En répétant cette méthode, on détruit tous les cycles et on obtient une coloration admissible telle que le graphe correspondant soit une forêt.

■

Le théorème suivant est le résultat combinatoire central permettant d'obtenir un algorithme fortement polynomial.

Théorème 7.3.12
Soient $A = (H = (V, I), C)$ une instance de P_k^l et $i \in I$ le plus grand des intervalles. Si cette instance est satisfaisable, alors il existe une solution telle que i épuise toutes les couleurs qu'il voit sauf au plus une.

Preuve :

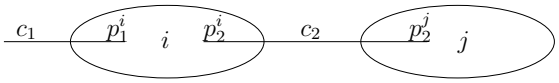
Supposons que l'instance est satisfaisable. Considérons le graphe G obtenu grâce au théorème 7.3.11. G est une forêt. Si i est le seul intervalle dans sa composante connexe, il épuise toutes les couleurs qu'il voit donc c'est gagné. Sinon, on veut faire en sorte que i soit une pseudo-feuille, c-à-d. que toutes les couleurs adjacentes à i sauf une soient des feuilles. Supposons que i n'est pas seul dans sa composante connexe et qu'il n'est pas une pseudo-feuille. Afin que i devienne une pseudo-feuille, on va mettre tous les autres intervalles de la composante du même côté de i, c-à-d. dans une unique branche partant de i.

Considérons les couples (cc, gbi) où, pour une coloration donnée et un intervalle i, cc est le nombre de composantes connexes du graphe de coloration et gbi (plus grosse branche) est le nombre maximum de sommets correspondant à des intervalles dans une branche partant de i. On ordonne les couples (cc, gbi) par ordre lexicographique. Nous allons choisir une branche dont le nombre de sommets correspondant à des intervalles est égal à gbi ; une telle branche sera appelée branche gbi. Ainsi, les transformations suivantes fourniront des graphes de coloration qui restent des forêts et pour lesquels la valeur de (cc, gbi) croît strictement.

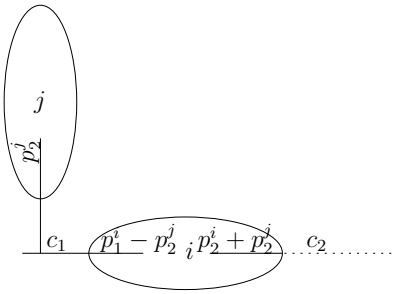
Pour illustrer cette preuve, nous allons visualiser les couleurs comme des fils semi-rigides reliant les intervalles. Les couleurs seront donc représentées par des lignes et les intervalles par des ellipses sur les schémas qui suivent. Les phrases en italiques correspondront par la suite à l'explication des déformations continues que je visualise. La preuve peut être lue sans ces explications.

Soient c_1 et c_2 deux couleurs vues par i et ses voisins (les intervalles voyant une couleur vue par i). Nous supposons que c_1 est dans une

branche gbi partant de i. Soit j un voisin de i qui voit c_2. *Nous allons déplacer i de c_1 vers c_2*. On note $p(i, c_1) = p_1^i$, $p(i, c_2) = p_2^i$ et $p(j, c_2) = p_2^j$.

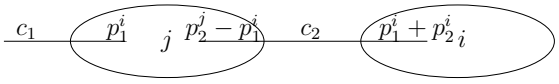


Si $p_2^j \leqslant p_1^i$, alors on peut pousser toutes les unités de la couleur c_2 qui sont dans j à l'intérieur de i et faire ressortir p_2^j unités de c_1 afin de rebrancher j derrière i. Plus formellement, si $p_2^j \leqslant p_1^i$ on échange p_2^j unités de c_1 dans i contre les p_2^j unités de c_2 dans j. De plus, si c_2 n'est vue que par i et j, alors elle est totalement dans i. On a fait augmenter d'au moins un le nombre d'intervalles qui sont du bon côté de i.

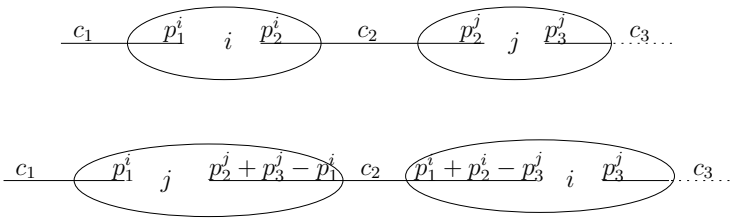


On peut remarquer que si $p_2^j = p_1^i$, alors i ne voit plus la couleur c_1, on a réussi à casser la composante connexe de i en deux et faire baisser d'un le nombre de couleurs vues par i. Si c'est le cas et que i n'est ni une pseudo-feuille ni l'unique intervalle de sa nouvelle composante connexe, on recommence la procédure sur la nouvelle composante connexe qui est strictement plus petite.

Si $p_2^j > p_1^i$ et j voit moins de l couleurs, alors *on peut échanger les positions de i et j*, c'est à dire donner à j les p_1^i unités de c_1 dans i en échange de p_1^i unités de c_2 dans j. De cette manière, j est à nouveau dans la branche gbi partant de i. j voit une couleur de plus et i une de moins qu'avant.



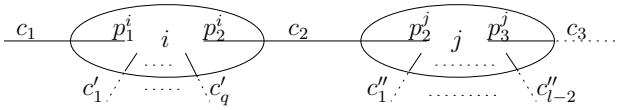
Supposons que j voit l couleurs, alors il existe c_3 une couleur vue par j autre que c_2. On note $p(j, c_3) = p_3^j$. Si $p_2^j > p_1^i$ et $p_1^i + p_2^i \geqslant p_3^j$, alors *on peut échanger les positions de i et j*, c'est à dire donner à j les p_1^i unités de c_1 dans i et donner à i les p_3^j unités de c_3 dans j, puis compenser un éventuel déséquilibre en échangeant des unités de c_2. Formellement, j contient à présent p_1^i unités de c_1 et $p_2^j + p_3^j - p_1^i$ unités de c_2 ($p_2^j + p_3^j - p_1^i > 0$ car $p_2^j > p_1^i$) ; tandis que i contient à présent p_3^j unités de c_3 et $p_1^i + p_2^i - p_3^j$ unités de c_2 ($p_1^i + p_2^i - p_3^j \geqslant 0$ car $p_1^i + p_2^i \geqslant p_3^j$).



devient

L'échange a porté sur 3 couleurs et on s'est assuré que i et j étaient les seuls à voir l'une des 3, donc chacun des deux voit autant de couleurs qu'avant. On peut constater à nouveau que si $p_1^i + p_2^i = p_3^j$, on a réussi à casser la composante connexe de i en deux et faire baisser d'un le nombre de couleurs vues par i qui ne voit plus c_2. Il faut remarquer ici que si i ne voit initialement que les couleurs c_1 et c_2, alors $t(i) = p_1^i + p_2^i > p_3^j$ car i est le plus grand des intervalles. Le problème est donc réglé si $l = 2$.

Supposons à présent que $p_2^j > p_1^i$ et $p_1^i + p_2^i < p_3^j$. D'après les remarques précédentes, on peut supposer que i voit un nombre q entre 1 et $l-2$ de couleurs c'_x autres que c_1 et c_2 et j voit $l-2$ couleurs c''_x autres que c_2 et c_3.



Il faut à présent remarquer que q vérifie

$$\sum_{x=1}^{q} i(c'_x) + \sum_{x=q+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j > t(j) \text{ car}$$

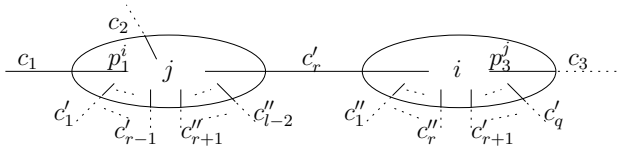
$$\sum_{x=1}^{q} i(c'_x) + p_1^i + p_2^i = t(i) \geqslant t(j).$$

Soit r le plus petit entier tel que

$$\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j \geqslant t(j).$$

Comme $t(j) = p_2^j + p_3^j + \sum_{x=1}^{l-2} j(c''_x)$ et $p_1^i + p_2^i < p_3^j$, on va donner à i toutes les unités des couleurs c_3,$c''_1, \ldots, c''_r$ dans j en échange des unités de c_1, c_2, $c'_1, \ldots, c'_{r-1}$ et d'une partie (peut-être l'ensemble) des unités de c'_r dans i.

Formellement, i contient à présent p_3^j unités de c_3, $j(c''_x)$ unités de c''_x, pour x allant de 1 à r, $i(c'_x)$ unités de c'_x, pour x allant de $r+1$ à q, et $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j - t(j)$ unités de c'_r (on a bien $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j - t(j) \geqslant 0$ par définition de r) ; tandis que j contient à présent p_1^i unités de c_1, $p_2^i + p_2^j$ unités de c_2, $j(c''_x)$



unités de c''_x pour x allant de $r+1$ à $l-2$, $i(c'_x)$ unités de c'_x pour x allant de 1 à $r-1$ et $t(j) - (\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j)$ unités de c'_r (on a bien $t(j) - (\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j) \geqslant 0$ car, par définition de r, $\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j < \sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j < t(j)$.).

Ainsi, soit $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j = t(j)$ et nous avons réussi à casser la composante connexe de i en deux parties et i voit une couleur de moins qu'avant, soit $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + p_1^i + p_2^i + p_2^j > t(j)$, i et j voient tous deux autant de couleurs qu'avant et j est encore passé dans la branche gbi partant de i.

Grâce à ces quatre transformations, on peut choisir une branche adjacente à i et mettre n'importe quel intervalle dans cette branche, jusqu'à ce qu'i soit une pseudo-feuille. CQFD.

■

Corollaire 7.3.13
Soit $A = (H = (V, I), C)$ une instance de P_k^l qui possède un gêneur et soit $g \in I$ le plus grand des gêneurs. Si cette instance est satisfaisable, alors il existe une solution telle que g épuise toutes les couleurs qu'il voit sauf au plus une et telle qu'il en voit au moins deux. De plus, on peut supposer que la couleur non épuisée est de taille maximale parmi les couleurs vues par g.

Preuve :

Grâce au théorème 7.3.12, on sait qu'il existe une solution telle que g épuise toutes les couleurs qu'il voit sauf au plus une. Supposons qu'il n'en épuise aucune car il n'en voit qu'une c_g. Puisque g est un gêneur, on peut échanger toutes les unités des $l-1$ plus petites couleurs contre des unités de c_g. De cette manière, g voit l couleurs et en épuise $l-1$. Les intervalles qui voyaient une des $l-1$ petites couleurs voient autant de couleurs qu'avant et ceux qui en voyaient deux ou plus en voient moins qu'avant. Remarque : avec ces échanges, on a peut-être créé des cycles dans le graphe de coloration.

Supposons que la couleur non épuisée, c_1, n'est pas une des couleurs vues par g de taille maximale. Soit c_2 une couleur vue par g de taille maximale. Comme $t(c_2) > t(c_1)$, on peut échanger toutes les billes de c_1 hors de g avec des billes de c_2 dans g. Le nombre de couleurs vues par g ou un autre intervalle n'est pas modifié. On obtient donc à nouveau une coloration acceptable.

■

Le corollaire précédent résume quatre points ayant un impact sur la complexité de l'algorithme :

- g épuise au moins une couleur (il en voit au moins deux) ; ceci permet d'obtenir un algorithme polynomial en m ;
- g épuise toutes les couleurs qu'il voit sauf au plus une ; cet argument permet de prouver que l'algorithme est fortement polynomial ;
- on peut se limiter à g pour épuiser une couleur et éviter d'essayer tous les intervalles ; nous en déduirons une complexité linéaire en m ;
- la couleur non épuisée est la plus grande des couleurs vues par g, donc chaque ensemble d'au plus l couleurs qui colore g n'engendre qu'un seul sous-problème. Cela diminue légèrement la complexité d'un facteur $O(kl)$.

Les résultats obtenus jusqu'ici nous donnent le schéma d'algorithme suivant pour décider si une instance est positive :

Début Programme
Soit (I, C) une instance de P_k^l
Si il existe un mégalomane
  Alors retourner Faux (d'après la définition 7.3.3)
Si il n'existe pas de gêneur
  Alors retourner Vrai (d'après la remarque 1)
Si $k - 2l < 0$
  Alors retourner Vrai (d'après le théorème 7.3.7)
Soit g le plus grand intervalle (c'est un gêneur)
Pour tout ensemble d'au moins deux couleurs C' tel que i peut les épuiser toutes sauf au plus une
  Faire
    Soit c' la taille de C' moins la taille de i
    Si $c' = 0$
      Alors Si l'instance $(I \backslash i, C \backslash C')$ est positive pour $P^l_{k-|C'|}$
        Alors retourner Vrai
      Sinon Si l'instance $(I \backslash i, C \backslash C' \cup c')$ est positive pour $P^l_{k-|C'|+1}$
        Alors retourner Vrai
Fin pour tout
retourner Faux (d'après le corollaire 7.3.13)
Fin Programme

L'algorithme 1 est une implémentation plus détaillée de ce schéma d'algorithme dont la complexité est donnée dans le théorème suivant. On y utilise la notation $\mathcal{P}_{\geqslant a, \leqslant b}(X)$ pour l'ensemble des parties de X dont le cardinal est compris entre a et b.

Théorème 7.3.14
L'algorithme 1 décide si une instance de P_k^l est satisfaisable. De plus, sa complexité est en $O(1)$ pour l et k fixés à condition que les $\max(1, k - 2l + 1)$ plus grands intervalles soient donnés triés à la fin du tableau. On peut donc décider le problème P_k^l en temps $O(m)$, où m est le nombre d'intervalles de l'instance.

Preuve :

Algorithme 1 `Colore` : Algorithme de décision pour P_k^l

entrée:

(k, l, I, C, m) où (I, C) est une instance de P_k^l, $|I| \geqslant m$, C est trié dans l'ordre croissant des tailles et I est partiellement trié dans cet ordre, c-à-d. qu'au moins les $k - 2l + 1$ plus grands intervalles sont triés à la fin du tableau.

sortie:

VRAI : s'il existe une coloration telle que chaque intervalle voit au plus l couleurs.

début

si $m = 0$ ou $k \leqslant l$ **alors**
  rendre VRAI
L' :=2 ;
pour chaque r=1..l-1 **faire**
  L'+=C[r] ;
si I[m] < L' **alors**
  rendre VRAI
L :=0
pour chaque r=0..l-1 **faire**
  L+=C[k-r] ;
si I[m] > L **alors**
  rendre FAUX
si $k - 2l < 0$ **alors**
  rendre VRAI
pour chaque $C' \in \mathcal{P}_{\geqslant 2, \leqslant l}(C)$ **faire**
  c := la plus grande des couleurs de C'
  si $t(C') - t(c) < I[m]$ **alors**
    si $t(C') > I[m]$ **alors**
      c' := le reste de la couleur c de taille $t(C') - I[m]$
      si Colore($k - |C'| + 1$,l,I,$C \backslash C' \cup c'$,$m - 1$) **alors**
        rendre VRAI
    si $t(C') = I[m]$ **alors**
      si Colore($k - |C'|$,l,I,$C \backslash C'$,$m - 1$) **alors**
        rendre VRAI
rendre FAUX
fin

L'algorithme 1 commence par regarder si le nombre d'intervalles est 0 ou si le nombre total de couleurs est inférieur à l. Si c'est le cas, on renvoie «vrai» car toute coloration convient. Pour décider en temps constant s'il existe un gêneur, on regarde ensuite si le plus grand des intervalles g en est un. S'il n'y a pas de gêneur, on renvoie «vrai» car une coloration linéaire convient. Puis, on décide s'il existe l couleurs permettant de couvrir g. Si g est mégalomane, on renvoie «faux». Si g ne l'est pas et que $k - 2l < 0$, alors on renvoie «vrai» car d'après le théorème 7.3.7 on peut épuiser $l - 1$ couleurs dans g et il en restera au plus l. Cette première phase s'effectue en temps constant $\alpha \times l$.

On sait, d'après le corollaire 7.3.13, que s'il existe une solution, on peut en trouver une où g voit au moins deux couleurs et les épuise toutes sauf peut-être la plus grande. Dans le pire des cas, g n'épuise jamais la plus grande des couleurs qu'il voit, on doit tester toutes les possibilités et l'on ne s'arrête pas avant d'avoir $k - 2l < 0$ pour toutes les possibilités. Soit $T(k, l)$ la complexité dans le pire des cas de l'algorithme pour les valeurs k et l. Si $k - 2l < 0$, $T(k, l) = \alpha l$ où α est une constante. Sinon, $T(k, l) = \alpha l + \sum_{p=2}^{l} C_k^p T(k-p+1, l)$, $\sum_{p=2}^{l} C_k^p T(k-p+1, l)$ étant le coût des appels récursifs dans la troisième et dernière boucle de l'algorithme. Comme $k \geqslant 2l$, $C_k^p \leqslant C_k^l$ pour $p = 2..l$. On a donc $T(k, l) \leqslant \alpha l + \sum_{p=2}^{l} C_k^l T(k-p+1, l) \leqslant \alpha l + \sum_{p=2}^{l} C_k^l T(k-1, l) = \alpha l + (l-1) C_k^l T(k-1, l) < \alpha l + (l-1) k^l T(k-1, l)$. Si $k = 2l + q$, $T(k, l) < \alpha l \sum_{p=0}^{q} ((l-1)k^l)^q < \alpha l ((l-1)k^l)^k$. En fait, chaque appel récursif est précédé d'un temps $\alpha' l$ pour calculer le reste de la couleur entamée. En faisant porter ce coût à l'appelé plutôt qu'à l'appelant, il est confondu avec le coût initial de αl car α et α' sont des constantes. On obtient bien une constante par rapport à m car on ne touche jamais au tableau contenant les intervalles ; chaque appel récursif se contente de s'intéresser à une case différente du tableau en fonction de sa profondeur de récursion.

Finalement, on obtient une complexité linéaire car il suffit de $k - 2l + 1$ passages pour trier partiellement les intervalles. ∎

On obtient une complexité linéaire $am + b$ avec deux constantes. La constante multiplicative $a \approx k - 2l$ est linéaire en k et l ; de plus pour k fixé elle augmente quand l décroît, c.-à-d. quand l'on s'approche de la partie NP-complète. La constante additive $b \approx ((l-1)k^l)^k$ est exponentielle en k et l mais cette fois elle décroît quand l'on s'approche de la partie NP-complète. Je ne sais pas si ce phénomène est dû à une mauvaise approximation ou si c'est réellement ce qui se passe dans le pire des cas. Néanmoins, je ne pense pas que ce phénomène se produise toujours quand on étudie la complexité dans le cas moyen. En effet, il est probable qu'il existe une variante du corollaire 7.3.13 affirmant qu'en moyenne g voit au moins $O(l)$ couleurs.
Grâce au théorème 7.3.14, on peut donc conclure que l'on a la répartition de complexité représentée sur la figure 7.4.

Il existe donc un effet de seuil violent pour la complexité du problème entre $l = 1$ et $l > 1$.

Théorème 7.3.15
Le problème P_k^l est NP-complet si $l = 1$, linéaire sinon.

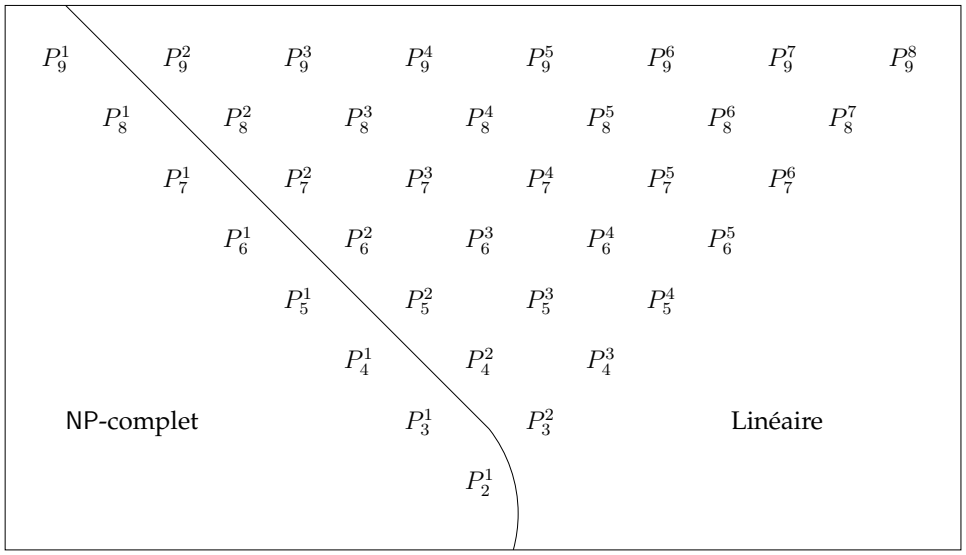


FIGURE 7.4 – Complexité sur les hypergraphes d'épaisseur 1.

7.3.4 Applications de l'algorithme

L'algorithme 1 peut facilement être étoffé pour fournir une coloration acceptable quand elle existe, au lieu de se contenter de répondre au problème de décision. De plus, il peut être utilisé comme sous-procédure pour plusieurs problèmes liés aux problèmes P_k^l.

Il peut par exemple être utilisé pour trouver pour une instance donnée le plus petit $l \neq 1$ tel qu'il existe une coloration acceptable. Il suffit de $\log(k)$ étapes de recherche dichotomique. Pour cette recherche, nous utilisons la même instance mais différents problèmes P_k^l. Cette procédure nous fournit un algorithme d'approximation absolue (à une constante additive près égale à 1) pour le problème de minimisation du l tel que les programmes peuvent être dispersés sur au plus l processeurs. Ce problème est NP-complet de par la NP-complétude de P_k^1 mais l'algorithme nous fournit la valeur exacte sauf dans le cas où il obtient $l = 2$, auquel cas la valeur exacte peut être 1 ou 2.

Une autre application peut être de trouver une estimation du meilleur temps pour l'exécution des programmes I et une solution correspondante quand on impose que chaque programme soit exécuté sur au plus l processeurs ($l > 1$). Pour ce problème, on suppose qu'on nous fournit les « puissances de calcul » $pu_1, \ldots, pu_k$ des k processeurs et non leurs « capacités de calcul ». La puissance de calcul pu_r représente le nombre de tâches unitaires pouvant être traitées par le processeur r en une unité de temps. On fera donc varier les valeurs de $t_1, \ldots, t_k$ en faisant varier le temps laissé aux processeurs. Cette fois, on utilise toujours le même problème P_k^l mais on modifie l'instance en changeant le nombre de sommets non-couverts par des hyperarêtes (en changeant n pour que $\sum_{r=1}^{k} t_r = n$). On peut remarquer que si $n > kt(I)$, alors tous les intervalles peuvent être couverts par une couleur. On peut donc faire une recherche dichotomique entre $t(I)$ et $kt(I)$ et en $\log(t(I)) + \log(k-1)$ étapes

trouver cette estimation.

Pour toutes ces applications, il serait intéressant d'étudier la complexité en moyenne car l'analyse dans le pire cas ne nous fait épuiser qu'une couleur à la fois alors qu'il est vraisemblable que le nombre moyen de couleurs épuisées à chaque étape augmente quand l augmente.

7.4 Résultats partiels de complexité en épaisseur 2

Historiquement, mon étude des problèmes P_k^l a débuté sur les hypergraphes d'intervalle de degré maximum deux. J'étudiais la largeur de branche des graphes de cordes et je voulais savoir s'il était NP-difficile ou non de décider si la largeur de branche d'un splitgraphe de corde est égale à celle de sa clique sous-jacente (cette question sur les splitgraphes généraux est NP-complète par le résultat de Kloks, Kratochvíl et Müller [44]). J'avais ramené ce problème au problème P_3^2 en épaisseur 2 sur les instances presque-équilibrées. Mon premier résultat fut donc de montrer dans mon mémoire de DEA [49] que le problème P_3^2 en épaisseur 2 sur les instances presque-équilibrées est linéaire. Sa complexité sur les instances quelconques est toujours ouverte et les preuves de ce mémoire sont à mon avis plus techniques que celles qui précèdent (il faut dire aussi que j'ai passé moins de temps à les simplifier).

Bien entendu, les problèmes P_k^1 en épaisseur 2 sont NP-complets puisqu'ils contiennent les problèmes en épaisseur 1. Nous allons à présent chercher à démontrer que le problème P_k^{k-1} en épaisseur 2 est linéaire sur les instances quelconques quand $k \geqslant 4$. Pour cela, il nous faut introduire de nouveaux types d'obstructions.

Définition 7.4.1 (gêneur exigeant)
Un gêneur est exigeant s'il ne peut voir que les l plus grandes couleurs, c.-à-d. de taille supérieure à $t_{l-1}^{max} + t_{k-l}$.

Les gêneurs exigeants sont un nouveau type d'obstruction conditionnelle qui n'apporte pas grand chose si l'on considère un gêneur isolé. Par contre, combinés, ils fournissent un nouveau type d'obstruction inconditionnelle.

Définition 7.4.2 (guerre)
On dira que 2 gêneurs exigeants i et j sont en guerre *si $|i \cup j| > t_l^{max}$.*

Lemme 7.4.3
S'il existe une guerre, alors il n'y a pas de coloration acceptable.

Preuve :

En effet, les deux gêneurs exigeants ne doivent voir que les l plus grandes couleurs donc leur union aussi, ce qui est impossible car elle est trop grosse. ■

Il nous reste à présent à introduire un dernier type d'obstruction inconditionnelle.

Définition 7.4.4 (pré-conflit,conflit)
On dira que 2 gêneurs i et j sont en pré-conflit *de niveau $p \leqslant l$ si $|i \cap j| > t^{max}_{p-1}$. En français, il faut au moins p couleurs dans l'intersection. On dira que 2 gêneurs i et j sont en* conflit *de niveau p si en plus d'être en pré-conflit de niveau p on a $|i \cup j| > t^{max}_{2l-p}$.*

Lemme 7.4.5
S'il existe un conflit, alors il n'y a pas de coloration acceptable.

Preuve :

En effet, supposons que l'on ait un conflit de niveau p. Il faut au moins p couleurs dans l'intersection donc chacun des deux gêneurs a droit à au plus $l - p$ couleurs supplémentaires pour le couvrir. On a donc droit à $p+2(l-p) = 2l-p$ couleurs au total pour couvrir l'union mais la deuxième inégalité nous dit que les $2l - p$ plus grandes couleurs sont insuffisantes. CQFD. ■

Remarques sur les conflits : si $2l - p \geqslant k$, alors $|i \cup j| \leqslant t^{max}_{2l-p}$ donc il ne peut y avoir de conflit. Si $l = k - 1$, alors on regarde les valeurs de p telles que $2k - 2 - p < k$ et $p \leqslant k - 1$, c.-à-d. $k - 2 < p \leqslant k - 1 \Leftrightarrow p = k - 1$.

Proposition 7.4.6
Soit (I, C) une instance de P_k^{k-1} avec épaisseur 2 où $k \geqslant 4$. Cette instance admet une coloration admissible si et seulement si elle n'a ni mégalomane, ni conflit, ni guerre.

Preuve :

Dans la preuve qui suit, on considère que les couleurs sont indexées par ordre de taille croissante, $c_1, c_2, \ldots, c_k$ tel que $t(c_1) \leqslant t(c_2) \leqslant \cdots \leqslant t(c_k)$.

$\Rightarrow$: Par contraposée, les lemmes précédents nous montrent que s'il y a un mégalomane, un conflit ou une guerre, il n'y a pas de coloration admissible.

$\Leftarrow$: 1er cas : S'il n'y a pas de gêneur, toute coloration continue est admissible.

2e cas : Il existe un gêneur isolé (sans pré-conflit). Comme avec épaisseur 1, on peut le colorer de manière continue en épuisant $k - 2$ couleurs à l'intérieur. Les éventuels intervalles qui le chevauchent ne posent pas de problème car ce ne sont pas des gêneurs et on utilise une coloration continue.

3e cas : Il existe deux gêneurs exigeants i et j. Comme il n'y a pas de guerre, $|i \cup j| = |i| + |j| - |i \cap j| \leqslant t^{max}_{k-1}$ donc $|i \cap j| \geqslant |i| + |j| - t^{max}_{k-1}$. De plus i, j gêneurs exigeants implique $t^{max}_{k-2} + t_1 < |i|, |j|$. D'où $|i \cap j| \geqslant 2t^{max}_{k-2} + 2t_1 + 2 - t^{max}_{k-1} = t^{max}_{k-2} + 2t_1 - t_2 + 2$. En conséquence, si $t_2 \leqslant 2t_1 + 2$, on peut épuiser les $k-2$ plus grandes couleurs dans l'intersection. Sinon, comme $t^{max}_{k-2} = t^{max}_{k-3} + t_3$ et $t_3 \geqslant t_2$, on peut épuiser les $k - 3$ plus grandes couleurs dans l'intersection. Comme $k \geqslant 4$, on arrive donc toujours à colorer i et j avec les $k - 1$ plus grandes couleurs en en épuisant au moins une dans l'intersection.

4e cas : Tout gêneur est impliqué dans un pré-conflit. Soit i le gêneur le plus à gauche et j le second plus à gauche donc l'unique gêneur avec lequel i est en pré-conflit. Si $i \cup j \leqslant t_{k-1}^{max}$, on peut faire une coloration continue de $i \cup j$ en partant du bord gauche de i et en utilisant les $k-1$ plus grandes couleurs par ordre croissant de taille. Comme i est un gêneur et $k \geqslant 4$, $|i| > t_2$ donc on aura épuisé au moins une couleur.

Sinon on a besoin des k couleurs pour colorer i et j.

1er sous-cas : Si i n'est pas exigeant et $|i \backslash j| \geqslant t_1$, on fait une coloration continue commençant par la couleur c_1 au bord gauche de i, ainsi on aura épuisé cette couleur dans $i \backslash j$. Si $|i \backslash j| < t_1$ comme $|i| \geqslant t_1 + t_2 + 2$, on a $|i \cap j| > t_2 + 2$. On peut donc colorer $i \backslash j$ avec un bout de t_1 et finir par une coloration continue décroissante de j avec les $l-1$ plus grandes couleurs. Comme il n'y a pas de conflit, $|i \cap j| \leqslant t_{l-2}^{max}$ donc i voit bien moins de $l-1$ couleurs. Si cette coloration n'épuise pas de couleur dans $i \cap j$, on peut prendre c_2 et l'insérer au niveau du bord gauche de j en décalant toutes les couleurs à sa droite. Comme $|i \cap j| > t_2 + 2$, on aura bien épuisé une couleur dans $i \cap j$ et i qui ne voyait que 2 couleurs en verra 3.

2e sous-cas : Si i est exigeant et $|i \backslash j| \geqslant t_2$, on fait une coloration continue commençant par la couleur c_2 au bord gauche de i, ainsi on aura épuisé cette couleur dans $i \backslash j$. Sinon on colore $i \backslash j$ avec un bout de t_2 et on finit par une coloration continue décroissante de j avec les autres couleurs (j n'est pas exigeant). Comme il n'y a pas de conflit, $|i \cap j| \leqslant t_{l-2}^{max}$ donc i voit bien moins de $l-1$ couleurs. Comme i est exigeant, $|i| > t_l + t_{l-1} + t_1$ donc $|i \cap j| > t_l$ d'où on épuise t_l dans $i \cap j$. CQFD. ∎

7.5 Conclusion

Dans ce chapitre, nous avons mis en évidence l'existence d'un effet de seuil pour la complexité de problèmes combinatoires voisins. Ce phénomène, formalisé par le théorème 7.3.15 pour les hypergraphes avec des hyperarêtes disjointes, peut être vu comme une sorte d'analogue combinatoire au théorème de Schaefer [62] qui sépare les sous-problèmes de SAT en problèmes NP-complets ou linéaires. Il est plus surprenant de constater l'inversion de complexité entre la classe des hypergraphes et la première sous-classe considérée.

Dans le cas des hypergraphes d'épaisseur 2, nous n'avons que des résultats partiels de linéarité. Il serait intéressant d'étendre ces résultats et de poursuivre l'étude du problème quand l'épaisseur croît, afin de constater si la frontière entre problèmes polynomiaux et NP-difficiles évolue ou non. La question se pose aussi de savoir si les problèmes P_k^l sont toujours linéaires ou NP-complets, ou si une complexité intermédiaire apparaît. Enfin la complexité sur les hypergraphes d'intervalles est aussi ouverte. Un aspect intéressant à mes yeux est que le nombre d'obstructions à considérer augmente avec le degré maximum ; il n'est pas clair que l'ensemble global des obstructions à considérer lorsqu'on étudie la classe des hypergraphes d'intervalles puisse être facilement défini. Mon sentiment est que tous les problèmes P_k^l sont NP-difficiles sur la classe des hypergraphes d'intervalles.

Conclusion et perspectives

Dans le chapitre précédent, j'ai étudié la complexité du problème P_k^l de partition des sommets d'un hypergraphe en k classes de taille donnée telle que chaque hyperarête ait des sommets dans au plus l classes de la partition. La preuve y était notamment apportée que, sur des hypergraphes quelconques, les problèmes P_k^l sont polynomiaux quand $l = 1$ et NP-complets quand $l > 1$, alors que, sur des hypergraphes de degré maximum 1, les problèmes P_k^l sont NP-complets quand $l = 1$ et linéaires sinon, ce phénomène d'inversion résultant en partie du codage plus compact d'une instance sur cette sous-classe. Ces recherches sur le partitionnement d'hypergraphes mériteraient d'être poursuivies dans plusieurs directions. À court terme, il serait intéressant de savoir si les problèmes P_k^l sont toujours NP-complets quand $l = 1$ et linéaires sinon, sur les hypergraphes d'intervalles de degré maximum d. Ou bien, si rapidement les problèmes P_k^l pour $l \neq 1$ deviennent NP-complets comme sur les hypergraphes quelconques.

À long terme, ce même problème de partitionnement pourrait être étudié sur la classe des hypergraphes d'arcs de cercle[1] (voire de cercles) ou sur les hypergraphes de rectangles[2]. Ces classes[3] ayant toutes un codage compact en espace logarithmique par rapport au nombre de sommets, il serait intéressant de savoir si elles présentent elles aussi le phénomène d'inversion de complexité avec la classe des hypergraphes quelconques quand on restreint leur degré maximum[4] ou non. (On sait déjà que P_k^1 sera NP-complet pour tout k.)

Concernant la complexité algébrique, les perspectives à court terme sont évidentes. Il s'agit premièrement de chercher à refermer l'écart entre la borne inférieure ($\mathsf{VP_e}$) et la borne supérieure (VP) que nous avons montrées pour la complexité des trois interprétations combinatoires du permanent et de l'hamiltonien sur les graphes de largeur de clique pondérée bornée. La seconde

[1] Dans un tel hypergraphe, les sommets sont placés de manière cyclique et les hyperarêtes sont des arcs de ce cercle. Un hypergraphe d'arcs de cercles place quant à lui les sommets sur plusieurs cercles (chaque sommet sur un seul cercle) et chaque hyperarête est un arc de l'un des cercles. On peut aussi imaginer des hypergraphes d'arcs de multi-cercles où chaque sommet peut être présent dans plusieurs cercles. Je pense que le lecteur aura compris comment définir les hypergraphes de multi-arcs de cercles ou les hypergraphes de multi-arcs de multi-cercles.

[2] Les sommets sont placés selon une grille et chaque hyperarête est une sous-grille rectangulaire.

[3] On peut en imaginer d'autres.

[4] On n'osera tout de même pas se restreindre à être inférieur ou égal à 1…

perspective est quant à elle d'étudier la complexité d'autres couvertures et polynômes de graphes, tels que le polynôme coloré de Tutte ou les couplages partiels connus tous deux pour être VNP-complets sur les graphes quelconques. Il s'agira donc d'observer si, là encore, leur complexité est celle de $\mathsf{VP_e}$ sur les graphes de largeurs linéaire ou arborescente bornée. Leurs complexités sur la classe des graphes planaires seront aussi d'un grand intérêt car elles permettront de juger du caractère accidentel de la VDET-complétude des couplages parfaits face à la VNP-complétude des couvertures par circuits et des circuits hamiltoniens.

Arrivé à la fin de ce mémoire, il me faut constater que je n'ai que très partiellement défendu la thèse, présentée en introduction, selon laquelle les classes de graphes les plus significatives capturent intrinsèquement une part de la complexité des graphes et par extension des complexités algorithmique et algébrique. En effet, nos résultats de classes de complexité algébrique capturées par des couvertures de graphes sur les classes de graphes de largeur bornée ou sur les graphes planaires ne concernent que trois types de couverture, même si celles-ci sont les plus fondamentales du domaine. Les opérateurs algébriques ou couvertures de graphe qui nous ont permis de passer de la complexité d'une classe de graphe à celle d'une classe de complexité algébrique ne sont pas absolus et les perspectives à long terme de ce travail consistent à encadrer, à mesurer ce biais. Je vais donc tenter de dessiner quels types de résultats seraient une défense convenable de cette thèse.

Dans son habilitation, Bürgisser présente le concept unificateur de fonction génératrice d'une propriété de graphe. Il définit une propriété de graphe comme un ensemble $\mathcal{E}$ de graphes finis stable par isomorphisme. La fonction génératrice d'une telle propriété sur un graphe $G = (V, E)$ est alors définie par

$$\mathrm{GF}(G, \mathcal{E}) = \sum_{E' \subseteq E} w(E'),$$

où la somme porte sur tous les sous-ensembles d'arêtes E' tels que $(V, E') \in \mathcal{E}$. Là encore le poids $w(E')$ d'un ensemble d'arêtes est défini comme le produit des poids de ses arêtes.

On peut ainsi définir le permanent comme la fonction génératrice de la propriété de graphe « être une union disjointe de circuits » prise sur la famille de cliques pondérées G_n^X dont l'arc de i vers j est de poids $X_{i,j}$. De même, l'hamiltonien peut être défini comme la fonction génératrice de la propriété de graphe « être un circuit » prise sur la famille de cliques pondérées G_n^X.

Par extension, on peut considérer la fonction génératrice d'une matrice sur $\mathbb{K}$ en remplaçant un graphe pondéré par sa matrice d'adjacence mais surtout, de manière plus intéressante, on peut considérer la fonction génératrice d'une formule logique ϕ sur le vocabulaire τ_1 ou τ_2 :

$$\mathrm{GF}(G, \phi) = \sum_{E' \subseteq E} w(E'),$$

où la somme porte sur tous les sous-ensembles d'arêtes E' tels que (V, E') est un modèle de ϕ. Cette possibilité nous permet de définir le permanent comme une fonction génératrice du premier ordre sur le vocabulaire τ_1. En effet, la condition « être une union disjointe de circuits » peut s'exprimer par la condition locale que tout sommet a un unique voisin intérieur et un unique voisin

extérieur, ce qui se traduit par la formule

$$\phi_1 = \forall x, [(\exists y, \mathrm{Adj}(x,y) \wedge (\forall z, \mathrm{Adj}(x,z) \Rightarrow y = z))$$
$$\wedge \quad (\exists u, \mathrm{Adj}(u,x) \wedge (\forall v, \mathrm{Adj}(v,x) \Rightarrow u = v))].$$

On peut aussi définir le permanent à l'aide des couplages parfaits pris sur les bipartis complets. Là encore, il s'agit d'une fonction génératrice du premier ordre sur le vocabulaire τ_1. La formule en est encore plus simple puisqu'il suffit de vérifier que tout sommet a un unique voisin :

$$\phi_2 = \forall x, [\exists y, \mathrm{Adj}(x,y) \wedge (\forall z, \mathrm{Adj}(x,z) \Rightarrow y = z)].$$

Dans le cas non-orienté, on suppose bien sûr que la relation Adj d'adjacence est symétrique.

L'hamiltonien est un peu plus complexe du point de vue logique car la connexité entre deux sommets ne peut être vérifiée qu'au moyen d'un chemin, c'est-à-dire d'un ensemble de sommets ou d'arcs, ce qui oblige à avoir recours à la logique monadique du second ordre sur le vocabulaire τ_1 ou τ_2.[5]

Avec tous ces concepts, on peut à présent donner l'idée de résultats capables d'appuyer réellement ma thèse. Ceux-ci sont du type suivant.

Théorème imaginaire 1
Toute fonction génératrice de la logique XO vérifiant les conditions C est VX*-complète sur les graphes de largeur XL bornée.*

Ce type d'énoncé à trou est un peu confus donc pour prendre un exemple un peu plus concret, on peut choisir comme logique XO la logique monadique du second ordre sur le vocabulaire τ_2, un ensemble de conditions C égal à « être VNP-complet sur les graphes quelconques », $\mathsf{VX} = \mathsf{VP_e}$ et $XL =$ arborescente. Ce qui nous donne :

Théorème imaginaire 2
Toute fonction génératrice de la logique monadique du second ordre sur le vocabulaire τ_2, VNP*-complète sur les graphes quelconques est* $\mathsf{VP_e}$*-complète sur les graphes de largeur arborescente bornée.*

[5] On obtient ainsi les formules :

$$\phi_3 = \forall x \in V, [(\exists y \in V, \mathrm{Adj}(x,y) \wedge (\forall z \in V, \mathrm{Adj}(x,z) \Rightarrow y = z))$$
$$\wedge \quad (\exists u \in V, \mathrm{Adj}(u,x) \wedge (\forall v \in V, \mathrm{Adj}(v,x) \Rightarrow u = v))]$$
$$\wedge \quad \forall x, y \in V, \neg[\exists C \subset V, x \in C \wedge y \notin C \wedge \forall z \in C,$$
$$(\mathrm{Adj}(z,t) \Rightarrow t \in C)].$$

$$\phi_4 = \forall x \in V, [(\exists y \in V, \mathrm{Adj}(x,y) \wedge (\forall z \in V, \mathrm{Adj}(x,z) \Rightarrow y = z))$$
$$\wedge \quad (\exists u \in V, \mathrm{Adj}(u,x) \wedge (\forall v \in V, \mathrm{Adj}(v,x) \Rightarrow u = v))]$$
$$\wedge \quad \forall x, y \in V, [\exists C \subseteq E, \forall e \in C,$$
$$((\mathrm{Inc}(x,e) \vee (\exists f \in C, z \in V, \mathrm{Inc}(z,e) \wedge \mathrm{Inc}(f,z)))$$
$$\wedge((\mathrm{Inc}(e,y) \vee (\exists f \in C, z \in V, \mathrm{Inc}(e,z) \wedge \mathrm{Inc}(z,f))))].$$

Dans la formule ϕ_4 ci-dessus, je triche un peu car j'utilise la relation d'adjacence alors que je ne suis plus sensé avoir que la relation d'incidence « Inc ». Néanmoins, il est facile de redéfinir l'adjacence à partir de l'incidence sous la forme d'une « macro ».

Notons que ce genre d'énoncé semble dramatiquement compromis pour les graphes planaires, même pour la logique du premier ordre. En effet, les résultats du chapitre 6 montrent que les deux interprétations combinatoires du permanent sont l'une VNP-complète, l'autre VDET-complète dans ce cas. En conséquence, sauf si VDET $=$ VNP ou si l'on trouve un ensemble pertinent de conditions C capables de distinguer la formule ϕ_1 de la formule ϕ_2, il faut abandonner cette optique pour les graphes planaires. (Il est possible qu'un ensemble de conditions C qui intègre tout bêtement le fait que la relation Adj soit symétrique ou non fonctionne, mais cela reste à vérifier.)

Pour ces théorèmes imaginaires, l'une des difficultés résidera sans doute dans la variation de complexité d'une fonction génératrice selon la caractéristique du corps $\mathbb{K}$, tel le permanent qui est VDET-complet en caractéristique 2 et VNP-complet sinon. Peut-être faudra-t-il se limiter aux corps de caractéristique nulle.

Bien sûr, se restreindre à la complexité algébrique n'est pas suffisant ; il faudrait aussi des résultats similaires dans le cadre de la complexité booléenne. Ceux-ci seraient du type :

Théorème imaginaire 3
Tout problème de décision exprimable dans la logique XO vérifiant les conditions C est X*-complet sur les graphes de largeur XL bornée.*

Avec une fois encore comme exemple, XO égal à la logique monadique du second ordre sur le vocabulaire τ_2, un ensemble de conditions C égal à « être NP-complet sur les graphes quelconques », $\mathsf{X} = \mathsf{NC}^1$ et $XL =$ arborescente, on obtient :

Théorème imaginaire 4
Tout problème de décision exprimable dans la logique monadique du second ordre sur le vocabulaire τ_2, NP*-complet sur les graphes quelconques est* NC^1*-complet sur les graphes de largeur arborescente bornée.*

Bien entendu, dans l'exemple précédent, je n'ai pas choisi NC^1 par hasard. Ce choix découle de l'analogie avec le cas algébrique où $\mathsf{VP_e} = \mathsf{VNC}^1$ et surtout du résultat de Bodlaender permettant d'obtenir une décomposition arborescente approchée de profondeur logarithmique.

Pour en finir avec cette série de théorèmes imaginaires, il serait peut-être possible de montrer un théorème de dichotomie pour les fonctions génératrices du premier ordre, ce qui fournirait un analogue algébrique au théorème de Schaeffer. Ce résultat serait du type :

Théorème imaginaire 5
Toute fonction génératrice du premier ordre est VNP*-complète (en caractéristique nulle ?) sur les graphes quelconques si et seulement si elle vérifie les conditions C. De plus, dans le cas contraire celle-ci est dans* VP*(*$\mathsf{VP_e}$ *?).*

Tous ces théorèmes imaginaires seront l'objet, je l'espère, de mes recherches futures.

Annexe A

Un peu plus de NP-complétude

L'objet de cette annexe est de montrer que le problème P_k^l avec couleurs équilibrées est NP-complet pour $l = k - 1$ et k premier à l'aide d'une réduction à partir de la k-colorabilité d'un graphe. Cette preuve est une généralisation de la preuve donnée par Kloks, Kratochvíl et Müller dans [44] pour $k = 3$. Comme il est précisé dans le chapitre 7, celle-ci n'est pas utile mais jolie. On peut donc considérer que cette annexe est un peu mon caprice de thésard.

Soit $G = (V, E)$ un graphe avec $|V| = n > 0$ et $|E| = m$. On choisit un sommet $v_0 \in V$ et un ensemble W tel que $|W| = (k-1)n$ et $V \cap W = \{v_0\}$. On pose $Y[k] = \{1, 2, \ldots, k\}$. L'hypergraphe $H_G = (X, S)$ est défini par $X = (V \cup W) \times Y[k]$ et

$$S = \{s_1^D = \{(v,i)/i \in D\} \cup \{(v',j)/j \in Y[k] \backslash D\}/\{v,v'\} \in E \text{ et } \emptyset \subset D \subset Y[k]\}$$

$$\cup \{s_2^j = \{(w,i)/w \in W \text{ et } i \neq j\}/j \in Y[k]\}.$$

Intuitivement, on a fait k copies de G muni d'une extension W et l'on a ajouté deux types d'hyperarêtes : le premier comprend des hyperarêtes qui dominent les k copies d'une arête de G en contenant k sommets parmi les $2k$ copies des extrémités de l'arête ; de plus ces k sommets ne sont pas les k copies d'une seule extrémité. Le second est composé de k grosses hyperarêtes couvrant $k-1$ des k copies de tout sommet de W.

Remarque : $|X| = k^2 n - k$ et $|S| = (2^k - 2)m + k$ donc $|H_G|$ est bien polynomiale en la taille de G quand k est fixé.

Lemme A.0.1
Si G est k-colorable, alors H_G a une coloration admissible.

Preuve :

> Soit $f : V \to \{1, 2, \ldots, k\}$ une coloration propre de G. On définit une coloration admissible $c : X \to \{1, 2, \ldots, k\}$ par $c(v, i) = j$ si et seulement si $v \in V$ et $f(v) + i \equiv j[k]$ et $c(w, i) = c(v_0, i)$ pour tout $w \in W$.

En effet, chaque copie de v a une couleur différente donc chaque couleur est de taille $kn-1$. Pour les hyperarêtes du premier type, considérons une arête $vv' \in E$ et les sommets $(v,i),(v',i)$ qui interviennent dans les hyperarêtes correspondantes. Essayons de trouver un sous-ensemble M de ces sommets qui, pour chaque valeur de i, ne contient qu'un des deux sommets $(v,i),(v',i)$ et voit les k couleurs. Supposons, sans perte de généralité, que $(v,i_0) \in M$; comme $f(v) \neq f(v') \Rightarrow f(v) \equiv f(v') + a[k], 0 < a < k \Rightarrow f(v)+i_0 \not\equiv f(v')+i_0[k]$ et que la couleur $f(v')+i_0 \mod k = c(v',i_0)$ n'apparaît qu'une fois du côté de v', on doit prendre dans M le sommet (v,i_0-a). De même, si $c(v',i_0-a)$ n'est pas parmi $c(v,i_0), c(v,i_0-a)$, on doit prendre (v,i_0-2a) dans M. Soit q le premier indice tel que $c(v',i_0-qa)$ soit égal à $c(v,i_0-q'a)$ pour $0 \leqslant q' < q$; on a alors $f(v)+i_0-q'a \equiv f(v')+i_0-qa \equiv f(v)+i_0-(q+1)a[k]$ d'où $((q+1)-q')a \equiv 0[k]$. Comme k est premier et $a<k$, on a $(q+1) \equiv q'[k]$ d'où $q \geqslant k-1$ donc on doit prendre au moins k éléments (v,i_0-ja), $0 \leqslant j \leqslant q$ du côté de v, c.-à-d. la totalité, dans M. En conséquence, M ne peut être une des hyperarêtes donc celles-ci voient bien moins de k couleurs.

Pour les hyperarêtes du second type, comme elles englobent $k-1$ copies de W et que chaque copie ne voit qu'une couleur, elles ne voient bien que $k-1$ couleurs.

■

Remarque : avec la construction de l'hypergraphe choisie ainsi que le type de coloration à partir d'une coloration propre de G, la condition k premier est nécessaire car si $f(v) = f(v')+a$ où $a \mid k$ et $k/a = r$ (ce qui se produit forcément pour au moins une arête si le graphe est parfait, est-ce qu'on peut montrer que ça se produit forcément dans toute coloration d'un graphe appartenant à une classe pour laquelle la k-coloration est NP-complète ?), alors l'hyperarête $s_1^D = \{(v,i)/i \in D\} \cup \{(v',j)/j \in Y[k] \backslash D\}$ avec $D = \{1, 1+a, \ldots, 1+(r-1)a\}$ voit les k couleurs.

On pourrait toutefois envisager un autre type de coloration ; toujours sur le principe d'une permutation des k couleurs sur les k copies d'un sommet mais sans que cette permutation soit induite par un décalage. La vraie condition nécessaire et suffisante dans la preuve ci-dessus est en effet d'avoir un ensemble de k permutations $\sigma_1, \ldots, \sigma_k$ des k couleurs telles que pour tout couple (σ_i, σ_j), le plus petit ensemble d'indices D non vide tel que $\bigcup_{d \in D}\{\sigma_i(d)\} = \bigcup_{d \in D}\{\sigma_j(d)\}$ est nécessairement égal à $Y[k]$. Formalisons tout ça à l'aide de quelques définitions.

Définition A.0.2 (permutations disjointes)
Deux permutations σ, σ' sont dites disjointes si $\sigma(d) \neq \sigma'(d), \forall d \in Y[k]$.

Trouver k permutations disjointes est extrêmement facile puisque les k décalages fonctionnent toujours. Remarquons qu'il existe déjà une définition standard de permutations disjointes où l'on demande que les orbites des deux permutations soient disjointes. Ces deux définitions s'intersectent proprement puisque les permutations $(2,1,3,4,5)$ et $(1,2,4,3,5)$ sont disjointes au sens standard et pas au nôtre, tandis que les permutations $(2,1,3,4)$ et $(1,2,4,3)$ sont disjointes dans les deux sens et les permutations $(2,3,1)$ et $(3,1,2)$ ne sont pas disjointes au sens standard et le sont au nôtre.

Définition A.0.3 (permutations fortement disjointes)
Deux permutations σ, σ' sont dites fortement disjointes si le plus petit ensemble d'indices D non vide tel que $\bigcup_{d\in D}\{\sigma(d)\} = \bigcup_{d\in D}\{\sigma'(d)\}$ est nécessairement égal à $Y[k]$.

Dans le cas où k est premier, ces k permutations fortement disjointes nous sont fournies par les k décalages. Je ne sais pas s'il existe d'autres solutions à ce problème, pour k premier ou non ; ni même si ce problème a déjà été regardé. Il pourrait être amusant de chercher la cardinalité maximum d'un tel ensemble de permutations en fonction de k.

Lemme A.0.4
Pour toute coloration admissible c de H_G, on a $c(w,i) = c(w',i)$ pour tous $w, w' \in W$ et $i \in Y[k]$.

Preuve :

Si une couleur n'est pas présente dans les W_i alors elle est de taille au plus $kn - k < kn - 1$ donc la coloration n'est pas équilibrée. Si l'un des W_i voit au moins 2 couleurs, comme chaque couleur apparaît dans au moins un des W_i, on peut trouver un ensemble d'au plus $k-1$ W_i qui voit toutes les couleurs ; mais cet ensemble est dans l'une des hyperarêtes donc la coloration n'est pas admissible. Absurde. ■

Le caractère joli que j'attribue à cette preuve est grandement lié au lemme suivant dont j'ignore s'il est original ou une conséquence directe d'un autre résultat plus puissant. Par souci de simplicité, on notera $(v, I) = \{(v, i)/i \in I\}$.

Lemme A.0.5 (Lemme de convergence chromatique)
Soient $X = (u, I) \cup (v, I)$, où I est un ensemble fini non vide, et $c : X \to C$ une coloration de X. Si $c(u, I) \subseteq c(v, I)$, alors $\exists\, \emptyset \subset J \subseteq I$ tel que $c(u, J) = c(v, J)$.

Version infinie (bonus) : Soient $X = (u, I) \cup (v, I)$, où I est un ensemble non vide, et $c : X \to C$ une coloration de X. Si $c(u, I) \subseteq c(v, I)$ et $c(v, I)$ est de taille finie, alors $\exists\, \emptyset \subset J$ fini $\subseteq I$ tel que $c(u, J) = c(v, J)$.

Preuve :

Par récurrence sur $|I|$, si $|I| = 1$, alors $c(u, I) = c(v, I)$ et donc $J = I$ convient. Supposons que la propriété soit vérifiée au rang r et considérons I tel que $|I| = r + 1$. Si $c(u, I) = c(v, I)$, I convient. Sinon, soit $i \in I$ tel que $c(v, i) \notin c(u, I)$, alors $I\backslash\{i\}$ vérifie $c(u, I\backslash\{i\}) \subseteq c(v, I\backslash\{i\})$ et $|I\backslash\{i\}| = r$. Donc, par hypothèse de récurrence, $\exists J \subseteq I\backslash\{i\} \subset I$ tel que $c(u, J) = c(v, J)$.

Version infinie (preuve qui marche aussi pour le cas fini) : par récurrence sur $|c(v, I)|$, si $|c(v, I)| = 1$, alors $c(u, I) = c(v, I) = \{c_1\}$ et donc tout $\emptyset \subset J$ fini $\subseteq I$ convient. Supposons que la propriété soit vérifiée au rang r et considérons I tel que $|c(v, I)| = r + 1$. Si $c(u, I) = c(v, I)$, alors il existe un ensemble $\emptyset \subset J \subseteq I$ d'au plus $2r + 2$ éléments tel que $c(u, J) = c(v, J)$. Sinon, soient $c_1 \in c(v, I)\backslash c(u, I)$ et $I' = I\backslash\{i \in I/c(v, i) = c_1\}$. Comme $|c(v, I)| \geqslant 2$, $I' \neq \emptyset$. De plus, $|c(v, I')| = r$ donc, par hypothèse de récurrence, $\exists\, \emptyset \subset J$ fini $\subseteq I' \subset I$ tel que $c(u, J) = c(v, J)$. CQFD. ■

Remarque : une généralisation avec un nombre infini de couleurs est trivialement fausse (considérer $I = \mathbb{N}$, $C = \mathbb{N}$, $c(u,i) = i+1$ et $c(v,i) = i$ pour tout $i \in I$). De même, on ne peut exiger que l'ensemble J soit infini (considérer $I = \mathbb{N}$, $C = \{c_1, c_2\}$, $c(u,i) = c_1$ pour tout $i \in I$, $c(v,1) = c_1$ et $c(v,i) = c_2$ pour tout $i > 1$). Sauf peut-être si on impose d'avoir une couleur c_i qui apparait un nombre infini de fois dans (u,I) et dans (v,I) (on peut peut-être avoir J de cardinal égal au max du min des cardinaux d'une couleur dans (u,I) et (v,I)).

Lemme A.0.6
Si c est une coloration admissible de H_G et $\{u,v\}$ est une arête de G telle que les k couleurs apparaissent sur $(v,1),\ldots,(v,k)$, alors les k couleurs apparaissent sur $(u,1),\ldots,(u,k)$.

Preuve :

Supposons sans perte de généralité que $c(v,i) = i$ pour tout $i \in Y[k]$. On a alors $c(u,i) \neq i$ pour tout $i \in Y[k]$ car si $c(u,i_0) = i_0$, l'hyperarête $\{(u,i_0)\} \cup \{(v,i)/i \neq i_0\}$ voit k couleurs. On peut généraliser cet argument de la manière suivante : soient $I \subseteq Y[k]$ un ensemble d'indices, $c(u,I)$ (respectivement $c(v,I)$) l'ensemble des couleurs vues en I du côté de u (respectivement v) ; si $c(u,I) = c(v,I)$, alors $I = Y[k]$. En effet, si $I \subset Y[k]$, alors $(u,I) \cup (v, Y[k]\backslash I)$ est une hyperarête du premier type qui voit les k couleurs. Comme le lemme précédent nous assure l'existence d'un tel sous-ensemble d'indices, on en conclut immédiatement que les k couleurs apparaissent sur $(u,1),\ldots,(u,k)$. ■

Remarque : il faut que toutes les hyperarêtes du premier type soient dans la construction de l'hypergraphe pour avoir le lemme précédent. En effet, supposons qu'une de ces hyperarêtes n'existe pas, c.-à-d. $\exists\, \emptyset \subset D \subset Y[k]$ tel que $s_1^D = \{(u,i)/i \in D\} \cup \{(v,j)/j \in Y[k]\backslash D\} \notin S$. On note $D = \{i_1,\ldots,i_l\}$ où $i_1 < i_2 < \cdots < i_l$. Considérons la coloration suivante : la couleur de (v,i) est i, celle de (u,i) est i_1 si $i \notin D$ et celle de (u,i_r) est i_{r+1}. Une hyperarête sur ces sommets qui voit les k couleurs contient $\{(v,j)/j \in Y[k]\backslash D\}$ car sinon elle ne voit pas les couleurs qui ne sont pas dans D. Comme elle ne peut être complétée avec $\{(u,i)/i \in D\}$, elle possède un autre élément du côté de v, disons (v,i_r). Mais comme la couleur i_{r+1} n'apparaît qu'en (u,i_r) du côté de u restreint à D, on doit prendre (v,i_{r+1}) dans l'hyperarête. En répétant cet argument, on montre que l'hyperarête autre que s_1^D qui voit k couleurs est le côté de v complet qu'on n'a pas pris comme hyperarête dans la construction car elle verrait toujours k couleurs. On a donc bien une coloration qui met en défaut le lemme précédent sur une construction utilisant moins d'hyperarêtes.

Lemme A.0.7
Si G est connexe et H_G a une coloration admissible, alors G a une k-coloration propre.

Preuve :

D'après le lemme A.0.4, on déduit que les k couleurs apparaissent sur $(v_0,1),\ldots,(v_0,k)$. Par un argument inductif utilisant le lemme A.0.6, on déduit que les k couleurs apparaissent sur $(v,1),\ldots,(v,k)$ pour tout sommet $v \in V$. On définit une k-coloration $f : V \to Y[k]$ de G par $f(v) = c(v,1)$

pour tout $v \in V$. Pour toute arête $\{u, v\} \in E$, on a $f(u) \neq f(v)$ car sinon l'hyperarête s_1^D avec $D = \{1\}$ voit k couleurs. ■

Théorème A.0.8
Le problème P_k^{k-1} dans le cas équilibré est NP*-complet si k est premier.*

Preuve :

La preuve découle directement des lemmes A.0.1 et A.0.7 qui montrent que la k-coloration se réduit au problème P_k^{k-1}. ■

Annexe B

Liens entre les largeurs pondérées

L'objet de cette annexe est de fournir d'une part la preuve qu'il existe des graphes de largeur arborescente bornée et de largeur de clique pondérée non bornée, d'autre part de montrer que tout graphe de largeur linéaire bornée est de largeur de clique pondérée bornée et d'en redéduire de manière triviale que toute formule de taille polynomiale est équivalente au permanent, à l'hamiltonien ou à la somme des poids des couplages parfaits d'un graphe de largeur de clique pondérée bornée. Nous en déduisons aussi que l'écart entre largeur arborescente et largeur de clique pondérée est au plus logarithmique en le nombre de sommets.

B.1 Liens entre largeur arborescente et largeur de clique pondérée

Dans cette section, nous exhibons une famille de graphes planaires de largeur arborescente 2 dont la largeur de clique pondérée n'est pas bornée. Ce résultat a été obtenu en collaboration avec Ioan Todinca.

Cette famille est la suivante : soit T_n l'arbre binaire complet de profondeur n dont toutes les arêtes sont de poids 1. On note T'_n le graphe obtenu en ajoutant un sommet universel u à T_n. On fixe le poids p_x de l'arête entre u et tout sommet x de T_n de sorte que les poids p_x soient tous distincts. Il est évident que tous les graphes de la famille $(T'_n)_{n\in\mathbb{N}}$ sont de largeur arborescente 2. La planarité est aussi triviale à démontrer ; il suffit de montrer par récurrence que tout graphe T'_n admet une représentation planaire sans croisement où la racine de l'arbre T_n induit et le sommet universel u sont sur la face extérieure ; c'est vrai quand $n = 1$ et l'étape de récurrence consiste juste à mettre côte à côte les plongements planaires de deux copies de T'_n, puis d'ajouter un sommet sur la face extérieure qu'on relie aux deux racines des T_n induits et, enfin, de fusionner les sommets universels des deux copies. (On peut aussi voir que tout graphe planaire extérieur, comme T_n, auquel on rajoute un sommet universel

est planaire. J'ignore laquelle des deux preuves se visualise le mieux.)

Pour montrer que cette famille de graphes n'est pas de largeur de clique pondérée bornée, nous empruntons le résultat suivant de Lengauer [48] et un bout de raisonnement de Gurski et Wanke [32].

Définition B.1.1 (largeur min-cut)
Un graphe $G = (V, E)$ possède un arrangement linéaire de largeur min-cut au plus l, s'il existe un ordre total sur les sommets $f : V \to \{1, \ldots, |v|\}$ tel que $\max_{1 \leqslant i < |V|} |\{uv \in E / f(u) \leqslant i < f(v)\}| \leqslant l$. On appelle largeur min-cut d'un graphe G le minimum des largeurs min-cut des arrangements linéaires de G.

Proposition B.1.2 (Lengauer 1982)
Tout arbre binaire complet de profondeur n est de largeur min-cut n .

C'est le « au moins n » de la proposition précédente qui va nous être utile pour prouver par l'absurde que la famille (T'_n) n'est pas de largeur de clique pondérée bornée.

Lemme B.1.3
Si T'_n est de largeur W-NLC au plus k, alors T_n est de largeur min-cut au plus $k \cdot (2\Delta^2 + \frac{k-1}{2}) = k \cdot (18 + \frac{k-1}{2})$ (Δ est le degré maximum de T_n qui vaut 3).

Preuve :

Considérons une décomposition W-NLC de T'_n de largeur k. Soit Ch le chemin dans la décomposition qui va de la racine à la feuille qui correspond au constructeur du sommet universel u de T'_n. Comme les poids des arêtes entre u et les autres sommets sont tous distincts, lors de toute opération d'union de 2 graphes avec ajout d'arêtes $G_1 \times_S G_2$ le long de Ch, si u est dans G_1, alors tous les sommets de G_2 ont nécessairement des étiquettes distinctes. G_2 est donc nécessairement de cardinalité au plus k. Cette décomposition a donc une structure de « peigne » le long de Ch où vient se greffer des petits graphes.

En supprimant la feuille qui correspond au constructeur de u, on obtient une décomposition W-NLC de T_n qui garde cette structure. Elle nous fournit donc une partition $(V_1, V_2, \ldots, V_r)$ des sommets de T_n en blocs d'au plus k sommets qui sont rattachés au « peigne » dans cet ordre. On considère alors un arrangement linéaire des sommets tel que $x \in V_i, y \in V_j, i < j$ implique que $f(x) < f(y)$. L'ordre des sommets à l'intérieur d'un V_i est sans importance. On note $V'_i = \bigcup_{j=1}^{i} V_j$ et $V''_i = \bigcup_{j=r}^{i+1} V_j$, $1 \leqslant i \leqslant r-1$.

Nous utilisons à présent le raisonnement de Gurski et Wanke. Combien de sommets de V'_i peuvent être adjacents à au moins un sommet de V''_i (on notera ce nombre N_i) ? La réponse est au plus $k \cdot \Delta$. En effet, un sommet de V''_i adjacent à un sommet d'étiquette a dans V'_i (on considère l'étiquette du sommet au niveau du noeud de la décomposition où V'_i est intégralement présent dans les feuilles du sous-arbre) est obligé de voir tous les sommets de V'_i d'étiquette a. Comme le degré maximum est Δ, il ne peut y avoir plus de Δ sommets d'étiquette a dans V'_i si un sommet d'étiquette a dans V'_i est relié à au moins un sommet de V''_i. On obtient bien que $N_i \leqslant k \cdot \Delta$ sommets.

À présent, chacun de ces N_i sommets peut voir au plus Δ sommets dans V_i''. Donc le nombre d'arêtes E_i allant de V_i' à V_i'' est nécessairement inférieur à $\Delta^2 \cdot k$.

Notons $x_1, \ldots, x_n$ l'arrangement choisi conformément aux contraintes ci-dessus. Soit x_{j_i} le dernier sommet de V_i dans cet arrangement. Par l'argument précédent, il est clair que $|\{uv \in E / f(u) \leqslant j_i < f(v)\}| \leqslant \Delta^2 \cdot k$. Il reste donc à vérifier cette condition pour les autres x_j. C'est à ce stade que la borne k sur la taille des V_i est importante. En effet, si $x_j \in V_i$ et E_i' est le nombre d'arêtes entre sommets de V_i, $|\{uv \in E / f(u) \leqslant j < f(v)\}| \leqslant E_{i-1} + E_i + E_i' \leqslant \Delta^2 \cdot k + \Delta^2 \cdot k + \frac{k(k-1)}{2} = k \cdot (2\Delta^2 + \frac{k-1}{2})$.

■

Corollaire B.1.4
La famille (T_n') n'est pas de largeur de clique pondérée bornée. Il existe donc des familles de graphes de largeur arborescente bornée et de largeur de clique pondérée non bornée.

B.2 Liens entre largeur linéaire et largeur de clique pondérée

Nous démontrons à présent le lemme suivant prouvant que tout graphe de largeur linéaire bornée est de largeur de clique pondérée bornée. Nous l'utilisons ensuite pour redémontrer que toute formule arithmétique est équivalente au permanent, à l'hamiltonien ou à la somme des poids des couplages parfaits d'un graphe de largeur de clique pondérée bornée. Une autre conséquence en est que tout graphe de largeur arborescente k est de largeur de clique pondérée au plus $O(k \log(n))$ et cette borne est optimale de par la construction de la section précédente.

Lemme B.2.1
Soit G un graphe pondéré (orienté ou non). Si G est de largeur linéaire k, alors G est de largeur de clique pondérée au plus $k + 2$.

Preuve :

Soit $\langle T, (X_t)_{t \in V(T)} \rangle$ une décomposition linéaire de largeur k de G. On appelle *feuille* et *racine* les deux extrémités de la décomposition linéaire. Soit G_t le sous-graphe de G induit par les sommets dans les sacs sous X_t.

On prouve par induction sur la hauteur de $\langle T, (X_t)_{t \in V(T)} \rangle$ que tout graphe G_t peut être construit par des opérations de clique pondérée utilisant au plus $k + 2$ étiquettes distinctes. De plus, à la fin de la construction tous les sommets du sac X_t ont des étiquettes distinctes et tous les autres sommets ont une étiquette *puits* (ou poubelle).

Si $|V(T)| = 1$, alors G a au plus $k + 1$ sommets. On peut les créer avec $k + 1$ étiquettes distinctes et ajouter indépendamment chaque arête entre deux sommets en utilisant des opérations de clique pondérée.

Supposons que $|V(T)| > 1$, soient r la racine et t son fils. Par induction, G_t peut être construit à l'aide d'opérations de clique pondérée utilisant au plus $k+2$ étiquettes distinctes. Pour tout sommet $v \in X_t \backslash X_r$, nous ajoutons une opération de renommage qui donne l'étiquette *puits* à v (cette opération de renommage ne renomme que v puisque, par induction, v a une étiquette distincte de celles des autres sommets). Puisque $|X_r| \leqslant k+1$ et tous les sommets de $V(G) \backslash X_r$ ont l'étiquette *puits*, on peut créer les sommets de $X_r \backslash X_t$ avec des étiquettes distinctes et les ajouter par union disjointe à la construction courante. Il est clair à présent que tous les sommets de X_r ont des étiquettes distinctes et l'on peut ainsi ajouter indépendamment chaque arête entre deux sommets. D'où la conclusion. ■

Proposition B.2.2
Toute formule arithmétique peut s'exprimer comme le permanent d'une matrice de largeur de clique pondérée au plus 22 *et de taille polynomiale en* t*, où* t *est la taille de la formule. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.*

Preuve :

Soit φ une formule de taille t. Grâce à la preuve du théorème 4.3.3, nous savons qu'elle peut être calculée par un circuit asymétrique de largeur 6 et de taille $O(t^{O(1)})$. En conséquence, elle est égale au permanent d'un graphe de taille $O(t^{O(1)})$, de largeur linéaire au plus $\lfloor \frac{7 \cdot 6}{2} \rfloor - 1 = 20$ d'après la proposition 4.1.2, et de largeur de clique pondérée au plus $20 + 2 = 22$, par le lemme B.2.1. ■

Proposition B.2.3
Toute formule arithmétique peut s'exprimer comme l'hamiltonien d'une matrice de largeur de clique pondérée au plus 45 *et de taille polynomiale en* t*, où* t *est la taille de la formule. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.*

Preuve :

Soit φ une formule de taille t. Grâce à la preuve du théorème 4.3.3, nous savons qu'elle peut être calculée par un circuit asymétrique de largeur 6 et de taille $O(t^{O(1)})$. En conséquence, elle est égale au permanent d'un graphe de taille $O(t^{O(1)})$, de largeur linéaire au plus $((7 \cdot 6 + 2) - 1 = 43$ d'après la proposition 4.1.3, et de largeur de clique pondérée au plus $43 + 2 = 45$, par le lemme B.2.1. ■

Proposition B.2.4
Toute formule arithmétique peut s'exprimer comme la somme des poids des couplages parfaits d'une matrice symétrique de largeur de clique pondérée au plus 44 *et de taille polynomiale en* t*, où* t *est la taille de la formule. Toutes les entrées de la matrice sont 0, 1, des constantes ou des variables de la formule.*

Preuve :

C'est une conséquence directe de la proposition B.2.2 et du lemme 2.3.5. ■

Nous utilisons à présent le résultat suivant de Bodlaender [9] pour déduire que l'écart entre largeur arborescente et largeur de clique pondérée est au plus logarithmique.

Théorème B.2.5
Tout graphe G de largeur arborescente k est de largeur linéaire au plus $O(k \log(n))$, où n est le nombre de sommets du graphe.

Corollaire B.2.6
Tout graphe G de largeur arborescente k est de largeur de clique pondérée au plus $O(k \log(n))$, où n est le nombre de sommets du graphe.

Bibliographie

[1] C. Alpert and A. Kahng. Recent directions in netlist partitionning : A survey. *VLSI Journal*, 19(1-2) :1–81, 1995.

[2] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Number 11 in EATCS monographs on theoretical computer science. Springer-Verlag, 1988.

[3] D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *STOC*, pages 1–5. ACM, 1986.

[4] D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1) :150–164, 1989.

[5] R. Beigel and J. Gill. Counting classes : Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1) :3–23, 1992.

[6] M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. In *STOC*, pages 254–257. ACM, 1988.

[7] M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1) :54–58, 1992.

[8] H. L. Bodlaender. NC-algorithms for graphs with small treewidth. In J. van Leeuwen, editor, *WG*, volume 344 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1988.

[9] H. L. Bodlaender. A partial *k*-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2) :1–45, 1998.

[10] H. L. Bodlaender and T. Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. In Z. Fülöp and F. Gécseg, editors, *ICALP*, volume 944 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1995.

[11] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Parallel algorithms for series parallel graphs and graphs with treewidth two. *Algorithmica*, 29(4) :534–559, 2001.

[12] J. Bradley, N. Dingle, W. Knottenbelt, and H. Wilson. Hypergraph-based parallel computation of passage time densities in large semi-Markov models. *Linear Algebra and Its Applications*, July 2004.

[13] R. P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2) :201–206, 1974.

[14] G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *STOC*, pages 175–181. ACM, 1991.

[15] P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Number 7 in Algorithms and Computation in Mathematics. Springer, 2000.

[16] P. Bürgisser. Completeness and reduction in algebraic complexity theory. Habilitationsschrift Universität Zürich, Aug. 1998.

[17] Complexity Zoo. `http://qwiki.caltech.edu/wiki/Complexity_Zoo`.

[18] R. Cooley, B. Mobasher, and J. Srivastava. Web mining : Information and pattern discovery on the world wide web. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI '97)*, pages 558–567, Newport Beach, 1997.

[19] D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4) :825–847, 2005.

[20] B. Courcelle. The monadic second-order logic of graphs I : Recognizable sets of finite graphs. *Inf. Comput.*, 85(1) :12–75, 1990.

[21] B. Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In N. Robertson and P. D. Seymour, editors, *Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1991.

[22] B. Courcelle. *Décompositions arborescentes*. chapitre d'un ouvrage à paraître. Éditions Hermès, mai 2006.

[23] B. Courcelle, J. Engelfriet, and G. Rozenberg. Context-free handle-rewriting hypergraph grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 253–268. Springer, 1990.

[24] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2) :23–52, 2001.

[25] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3) :77–114, 2000.

[26] B. Courcelle and A. Twigg. Compact forbidden-set routing. In W. Thomas and P. Weil, editors, *STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2007.

[27] S. Datta, R. Kulkarni, N. Limaye, and M. Mahajan. Planarity, determinants, permanents, and (unique) perfect matchings. In *Proceedings of 2nd International Computer Science Symposium in Russia CSR*, volume 4649 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[28] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In A. Brandstädt and V. B. Le, editors, *WG*, volume 2204 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2001.

[29] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73, Providence, RI, 1974. American Mathematical Society.

[30] M. E. Fisher. Statistical mechanics of dimers on a plane lattice. *Phys. Rev.*, 124 :1664–1672, 1961.

[31] U. Flarup, P. Koiran, and L. Lyaudet. On the expressive power of planar perfect matching and permanents of bounded treewidth matrices. In *Proceedings of ISAAC*, pages 124–136, 2007.

[32] F. Gurski and E. Wanke. On the relationship between nlc-width and linear nlc-width. *Theor. Comput. Sci.*, 347(1-2) :76–89, 2005.

[33] R. Halin. S-functions for graphs. *Journal Geometry*, 8 :171–186, 1976.

[34] L. A. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion*. EATCS. Springer, 2002.

[35] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.* (3), 2 :326–336, 1952.

[36] D. S. Hochbaum and A. Pathria. The bottleneck graph partition problem. *Networks*, 28(4) :221–225, 1996.

[37] S. il Oum and P. D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4) :514–528, 2006.

[38] D. Karger. Global min cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proc. ACM/SIAM Symp. on Discrete Algorithms (SODA 93)*, pages 21–30, 1993.

[39] R. M. Karp and R. J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28 :191–209, 1982.

[40] G. Karypis, E. Han, and V. Kumar. Chameleon : A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8) :68–75, 1999.

[41] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. Technical Report 98-036, University of Minnesota, 1998.

[42] P. W. Kasteleyn. The statistics of dimers on a lattice. *Physica*, 27 :1209–1225, 1961.

[43] P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, New York, 1967.

[44] T. Kloks, J. Kratochvíl, and H. Müller. New branchwidth territories. In *Proceedings 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 173–183. Springer-Verlag, 1999.

[45] D. E. Knuth. Overlapping pfaffians. *Electr. J. Comb.*, 3(2), 1996.

[46] J. B. Kruskal. Well-quasi-ordering, the tree theorem and Vázsonyi's conjecture. *Transactions of the American Mathematical Society*, 95 :210–225, 1960.

[47] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15 :271–283, 1930.

[48] T. Lengauer. Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees. *SIAM Journal on Algebraic and Discrete Methods*, 3(1) :99–113, 1982.

[49] L. Lyaudet. Largeur de branche des graphes de cordes. Master's thesis, ENSLyon, LIP, 2004. http ://www.ens-lyon.fr/LIP/Pub/DEA2004.php.

[50] L. Lyaudet. Threshold effect in algorithmic complexity : NP-hard and linear variants of hypergraph partitioning. Technical report, ENSLyon, LIP, September 2005. http ://www.ens-lyon.fr/LIP/Pub/rr2005.php.

[51] M. Mahajan, P. R. Subramanya, and V. Vinay. The combinatorial approach yields an NC algorithm for computing pfaffians. *Discrete Applied Mathematics*, 143(1-3) :1–16, 2004.

[52] G. Malod. *Polynômes et coefficients*. PhD thesis, Université Claude Bernard Lyon 1, July 2003.

[53] G. Malod and N. Portier. Characterizing Valiant's algebraic complexity classes. In *Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 704–716. Springer-Verlag, 2006.

[54] G. L. Miller, V. Ramachandran, and E. Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM J. Computing*, 17(4) :687–695, 1988.

[55] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[56] N. Robertson and P. D. Seymour. Graph minors I : Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1) :39–61, 1983.

[57] N. Robertson and P. D. Seymour. Graph minors III : Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1) :49–64, 1984.

[58] N. Robertson and P. D. Seymour. Graph minors X : Obstructions to tree-decomposition. *J. Comb. Theory, Ser. B*, 52(2) :153–190, 1991.

[59] N. Robertson and P. D. Seymour. Graph minors XX : Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2) :325–357, 2004.

[60] D. J. Rose. On simple characterizations of k-trees. *Discrete Math.*, 7(2) :317–322, 1974.

[61] M. Salavatipour. A (1+ε)-approximation algorithm for partitioning hypergraphs using a new algorithmic version of the Lovasz local lemma. *Random Structures and Algorithms*, 25(1) :68–90, 2004.

[62] T. J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226. ACM, 1978.

[63] S. Shekhar and D. Liu. Partitioning similarity graphs : A framework for declustering problems. *Information Systems Journal*, 21(4), 1996.

[64] H. N. V. Temperley and M. E. Fisher. Dimer problems in statistical mechanics – an exact result. *Philosophical Magazine*, 6 :1061–1063, 1961.

[65] S. Toda. On the computational power of PP and $\oplus$P. In *Proc. 30th IEEE Symposium on the Foundations of Computer Science*, pages 514–519, 1989.

[66] S. Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Information and Systems*, E75-D(1) :116–124, 1992.

[67] L. G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.

[68] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2) :189–201, 1979.

[69] L. G. Valiant. Holographic algorithms (extended abstract). In *FOCS*, pages 306–315. IEEE Computer Society, 2004.

[70] L. G. Valiant. Completeness for parity problems. In L. Wang, editor, *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 1–8. Springer, 2005.

[71] L. G. Valiant. Holographic algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, (099), 2005.

[72] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3) :325–356, 1986.

[73] E. Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54(2-3) :251–266, 1994.

[74] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *ACM International Conference on Information and Knowledge Management (CIKM 2001)*, 2001.

[75] Y. Zhang and A. Mackworth. Parallel and distributed finite constraint satisfaction : Complexity, algorithms and experiments. In V. K. L. Kanal, H. Kitano and C. Suttner, editors, *Parallel Processing for Artificial Intelligence*, volume 1, pages 305–334. Elsevier, Amsterdam, 1994.

[76] U. Çatalyürek. *Hypergraph models for sparse matrix partitioning and reordering*. Computer engineering and information science, Bilkent University, November 1999.

[77] U. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10 :673–693, 1999.

[78] Öjvind Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition - relationships and results for random graphs. *Congressus Numerantium*, 132 :39–60, 1998.

Index