Décompositions de graphes : quelques limites et obstructions

Mathieu Chapelle

▶ **To cite this version:**

Mathieu Chapelle. Décompositions de graphes : quelques limites et obstructions. Complexité [cs.CC]. Université d'Orléans, 2011. Français. NNT : . tel-00659666v1

HAL Id: tel-00659666

https://theses.hal.science/tel-00659666v1

Submitted on 13 Jan 2012 (v1), last revised 5 Mar 2012 (v2)

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

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

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE SCIENCES ET TECHNOLOGIES

Laboratoire d'Informatique Fondamentale d'Orléans

THÈSE

présentée par :

Mathieu CHAPELLE

soutenue le : **5 décembre 2011**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/Spécialité : **Informatique**

Décompositions de graphes : quelques limites et obstructions

THÈSE DIRIGÉE PAR :

Ioan TODINCA	Professeur des Universités, Université d'Orléans

RAPPORTEURS :

Cyril GAVOILLE	Professeur des Universités, Université de Bordeaux I
Christophe PAUL	Directeur de Recherche CNRS, Montpellier

JURY :

Michel HABIB	Professeur des Universités, Université Paris 7
Cyril GAVOILLE	Professeur des Universités, Université de Bordeaux I
Mathieu LIEDLOFF	Maître de Conférences, Université d'Orléans
Christophe PAUL	Directeur de Recherche CNRS, Montpellier
Ioan TODINCA	Professeur des Universités, Université d'Orléans
Yann VAXÈS	Professeur des Universités, Univ. de la Méditerranée, Marseille

Version du 22/11/2011 pour jury

Sommaire

Introduction

Modéliser

À la base de toute science, les scientifiques appréhendent des *observations*, dont ils essayent de comprendre le sens, et des *problématiques*, auxquelles ils s'efforcent de trouver des *réponses*.

Face à leur très grande complexité, divers outils ont été développés pour *modéliser* les nombreuses situations rencontrées auxquelles font face les scientifiques, afin de mieux les étudier. La « théorie des graphes » apporte un cadre mathématique permettant de représenter formellement ces situations.

Un « graphe » est une structure mathématique simple, composée de « sommets », pouvant représenter les éléments d'une problématique, et d'« arêtes » reliant ces sommets entre eux, pouvant représenter les liens existant entre les différents éléments. Un graphe est noté $G = (V, E)$, où V est l'ensemble des sommets, et E l'ensemble des arêtes, chacune reliant deux sommets du graphe. Si besoin, les sommets et arêtes du graphe peuvent se voir attribuer des « étiquettes », leur associant ainsi des *informations* supplémentaires telles que des entiers.

Cette apparente simplicité est ce qui fait la force des graphes : de très nombreuses situations peuvent être modélisées sous la forme d'un graphe, dans des domaines aussi divers que la biologie, les transports, les mathématiques, l'informatique, les télécommunications, *etc*.

Prenons pour exemple la carte d'une ville, et associons-lui un graphe (voir FIGURE 0.1). Dans ce graphe, les sommets correspondent aux carrefours, et une arête relie deux sommets s'il existe une rue reliant les deux carrefours correspondants. De plus, ce graphe est étiquetté : à chaque arête est associée un entier, correspondant à la longueur (en mètres) de la rue reliant les deux carrefours.

Il existe de nombreux problèmes théoriques sur les graphes, pour autant de problématiques modélisées par des graphes. Ces problèmes peuvent prendre différentes formes : recherche d'un sous-ensemble d'éléments du graphe, association d'informations à ces éléments, transformation du graphe, *etc*.

Pour illuster cela, donnons deux exemples de problèmes appliqués à notre ville.

Le premier est la problématique que nous rencontrons dès lors que nous souhaitons circuler dans une ville : quel est le chemin le plus court pour aller d'un endroit à un autre ? Dans le graphe, un tel chemin entre deux sommets correspond à un parcours d'une partie du graphe, en traversant un certain nombre d'arêtes, et sa *longueur* correspond à la somme des étiquettes associées aux arêtes parcourues. Bien entendu, il peut

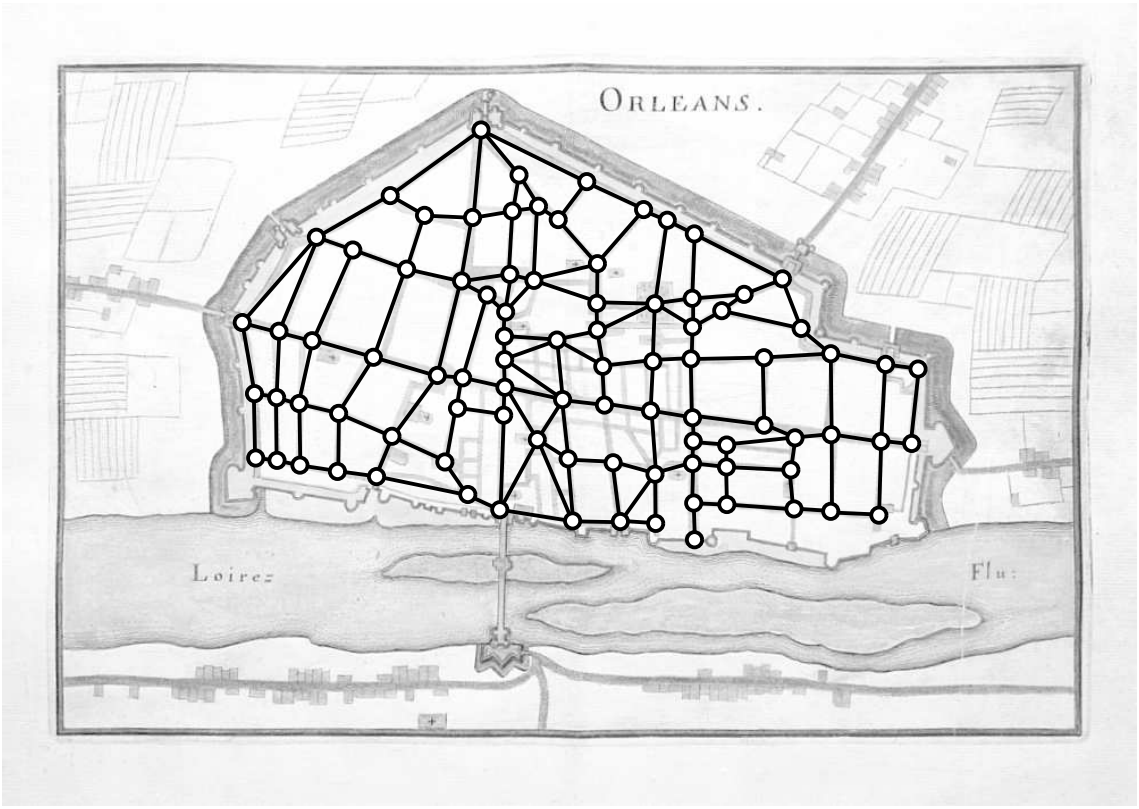


FIGURE 0.1 — Modélisation par un graphe de la ville d'Orléans au XVII^{e} siècle.

exister plusieurs chemins de même longueur, et dans ce cas nous souhaitons simplement en trouver *un* de plus petite longueur.

PLUS COURT CHEMIN

Entrée : Un graphe étiqueté $G = (V, E)$, deux sommets $d, a \in V$.
Question : Trouver un plus court chemin dans G entre les sommets d et a.

Le second problème peut être motivé ainsi : pour rendre la circulation des secours plus rapide en cas d'urgence, un système haute-fréquence est installé sur chaque carrefour, et lorsqu'un véhicule de secours arrive à proximité, tous les feux passent au rouge. Toutefois, nous ne voulons pas que ce système paralyse la circulation de tout un quartier, en passant au rouge les feux des autres carrefours à proximité, et devons donc assigner des fréquences différentes aux carrefours proches. Enfin, les bandes de fréquences étant une ressource précieuse, et pour que ce système ne soit pas trop complexe à gérer, nous souhaitons minimiser autant que possible le nombre fréquences différentes utilisées.

En terme de graphe, ce type de problème est généralement vu comme un problème de coloration, chaque couleur représentant une fréquence différente. Une assignation des fréquences aux carrefours de la ville correspond alors à une coloration des sommets

du graphe, et le fait d'éviter les interférences entre différents carrefours se traduit par des couleurs différentes associées aux sommets *adjacents* du graphe.

COLORATION

Entrée : Un graphe $G = (V, E)$, un entier $K \in \mathbb{N}$.
Question : Peut-on colorier les sommets de G avec au plus K couleurs différentes, de sorte que les couleurs associées à deux sommets *adjacentes* soient distinctes ?

Ces deux problèmes modélisent ainsi des problématiques réelles, sous la forme de problèmes sur un graphe. Et tout naturellement, nous souhaitons trouver une solution optimale à ces problèmes, si de telles solutions existent. Mais sur des problématiques réelles de taille conséquente, chercher une solution ne peut se faire manuellement ; nous préférons pour cela écrire un algorithme capable de calculer cette solution. Dès lors, est-il possible d'écrire un algorithme pouvant effectuer ce calcul en un temps *raisonnable* ?

Comparer

Tous les problèmes n'ont pas la même *difficulté* : certains peuvent être résolus *facilement* (tel par exemple le problème PLUS COURT CHEMIN [Dik59]), tandis que d'autres nécessitent un travail plus conséquent pour trouver une solution si celle-ci existe (c'est le cas du problème COLORATION [Kar72]). Il est dès lors important de disposer d'un cadre formel permettant d'étudier et comparer la difficulté intrinsèque de ces problèmes.

Complexité classique. La « théorie de la complexité classique » est l'un des principaux cadres permettant d'étudier la « complexité » des problèmes théoriques. L'essor de cette théorie trouve sa motivation dans la volonté des scientifiques de comprendre et améliorer les besoins en ressources des algorithmes permettant de résoudre des problèmes.

Basée sur les « machines de TURING », l'un des modèles mathématiques représentant le mieux les ordinateurs que nous utilisons, la théorie de la complexité classique a pour but de classifier les problèmes dans différentes « classes de complexité », chacune définissant un niveau de complexité en fonction de la taille globale de l'instance. Citons en particulier la classe P, contenant les problèmes pouvant être résolus en temps polynomial sur une machine de TURING déterministe, la classe NP, contenant les problèmes dont la *validité* d'une solution peut être vérifiée en temps polynomial, la classe PSPACE, contenant les problèmes pouvant être résolus en espace polynomial, et la classe EXPTIME, contenant les problèmes pouvant être résolus en temps (*simplement*) exponentiel. Intuitivement, la classe P contient les problèmes que l'on peut résoudre *facilement* et *efficacement*, tandis que les autres classes citées contiennent les problèmes *difficiles*, ordonnées par difficulté croissante.

L'un des objectifs étant de comparer la complexité des différents problèmes, ces classes sont ordonnées, et nous savons notamment que :

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXPTIME}$$

Mais encore aujourd'hui, nous ne savons pas si ces inclusions sont strictes ou non. La grande question ouverte dans ce domaine, et plus généralement en informatique,

concerne l'inclusion des classes P et NP, et est traditionnellement formulée ainsi : « Est-ce que P = NP ? ». En d'autres termes, si l'on peut vérifier la validité d'une solution en temps polynomial (classe NP), peut-on également trouver une solution en temps polynomial (classe P) ?

Bien que cette question n'ait pas encore trouvé de réponse, elle suscite toujours beaucoup d'intérêt de la part des scientifiques, car de très nombreux problèmes fondamentaux s'avèrent être « NP-difficiles », et aucun algorithme polynomial n'existe pour les résoudre, sauf si P = NP. Puisqu'il est tout de même important de pouvoir les résoudre, les scientifiques introduisent et utilisent divers méthodes pour *attaquer* ces problèmes.

Complexité paramétrée. La théorie de la complexité classique est un bon outil pour comparer la complexité des problèmes théoriques. Toutefois, la *mesure* de cette complexité se fait toujours en fonction de la taille globale de l'instance, alors que la réelle difficulté du problème peut résider ailleurs. Cette restriction conduit à considérer tous les problèmes NP-complets comme étant de même complexité, quand bien même certains semblent plus *difficiles* que d'autres.

La « théorie de la complexité paramétrée » a été introduite pour étudier et comparer plus finement la complexité des problèmes théoriques, en considérant un *paramètre* plus spécifique aux instances. La classification de la complexité des problèmes s'effectue au travers de « classes de complexité paramétrée », ordonnées selon la hiérarchie W et en fonction du paramètre considéré.

Il est pertinent d'étudier un problème du point de vue paramétré dès lors que l'explosion combinatoire engendrée lors de la résolution du problème peut être confinée au paramètre considéré, c'est-à-dire que le problème peut être résolu en temps $\mathcal{O}(n^{f(k)})$, où f est une fonction calculable, k est le paramètre et n est la taille de l'instance. La classe XP regroupe l'ensemble de ces problèmes.

L'une des plus importantes classes de complexité paramétrée, si ce n'est la plus importante, est la classe FPT. Celle-ci contient les problèmes pouvant être résolus en temps $\mathcal{O}\big(f(k) \cdot poly(n)\big)$, où f est une fonction calculable, k est le paramètre et n est la taille de l'instance, c'est-à-dire que l'explosion combinatoire est totalement indépendante de la taille de l'instance.

Tous les problèmes n'appartiennent pas à la classe FPT. Toutefois, contrairement à la complexité classique, il est tout de même possible, dans de nombreux cas, de comparer et différencier la complexité de problèmes n'étant pas FPT, en montrant que certains sont plus *difficiles* que d'autres.

Pour illustrer cela, prenons le cas de deux problèmes fondamentaux en théorie des graphes, dont l'étude plus précise de la complexité a motivé l'introduction de la théorie de la complexité paramétrée : ENSEMBLE DOMINANT, et ENSEMBLE STABLE.

Le premier consiste à trouver un ensemble dominant dans un graphe $G = (V, E)$, c'est-à-dire un sous-ensemble de sommets $D \subseteq V$ tel que tout sommet du graphe est soit dans cet ensemble, soit a un voisin dans cet ensemble. Et plutôt que de considérer la taille globale du graphe pour *mesurer* la complexité de ce problème, nous allons considérer la cardinalité de l'ensemble dominant recherché.

k-ENSEMBLE DOMINANT

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Le graphe G admet-il un ensemble dominant $D \subseteq V$ de cardinalité au plus k ?

Le second problème consiste à trouver un ensemble stable dans un graphe $G = (V, E)$, c'est-à-dire un sous-ensemble de sommets $S \subseteq V$ tel qu'aucun sommet de S n'est voisin d'un autre sommet de ce sous-ensemble. Là encore, nous allons considérer non pas la taille globale du graphe, mais plutôt la cardinalité de l'ensemble stable recherché.

k-ENSEMBLE STABLE

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Le graphe G admet-il un ensemble stable $S \subseteq V$ de cardinalité au moins k ?

En théorie de la complexité classique, où seule la taille globale de l'instance (taille du graphe G et entier k) est prise en compte pour la *mesure* de la complexité, ces problèmes sont tous deux NP-complets. De ce fait, ils peuvent indifféremment se réduire l'un à l'autre, c'est-à-dire que l'on peut résoudre l'un de ces problèmes à l'aide de l'autre en effectuant une transformation polynomiale sur le graphe et l'entier k.

Si l'on considère maintenant l'entier k comme *mesure* de la complexité de ces problèmes, la situation est différente : il est possible de réduire k-ENSEMBLE STABLE vers k-ENSEMBLE DOMINANT en temps FPT, c'est-à-dire $\mathcal{O}\big(f(k) \cdot poly(n)\big)$ où n est la taille du graphe, mais pas l'inverse. Dans ce cas, les deux problèmes ne semblent pas être de même complexité.

Cette différente est formalisée par la hiérarchie W : le problème k-ENSEMBLE STABLE appartient à la classe $\mathrm{W}[1]$, tandis que le problème k-ENSEMBLE DOMINANT appartient à la classe $\mathrm{W}[2]$. Il existe en fait une infinité de classes $\mathrm{W}[t]$, ordonnées ainsi :

$$\mathsf{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots \subseteq \mathsf{XP}$$

À l'instar de la hiérarchie des classes de complexité classique, nous ne savons pas à l'heure actuelle si ces inclusions sont strictes ou non. En particulier, l'une des grandes questions ouvertes dans ce domaine est la suivante : « Est-ce que $\mathsf{FPT} = \mathrm{W}[1]$? ».

Remarquons enfin qu'il n'est pas toujours pertinent de considérer un problème du point de vue paramétré. Prenons le cas du problème COLORATION, que nous avons présenté dans la section précédente. Il serait naturel de paramétrer ce problème par le nombre maximum k de couleurs utilisées pour colorier les sommets du graphe. Or, ce problème est connu pour être NP-complet pour toute valeur fixée du paramètre $k \geq 3$ [Kar72]. Il n'est donc pas possible de résoudre ce problème même en temps $\mathcal{O}(n^{f(k)})$, puisque dans le cas où k est fixé, cela impliquerait que le problème est dans la classe P, et donc que $\mathsf{P} = \mathsf{NP}$. Les problèmes dont l'étude du point de vue paramétré n'est pas pertinente sont dits paraNP-complets.

Attaquer

De très nombreux problèmes en théorie des graphes sont NP-difficiles lorsque l'on considère des graphes quelconques. Bien que, sous l'hypothèse P $\neq$ NP, des algorithmes polynomiaux soient impossibles à obtenir pour résoudre les problèmes NP-difficiles [Wil10], il est tout de même important de pouvoir trouver des solutions à ces problèmes.

Il existe plusieurs approches pour attaquer un problème NP-difficile :

– obtenir un algorithme exact, en temps exponentiel, pour trouver une solution optimale ;
– décrire un algorithme d'approximation, souvent en temps polynomial, permettant de calculer une *bonne* solution (sans qu'il s'agisse d'une solution optimale) ;
– restreindre les instances à certaines classes de graphes, telles que les graphes planaires, r-réguliers, de largeur arborescente bornée,... ;
– *etc*.

Dans cette thèse, nous nous intéressons à quelques limites pouvant survenir lorsque l'on considère des instances de largeur de graphes bornée, et en particulier dans le cas de la largeur arborescente (*tree-width*).

La plupart de ces largeurs consiste à décomposer l'ensemble des sommets ou arêtes du graphe, de telle sorte que cette décomposition soit de petite largeur comparée à la taille globale du graphe. Par exemple, dans le cas de la largeur arborescente, la décomposition arborescente d'un graphe $G = (V, E)$ est un arbre T, où chaque noeud est associé à un sous-ensemble de sommets de G (appelé *sac*), chaque arête de G est représentée dans la décomposition par ses deux extrémités dans au moins un sac de T, et l'ensemble des sacs contenant un sommet de G forme un sous-arbre de T. Cette décomposition est de petite largeur si les sacs de l'arbre sont tous de petite cardinalité.

Pour de nombreux problèmes, NP-difficiles dans le cas de graphes quelconques, disposer d'une décomposition de petite largeur du graphe permet de résoudre efficacement le problème, en tirant avantage de cette décomposition. De ce fait, il est important de pouvoir construire une décomposition de petite largeur d'un graphe, mais aussi dans le cas contraire de pouvoir certifier qu'une telle décomposition n'existe pas.

Pour chacune des largeurs de graphes que nous considérerons dans cette thèse, un algorithme polynomial existe pour construire une décomposition de largeur fixée. Toutefois, lorsqu'une telle décomposition n'existe pas, ces algorithmes ne donnent aucun *certificat* démontrant l'impossibilité de construire la décomposition de petite largeur. Un tel *certificat* est pourtant important, puisqu'il permet de comprendre les raisons de la non existence de la décomposition. Nous présentons dans le CHAPITRE 2 un algorithme construisant explicitement un certificat pour différentes largeurs de graphes.

Considérer la largeur arborescente d'un graphe permet de résoudre efficacement la plupart des problèmes de graphe, lorsqu'ils sont paramétrés par cette largeur. Mais est-ce le cas de tous les problèmes ? Quelques problèmes sont d'ors-et-déjà connus pour ne pas être FPT lorsqu'ils sont paramétrés par la largeur arborescente du graphe donné en entrée. Nous donnons dans le CHAPITRE 3 une nouvelle collection de problèmes n'étant pas FPT lorsque paramétrés par la largeur arborescente.

Résumé de la thèse

Chapitre 1 : Préliminaires

Le premier chapitre est dédié à l'introduction des principales notions que nous utiliserons tout au long de cette thèse. Nous y définirons formellement les notions de complexité classique et paramétrée, la terminologie des graphes que nous considérons, la largeur arborescente formellement et vue comme un jeu de course-poursuite, et la logique monadique du second ordre ainsi qu'un résultat théorique de COURCELLE lié à la complexité paramétrée des problèmes paramétrés par la largeur arborescente. Pour illustrer certaines de ces notions, nous présenterons également quelques problèmes classiques de graphes parmi les plus connus.

Chapitre 2 : Construction d'obstructions aux décompositions

La plupart des algorithmes connus, pour déterminer si un graphe est de petite largeur ou non, le font en construisant (si possible) une décomposition du graphe. Lorsqu'une telle décomposition est obtenue, cela certifie que le graphe est effectivement de petite largeur. Mais si une telle décomposition n'existe pas, ces algorithmes se contentent généralement d'affirmer que le graphe est de grande largeur, sans fournir de certificat.

Pour la plupart des largeurs de graphes, des objets combinatoires, appelés « obstructions », existent pour certifier qu'un graphe est de grande largeur. Cependant, leur structure ne permet généralement pas de les construire directement à l'aide d'un algorithme. Partant de ce constat, de récents travaux se sont penchés sur ces obstructions, en étudiant sous des axes différents leurs structures et les propriétés qui les caractérisent.

Notre objectif principal dans ce chapitre est d'écrire un algorithme construisant explicitement et de manière exacte des obstructions pour diverses largeurs de graphes. Pour cela, nous reprenons et étendons la plupart des notions introduites par AMINI *et al.* [AMNT09] et LYAUDET *et al.* [LMT10].

Nous obtenons un algorithme paramétré générique, s'exécutant en temps XP, et construisant une obstruction optimale pour toute largeur de graphe respectant certaines propriétés. Pour illustrer ce résultat, nous en présentons l'application pour quelques unes des principales largeurs de graphes connues : la largeur arborescente (*tree-width*), la largeur de branches (*branch-width*), et la largeur de découpe (*carving-width*).

Chapitre 3 : Problèmes de domination

Comme dans le cas de la complexité classique, la largeur arborescente est souvent utilisée comme paramètre pour résoudre efficacement des problèmes *difficiles* dans le cas de graphes quelconques. En ce sens, j'ai étudié dans mon mémoire de Master [Cha08] le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, une généralisation des problèmes de domination où les ensembles d'entiers σ et ρ correspondent aux contraintes sur les sommets du graphe, et montré que ce problème peut être résolu en temps FPT lorsque paramétré par la largeur arborescente, si σ et ρ sont tous deux ultimement périodiques.

Cependant, ce problème est-il toujours FPT, quelques soient les ensembles σ et ρ considérés ? Nous montrons dans ce chapitre que pour certains cas d'ensembles σ et ρ, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT n'est plus FPT : il est W[1]-difficile. L'étude de la

complexité paramétrée de ce problème est tout de même pertinente, et nous montrons en ce sens que le problème appartient à la classe XP pour tout ensembles σ et ρ récursifs.

Chapitre 4 : Coloration additive

Le travail de ce chapitre est le fruit d'une collaboration effectuée dans le cadre d'un projet franco-chilien, et s'intéresse à la complexité d'un nouveau problème de coloration.

Ce nouveau problème de coloration, appelé COLORATION ADDITIVE, combine théorie des graphes et théorie des nombres : les couleurs associées aux sommets du graphe, qui de manière équivalentes correspondent à des entiers, doivent être telles que pour trois sommets formant un chemin, leurs couleurs ne créent pas une 3-progression arithmétique. En d'autres termes, si la valeur associée à une arête du graphe correspond à la différence (en valeur absolue) entre les couleurs de ses deux sommets incidents, alors les valeurs associées à deux arêtes adjacentes doivent être distinctes.

Nous montrons qu'à l'instar d'autres problèmes de coloration connus, ce nouveau problème est NP-complet, même lorsque le nombre $k \geq 3$ de couleurs est fixé, et ce même pour les graphes de degré au plus k. De ce fait, il est paraNP-complet, et il n'est donc pas pertinent de l'étudier du point de vue paramétré.

Toutefois, de très nombreux problèmes peuvent être résolus efficacement lorsqu'ils sont paramétrés par la largeur arborescente. Il est donc naturel d'étudier ce cas pour le problème COLORATION ADDITIVE. En ce sens, nous montrons que ce problème peut être résolu en temps polynomial sur les arbres, qui sont les graphes de largeur arborescente égale à 1. Toutefois, étendre ce résultat même aux graphes de largeur arborescente au plus 2 semble déjà difficile, et pourra faire l'objet de travaux futurs.

Conclusion

Enfin, dans la conclusion de cette thèse, nous reprendrons les principaux résultats présentés dans ce manuscrit, et discuterons de perspectives liées aux limites des décompositions de graphes et aux différents travaux effectués.

Préliminaires 1

Sommaire

Avant de présenter tout résultat, il convient de rappeler les principales définitions et notions usuelles relatives à la complexité, aux graphes, à la logique, et quelques autres notions pouvant nous être utiles dans la suite de cette thèse. En particulier, nous introduisons la notion de largeur arborescente, fil rouge de nos travaux et résultats. Ce chapitre est également l'occasion d'introduire le vocabulaire et les notations que nous utiliserons par la suite.

1.1 Complexité

1.1.1 Algorithme et complexité

Algorithme. Un *algorithme* est un *processus de calcul* bien spécifié qui, à partir d'une ou plusieurs données fournies en entrée, effectue un certain nombre d'opérations déterminées, puis retourne une ou plusieurs nouvelles données. La manière dont chacune de ces opérations est représentée et effectuée dépendra du modèle de calcul considéré.

La notion de *processus de calcul* existe informellement depuis des centaines d'années. Par exemple, le produit de deux entiers est un processus de calcul.

Durant la première moitié du XX^{e} siècle, cette notion a reçu une attention particulière dans le but d'être formalisée, notamment au travers des travaux de TURING, GÖDEL, CHURCH, KLEENE, ou encore POST (voir [Odi89]). Ces travaux ont introduit divers *modèles de calcul*, sur lesquels la notion de processus de calcul a été définie. Plus tard, il a été montré que tous ces modèles sont en fait de même expressivité, c'est-à-dire qu'un processus de calcul peut être soit modélisé par tous ces modèles, soit par aucun d'entre eux.

Complexité. Suite à l'introduction de ces différents modèles de calcul, deux notions fondamentales liées à ces modèles se sont montrées d'importance : la *calculabilité* et la *complexité*.

La calculabilité concerne l'existence et la possibilité d'exprimer des algorithmes dans le modèle considéré pour résoudre certains problèmes, tandis que la complexité concerne les ressources nécessaires dans le modèle considéré pour *exécuter* un algorithme (principalement temps et espace mémoire) dès lors que celui-ci peut être exprimé dans ce modèle.

Dans cette thèse, nous nous focalisons sur la notion de complexité, liée aux graphes et en rapport avec leur *largeur arborescente* (notion définie dans la SECTION 1.3). Le lecteur souhaitant approfondir la notion de calculabilité pourra se référer aux divers ouvrages de référence, notamment « *Classical recursion theory* » de ODIFREDDI [Odi89], « *Models of computation* » de SAVAGE [Sav98], ou encore « *Introduction to the theory of computation* » de SIPSER [Sip06].

La *complexité d'un algorithme* est en quelque sorte une *mesure* du coût de celui-ci. L'étude de cette complexité se focalise essentiellement sur deux ressources liées au modèle de calcul considéré : le temps d'exécution, exprimé en fonction du nombre d'opérations/instructions utilisées sur ce modèle lors de l'exécution de l'algorithme, et l'espace mémoire, dont l'expression dépend du modèle. La complexité d'un algorithme dépend donc fortement du modèle considéré.

Dans le cadre de cette thèse, nous nous intéressons principalement au temps d'exécution *au pire des cas* de nos algorithmes, c'est-à-dire le plus long temps d'exécution que peuvent prendre ces algorithmes sur toutes entrées de même taille.

Notations de complexité. Les notations $\mathcal{O}$, Ω et Θ, popularisées dans le domaine informatique par KNUTH [Knu76], sont généralement utilisées lors de l'évaluation de la complexité d'un algorithme, qu'il s'agisse de son temps d'exécution ou de l'espace mémoire qu'il utilise. Nous allons présenter ces notations en terme de temps d'exécution, leur formulation en terme d'espace mémoire étant similaire.

Supposons que le temps d'exécution d'un algorithme donné soit désigné par $T(n)$, où n est la taille globale des données fournies en entrée à l'algorithme. Les trois notations $\mathcal{O}$, Ω et Θ ont pour rôle de donner une borne asymptotique de ce temps d'exécution.

Ainsi, $T(n) = \mathcal{O}\big(f(n)\big)$, pour une certaine fonction f croissante, s'il existe deux constantes $c, n_0 \in \mathbb{N}^*$ telles que, pour tout $n \geq n_0$, $T(n) \leq c \cdot f(n)$. En d'autres termes, le temps d'exécution $T(n)$ de l'algorithme est borné supérieurement par la fonction f, à un facteur multiplicatif constant près.

De même, $T(n) = \Omega\big(f(n)\big)$ s'il existe deux constantes $c, n_0 \in \mathbb{N}^*$ telles que, pour tout $n \geq n_0$, $T(n) \geq c \cdot f(n)$, c'est-à-dire que le temps d'exécution $T(n)$ de l'algorithme est borné inférieurement par la fonction f, à un facteur multiplicatif constant près.

Enfin, $T(n) = \Theta\big(f(n)\big)$ si $T(n) = \mathcal{O}\big(f(n)\big)$ et $T(n) = \Omega\big(f(n)\big)$.

Dans la suite de cette thèse, nous utiliserons presque exclusivement la notation $\mathcal{O}$, car il nous importe principalement d'obtenir une borne supérieure de la complexité.

L'objectif de ces notations est de donner une estimation compréhensible de la complexité d'un algorithme. C'est pourquoi WOEGINGER [Woe03] a récemment introduit la notation $\mathcal{O}^*$. Lorsque la fonction f est composée d'une partie exponentielle, la notation $\mathcal{O}^*$ autorise à masquer les facteurs polynomiaux de f afin de ne conserver que la partie exponentielle.

Ainsi, une complexité de la forme $\mathcal{O}\big(exp(n) \cdot poly(n)\big)$ pourra s'écrire $\mathcal{O}^*\big(exp(n)\big)$. Par exemple, si $f(n) = n + n^3 \cdot 1.41^n$ et $T(n) = \mathcal{O}\big(f(n)\big)$, on pourra écrire $T(n) = \mathcal{O}^*(1.41^n)$ au lieu de $T(n) = \mathcal{O}\big(n + n^3 \cdot 1.41^n\big)$. Cette simplification se justifie par le fait que pour une valeur de n suffisamment grande, la fonction f est comprise entre 1.41^n et 1.42^n, et donc $f(n) = \mathcal{O}(1.42^n)$.

Complexité des problèmes. Un *problème* pose une *question* qui, à partir d'une ou plusieurs données fournies en entrée, aussi appelées *instance* et spécifiant entièrement l'entrée du problème, demande s'il est possible d'obtenir une ou plusieurs nouvelles données dont les spécifications sont bien définies, également appelées *solution* du problème. Un tel problème peut être posé sous différentes formes :

- Problème de décision : décider de l'existence (ou non) d'une solution telle que spécifiée, sans nécessairement la construire.
- Problème de construction : construire explicitement une solution au problème, si une telle solution existe.
- Problème d'optimisation : construire explicitement une solution optimale au problème, l'optimalité étant souvent mesurée en terme de cardinalité.

Un algorithme est alors la description d'un processus de calcul permettant de résoudre un problème donné.

Pour chaque problème, il existe une infinité d'algorithmes permettant de le résoudre, mais tous n'ont pas la même complexité (en temps ou en espace mémoire). La *complexité d'un problème* correspond à la complexité du *meilleur* algorithme pour le résoudre, selon le modèle de calcul considéré, c'est-à-dire à la complexité en temps de l'algorithme le plus rapide, ou la complexité en espace mémoire de l'algorithme le moins *gourmand*.

L'étude de la complexité d'un problème consiste, soit à montrer l'existence d'un algorithme de complexité donnée, soit à démontrer que tout algorithme pour résoudre ce problème nécessite au minimum une certaine complexité.

1.1.2 Complexité classique

Nous allons présenter quelques notions importantes relatives à la complexité classique de calcul. Dans le cadre de notre travail de thèse, nous nous focalisons sur la complexité en terme de temps d'exécution, et allons donc nous concentrer sur cet aspect de la complexité des algorithmes et problèmes que nous considérerons.

Le lecteur souhaitant approfondir la notion de complexité classique pourra se référer aux divers ouvrages de référence, notamment « *Computers and intractability: a guide to the theory of* NP*-completeness* » de GAREY et JOHNSON [GJ79], « *Computational complexity* » de PAPADIMITRIOU [Pap94], « *Introduction to automata theory, languages, and computation* » de HOPCROFT *et al.* [HMU01], « *Introduction to the theory of computation* » de SIPSER [Sip06], ou encore « *Computational complexity: a modern approach* » de ARORA et BARAK [AB09].

Dès lors qu'un problème est décidable (et donc calculable) dans un certain modèle donné, il existe au moins un algorithme permettant de le résoudre : il s'agit de l'algorithme dit de *force brute*, qui effectue une recherche exhaustive d'une solution au problème. On dispose dès lors d'une estimation, parfois grossière, de la complexité du problème.

Historiquement, l'un des éléments déclencheurs de l'étude plus fine de la complexité des problèmes fut une lettre envoyée par GÖDEL à VON NEUMANN, et dont voici un court extrait (la version originale étant en allemand, nous en donnons ici une traduction libre ; voir également [Har89]) :

> « Il serait intéressant de savoir à quel point, en général, le nombre d'étapes nécessaires pour la résolution de problèmes finis de combinatoire peut être réduit par rapport à une simple recherche exhaustive. »
>
> GÖDEL à VON NEUMANN, le 20 mars 1956.

Dès lors, il est devenu d'importance de pouvoir déterminer la complexité de chaque problème, en terme des ressources nécessaires pour le résoudre (temps d'exécution et espace mémoire utilisé). De très récents travaux se penchent à nouveau sur les aspects théoriques de cette question, notamment ceux de WILLIAMS [Wil10] montrant qu'une réduction même faible de ce nombre d'étapes semble improbable.

La notion de complexité d'un problème n'a de réel sens que lorsqu'elle est liée à un modèle de calcul déterminé.

De nombreux modèles classiques de calcul existent. Nous pouvons notamment citer la « machine de TURING » introduite par TURING, le « λ-calcul » introduit par CHURCH, les « fonctions récursives » utilisées notamment par GÖDEL et KLEENE, ou encore la

« machine de POST » introduite par POST (voir [Odi89] pour une introduction à de nombreux problèmes de calcul).

Parmi tous ces modèles théoriques, celui le plus couramment utilisé est la machine de TURING, car les opérations *unitaires* sont simples et relativement intuitives, c'est pourquoi les classes de complexité les plus connues sont définies à partir de ce modèle.

Machine de TURING. Nous allons présenter succinctement la définition d'une machine de TURING. Le lecteur souhaitant approfondir cette notion pourra se référer aux nombreux ouvrages de référence que nous avons cités précédemment.

Une *machine de* TURING *déterministe* à une bande est composée d'une *bande* formée par une séquence doublement infinie de *cases*, d'une *tête de lecture-écriture* se déplaçant sur cette bande, et d'une *table finie de transitions* contrôlant le déplacement et l'écriture de la tête. Une telle machine est schématisée sur la FIGURE 1.1.

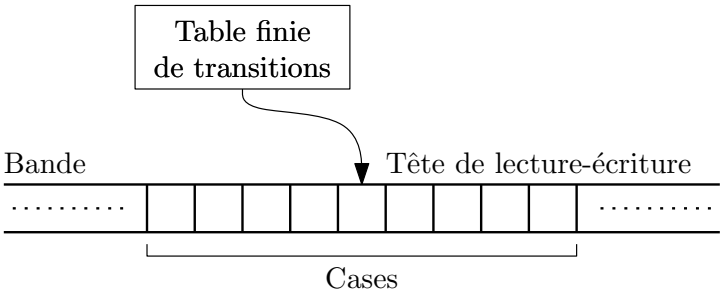


FIGURE 1.1 — Schéma représentant une machine de TURING déterministe à une bande.

D'un point de vue théorique, il n'est pas nécessaire d'ajouter des têtes de lecture-écriture ou des bandes supplémentaires ; il a en effet été montré que toute machine disposant de plusieurs têtes de lecture-écriture ou plusieurs bandes n'est pas plus expressive, et qu'elle peut être simulée dans le même ordre de complexité par la machine que nous venons de définir précédemment (voir par exemple [HMU01]).

Un *programme* sur une machine de TURING déterministe à une bande, qui correspond à un algorithme exécuté sur ce modèle de calcul pour résoudre un problème de décision, se compose des informations suivantes :

- un ensemble fini Γ de *symboles de bande*, contenant notamment un sous-ensemble $\Sigma \subseteq \Gamma$ de *symboles d'entrée*, et un *symbole blanc* distinct $B \in \Gamma \setminus \Sigma$;
- un ensemble fini Q d'*états*, contenant notamment un *état initial* $q_0 \in Q$, et deux *états acceptants* $q_Y, q_N \in Q$;
- une *fonction de transition* $\delta : \big(Q \setminus \{q_Y, q_N\}\big) \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

À un instant donné de l'exécution, en fonction de l'état dans lequel la tête se trouve et du symbole de bande présent sur la case actuellement lue par la tête, la fonction de transition définit le nouveau symbole à écrire sur cette case, le déplacement que la tête doit effectuer sur la bande (L pour un déplacement d'une case vers la gauche, R pour un déplacement d'une case vers la droite), et le nouvel état de la tête de lecture-écriture.

L'état q_0 correspond à l'état dans lequel se trouve la tête au début de l'exécution du programme. Le programme s'arrête lorsque la tête arrive dans l'un des deux états q_Y ou q_N, correspondant respectivement à une réponse positive ou négative du programme. À noter qu'un programme peut ne jamais s'arrêter.

Une *machine de* TURING *non-déterministe* à une bande est composée des mêmes éléments que ceux d'une machine déterministe. La différence tient dans la définition d'un programme pour une telle machine : dans un *programme* sur une machine de TURING non-déterministe à une bande, la fonction de transition est remplacée par une *relation de transition* $\delta : \big(Q \setminus \{q_Y, q_N\}\big) \times \Gamma \to Q \times \Gamma \times \{L, R\}$. La différence notable par rapport à une fonction de transition, est qu'il existe potentiellement plusieurs transitions possibles à partir d'un même état et d'un même symbole lu sur la bande. Lors de l'exécution du programme, le choix d'une transition est effectué de manière non-déterministe par la machine.

Classes de complexité P **et** NP**.** Deux des plus importantes classes de complexité sont la classe P et la classe NP. La classe de complexité P (pour *deterministic Polynomial*) contient tous les problèmes pouvant être résolus en *temps polynomial*, par un programme adéquat, sur une machine de TURING *déterministe*. La classe de complexité NP (pour *Non-deterministic Polynomial*) contient quant à elle tous les problèmes pouvant être résolus en *temps polynomial*, par un programme adéquat, sur une machine de TURING *non-déterministe*.

Ces deux classes sont en fait définies sur des problèmes de décision uniquement : la question posée est simplement de déterminer si une solution au problème existe, sans nécessairement la construire. Bien entendu, la version d'optimisation de ces problèmes est au moins aussi *difficile* que la version de décision, puisque parvenir à construire une solution au problème signifie qu'une telle solution existe.

En pratique, un problème d'optimisation peut souvent être transformé en un problème de décision, en ajoutant à l'entrée un entier $K \in \mathbb{N}$, et en demandant s'il existe une solution dont la *taille* est au plus (ou au moins) égale à K.

Il est aisé de montrer que $\mathsf{P} \subseteq \mathsf{NP}$: en effet, un programme s'exécutant en temps polynomial sur une machine de TURING déterministe peut être également exécuté dans le même temps sur une machine non-déterministe, sans utilisation de l'aspect « non-déterministe » de la machine (la relation de transition étant alors une fonction de transition).

Mais est-il vrai que $\mathsf{NP} \subseteq \mathsf{P}$ et donc $\mathsf{P} = \mathsf{NP}$, ou bien $\mathsf{P} \subsetneq \mathsf{NP}$ et donc $\mathsf{P} \neq \mathsf{NP}$? Il s'agit en fait de l'un des problèmes ouverts les plus importants et fondamentaux en informatique, « $\mathsf{P} = \mathsf{NP}$? », dont la résolution serait une avancée majeure en informatique fondamentale, et aurait des conséquences très fortes sur de très nombreux domaines scientifiques.

Il existe d'autres classes importantes de complexité, qui ne concernent toutefois pas notre travail de thèse. Nous pouvons notamment citer la classe PSPACE, contenant tous les problèmes pouvant être résolus par un programme utilisant un espace mémoire polynomial sur une machine de TURING déterministe, la classe EXPTIME, contenant tous les problèmes pouvant être résolus en temps (simplement) exponentiel sur une machine de

TURING déterministe, et leurs équivalents sur une machine de TURING non-déterministe (NPSPACE et NEXPTIME). Pour plus d'informations sur les différentes classes de complexité existantes, le lecteur pourra se référer aux divers ouvrages de référence que nous avons cité au début de cette section.

NP-complétude. La classe de complexité NP contient une sous-classe importante de problèmes : les problèmes « NP-complets ». Une part conséquente de l'étude du problème « P = NP ? », et l'une des raisons pour lesquelles de nombreux chercheurs penchent pour une inegalité stricte entre ces deux classes, est l'existence de ces problèmes NP-complets.

DÉFINITION 1.1.1 (réduction polynomiale (réduction de KARP))
Un problème Π_1 *peut être* réduit en temps polynomial *(ou* réduit polynomialement*) à un problème* Π_2*, s'il existe une fonction de réduction* R *d'une instance du problème* Π_1 *vers une instance du problème* Π_2*, calculable en temps polynomial, et telle que le problème* Π_1 *admet une solution à partir de l'instance* I_1 *si et seulement si le problème* Π_2 *admet une solution à partir de l'instance* $I_2 = R(I_1)$*. Cette réduction polynomiale est notée* $\Pi_1 \preceq_p \Pi_2$*.*

La fonction de réduction R effectue donc une transformation d'une instance du problème Π_1 vers une instance du problème Π_2, en temps polynomial, tout en s'assurant de l'équivalence des solutions. Nous utilisons ici la notion de réduction polynomiale telle que définie par KARP [Kar72].

Nous pouvons maintenant définir la notion de NP-complétude d'un problème.

DÉFINITION 1.1.2 (NP-complet)
Un problème Π *est dit* NP-complet *si et seulement si :*

1. $\Pi \in \mathsf{NP}$*;*
2. *pour tout problème* $\Pi' \in \mathsf{NP}$*,* $\Pi' \preceq_p \Pi$*.*

Le problème Π est dit NP-*difficile* lorsque la seconde propriété est vraie, sans nécessairement que Π appartienne à la classe NP. Intuitivement, cela signifie que le problème Π est au moins aussi *difficile* que tous les problèmes Π' qui sont dans la classe NP. La notion de réduction polynomiale permet donc de *comparer* la complexité des problèmes.

COOK [Coo71] et LEVIN [Lev73, Tra84] ont indépendamment donné la première démonstration de NP-complétude d'un problème, connu sous le nom de SATISFIABILITÉ, ou plus simplement SAT :

SAT

Entrée : Une formule booléenne $\varphi(X)$ sur un ensemble de variables booléennes X.
Question : Existe-t-il une affectation de valeurs aux variables de X de telle sorte que la formule booléenne $\varphi(X)$ soit satisfaite ?

Rapidement, de nombreux autres problèmes ont été montrés NP-complets, notamment par KARP [Kar72] et par GAREY et JOHNSON [GJ79].

En fait, lorsque l'on souhaite montrer qu'un problème Π est NP-complet, il n'est pas nécessaire de réduire tous les problèmes de la classe NP au problème Π. En utilisant

la notion de réduction polynomiale, il suffit en fait de considérer un autre problème Π' connu comme étant NP-complet, puis de montrer que $\Pi' \preceq_p \Pi$. C'est cette méthode qui est généralement utilisée pour montrer qu'un problème Π est NP-complet, et qui a facilité la preuve de NP-complétude de nombreux problèmes à partir de la NP-complétude du problème SAT.

1.1.3 Complexité paramétrée

Notre travail de thèse s'est principalement articulé autour de la notion de complexité paramétrée. Cette nouvelle notion de complexité, introduite récemment par DOWNEY et FELLOWS [DF95a, DF95b], permet d'étudier et comparer plus précisément la complexité des problèmes, en particulier des problèmes NP-difficiles. À l'instar de la théorie de la complexité classique, et notamment de l'étude des classes P et NP, la complexité paramétrée s'appuie sur les « machines de TURING ».

Le lecteur souhaitant approfondir la notion de complexité paramétrée pourra se référer aux quelques ouvrages de référence d'ors-et-déjà existants, notamment « *Parameterized complexity* » de DOWNEY et FELLOWS [DF99], « *Introduction to fixed-parameter algorithms* » de NIEDERMEIER [Nie06], et « *Parameterized complexity theory* » de FLUM et GROHE [FG06].

En complexité, il est intéressant de déterminer l'appartenance d'un problème à l'une ou l'autre des différentes classes de complexité classique existantes. Toutefois, la mesure de la complexité classique se fait toujours en fonction de la taille de l'instance fournie en entrée, et ne tient donc pas compte de la possible structure particulière de cette instance.

De ce fait, de très nombreux problèmes s'avèrent être NP-complets, et la théorie de la complexité classique considère que tous ces problèmes sont de même *difficulté*.

Partant de ce constat, DOWNEY et FELLOWS ont introduit la notion de complexité paramétrée.

En complexité paramétrée, la complexité d'un problème n'est pas mesurée seulement en fonction de la taille de l'instance fournie en entrée, mais aussi (et surtout) en fonction d'un « paramètre » dépendant arbitrairement de l'instance, mais qui généralement supposé petit par rapport à la taille de cette instance. L'idée est notamment de pouvoir mesurer cette complexité de manière plus fine, en particulier lorsque l'on sait que le paramètre considéré est de petite taille par rapport à la taille globale de l'instance.

Un problème est dit *paramétré* lorsqu'un paramètre dépendant arbitrairement de l'instance, généralement considéré petit par rapport à la taille globale de l'instance, est pris en considération pour la mesure de sa complexité. Par exemple, le problème SAT peut être paramétré par le nombre maximum de variables que l'on affecte à vraie :

k-WSAT

Entrée : Une formule booléenne $\varphi(X)$ sur un ensemble de variables booléennes X, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Existe-t-il une affectation de valeurs aux variables de X, avec au plus k variables affectées à vraie, de telle sorte que la formule booléenne $\varphi(X)$ soit satisfaite ?

La notion de complexité paramétrée a notamment été motivée par l'existence des deux problèmes k-ENSEMBLE DOMINANT et k-ENSEMBLE STABLE qui, lorsque paramétrés par la cardinalité k de la solution recherchée, semblent ne pas être de même complexité. En effet, dans ce cas, le problème k-ENSEMBLE STABLE peut être réduit au problème k-ENSEMBLE DOMINANT, mais pas l'inverse.

La notion centrale de la théorie de la complexité paramétrée est la « tractabilité à paramètre fixe » (*fixed-parameter tractability*) : l'explosion combinatoire du problème est confinée à un paramètre sur l'instance significativement plus petit que la taille globale de cette instance. La FIGURE 1.2 schématise cette idée, où l'explosion combinatoire diffère selon que le paramètre considéré soit global (cas de la complexité classique) ou plus spécifique (cas de la complexité paramétrée).

(a) Complexité classique (b) Complexité paramétrée

FIGURE 1.2 — Schématisation de l'explosion combinatoire d'un problème selon le paramètre choisi. En complexité classique (figure de gauche), le paramètre est global et correspond généralement à la taille n de l'instance ; l'explosion combinatoire est globale. En complexité paramétrée (figure de droite), le paramètre est plus spécifique ; l'explosion combinatoire est alors confinée et mieux contrôlée.

Il existe de nombreuses classes de complexité paramétrée. Nous n'allons effectuer ici qu'un léger survol de quelques unes des principales classes de complexité paramétrée. Le lecteur souhaitant approfondir les différentes classes de complexité paramétrée, ainsi que les nombreuses propriétés qui les caractérisent, pourra se référer aux différents ouvrages de référence cités précédemment.

Classes de complexité paramétrée. L'explosion combinatoire confinée au paramètre peut varier, selon le paramètre considéré ou le problème étudié. De ce fait, et à l'instar de la complexité classique, il existe de nombreuses classes de complexité paramétrée, hiérarchisées en fonction de leurs inclusions respectives.

Rappelons que l'étude en complexité paramétrée n'est réellement pertinente que lorsque le paramètre considéré est significativement plus petit que la taille globale de l'instance, et donc que le paramètre considéré en complexité classique.

La classe FPT (pour *fixed-parameter tractable*), centrale en théorie de la complexité paramétrée, contient l'ensemble des problèmes pouvant être résolus par un algorithme dont la complexité en temps est de la forme :

$$\mathcal{O}\big(f(k) \cdot poly(n)\big)$$

où k est le paramètre du problème, f est une fonction calculable quelconque, et n est la taille globale de l'instance.

Par exemple, un problème pouvant être résolu en temps $\mathcal{O}(3^{2^k} \cdot n^4)$, où k est le paramètre du problème, appartient à la classe FPT.

La classe XP contient l'ensemble des problèmes pouvant être résolus par un algorithme dont la complexité en temps est de la forme :

$$\mathcal{O}\big(poly(n)^{f(k)}\big)$$

où k est le paramètre du problème, f est une fonction calculable quelconque, et n est la taille globale de l'instance. Intuitivement, la classe XP en complexité paramétrée joue le rôle de la classe EXPTIME en complexité classique.

Tout problème appartenant à la classe FPT est de surcroît membre de la classe XP : puisqu'il appartient à la classe FPT, il existe un algorithme permettant de résoudre ce problème en temps $\mathcal{O}\big(f(k) \cdot poly(n)\big) \leq \mathcal{O}\big(poly(n)^{f(k)}\big)$. Nous avons donc :

$$\mathsf{FPT} \subseteq \mathsf{XP}$$

Comme nous l'affirmions, la théorie de la complexité paramétrée permet de mesurer plus finement la complexité de problèmes NP-difficiles. Il existe en effet une infinité de classes de complexité paramétrée entre ces deux classes fondamentales : il s'agit de la hiérarchie W, composée des classes W[1], W[2], W[3],... La hiérarchie W, incluant les classes FPT et XP, se présente ainsi :

$$\mathsf{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots \subseteq \mathsf{XP}$$

La définition de ces différentes classes, nécessitant notamment des notions relatives aux machines de TURING alternantes [CKS81] et à la logique du second ordre, sort quelque peu du cadre de cette thèse. C'est pourquoi nous invitons le lecteur souhaitant approfondir ces notions à se référer aux quelques ouvrages de référence cités au début de cette SECTION [DF99, Nie06, FG06].

Il existe une classe de complexité paramétrée un peu particulière : la classe paraNP, introduite récemment par FLUM et GROHE [FG03]. Tout problème qui, en complexité classique, est NP-difficile lorsque le paramètre est fixé (c'est-à-dire considéré comme étant une constante), est paraNP-complet en complexité paramétrée. Pour un problème, le fait d'être paraNP-complet signifie en fait qu'il n'est pas pertinent d'en étudier la complexité du point de vue paramétré, puisque la complexité en temps d'un algorithme permettant de le résoudre sera de la forme $\mathcal{O}\big(c^{f(n)}\big)$ (sauf si $\mathsf{P} = \mathsf{NP}$), où c est une constante pouvant contenir le paramètre fixé et f une fonction quelconque.

Par exemple, le problème COLORATION (voir SECTION 1.5), qui consiste à assigner des couleurs différentes aux sommets adjacents d'un graphe telles qu'au plus $K \in \mathbb{N}^*$ couleurs différentes soient utilisées pour tout le graphe, est NP-complet pour tout entier $K \geq 3$ fixé. De ce fait, le problème k-COLORATION, version paramétrée du problème demandant d'utiliser au plus k couleurs où k est le paramètre, est paraNP-complet.

Notons que le problème « $\mathsf{P} = \mathsf{NP}$? » en théorie de la complexité classique, fondamental en informatique, a son équivalent en théorie de la complexité paramétrée :

THÉORÈME (voir [FG06])
$\mathsf{P} = \mathsf{NP}$ *si et seulement si* $\mathsf{FPT} = \mathsf{paraNP}$.

L'état exact de chacune des inclusions de la hiérarchie W n'est actuellement pas connu : nous ne savons pas si ces inclusions sont strictes ou non, et en particulier si $\mathsf{FPT} = \mathrm{W}[1]$ ou $\mathsf{FPT} \subsetneq \mathrm{W}[1]$. Toutefois, par un résultat de DOWNEY et FELLOWS [DF99], nous savons que $\mathsf{FPT} \subsetneq \mathsf{XP}$, conséquence du « théorème de hiérarchie polynomiale sur machine de TURING déterministe » [HS65, HS66].

Réduction FPT et complétude. En théorie de la complexité paramétrée, il existe une notion similaire à celle de la « réduction polynomiale » en complexité classique (voir SECTION 1.1.2) : la « réduction FPT ».

DÉFINITION 1.1.3 (réduction FPT)
Un problème paramétré $\Pi = (I, k)$ *admet une* réduction FPT *vers un problème paramétré* $\Pi' = (I', k')$*, s'il existe une fonction de réduction* R *d'une instance* I *de* Π *vers une instance* I' *de* Π' *telle que, pour toute instance* X *du problème* Π *:*
- X *est une instance positive du problème* Π *si et seulement si* $R(X)$ *est une instance positive du problème* Π' *;*
- *la fonction* R *peut être calculée en temps* FPT*, c'est-à-dire qu'il existe une fonction* f *calculable et un polynôme* $p(|X|)$ *tels que* $R(X) = \mathcal{O}\big(f(k) \cdot p(|X|)\big)$ *;*
- *il existe une fonction* $g : \mathbb{N} \to \mathbb{N}$ *calculable telle que* $k' \leq g(k)$.

Cette réduction FPT *est notée* $\Pi \preceq_{fpt} \Pi'$.

En d'autres termes, le problème paramétré Π peut être transformé en temps polynomial en le problème Π', de sorte que les solutions de ce nouveau problème correspondent exactement, après transformation, aux solutions du problème initial.

Étant donnée C une classe de complexité paramétrée (FPT, $\mathrm{W}[1]$, XP, ...), les notions de C-difficulté et C-complétude d'un problème paramétré Π sont définies de manière similaire à celles en complexité classique, en utilisant cette fois la notion de réduction FPT :
- Π est *C-difficile* (par réduction FPT) si tout problème appartenant à la classe C admet une réduction FPT vers Π ;
- Π est *C-complet* (par réduction FPT) si $\Pi \in C$ et Π est C-difficile.

Chaque classe de complexité paramétrée $\mathrm{W}[1]$, $\mathrm{W}[2]$, *etc.*, admet un problème C-complet, défini par une variante du problème WSAT que nous ne détaillons pas ici (voir notamment [FG06]). À l'instar de la complexité classique, la C-complétude d'un nouveau problème peut être démontrée en construisant une réduction FPT depuis un autre problème dont la C-complétude est déjà connue.

Depuis l'introduction récente de la théorie de la complexité paramétrée, de nombreux problèmes paramétrés ont été étudiés, soit pour déterminer à quelle classe de complexité paramétrée ils appartiennent (principalement FPT, $\mathrm{W}[1]$ et $\mathrm{W}[2]$), soit pour écrire de nouveaux algorithmes efficaces permettant de les résoudre. Là encore, nous invitons le lecteur souhaitant approfondir ce domaine à se référer aux quelques ouvrages de référence cités au début de cette SECTION [DF99, Nie06, FG06], ainsi qu'aux très nombreux articles scientifiques liés à ce domaine.

1.2 Terminologie des graphes

1.2.1 Définitions et notations usuelles

Nous allons tout d'abord définir les principales définitions et notations relatives aux graphes, et dont nous aurons besoin dans la suite de cette thèse. Pour plus de détails sur ces notions, le lecteur pourra se référer à des ouvrages de référence sur la théorie des graphes, notamment « Graphes et Hypergraphes » de BERGE [Ber70] et « Graph Theory » de DIESTEL [Die10].

Graphe non orienté. Un *graphe non orienté* G est la donnée d'un couple (V, E), où V est un ensemble de *sommets*, et E un ensemble d'*arêtes* formées de paires de sommets distincts. S'il est besoin de lever une ambiguïté, on les note $V(G)$ et $E(G)$. On définit la taille de ces deux ensembles par $n(G) = |V(G)|$ et $m(G) = |E(G)|$, que l'on écrit n et m s'il n'y a pas d'ambiguïté. Nous ne considérons dans cette thèse que des graphes finis.

Une arête entre les sommets u et v est notée $\{u, v\} \in E(G)$. Deux sommets $u, v \in V(G)$ sont dits *adjacents*, ou *voisins*, s'il existe une arête $e = \{u, v\} \in E(G)$. Dans ce cas, chacun de ces deux sommets u et v est dit *incident* à l'arête e. Deux arêtes $e, f \in E(G)$ sont dites *adjacentes* si elles partagent un sommet incident commun v, c'est-à-dire $e = \{u, v\}$ et $f = \{v, w\}$ avec $u, w \in V(G)$.

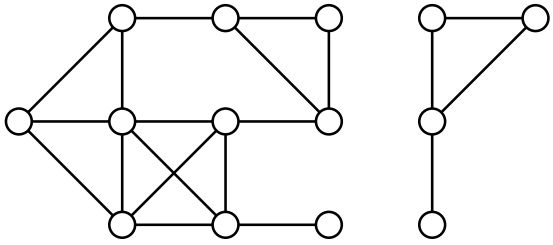


FIGURE 1.3 — Exemple d'un graphe G quelconque.

Le *voisinage ouvert* d'un sommet $v \in V(G)$, noté $N_G(v)$, ou simplement $N(v)$ lorsqu'il n'y a pas d'ambiguïté, est l'ensemble des sommets adjacents à v, c'est-à-dire $N(v) = \big\{u \mid \{u, v\} \in E(G)\big\}$. Le *voisinage fermé* de v, noté $N[v]$, inclut également le sommet lui-même, c'est-à-dire $N[v] = N(v) \cup \{v\}$. Le *voisinage ouvert d'un sous-ensemble* de sommets $S \subseteq V(G)$, noté $N(S)$, est l'ensemble des sommets de $V(G)$ ayant un voisin dans S, c'est-à-dire $N(S) = \big\{u \in V(G) \mid \exists v \in S : \{u, v\} \in E(G)\big\}$. Le *voisinage fermé* de S est alors $N[S] = N(S) \cup S$.

Le *degré* d'un sommet $v \in V(G)$, noté $d_G(v)$, ou simplement $d(v)$ s'il n'y a pas d'ambiguïté, est égal au nombre de voisins qu'il a dans G, c'est-à-dire $d_G(v) = |N(v)|$. Le *degré maximum* de G, noté $\Delta(G)$, est alors défini par : $\Delta(G) = \max_{v \in V(G)}\{d_G(v)\}$.

Un sommet est dit *isolé* s'il est de degré 0, et n'est donc voisin avec aucun autre sommet du graphe. Une arête est dite *isolée* si ses deux sommets incidents sont de degré 1, et ont donc comme seul voisin l'autre sommet incident à cette même arête.

Quelques éléments d'un graphe. Un *chemin* P dans un graphe G est une suite $(v_1, v_2, \ldots, v_k)$ de sommets dans $V(G)$, telle que $\{v_i, v_{i+1}\} \in E(G)$ pour tout $1 \leq i \leq k-1$. La *longueur* de ce chemin est alors égale à $k-1$. Un chemin de longueur $l+1$ est noté P_l. Un chemin est dit *élémentaire* s'il ne repasse pas plusieurs fois par le même sommet, c'est-à-dire $v_i \neq v_j$ pour tout $1 \leq i \neq j \leq k-1$.

Un *cycle* C est un chemin *élémentaire* revenant au sommet de départ, c'est-à-dire $v_1 = v_k$. Un cycle n'a donc pas de sommet de départ ni d'arrivée désignés. Un cycle de longueur l est noté C_l. Une *corde* dans un cycle est une arête reliant deux sommets non consécutifs du cycle, par exemple $\{v_2, v_6\}$.

Deux chemins sont *sommets-disjoints* (ou simplement *disjoints*) s'ils ne partagent aucun sommet interne commun (ils peuvent partager leurs deux sommets de départ et d'arrivée). Ils sont *arêtes-disjoints* s'ils ne partagent aucune arête commune (mais ils peuvent dans ce cas partager des sommets internes). De même, deux cycles sont sommets-disjoints s'ils ne partagent aucun sommet commun.

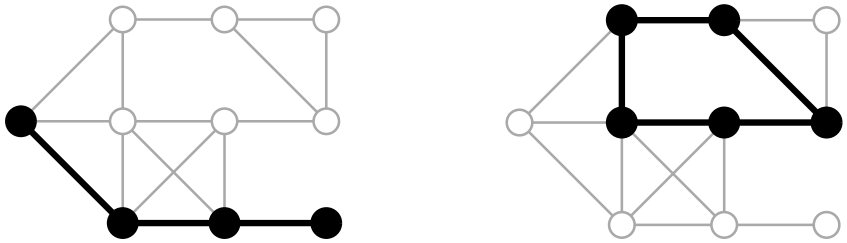


FIGURE 1.4 — Exemple d'un chemin (figure de gauche) et d'un cycle (figure de droite) dans un graphe G quelconque.

On appelle *sous-graphe* de G un graphe G' telle que $V(G') \subseteq V(G)$ et $E(G') \subseteq E(G)$. G' est un *sous-graphe induit* de G si l'ensemble des arêtes de G' correspond exactement à l'ensemble des arêtes de G dont les deux sommets incidents appartiennent à G', c'est-à-dire $E(G') = \big\{\{u, v\} \in E(G) \mid u, v \in V(G')\big\}$. On note $G[S]$ le sous-graphe induit par un sous-ensemble de sommets $S \subseteq V(G)$ de G. Un graphe G est *connexe* s'il existe un chemin entre toute paire de sommets du graphe. Une *composante connexe* $C(G)$ de G est un sous-graphe connexe de G *maximal par inclusion*, c'est-à-dire qu'aucun autre sommet du graphe ne peut être ajouté à $C(G)$ sans que ce dernier ne devienne alors un sous-graphe non connexe.

Le *complémentaire* d'un graphe G, noté $\bar{G}$, est tel que $V(\bar{G}) = V(G)$ et $E(\bar{G}) = \big\{\{u, v\} \mid \{u, v\} \notin E(G)\big\}$.

Un graphe est *complet* s'il existe une arête entre toute paire de sommets du graphe. Un sous-graphe complet d'un graphe G est appelé *clique*, et sa taille correspond au nombre de sommets qu'elle contient. Une clique contenant l sommets est notée K_l. Un sous-ensemble $S \subseteq V(G)$ de sommets de G est un *ensemble stable* (*independent set*) si $G[S]$ est le complémentaire d'une clique, c'est-à-dire qu'aucune arête n'existe entre toute paire de sommets du sous-graphe de G induit par S.

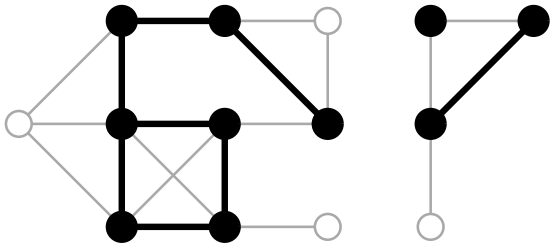


FIGURE 1.5 — Exemple d'un sous graphe G' d'un graphe G quelconque. Remarquons que ce sous-graphe n'est pas induit, car certaines arêtes de G apparaissant entre des sommets de G' n'appartiennent pas à ce sous-graphe.

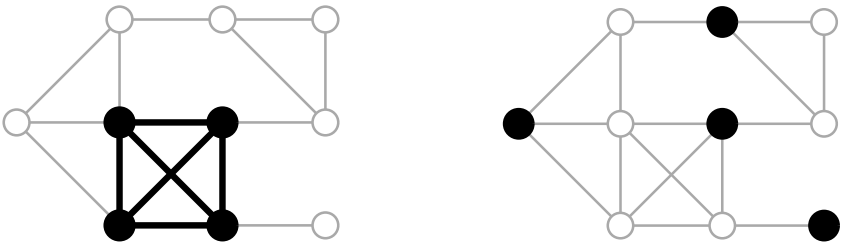


FIGURE 1.6 — Exemple d'une clique (figure de gauche) et d'un ensemble stable (figure de droite) dans un graphe G quelconque.

Soit $\mathcal{P}$ un ensemble de sous-ensembles d'éléments d'un univers U. Un *ensemble transversal* (*hitting set*) est un sous-ensemble $S \subseteq U$ d'éléments de U contenant au moins un élément de chaque sous-ensemble $\mu \in \mathcal{P}$.

Étant donné un graphe $G = (V, E)$, on appelle *couplage* (*matching*) un sous-ensemble $E' \subseteq E(G)$ d'arêtes disjointes de G, c'est-à-dire ne partageant aucun sommet incident commun. Un couplage est *parfait* (*perfect matching*) si tous les sommets du graphe sont incidents à exactement une arête du couplage. Un exemple de couplage parfait est donné dans la FIGURE 1.7.

Graphe d'incidence. Dans certains cas, il peut être utile de considérer les arêtes d'un graphe comme des sommets. Cela est rendu possible par l'utilisation du graphe d'incidence de ce graphe.

Le *graphe d'incidence* $I(G)$ d'un graphe G est défini par :

- $V\big(I(G)\big) = V(G) \cup E(G)$, c'est-à-dire que les sommets et les arêtes du graphe G sont des sommets du graphe d'incidence $I(G)$;
- $E\big(I(G)\big) = \big\{\{v, e\} \mid v \in V(G), e \in E(G)\big\}$, c'est-à-dire qu'une arête du graphe d'incidence $I(G)$ relie toujours un sommet de G à l'une de ses arêtes incidentes de G.

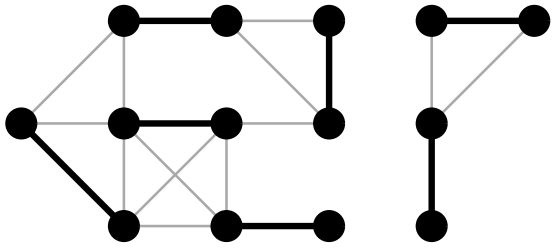


FIGURE 1.7 — Exemple d'un couplage parfait dans un graphe G quelconque.

Ce graphe d'incidence est en fait un graphe biparti, dans lequel l'une des parties correspond au sommets du graphe initial G, tandis que l'autre correspond aux arêtes du graphe G. On peut également remarquer que tous les sommets e du graphe d'incidence $I(G)$, correspondant à une arête du graphe G, sont de degré exactement 2, puisque chaque arête d'un graphe a deux sommets incidents.

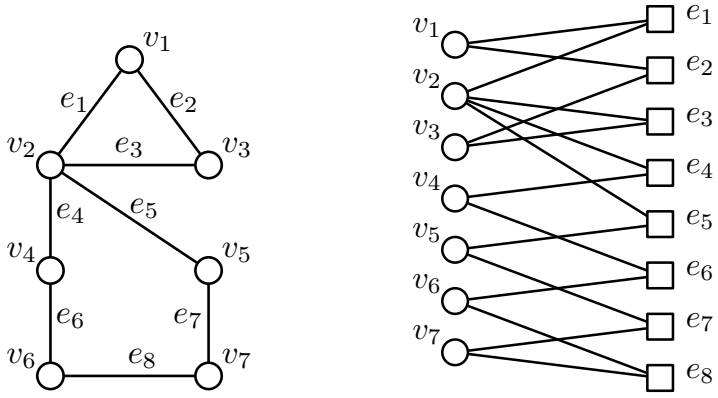


FIGURE 1.8 — Un graphe G (figure de gauche), et son graphe d'incidence $I(G)$ (figure de droite).

Pour rendre la description d'un graphe d'incidence plus aisée, nous appelons *sommet-originel* un sommet v de $I(G)$ correspondant à un sommet de G, et *sommet-arête* un sommet e de $I(G)$ correspondant à une arête de G.

1.2.2 Classes de graphes

La plupart des graphes peuvent être regroupés en « classes de graphes » partageant des propriétés ou structures communes. En les utilisant, certains problèmes de graphes, *difficiles* dans le cas général, peuvent devenir efficacement résolvables sur ces classes de graphes particulières.

Nous allons présenter ici quelques unes des principales classes de graphes connues, et que nous pourrons utiliser tout au long de cette thèse.

Arbres et forêts (*trees and forests*). Un graphe ne contenant aucun cycle est appelé *forêt*. Si de plus ce graphe est connexe, on parle alors d'*arbre*. Une forêt est donc composée de plusieurs composantes connexes, chacune étant un arbre.

La dénomination des éléments d'un arbre est quelque peu spécifique, et diffère de celle des graphes quelconques. Ainsi, les sommets d'un arbre T sont appelés *noeuds* (*nodes*), et les arêtes sont appelées *branches* (*branches*). On distingue dans un arbre deux types de noeuds : les *feuilles* (*leaves*), correspondant aux sommets de degré 1, et les *noeuds internes* (*internal nodes*), correspondant aux autres noeuds de l'arbre.

Un arbre peut être *enraciné*. Dans ce cas, un noeud particulier de l'arbre est désigné et appelé *racine* (*root*) de l'arbre, et une *hiérarchie* entre les noeuds s'applique. Chaque noeud de l'arbre dispose de zéro ou un noeud *père* (*father*), et de zéro ou plusieurs noeuds *fils* (*sons*). La racine est le seul noeud de l'arbre à n'avoir aucun père, et les feuilles sont les seuls noeuds à n'avoir aucun fils.

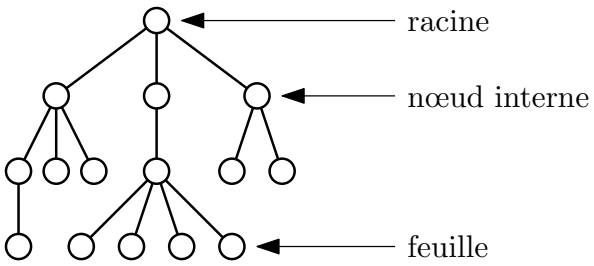


FIGURE 1.9 — Exemple d'un arbre enraciné.

De très nombreux problèmes de graphes, NP-complets pour les graphes quelconques, peuvent être résolus en temps polynomial sur les arbres (et forêts). En fait, très peu de problèmes sont connus pour rester NP-complets même pour les arbres (voir notamment [GJ79]).

Graphes de degré borné (*bounded degree*). Un graphe G est de *degré borné* par une constante $k \in \mathbb{N}$ si tout sommet du graphe est de degré au plus k. De manière équivalente, on a $\Delta(G) \leq k$.

En particulier, on peut noter que les graphes de degré borné par 1 sont composés uniquement de sommets et arêtes isolées, et les graphes de degré borné par 2 sont uniquement composés de chemins et de cycles disjoints (et éventuellement d'arêtes et sommets isolés).

Pour de nombreux problèmes sur les graphes, la complexité dépend du degré maximal du graphe. Nous pouvons citer comme exemple le problème DOMINATION (que nous définissons dans la SECTION 1.5), qui peut être résolu en temps polynomial sur les graphes de degré borné par 2 (en utilisant la structure particulière des chemins et cycles), tandis qu'il est NP-complet pour les graphes de degré borné par 3 (voir notamment [GJ79]).

Graphes k-réguliers (k-*regular*). Un graphe G est dit *k-régulier* si tout sommet du graphe est de degré exactement $k \in \mathbb{N}$. Les graphes 3-réguliers, de par l'importante littérature et les nombreux résultats de complexité relatifs à ce cas particulier, sont également appelés graphes *cubiques*.

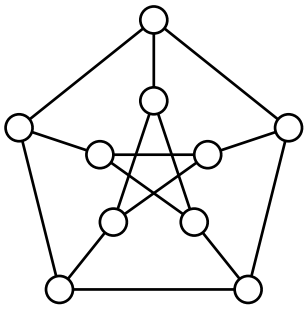


FIGURE 1.10 — Exemple d'un graphe 3-régulier. Ce graphe est aussi connu sous le nom de « graphe de PETERSEN ».

Là encore, la complexité de nombreux problèmes de graphes peut différer pour les graphes k-réguliers, selon la valeur de la constante k. Par exemple, le problème COLORATION peut être résolu en temps polynomial sur les graphes 2-réguliers (ce sont des cycles), tandis qu'il est NP-complets pour les graphes 3-réguliers (voir [Kar72]).

Graphes bipartis (*bipartite*). Un graphe G est *biparti* si l'ensemble de ses sommets peut être décomposé en deux ensembles stables, c'est-à-dire $V(G) = V_1 \cup V_2$ et $E(G) \subseteq \big\{\{u, v\} \mid u \in V_1, v \in V_2\big\}$. Nous pouvons remarquer qu'un graphe d'incidence, défini dans la SECTION 1.2.1, est un graphe biparti.

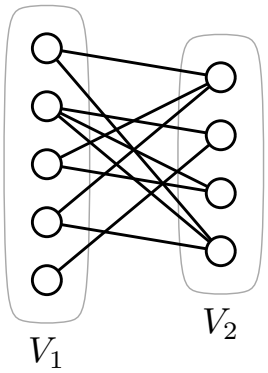


FIGURE 1.11 — Exemple d'un graphe biparti.

Graphes *split*. Un graphe G est split si l'ensemble de ses sommets peut être décomposé en deux sous-ensembles V_1 et V_2, tels que $G[V_1]$ est une clique, et V_2 est un ensemble stable. La notion de graphe *split* est donc similaire à celle d'un graphe biparti, à la différence près que l'un des deux sous-ensembles de sommets forme une clique.

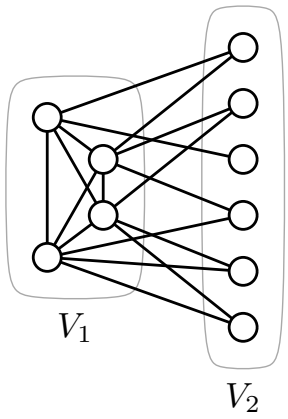


FIGURE 1.12 — Exemple d'un graphe *split*.

Graphes cordaux (*chordal*). Un graphe G est *cordal*, ou *triangulé*, si tout cycle (non-induit) de longueur au moins 4 dans G contient une corde. En d'autres termes, tout cycle induit d'un graphe cordal est de longueur 3.

Une *triangulation* d'un graphe G quelconque consiste à ajouter des arêtes à G jusqu'à obtenir un graphe G' cordal. On a donc $V(G) = V(G')$ et $E(G) \subseteq E(G')$.

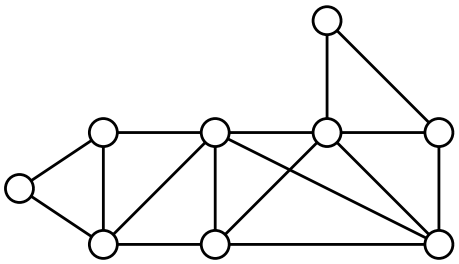


FIGURE 1.13 — Exemple d'un graphe cordal.

Les graphes cordaux sont étroitement liés à la « largeur arborescente », que nous définissons plus loin dans la SECTION 1.3.

Graphes planaires (*planar*). Un graphe G est *planaire* s'il existe un *dessin* de ce graphe sur une sphère sans aucun croisement d'arêtes.

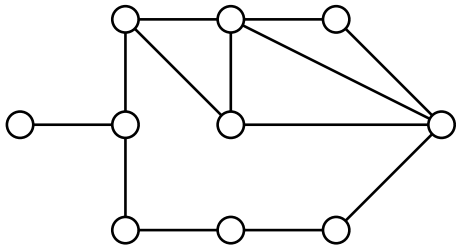


FIGURE 1.14 — Exemple d'un graphe planaire.

Les graphes planaires sont la base de l'un des théorèmes les plus importants et fondamentaux en théorie des graphes : le théorème des 4 couleurs (*four colors theorem*). Ce théorème, à l'énoncé simple mais pourtant très difficile à démontrer [AH76, RSST97, Tho98], affirme que tout graphe planaire admet une coloration de ses sommets n'utilisant que 4 couleurs au maximum (le problème COLORATION est défini plus loin, dans la SECTION 1.5).

1.3 Largeur arborescente (*tree-width*)

1.3.1 Un jeu de course-poursuite

Considérons le jeu de course-poursuite suivant, sur un graphe de taille finie. Celui-ci se joue à deux joueurs, l'un contrôlant un fugitif, et l'autre contrôlant des policiers tentant de le capturer. Le fugitif se trouve sur un sommet du graphe, et peut se déplacer d'un sommet à l'autre en suivant un chemin reliant ces deux sommets dans le graphe.

Les policiers ont pour tâche de capturer ce fugitif, et peuvent pour cela se placer sur certains sommets du graphe, sachant qu'un seul policier suffit à contrôler un sommet. Pour des raisons d'efficacité, les policiers utilisent des hélicoptères pour se déplacer d'un sommet vers n'importe quel autre sommet du graphe.

Le fugitif n'étant pas très discret, les policiers connaissent à tout instant le sommet sur lequel il se trouve. Mais il est tout de même vigilant, et voyant un hélicoptère s'approcher de sa position, il s'échappera vers un autre sommet du graphe (s'il le peut) avant qu'un policier ne se pose. Enfin, durant ses déplacements, le fugitif n'a pas intérêt à traverser un sommet occupé par un policier, car sinon il sera capturé immédiatement.

Le joueur contrôlant les policiers commence, puis chaque tour de jeu se déroule en trois étapes. Dans un premier temps, le joueur contrôlant les policiers retire du graphe les policiers qu'il souhaite déplacer, et indique à l'autre joueur les nouveaux sommets du graphe sur lesquels il va les poser. En tenant compte des sommets occupés par des policiers non déplacés et des sommets sur lesquels vont se poser les autres policiers, le joueur contrôlant le fugitif le déplace vers un sommet du graphe, si possible non occupé par un policier, en empruntant un chemin dont aucun sommet ne contient de policier. Enfin, le joueur contrôlant les policiers pose ceux qu'il avait retiré précédemment.

Puisqu'il s'agit d'un jeu, il est naturel de se demander s'il existe une stratégie permettant de gagner à coup sûr, qu'il s'agisse d'une stratégie pour que les policiers puissent capturer le fugitif, ou d'une stratégie permettant au fugitif de toujours échapper aux policiers. En étudiant plus en détails ce jeu, et en recherchant une telle stratégie gagnante pour l'un ou l'autre des deux joueurs, on s'aperçoit que l'élément crucial faisant la différence est le nombre de policiers à disposition. Si le nombre de policiers est illimité, il est facile de voir qu'une stratégie gagnante existe toujours pour le joueur contrôlant ces policiers : il lui suffit de placer un policier sur chaque sommet du graphe, et le fugitif sera nécessairement capturé par l'un deux. Si on limite ce nombre, combien faut-il au minimum de policiers pour qu'une stratégie toujours gagnante de capture du fugitif existe, et quelle est cette stratégie ? À l'inverse, si le nombre de policiers n'est pas suffisant, quelle est la stratégie permettant au fugitif de leur échapper indéfiniment ?

Disposer d'une stratégie gagnante pour les policiers, leur permettant de capturer à coup sûr le fugitif, consiste à savoir où placer les policiers afin de confiner le fugitif dans un « coin » du graphe pour finalement le capturer. Quant au fugitif, une stratégie gagnante consiste à savoir dans quelle partie du graphe se déplacer, et ce quelque soit la position des policiers.

Ces deux stratégies sont complémentaires : soit les policiers peuvent toujours confiner le fugitif dans un « coin » du graphe, soit le fugitif sait toujours comment leur échapper. Il apparaît donc clairement qu'une stratégie gagnante ne peut exister que pour un joueur à la fois. Mais d'un point de vue théorique, à quoi correspondent ces stratégies ?

Considérons le graphe G sur lequel se joue ce jeu de course-poursuite tel que défini ci-dessus. Le lien entre ce jeu de course-poursuite et la largeur arborescente du graphe G provient du résultat suivant de Seymour et Thomas [ST94] :

THÉORÈME 1.3.1 ([ST94])
Pour tout $k \geq 0$, les deux affirmations suivantes sont équivalentes :
- *$k+1$ policiers ont une stratégie de capture toujours gagnante sur le graphe G ;*
- *Le graphe G est de largeur arborescente au plus k.*

1.3.2 Définition formelle

La notion de « largeur arborescente » a été introduite par HALIN [Hal76]. Elle a été redécouverte plus tard par ROBERTSON et SEYMOUR [RS84, RS86a, RS86b], dans leur série d'articles ayant découlé au « théorème des mineurs de graphes », et indépendamment par ARNBORG *et al.* [ACP87].

En fait, ROBERTSON et SEYMOUR [RS84, RS86a, RS86b] ont étudié la notion de « largeur arborescente » (*tree-width*) telle que nous allons la définir, tandis que ARNBORG *et al.* [ACP87] se sont intéressés à la notion de « k-arbre partiel ». Ces deux notions ont par la suite été montrées équivalentes.

Il existe plusieurs définitions équivalentes de la notion de largeur arborescente ; nous reprenons ici la définition donnée par ROBERTSON et SEYMOUR [RS86a].

DÉFINITION 1.3.1 (décomposition arborescente, largeur arborescente)
Une décomposition arborescente *d'un graphe $G = (V, E)$ est une paire (T, χ), où T est un*

arbre, et $\chi = \{X_i \mid i \in I\}$ est une collection de sous-ensembles de V appelés sacs (bags)*, tels que :*

- *pour tout sommet $v \in V$, il existe un noeud $i \in T$ tel que $v \in X_i$;*
- *pour toute arête $\{u, v\} \in E$, il existe un noeud $i \in T$ tel que $u, v \in X_i$;*
- *pour tout sommet $v \in V$, l'ensemble des sacs $X_i \in T$ contenant le sommet v induit un sous-arbre T_v de T.*

La largeur *de la décomposition (T, χ) est égale à la taille du plus grand sac de cette décomposition moins un, c'est-à-dire $\max_{i \in T} |X_i| - 1$. La* largeur arborescente *du graphe G, notée $\mathrm{tw}(G)$, correspond à la plus petite largeur parmi toutes les décompositions arborescentes possibles du G.*

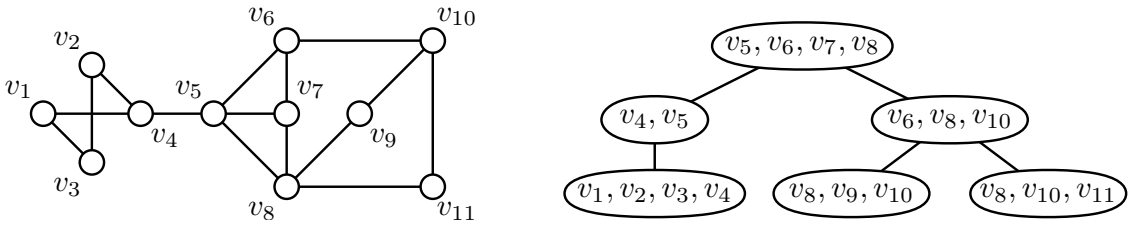


FIGURE 1.15 — Schéma d'un graphe (figure de gauche), et d'une décomposition arborescente de largeur 3 de ce graphe (figure de droite).

Le fait de mesurer la largeur d'une décomposition en retirant un à la taille du plus grand sac peut sembler étrange à première vue, mais la raison en est simple : les arbres sont alors de largeur arborescente 1. En effet, la notion de largeur arborescente d'un graphe G permet en quelque sorte de mesurer le degré de similitude du graphe G avec un arbre. Il est donc naturel de s'attendre à ce qu'un arbre soit de largeur arborescente 1. Une décomposition arborescente d'un graphe est schématisée dans la FIGURE 1.15.

De très nombreux problèmes de graphes, NP-difficiles dans le cas général, peuvent être résolus efficacement (c'est-à-dire en temps polynomial) lorsque le graphe fourni en entrée est de petite largeur arborescente. D'une certaine manière, la difficulté du problème est reportée au calcul de la largeur arborescente du graphe. Il est donc très intéressant de pouvoir déterminer si un graphe donné est de petite largeur arborescente ou non. Sans surprise, le problème de déterminer exactement la valeur de la largeur arborescente d'un graphe quelconque est NP-difficile [ACP87].

Pour déterminer la largeur arborescente d'un graphe, deux méthodes sont principalement utilisées :

- le calcul explicite de cette largeur arborescente ou d'une borne supérieure à cette largeur, par exemple en construisant une décomposition arborescente de petite largeur ; ou
- la mise en évidence d'éléments liés au graphe et donnant une borne inférieure sur cette largeur.

Dans un premier temps, suite à l'introduction de la notion de largeur arborescente, les différents travaux de recherche se sont principalement intéressés à la première méthode, et plusieurs techniques ont été créées pour calculer cette largeur, débouchant finalement sur un algorithme linéaire FPT s'exécutant en temps $\mathcal{O}(f(\mathrm{tw}) \cdot n)$, par un résultat de BODLAENDER [Bod96], permettant de déterminer si un graphe est d'une certaine largeur arborescente fixée. Notons toutefois que la fonction f est hautement exponentielle.

De par sa difficulté, la seconde méthode n'a reçu l'attention des chercheurs que très récemment. Dans ce second cas, il s'agit de trouver une forme de « certificat », sous la forme d'un objet mathématique lié au graphe, démontrant que la largeur arborescente du graphe est nécessairement grande. Cet objet mathématique est appelé *obstruction à la largeur arborescente*. Dans le CHAPITRE 2, nous définirons plus en détail les notions liées aux obstructions pour différentes largeurs, et définirons à cet occasion l'objet mathématique relatif à la largeur arborescente.

1.3.3 Algorithmique sur les décompositions

Dans la plupart des algorithmes appliqués aux graphes de petite largeur arborescente, il est supposé qu'une décomposition arborescente du graphe est fournie en entrée. En effet, cette décomposition arborescente peut être calculée en temps linéaire en la taille du graphe, à l'aide d'un algorithme de BODLAENDER [Bod96]. Nous allons expliquer succinctement comment la décomposition arborescente d'un graphe peut être utilisée par un algorithme.

En général, les algorithmes tirant avantage de la décomposition arborescente du graphe utilisent le paradigme de « programmation dynamique » sur cette décomposition.

Informellement, la décomposition arborescente (T, χ) d'un graphe est enracinée et utilisée comme suit. Dans un premier temps, pour chaque feuille de la décomposition arborescente, une collection de solutions partielles est calculée. Puis, pour chaque noeud interne i de la décomposition, la collection de solutions partielles associée à ce noeud, pour le sous-graphe de G induit par les sommets apparaissant dans des sacs du sous-arbres $T[i]$ enraciné en i, est calculée en *combinant* les collections de solutions partielles associées aux fils du noeud i dans la décomposition. Enfin, si une solution au problème existe, alors elle se trouvera parmi les solutions partielles calculées pour la racine de la décomposition.

Dans de nombreux algorithmes, la décomposition arborescente est au préalable transformée en une « décomposition arborescente jolie », équivalente à la décomposition initiale, mais dont les spécificités permettent de décrire plus facilement les opérations effectuées par l'algorithme. Un exemple de décomposition arborescente jolie est donné dans la FIGURE 1.16.

DÉFINITION 1.3.2 (décomposition arborescente jolie)
Une décomposition arborescente jolie (T, χ) *d'un graphe* $G = (V, E)$ *est une décomposition arborescente, où* T *est un arbre enraciné, et contenant seulement quatre types de noeuds :*

– *noeud feuille : une feuille* i *de l'arbre* T*, avec* $|X_i| = 1$*;*

– *noeud introduction : un noeud interne i avec un unique fils j, et tel que $X_i = X_j \cup \{v\}$ où $v \in V$;*
– *noeud suppression : un noeud interne i avec un unique fils j, et tel que $X_i = X_j \setminus \{v\}$ où $v \in V$;*
– *noeud jonction : un noeud interne i avec exactement deux fils j et k, et tel que $X_i = X_j = X_k$.*

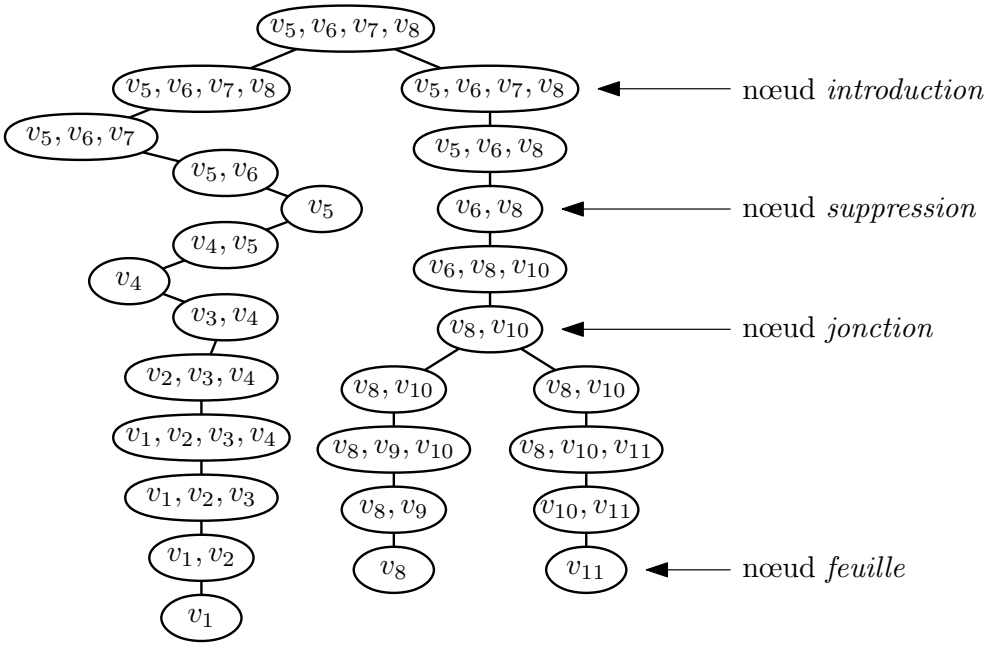


FIGURE 1.16 — Schéma d'une décomposition arborescente jolie, obtenue à partir de la décomposition arborescente quelconque schématisée dans la FIGURE 1.15.

La notion de « décomposition arborescente jolie » (*nice tree-decomposition*) a été introduite par KLOKS [Klo94]. Notons qu'une décomposition arborescente jolie d'un graphe G peut être calculée en temps polynomial à partir d'une décomposition arborescente quelconque de G.

1.4 Logique

La logique est un domaine très vaste, ayant fait l'objet de nombreux travaux de recherche, et étant le sujet de nombreux ouvrages. Nous allons effectuer un très léger survol des notions de logique liées aux graphes, en présentant l'une des structures logiques permettant de représenter les graphes, ainsi que la logique monadique du second ordre faisant l'objet d'un résultat important de COURCELLE. Le lecteur souhaitant approfondir les notions liées à la logique pourra se référer à différents ouvrages de référence, notamment « *Handbook of logic in computer science* » de ADAMSKY et MAIBAUM [AM92], et très récemment « *Graph structure and monadic second order logic, a language theoretic approach* » de COURCELLE et ENGELFRIET [CE11].

1.4.1 Structure des graphes en logique

Deux principales structures logiques existent pour représenter les graphes (et hypergraphes) dans une logique. Nous ne présentons ici que la structure que nous utiliserons tout au long de cette thèse, et appelée τ_2-structure.

Soit le vocabulaire τ_2, composé de deux relations unaires $VERT$ et $EDGE$, et d'une relation binaire I.

Nous pouvons représenter un graphe $G = (V, E)$ à l'aide d'une τ_2-stucture, notée $\mathcal{G} = (U, VERT, EDGE, I)$, où l'univers U regroupe l'ensemble des sommets et arêtes du graphe ($U := V \cup E$), les relations unaires $VERT$ et $EDGE$ permettent de différencier sommets et arêtes dans l'univers U ($VERT := V$ et $EDGE := E$), et $I := \{(v, e) \mid v \in V, e \in E, v \in e\}$ est la *relation d'incidence* du graphe.

En fait, cette τ_2-structure permet plus généralement de représenter tout *hypergraphe*, où les *hyperarêtes* peuvent relier un nombre arbitraire de sommets. Toutefois, nous n'utilisons dans cette thèse que les graphes, et c'est pourquoi nous avons utilisé le terme « graphe » pour dans la définition de cette structure logique.

Par abus de notation, les relations unaires $VERT$ et $EDGE$ seront parfois notées V et E dans une formule logique, sans qu'il n'y ait d'ambiguïté avec les ensembles de sommets V et d'arêtes E du graphe que ces relations représentent. De plus, pour simplifier l'écriture de certaines formules, nous utiliserons les notations classiques d'appartenance $x \in X \equiv X(x)$ et de sous-ensemble $X \subseteq Y \equiv \big(\forall x \in X : Y(x)\big)$, et nous noterons

$$E(x, y) \equiv \exists e \in U \ : \ E(e) \wedge I(x, e) \wedge I(y, e)$$

pour indiquer que deux sommets $x, y \in V(G)$ sont adjacents, c'est-à-dire incidents à une même arête $e \in E(G)$.

1.4.2 Logique monadique du second ordre

Logique du premier ordre. Rappelons tout d'abord quelques notions et notations de la logique du premier ordre.

Pour l'écriture de formules logiques du premier ordre, nous avons accès à un ensemble infini énumérable de *variables*, notées par des lettres latines minuscules, telles que x, y, z. Une *formule atomique sur un vocabulaire* τ est soit de la forme $x = y$, soit de la forme $R(x_1, \ldots, x_r)$ où $R \in \tau$ est une relation d'arité r et $x_1, \ldots, x_r$ sont des variables.

Une *formule* φ *du premier ordre sur le vocabulaire* τ est construite à partir de formules atomiques, composées à l'aide des connecteurs logiques $\vee, \wedge, \neg$, et des quantificateurs existentiel $\exists$ et universel $\forall$. En logique du premier ordre, ces quantificateurs ne peuvent s'appliquer qu'aux variables, c'est-à-dire aux éléments du vocabulaire τ.

La classe de toutes les formules du premier ordre est notée FO (*First-Order formulae*).

Par exemple, la formule suivante, appliquée à un graphe non orienté G dont la τ_2-structure est $\mathcal{G}(G, E^{\mathcal{G}})$, correspond au problème CLIQUE DE TAILLE k, qui est vraie si et seulement si le graphe G contient une clique de taille k :

$$\exists x_1, \ldots, x_k \in V(G)\big(\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \wedge \bigwedge_{1 \leq i < j \leq k} E(x_i, x_j)\big)$$

Pour rendre la description d'une formule plus compréhensible, nous nous autorisons l'utilisation d'abbréviations de sous-formules sous la forme de connecteurs logiques ne faisant pas partie à proprement parler du langage, tels que $\varphi \to \psi$ pour abréger $(\neg\varphi \vee \psi)$, et $\varphi \leftrightarrow \psi$ pour abréger $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Logique monadique du second ordre. La « logique monadique du second ordre » est une extension de la logique du premier ordre. Cette fois, pour un vocabulaire τ, nous pouvons non seulement sur des éléments de τ, mais également sur des ensembles d'élements de τ. Nous noterons ces ensembles d'éléments par des lettres latines majuscules, telles que X, Y, Z. La classe de toutes les formules monadiques du second ordre est notée MSO (*Monadic Second-Order formulae*).

Par exemple, la formule suivante appliquée à un graphe non orienté G correspond au problème COLORATION AVEC k COULEURS, qui est vraie si et seulement si le graphe G admet une coloration de ses sommets avec k couleurs :

$$\overbrace{\exists X_1, X_2, \ldots, X_k \subseteq V(G)}^{\text{« } X_i \text{ est l'ensemble des éléments de couleur } i. \text{ »}} \quad \forall x, y \in V(G) \quad \overbrace{\Big(\bigvee_{1 \leq i \leq k} X_i(x) \ \wedge \bigwedge_{1 \leq i < j \leq k} \neg(X_i(x) \wedge X_j(x)) \ \wedge \underbrace{\bigwedge_{1 \leq i \leq k} \big(E(x,y) \to \neg(X_i(x) \wedge X_i(y))\big)}_{\text{« Deux sommets adjacents reçoivent deux couleurs distinctes. »}} \Big)}^{\text{« Chaque sommet reçoit exactement une couleur. »}}.$$

1.4.3 Résultat de Courcelle et extensions

S'intéressant aux liens pouvant exister entre la logique et la largeur arborescente, COURCELLE [Cou97] a montré le résultat suivant, désormais connu sous le nom de « THÉORÈME de COURCELLE ». Ce résultat s'applique à toute structure logique $\mathcal{A}$ dont on sait calculer la largeur arborescente. Dans le cadre de notre thèse, la structure considérée sera celle des hypergraphes (τ_2-structure), telle que présentée dans la SECTION 1.4.1.

THÉORÈME 1.4.1 ([Cou97])
Supposons que l'on dispose d'une formule φ monadique du second ordre (MSO) pour exprimer un problème Π sur une structure quelconque $\mathcal{A}$. Alors ce problème peut être résolu en temps FPT *lorsque paramétré par la largeur arborescente de la structure $\mathcal{A}$ et la longueur de la formule φ.*

Ce résultat est très important, car il permet de montrer facilement qu'un problème est FPT lorsque paramétré par la largeur arborescente, en donnant simplement une formule monadique du second ordre de longueur polynomiale pour ce problème.

La logique monadique du second ordre (MSO) ne permet cependant pas de formuler des problèmes impliquant certains *comptages* d'éléments, et ajouter un prédicat permettant de compter exactement les éléments d'un ensemble ne semble pas possible.

Toutefois, pour aller dans ce sens, COURCELLE *et al.* [CMR01] ont introduit une extension de la logique monadique du second ordre, appelée *logique monadique du second*

ordre avec comptage (CMSO). Cette logique reprend les mêmes termes que la logique monadique du second ordre, et y ajoute un prédicat $Card_p(X)$ sur une relation d'ensemble X, qui est vrai si et seulement si la cardinalité de l'ensemble X est un multiple de l'entier $p \in \mathbb{N}^*$.

Par un résultat de COURCELLE *et al.* [CMR01], le « THÉORÈME de COURCELLE » présenté ci-dessus s'étend à cette nouvelle logique :

THÉORÈME 1.4.2 ([CMR01])
Supposons que l'on dispose d'une formule φ monadique du second ordre avec comptage (CMSO) pour exprimer un problème Π sur une structure quelconque $\mathcal{A}$. Alors ce problème peut être résolu en temps FPT *lorsque paramétré par la largeur arborescente de la structure $\mathcal{A}$ et la longueur de la formule φ.*

Ce résultat nous sera très utile dans le CHAPITRE 3 sur la DOMINATION GÉNÉRALISÉE, et en particulier dans la SECTION 3.2 où nous montrerons que dans certains cas, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps FPT lorsque paramétré par la largeur arborescente du graphe donné en entrée.

1.5 Quelques problèmes classiques de graphes

Il existe de nombreux problèmes sur les graphes, plus ou moins théoriques, et reflétant les problématiques que l'on souhaite résoudre en modélisant diverses situations par des graphes.

Pour illustrer les notions de complexité et de logique que nous venons d'introduire, nous allons présenter quelques uns des problèmes fondamentaux de la théorie des graphes, formulés selon ces différentes notions.

Pour les formulations en terme de complexité classique, nous reprenons la formulation de ces problèmes telles que données par GAREY et JOHNSON [GJ79], sous la forme de problèmes de décision. Dans le cas de la complexité paramétrée et de la logique, nous reprenons la formulation de ces problèmes telles que données par FLUM et GROHE [FG06].

Clique. Le problème CLIQUE consiste à trouver la plus grande clique d'un graphe G. Pour rappel, une *clique* dans un graphe G est un sous-graphe complet de G.

En complexité classique, le problème de décision associé peut être formulé ainsi :

CLIQUE

Entrée : Un graphe $G = (V, E)$, un entier $K \in \mathbb{N}$.
Question : Le graphe G contient-il une clique de taille au moins K ?

La complexité classique de ce problème est connu depuis les débuts de l'étude de la théorie de la complexité classique. En effet, celui-ci fait partie des vingt-et-un problèmes démontrés NP-complets par KARP [Kar72].

Du point de vue de la complexité paramétrée, il est naturel de paramétrer ce problème par le nombre de sommets composant la clique. Ainsi, le problème paramétré associé peut être formulé ainsi :

k-CLIQUE

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Le graphe G contient-il une clique de taille au moins k ?

Ce problème fait partie des premiers problèmes paramétrés ayant été démontrés W[1]-complet, par DOWNEY et FELLOWS [DF95a].

Ensemble stable. Le problème ENSEMBLE STABLE (INDEPENDENT SET), très similaire au problème CLIQUE, consiste à trouver le plus grand ensemble stable dans un graphe G. Pour rappel, un *ensemble stable* (*independent set*) dans un graphe G est un sous-ensemble S de sommets de G tel que $\bar{G}[S]$ est une clique, c'est-à-dire de manière équivalente qu'il n'existe dans G aucune arête entre les sommets appartenant à l'ensemble stable S.

En complexité classique, le problème de décision associé peut être formulé ainsi :

ENSEMBLE STABLE (INDEPENDENT SET)

Entrée : Un graphe $G = (V, E)$, un entier $K \in \mathbb{N}$.
Question : Le graphe G contient-il un ensemble stable $S \subseteq V(G)$ de cardinalité au moins K ?

Nous pouvons aisément voir que ce problème, appliqué à un graphe G quelconque, est équivalent au problème CLIQUE appliqué au complément $\bar{G}$ du graphe G. De ce fait, à l'instar du problème CLIQUE, le problème ENSEMBLE STABLE est NP-complet par KARP [Kar72].

Du point de vue de la complexité paramétrée, il est là encore naturel de paramétrer ce problème par le nombre de sommets composant l'ensemble stable que l'on recherche. Ainsi, le problème paramétré associé peut être formulé ainsi :

k-ENSEMBLE STABLE (k-INDEPENDENT SET)

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Le graphe G contient-il un ensemble stable $S \subseteq V(G)$ de cardinalité au moins k ?

Sans surprise, et parce que ce problème est équivalent au problème k-CLIQUE, le problème k-ENSEMBLE STABLE est W[1]-complet [DF95a].

Ensemble dominant. Un autre problème fondamental de recherche de sous-ensemble est le problème DOMINATION, qui consiste à trouver le plus petit ensemble dominant dans un graphe G. Un *ensemble dominant* $S \subseteq V(G)$ d'un graphe G est un sous-ensemble

de sommets tel que pour tout $v \in V(G)$, soit v est dans l'ensemble dominant S, soit au moins l'un de ses voisins est dans cet ensemble dominant.

En complexité classique, le problème de décision associé peut être formulé ainsi :

ENSEMBLE DOMINANT (DOMINATING SET)

Entrée : Un graphe $G = (V, E)$, un entier $K \in \mathbb{N}$.
Question : Le graphe G contient-il un ensemble dominant $S \subseteq V(G)$ de cardinalité au plus K ?

Dans le cadre de leur ouvrage relatif à la théorie de la NP-complétude, GAREY et JOHNSON [GJ79] ont montré que le problème ENSEMBLE DOMINANT est NP-complet.

Du point de vue de la complexité paramétrée, le paramètre qu'il est naturel de considérer est le nombre de sommets composant l'ensemble dominant que l'on recherche. Ainsi, le problème paramétré associé peut être formulé ainsi :

k-ENSEMBLE DOMINANT (k-DOMINATING SET)

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$.
Paramètre : k.
Question : Le graphe G contient-il un ensemble dominant $S \subseteq V(G)$ de cardinalité au moins k ?

Ce problème fait partie des premiers problèmes paramétrés ayant été démontrés $\mathrm{W}[2]$-complet, par DOWNEY et FELLOWS [DF95a].

Ensemble transversal. Nous avons vu précédemment la définition d'un tel ensemble dans le cas général, mais nous en donnons ici une définition propre aux graphes. Étant donnée une partition $\mathcal{P}$ de l'ensemble des sommets d'un graphe G, un *ensemble transversal* (*hitting set*) est un sous-ensemble $V' \subseteq V(G)$ contenant au moins un sommet de chaque partie $\mu \in \mathcal{P}$ de la partition.

En complexité classique, le problème de décision associé peut être formulé ainsi :

ENSEMBLE TRANSVERSAL (HITTING SET)

Entrée : Un graphe $G = (V, E)$, un entier $K \leq |V|$, un ensemble $\mathcal{P}$ de sous-ensembles de sommets de G.
Question : Existe-t-il un sous-ensemble $V' \subseteq V$ de sommets de G, de cardinalité au plus K, et tel que V' contienne au moins un sommet de chaque sous-ensemble dans $\mathcal{P}$?

À l'instar du problème CLIQUE, le problème ENSEMBLE TRANSVERSAL fait partie des vingt-et-un problèmes démontrés NP-complets par KARP [Kar72].

Du point de vue de la complexité paramétrée, il est naturel de paramétrer ce problème par le nombre de sommets composant l'ensemble transversal. Ainsi, le problème paramétré associé peut être formulé ainsi :

k-ENSEMBLE TRANSVERSAL (k-HITTING SET)

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}$, un ensemble $\mathcal{P}$ de sous-ensembles de sommets de G.
Question : Existe-t-il un sous-ensemble $V' \subseteq V$ de sommets de G, de cardinalité au plus k, et tel que V' contienne au moins un sommet de chaque sous-ensemble dans $\mathcal{P}$?

Comme le problème ENSEMBLE DOMINANT, ce problème est $\mathrm{W}[2]$-complet par un résultat de DOWNEY et FELLOWS [DF95a]. Notons que lorsque les sous-ensembles contenus dans $\mathcal{P}$ sont tous de cardinalité au plus 2, ce problème peut être résolu en temps FPT, car il correspond alors au problème k-COUVERTURE DE SOMMETS (k-VERTEX COVER) [DF99].

Coloration. Le problème classique COLORATION consiste à associer à chaque sommet d'un graphe une couleur (ou de manière équivalente, un entier naturel) tel que deux sommets voisins se voient associés deux couleurs distinctes. En fait, il est aisé de voir que ce problème revient à partitionner l'ensemble des sommets du graphe en ensembles stables disjoints.

Nous verrons dans le CHAPITRE 4 les définitions de quelques autres problèmes de coloration de graphe.

En complexité classique, le problème de décision associé peut être formulé ainsi :

COLORATION

Entrée : Un graphe $G = (V, E)$, un entier $K \leq |V|$.
Question : Le graphe G est-il K-coloriable, c'est-à-dire existe-t-il une fonction $\phi : V \to \{1, 2, \ldots, K\}$ telle que $\phi(u) \neq \phi(v)$ pour toute paire de sommets $u, v \in V$ adjacents ? De manière équivalente, l'ensemble des sommets de G peut-il être partitionné en K ensembles stables disjoints ?

Sans surprise, ce problème fait partie des vingt-et-un problèmes démontrés NP-complets par KARP [Kar72]. En fait, KARP [Kar72] a montré que ce problème est NP-complet, pour tout entier $K \geq 3$ fixé.

Du point de vue de la complexité paramétrée, l'un des paramètres naturels pour ce problème est le nombre maximum de couleurs utilisées. Ainsi, le problème paramétré associé peut être formulé ainsi :

k-COLORATION

Entrée : Un graphe $G = (V, E)$, un entier $k \in \mathbb{N}^*$.
Paramètre : k.
Question : Le graphe G est-il k-coloriable ?

Ce problème est paraNP-complet, puisque le problème COLORATION est NP-complet pour tout nombre fixé $K \geq 3$ de couleurs. Cela signifie donc qu'il n'est pas pertinent de considérer le problème k-COLORATION du point de vue de la complexité paramétrée.

Construction d'obstructions aux décompositions

2

Sommaire

2.1 Introduction

Nous allons présenter dans ce chapitre l'un de nos principaux résultats. Celui-ci concerne la construction exacte d'obstructions à certaines largeurs de graphes, c'est-à-dire le calcul d'un objet combinatoire sur un graphe permettant de certifier que ce graphe est de grande largeur.

Même si la thématique principale de cette thèse est la largeur arborescente, notre résultat est plus général puisqu'il permet de construire une obstruction pour d'autres largeurs de graphes, en particulier la largeur de branches (*branch-width*) et la largeur de découpe (*carving-width*). C'est pourquoi, dans ce chapitre, nous parlerons de « largeur », pouvant désigner n'importe laquelle des largeurs de graphes pour lesquelles notre résultat s'applique.

Les résultats de ce chapitre sont le fruit de travaux effectués avec FRÉDÉRIC MAZOIT et IOAN TODINCA. Ils ont fait l'objet d'un article en conférence [CMT09].

2.1.1 Certificat d'une grande largeur

De nombreux problèmes, NP-difficiles dans le cas général, peuvent être résolus efficacement sur des graphes de petites largeurs. Par ce biais, la complexité d'un problème est liée à une certaine mesure de la complexité du graphe, représentée par la largeur de ce graphe. Cela permet, d'une part de mieux comprendre les raisons de la NP-difficulté du problème, et d'autre part de transposer cette difficulté vers celle du calcul de la largeur du graphe.

Dès l'introduction de ces différentes largeurs, il est devenu d'importance de pouvoir déterminer si un graphe est de petite largeur ou non. Dans un premier temps, les travaux sur ce sujet se sont intéressés à la recherche d'éléments permettant de certifier qu'un graphe est de petite largeur, se traduisant, dans le cas par exemple de la largeur arborescente, par la construction d'une décomposition arborescente de petite largeur. Si le graphe est effectivement de petite largeur, alors on peut obtenir un objet mathématique permettant de résoudre efficacement ces nombreux problèmes NP-difficiles.

Toutefois, tous les graphes ne sont pas de petite largeur, et il n'est donc pas possible d'obtenir une bonne décomposition du graphe qui pourrait être utilisée par les algorithmes pré-cités. Or, la plupart des travaux de recherche s'intéressant aux largeurs de graphes ne font, dans ce cas, qu'affirmer l'impossibilité de trouver une telle décomposition, sans fournir de certificat pour cela.

Plus récemment, certains travaux se sont intéressés à l'étude et la recherche de cet objet mathématique, appelé « obstruction », permettant de certifier qu'un graphe est de grande largeur. Il ne s'agit donc plus de permettre de résoudre efficacement des problèmes NP-difficiles sur le graphe considéré, mais au contraire de constater qu'il ne sera pas possible d'utiliser un algorithme tirant partie d'une petite largeur puisqu'un certificat montrant que le graphe considéré est de grande largeur est fourni.

2.1.2 État de l'art

Lors de l'étude d'une largeur de graphes, deux axes principaux se présentent : la recherche d'algorithmes permettant de calculer cette largeur, qu'il s'agisse d'algorithmes exacts ou d'approximation, et la recherche d'éléments permettant de démontrer qu'un

graphe est de grande largeur. Les obstructions aux différentes largeurs de graphes entrent dans ce second axe, puisqu'elles permettent de certifier qu'un graphe est de grande largeur. De ce fait, leur étude est d'importance.

L'état de l'art relatif aux différentes largeurs de graphes connues est très vaste. Nous nous focalisons ici sur les travaux et résultats relatifs aux obstructions pour ces différentes largeurs.

Historiquement, la largeur arborescente (*tree-width*) est l'une des largeurs ayant reçu le plus d'attention de la part des scientifiques. Il est donc naturel que de nombreux travaux se soient penchés sur les obstructions pour cette largeur.

La notion usuelle d'obstruction pour la largeur arborescente a été introduite, sous deux notions différentes mais équivalentes, par SEYMOUR et THOMAS [ST93], sous les noms de « écran » (*screen*) et « havre » (*haven*), renommée par la suite « *bramble* » par REED [Ree97]. Dans cet article, SEYMOUR et THOMAS [ST93] ont montré un « théorème de dualité » pour la largeur arborescente, en démontrant qu'un graphe est de largeur arborescente au moins k si et seulement s'il contient un *bramble* d'ordre $k+1$. BELLENBAUM et DIESTEL [BD02] ont par la suite donné une démonstration plus simple de ce théorème.

Certains travaux s'intéressant aux obstructions pour la largeur arborescente se sont concentrés sur leur structure, leur construction, et sur différentes bornes pour les caractériser. RAMACHANDRAMURTHI [Ram97] s'est intéressé à la structure de ces obstructions, laissant apparaître une première borne inférieure exponentielle sur le nombre d'obstructions pour une largeur donnée. GUPTA *et al.* [GKS99] ont simplifié et étendu ce résultat, tandis que LAGERGREN [Lag98] a donné une borne supérieure sur ce nombre d'obstructions. CHLEBÍKOVÁ a également étudié la structure de ces obstructions, en donnant une caractérisation plus simple de l'ensemble des obstructions pour une largeur arborescente donnée. Enfin, GROHE et MARX [GM09] ont récemment montré que pour certains graphes, une obstruction d'ordre optimal est nécessairement de cardinalité exponentielle, empêchant donc de construire efficacement de telles obstructions optimales, tandis qu'une borne inférieure pour la largeur arborescente peut être obtenue à partir d'obstructions de cardinalité polynomiale (mais d'ordre non optimal).

D'autres travaux se sont penchés sur la construction de ces obstructions, afin de déterminer une borne inférieure à la largeur arborescente d'un graphe. D'une part, LUCENA [Luc07] a donné plusieurs constructions générales d'obstructions à la largeur arborescente, sous la forme de mineurs interdits. BODLAENDER *et al.* [BGK08] ont quant à eux présenté un algorithme polynomial construisant une obstruction (d'ordre non optimal) à la largeur arborescente d'un graphe, et permettant de ce fait d'obtenir une borne inférieure sur cette largeur. Très récemment, KREUTZER et TAZARI [KT10] ont amélioré ce résultat, en donnant un algorithme polynomial permettant de construire une obstruction de cardinalité polynomiale, et dont l'ordre est proche de la borne théorique donnée par GROHE et MARX [GM09] pour une obstruction de cardinalité polynomiale.

L'une des nouvelles pistes prometteuses concernant la construction d'obstructions pour la largeur arborescente est l'étude du « mineur pseudo-grille », dont la motivation provient de l'étude du « mineur grille ». En effet, ROBERTSON et SEYMOUR [RS91] ont montré une dualité entre la largeur arborescente d'un graphe et le fait que ce graphe contienne une grille comme mineur. Ce résultat a par la suite été amélioré par ROBERTSON *et al.* [RST94], et DIESTEL *et al.* [DJGT99] en ont donné une démonstration

plus simple. Toutefois, la meilleure borne obtenue sur la largeur arborescente du graphe, forçant le graphe à contenir une grille d'une certaine taille comme mineur, est exponentielle. À l'aide du « mineur pseudo-grille », version moins contrainte que le « mineur grille », REED et WOOD [RW08] ont obtenu une borne polynomiale sur la largeur arborescente d'un graphe contenant une pseudo-grille comme mineur. Notons que l'algorithme polynomial de KREUTZER et TAZARI [KT10] construit un tel mineur pseudo-grille, donnant une bonne borne inférieure à la largeur arborescente d'un graphe.

La largeur de branches (*branch-width*) est une autre largeur de graphes très étudiée. La notion usuelle d'obstruction pour cette largeur, appelée « enchevêtrement » (*tangle*), a été introduite par ROBERTSON et SEYMOUR [RS91]. Dans cet article, ils donnent notamment un « théorème de dualité » pour la largeur de branches, en démontrant qu'un graphe est de largeur de branches au moins k si et seulement s'il contient un enchevêtrement d'ordre $k+1$.

L'étude des obstructions à la largeur de branches n'est réapparue que plus récemment, notamment relancée par un résultat de HICKS [Hic04] donnant un algorithme calculant la largeur de branches d'un graphe en construisant un enchevêtrement d'ordre optimal du graphe.

De leur côté, GEELEN *et al.* [GGRW06] ont considéré la notion de largeur de branches des « matroïdes », une abstraction de certaines notions de théorie des graphes, et étudié une notion d'obstruction dans ce cadre. Sur cette base, GEELEN *et al.* [GGW09] ont étendu certains des résultats de ROBERTSON et SEYMOUR [RS91].

Notons qu'à la fin de ce chapitre, nous parlerons également de la largeur de découpe (*carving-width*), une autre notion de largeur de graphes introduite par SEYMOUR et THOMAS [ST94], qui ont également étudié une notion d'obstruction correspondante appelée « inclinaison » (*tilt*). Là encore, un « théorème de dualité » pour la largeur de découpe est donné, montrant qu'un graphe est de largeur de découpe au moins k si et seulement s'il contient une inclinaison d'ordre $k+1$.

Toutes ces largeurs de graphes possèdent une notion d'obstruction, et un théorème de dualité associé a été démontré. Dès lors, et à la vue de tous ces travaux, il est naturel de se demander s'il est possible de généraliser la notion d'obstruction, et d'obtenir un théorème général de dualité pour toute forme de *largeur* sur un graphe.

Pour ce faire, AMINI *et al.* [AMNT09] ont introduit la notion de « famille de partitions » et redéfini une notion de « fonction de partitionnement sous-modulaire ». Celles-ci généralisent certaines des notions de largeurs de graphes, et leur ont permis de montrer un théorème relativement général de dualité qui recouvre certains des différents théorèmes de dualité connus jusqu'à maintenant pour diverses largeurs de graphes.

Afin d'améliorer cette généralisation, et d'englober notamment le cas de la largeur de branches, LYAUDET *et al.* [LMT10] ont modifié légèrement les notions introduites par AMINI *et al.* [AMNT09], pour finalement montrer un théorème général de dualité recouvrant tous les théorèmes de dualité connus pour les largeurs de graphes.

2.1.3 Résultats de ce chapitre

Dans ce chapitre, nous allons présenter quelques résultats relatifs à la construction d'une obstruction pour quelques largeurs de graphes connues, c'est-à-dire la construction de certificats pour de grandes largeurs.

Dans un premier temps, nous montrons dans la SECTION 2.2 deux résultats classiques, appartenant au *folklore*, pour la construction d'une obstruction à la largeur arborescente qui est la principale largeur étudiée dans cette thèse.

Le premier résultat utilise le graphe des configurations, aussi appelé arène, prenant le point de vue « jeu de course-poursuite » que nous présentions dans les PRÉLIMINAIRES (voir SECTION 1.3.1). Ce graphe modélise toutes les situations pouvant être rencontrées dans ce jeu, et fait ainsi apparaître les coups à jouer pour que l'un ou l'autre des deux joueurs puisse gagner. Une obstruction à la largeur arborescente peut alors être indirectement construite à partir de ce graphe des configurations, en temps XP. Cette obstruction ne nous fournit cependant pas d'*objet mathématique* se rapprochant de la notion classique de *bramble*, obstruction à la largeur arborescente, et le graphe des configurations peut être de grande taille.

Le second résultat s'appuie sur le théorème des graphes mineurs de ROBERTSON et SEYMOUR [RS04], théorème fondamental de la théorie des graphes et de l'étude de la largeur arborescente. Grâce à ce théorème, un algorithme FPT peut être obtenu pour construire une obstruction à la largeur arborescente. Nous verrons toutefois que cet algorithme est purement théorique et non exploitable réellement.

Dans la SECTION 2.3, nous présentons un algorithme générique permettant de construire en temps XP une obstruction pour de nombreuses largeurs de graphes connues. Intuitivement, en partant d'une famille de partitions d'un ensemble d'éléments du graphe (l'ensemble des sommets ou des arêtes), et sous certaines contraintes sur cette famille), nous donnons un algorithme XP permettant de construire une obstruction, c'est-à-dire une collection d'éléments de ces partitions permettant (à l'aide d'un théorème de LYAUDET *et al.* [LMT10]) de certifier que la largeur considérée du graphe est grande. Nous montrons le résultat suivant :

THÉORÈME
Soit $\mathcal{P}$ une famille de partitions d'un ensemble U. Supposons que la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante et ne contient pas la partition en singletons. Alors il existe un algorithme construisant une $\hat{\mathcal{P}}$-obstruction (et donc une $\mathcal{P}$-obstruction), dont le temps d'exécution est polynomial en la cardinalité de l'ensemble U et de la famille $\mathcal{P}$.

Pour cela, nous présentons et étendons plusieurs notions introduites par AMINI *et al.* [AMNT09] et LYAUDET *et al.* [LMT10], généralisant les notions de la plupart des largeurs de graphes connues. Ces notions très générales s'intéressent au partitionnement (complet ou non) d'un ensemble quelconque d'éléments. Le résultat que nous utilisons concerne la dualité entre l'existence d'un partitionnement complet d'un ensemble quelconque d'éléments, et une obstruction certifiant qu'un tel partitionnement n'existe pas.

Enfin, dans la SECTION 2.4, nous donnons les détails de la construction d'une obstruction pour quelques unes de ces largeurs : largeur arborescente (*tree-width*), largeur

de branches (*branch-width*), et largeur de découpe (*carving-width*). Il nous suffit pour cela de définir les propriétés de partitionnement d'un graphe pour chacune de ces largeurs, l'existence et la construction d'une obstruction étant alors données par les notions et l'algorithme génériques présentés auparavant.

2.2 Folklore

2.2.1 Arène et jeu de course-poursuite

Dans les Préliminaires (Section 1.3), nous avons vu que la largeur arborescente peut être vue comme un jeu de course-poursuite, entre un fugitif se déplaçant dans un graphe G, et des policiers tentant de le capturer.

Il est alors possible de représenter l'intégralité de ce jeu sous la forme d'un « graphe des configurations », correspondant à toutes les configurations pouvant être rencontrées durant le déroulement du jeu dans ce graphe, et représentant les différents liens entre ces configurations.

Les arguments et détails de cette section sont tirés du récent livre de FOMIN et KRATSCH intitulé « *Exponential time algorithms* » [FK10].

Jeu de course-poursuite et largeur arborescente. Pour rappel, le jeu de course-poursuite que nous considérons est un jeu à deux joueurs, se déroulant sur un graphe $G = (V, E)$ non orienté. L'un des deux joueurs contrôle un fugitif, placé sur un sommet du graphe G et se déplaçant d'un sommet à l'autre du graphe, aussi vite qu'il le souhaite, en utilisant un chemin entre ces deux sommets. L'autre joueur contrôle un nombre limité de policiers, qu'il peut librement poser ou retirer des sommets du graphe (l'image couramment utilisée étant de considérer qu'ils se déplacent à l'aide d'hélicoptères).

Le joueur contrôlant les policiers a pour but de capturer le fugitif, tandis que le joueur contrôlant le fugitif a pour but d'échapper (indéfiniment) aux policiers. Le fugitif peut s'échapper au moment où un policier va être posé sur le sommet qu'il occupe, à la condition que les sommets du chemin qu'il emprunte pour s'échapper ne soient occupés par aucun policier. La capture du fugitif est donc réussie s'il ne peut pas échapper à un policier se posant sur le sommet qu'il occupe.

Comme nous le présentions dans les PRÉLIMINAIRES (voir SECTION 1.3.1), le lien entre ce jeu de course-poursuite et la largeur arborescente du graphe G provient du résultat suivant de SEYMOUR et THOMAS [ST94] :

THÉORÈME 2.2.1 ([ST94])
Pour tout $k \geq 0$, les deux affirmations suivantes sont équivalentes :
- *$k + 1$ policiers ont une stratégie de capture toujours gagnante sur le graphe G ;*
- *Le graphe G est de largeur arborescente au plus k.*

Il est relativement facile de montrer qu'une largeur arborescente d'au plus k pour le graphe G implique que $k + 1$ policiers suffisent pour capturer à coup sûr le fugitif. En effet, une décomposition arborescente de largeur k donne en fait une stratégie de capture : les policiers se posent sur les sommets composant un sac de la décomposition, puis se déplacent de sacs en sacs en suivant les déplacements du fugitif, jusqu'à le

capturer dans les noeuds composant le sac d'une feuille de la décomposition. L'autre implication du théorème, c'est-à-dire qu'une stratégie de capture avec $k+1$ policiers implique une largeur arborescente bornée par k, est moins évidente.

Nous allons montrer comment construire une stratégie de capture pour le minimum de policiers, ce qui par le théorème ci-dessus nous permettra d'en déduire la largeur arborescente du graphe G.

Graphe des configurations. Pour éviter toute confusion entre le graphe sur lequel se déroule le jeu et le graphe des configurations (que nous définissons ci-dessous), nous parlerons de « sommets » pour désigner les sommets du graphe sur lequel se déroule le jeu, et de « noeuds » pour désigner les sommets du graphe des configurations. Nous allons présenter ici la modélisation sous forme de jeu de course-poursuite, telle que définie par FOMIN et KRATSCH [FK10].

Le *graphe des configurations* sur G, aussi appelé *arène* de G, est un graphe $\mathcal{G}_G = (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{A})$ pour lequel :

- Chaque noeud $\mathcal{C} \in \mathcal{V}_1 \cup \mathcal{V}_2$ correspond à une configuration possible du jeu, c'est-à-dire un couple (C, R) tel que :
  - $C \subseteq V(G)$ est un sous-ensemble de sommets du graphe G sur lesquels se trouvent des policiers ;
  - $R \subseteq V(G)$ est une composante connexe du graphe $G \setminus C$ où le fugitif se trouve (puisqu'il peut se déplacer à une vitesse arbitrairement grande, sa position exacte n'a que peu d'importance).
- Les arcs $a \in \mathcal{A}$ relient deux noeuds $\mathcal{C}_1$ et $\mathcal{C}_2$ s'il est possible de passer de la configuration $\mathcal{C}_1$ à la configuration $\mathcal{C}_2$ en un tour, c'est-à-dire un déplacement des policiers (joueur 1) et du fugitif (joueur 2).

Ce graphe contient également deux noeuds spéciaux : le noeud $v \in \mathcal{V}_1$ correspondant à la configuration initiale du jeu, et $u \in \mathcal{V}_2$ correspondant à la capture du fugitif. Initialement, aucun policier n'est posé sur le graphe, donc la configuration initiale du jeu est $\big(\emptyset, V(G)\big)$. Il n'y a pas de configuration gagnante pour le fugitif, puisque son but est simplement d'échapper indéfiniment aux policiers.

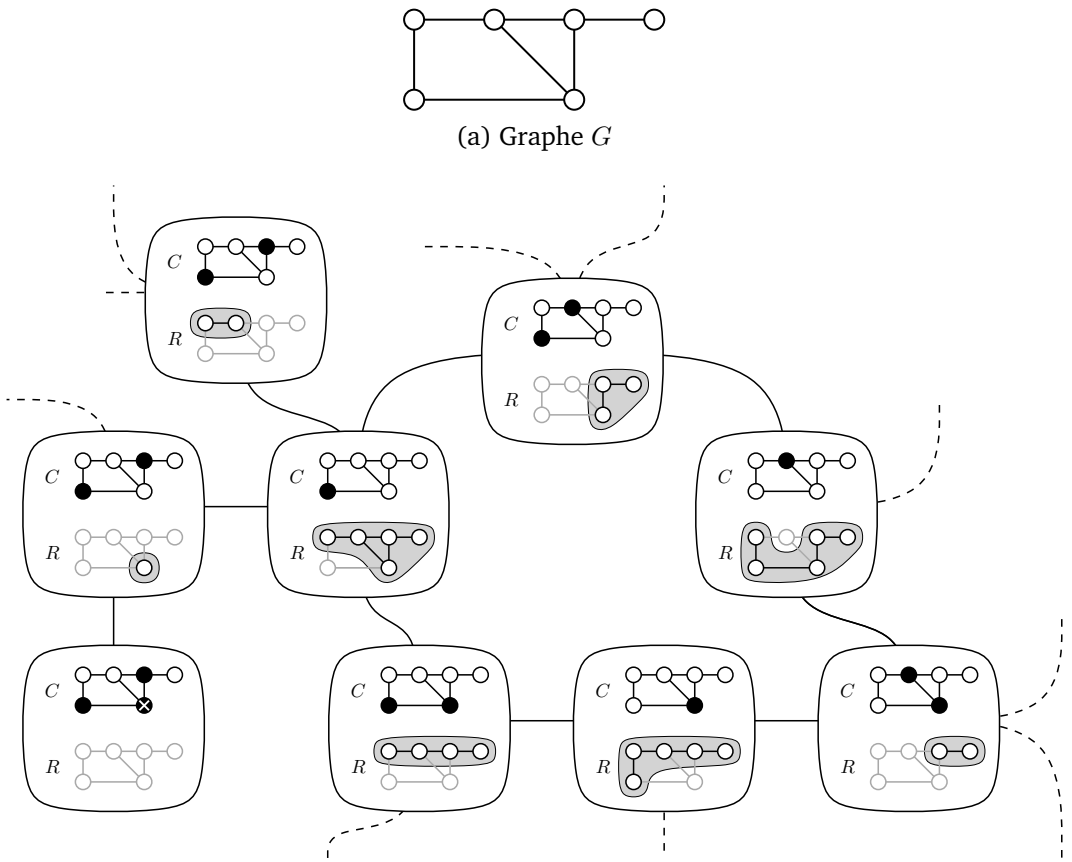
L'ensemble $\mathcal{V}_1$ contient l'ensemble des configurations pour lesquelles le tour de jeu appartient au joueur 1 contrôlant les policiers. Une configuration dans cet ensemble correspond à un « séparateur minimal », c'est-à-dire un sous-ensemble minimal de sommets séparant deux sommets du graphe G, et décomposant de ce fait G au moins deux sous-graphes connexes disjoints (voir notamment [BT01, BT02]).

L'ensemble $\mathcal{V}_2$ contient quant à lui l'ensemble des configurations pour lesquelles le tour de jeu appartient au joueur 2 contrôlant le fugitif. Une configuration dans cet ensemble correspond à un « bloc », composé d'une « clique maximale potentielle » Ω et d'une composante connexe C de $G \setminus \Omega$. Une *clique maximale potentielle* est un sous-ensemble de sommets de G pouvant apparaître comme « sac » d'une décomposition arborescente optimale du graphe G, et contient en particulier un séparateur minimal séparant la composante connexe C du reste du graphe. La composante connexe C correspond donc à l'endroit dans le graphe G où le fugitif s'échappe, lorsque les policiers sont placés sur un séparateur minimal de la clique maximale potentielle.

Deux types d'arcs composent ce graphe des configurations : les arcs « retrait », et les arcs « ajout ». Un arc *retrait* relie un noeud $(C_1, R_1) \in \mathcal{V}_1$ à un noeud $(C_2, R_2) \in \mathcal{V}_2$, où C_1

est un séparateur minimal, C_2 une clique maximale potentielle, $C_2 \subsetneq C_1$ et $R_1 = R_2$. Un tel arc correspond au retrait d'un ou plusieurs policiers de certains sommets du graphe initial G, le fugitif pouvant rester dans la composante connexe $R_1 = R_2$ qu'il occupait auparavant. Un arc *ajout* relie un noeud $(C_1, R_1) \in \mathcal{V}_1$ à un noeud $(C_2, R_2) \in \mathcal{V}_2$, où C_1 est cette fois une clique maximale potentielle, C_2 un séparateur minimal, $C_1 \subsetneq C_2$, et $R_2 \subseteq R_1$. Un tel arc correspond à l'ajout d'un ou plusieurs policiers sur certains sommets du graphe initial G, le fugitif devant choisir une composante connexe R_2 de $G \setminus C_2$.

La taille globale du graphe des configurations dépend donc du nombre de configurations possibles pour les k policiers et le fugitif, et est au plus de $\mathcal{O}\big(2 \cdot \binom{n}{k}\big) \leq \mathcal{O}(n^k)$. La FIGURE 2.1 schématise une partie du graphe des configurations H d'un petit graphe G.



(a) Graphe G

(b) Graphe des configurations H

FIGURE 2.1 — Graphe G, et extrait du graphe des configurations H correspondant.

Le jeu de course-poursuite peut se modéliser sur ce graphe des configurations : un jeton est initialement placé sur le noeud $v \in \mathcal{V}_1$, puis les deux joueurs font glisser à tour de rôle le jeton d'un noeud à un autre noeud voisin en suivant un arc de l'arène. Les noeuds du sous-ensemble $\mathcal{V}_1$ correspondent donc aux configurations où le joueur 1 contrôlant les policiers a la main, tandis que le sous-ensemble $\mathcal{V}_2$ correspond aux configurations où le joueur 2 contrôlant le fugitif a la main. Le joueur 1 gagne s'il parvient à un moment de la partie à faire glisser le jeton sur le sommet $u \in \mathcal{V}_2$, correspondant à la capture du fugitif. Sinon, c'est le joueur 2 qui gagne la partie.

En réalité, puisque le fugitif voit les déplacements des policiers et peut s'échapper avant que l'un d'eux ne se pose sur le sommet qu'il occupe, le jeu de course-poursuite se modélise sur ce graphe des configurations en considérant qu'un tour de jeu est composé d'un déplacement des policiers et du fugitif. Dans ce cas, les noeuds correspondant à un tour sont fusionnés pour ne former qu'un seul et même noeud du graphe des configurations.

Stratégie de capture. Le graphe des configurations contient toute l'information relative au jeu : les configurations possibles, ainsi que les mouvements pouvant être effectués d'un tour à l'autre. Dès lors, il est possible d'utiliser cette information pour déterminer si le nombre de policiers est suffisant pour capturer à coup sûr le fugitif, c'est-à-dire si une stratégie gagnante pour le joueur contrôlant les policiers existe.

Nous allons pour cela utiliser une procédure d'étiquetage des noeuds du graphe des configurations. Un noeud sera étiqueté si, à partir de cette configuration du jeu, les policiers ont une stratégie gagnante pour capturer le fugitif.

Au début de cette procédure, le noeud $u \in \mathcal{V}_2$ correspondant à la capture du fugitif est étiqueté. Un noeud $x \in \mathcal{V}_2$ est étiqueté si tous ses voisins $y \in \mathcal{V}_1$ (c'est-à-dire $(x, y) \in \mathcal{E}(\mathcal{G})$) sont étiquetés. Un noeud $x \in \mathcal{V}_1$ est étiqueté si au moins un de ses voisins $y \in \mathcal{V}_2$ (c'est-à-dire $(x, y) \in \mathcal{E}(\mathcal{G})$) est étiqueté.

À la fin de la procédure, si le sommet $v \in \mathcal{V}_1$ correspondant à la configuration initiale du jeu est étiqueté, alors il existe une stratégie permettant aux policiers de capturer à coup sûr le fugitif. Sinon, le fugitif pourra leur échapper indéfiniment.

Notons que cette procédure d'étiquetage s'effectue en temps linéaire en le nombre d'arêtes du graphe des configurations.

Stratégie d'échappement. Dans le cas du fugitif, il n'y a pas à proprement parler de stratégie gagnante. En effet, son objectif est d'échapper indéfiniment aux policiers, et ce quelques soient leurs déplacements. Du point de vue de la théorie des jeux, une stratégie d'échappement pour le fugitif (si elle existe) consiste à écrire un algorithme qui, pour toute configuration du jeu et en particulier toute position des policiers sur les noeuds du graphe, détermine un noeud vers lequel le fugitif doit se déplacer, c'est-à-dire une nouvelle configuration du jeu non gagnante pour les policiers. En particulier, cet algorithme doit utiliser avantageusement les cycles non gagnants dans le graphe des configurations, puisque ceux-ci permettent d'éviter indéfiniment une configuration gagnante pour les policiers.

Cette stratégie n'est clairement pas aussi *élégante* que la notion d'obstruction pour la largeur arborescente, appelée « *bramble* » et que nous définirons dans la SECTION 2.4. Celle-ci n'en est pas moins un certificat attestant que le graphe est de grande largeur arborescente.

2.2.2 Algorithme théorique en temps FPT pour une obstruction à la largeur arborescente

Théorème des mineurs de graphes. ROBERTSON et SEYMOUR ont montré, dans une vingtaine d'articles (voir notamment [RS86a, RS04]), l'un des résultats les plus importants en théorie des graphes et connu sous le nom de « Théorème des mineurs de graphes » (*Graph minors theorem*). Ce théorème montre que toute famille de graphes

close par mineurs peut être définie à partir d'une famille finie d'obstructions, et que l'appartenance d'un graphe à cette famille peut être décidée en temps FPT.

THÉORÈME 2.2.2 (Graph Minors Theorem, [RS04])
Soit $\mathcal{C}$ une famille de graphes close par mineurs, et soit $\mathcal{S}$ l'ensemble des graphes n'appartenant pas à $\mathcal{C}$. Alors $\mathcal{S}$ contient un ensemble fini $\mathcal{F}$ de graphes minimaux par mineur. De plus, étant donné un graphe G quelconque et l'ensemble $\mathcal{F}$ de mineurs minimaux, l'appartenance de G à la famille $\mathcal{C}$ peut être décidée en temps FPT *paramétré par la somme des tailles des graphes de la famille $\mathcal{F}$.*

Les éléments de $\mathcal{F}$ sont appelés « mineurs interdits » (*forbidden minors*) de la famille $\mathcal{C}$. Ces éléments peuvent également être rencontrés dans la littérature sous les noms de « mineurs exclus » (*excluded minors*) ou encore « obstructions minimales par mineur » (*minor-minimal obstructions*).

À partir de ce résultat, nous pouvons écrire un algorithme construisant une obstruction pour la largeur arborescente sous la forme d'un mineur interdit, en temps FPT paramétré par la largeur arborescente. Toutefois, le temps d'exécution de cet algorithme est *déraisonnable*, et de ce fait il est inutilisable en pratique, même pour des petites largeurs. Notons que si l'on dispose d'un *bramble* pour un certain mineur d'un graphe G, alors celui-ci donne immédiatement un *bramble* pour le graphe global.

Algorithme pour la largeur arborescente. BODLAENDER [Bod96] a donné un algorithme permettant de déterminer en temps FPT si un graphe G quelconque est de largeur arborescente au plus k. Si tel est le cas, cet algorithme renvoie également une décomposition arborescente de largeur au plus k du graphe. Dans le cas contraire, il indique seulement que le graphe est de largeur arborescente au moins $k+1$.

En utilisant le théorème des mineurs de graphes de ROBERTSON et SEYMOUR [RS04], nous pouvons écrire un algorithme permettant de compléter cette dernière information, en fournissant un certificat pour cette grande largeur arborescente.

La famille $\mathcal{C}_k^{\mathrm{tw}}$ des graphes de largeur arborescente au plus k, pour un entier k fixé, est close par mineurs. Cette famille dispose donc d'un ensemble fini $\mathcal{F}_k^{\mathrm{tw}}$ de mineurs interdits, et tout graphe G de largeur arborescente au moins $k+1$ admet au moins un élément de $\mathcal{F}_k^{\mathrm{tw}}$ comme mineur. De plus, l'ensemble $\mathcal{F}_k^{\mathrm{tw}}$ étant fini, il possède un élément de plus grande taille ne dépendant que du paramètre k. L'algorithme peut donc, en quelque sorte, énumérer par taille croissante tout graphe existant, et en particulier tous ceux appartenant à la famille $\mathcal{F}_k^{\mathrm{tw}}$, en un temps FPT paramétré par k, et trouver un mineur de G certifiant que le graphe est bien de largeur arborescente au moins $k+1$.

Cet algorithme FPT paramétré par k renvoie une obstruction de taille minimale certifiant que le graphe est bien de grande largeur arborescente, si le graphe donné en entrée est de largeur arborescente au moins $k+1$. Il s'agit toutefois d'un algorithme théorique et non exploitable dans la réalité, la fonction dépendant du paramètre k étant déraisonnable (voir notamment la série d'articles de ROBERTSON et SEYMOUR démontrant le « théorème des mineurs » [RS83, RS04]).

Données : Un graphe G, un entier k
Résultat : Le plus petit mineur H du graphe G tel que $\mathrm{tw}(H) > k$, si $\mathrm{tw}(G) > k$.
Début
  Utiliser l'algorithme de [Bod96] pour déterminer si $\mathrm{tw}(G) > k$
  Si $\mathrm{tw}(G) > k$ **Alors**
    Énumérer par taille croissante les graphes H jusqu'à trouver H **tel que** $H \lhd G$ **et** $\mathrm{tw}(H) > k$ **Retourner** H
  Sinon
    Retourner "$\mathrm{tw}(G) \leq k$"
Fin

ALGORITHME 2.1 — OBSTRUCTION MINIMALE DE GRANDE LARGEUR ARBORESCENTE (FOLKLORE)

2.3 Algorithme générique en temps XP

2.3.1 Quelques définitions

Nous allons définir les principales notions que nous utiliserons pour la construction exacte d'obstructions pour différentes largeurs de graphes. Nous adaptons ici les notions introduites par AMINI *et al.* [AMNT09] et LYAUDET *et al.* [LMT10], dans l'optique d'unifier et généraliser les différents théorèmes associés à la plupart des largeurs de graphes connues, et montrant une dualité entre décomposition du graphe selon une largeur et obstruction à cette largeur.

Idées et notations. Comme dans les travaux de LYAUDET *et al.* [LMT10], nous verrons les décompositions de graphes comme un affinage successif de partitions de l'ensemble de sommets ou d'arêtes du graphe. Pour chaque type de décomposition, nous partirons d'un ensemble de partitions *de base*, noté $\mathcal{P}$. L'affinage successif sera obtenu par une grammaire sur les partitions, la même pour tous les types de décomposition que nous traitons. Les nouvelles partitions ainsi obtenues formeront la famille étendue de partitions, notée $\hat{\mathcal{P}}$.

Il est plus confortable de voir une partition de base comme un arbre « étoile », c'est-à-dire ne contenant qu'un seul noeud interne, et dont les feuilles sont étiquetées par les parties de la partition. La grammaire d'affinage de partitions sera interprétée comme une grammaire de recollement d'arbres. Plus exactement une feuille (correspondant à un ensemble d'arêtes ou de sommets) pourra être *sous-partitionnée* en parties plus petites, ce qui en terme d'arbres se traduira par un remplacement de cette feuille par une étoile.

Étant donné un graphe et un type de décomposition, deux cas sont possibles :
- Dans la famille étendue de partitions, on trouve la partition en singletons. Ceci correspondra au fait que le graphe est décomposable entièrement (par exemple, de largeur arborescente au plus k).
- La famille étendue ne contient pas la partition en singletons. Ceci indiquera que le graphe n'est pas entièrement décomposable (par exemple, il est de largeur arborescente au moins $k+1$).

On sait que dans le deuxième cas, il existe une notion naturelle d'obstruction que l'on aimerait construire. Notre objectif est d'obtenir un algorithme qui, étant donné en entrée la famille de partitions de base $\mathcal{P}$ sur un certain univers (dans le cas de la largeur arborescente, l'ensemble des arêtes du graphe), construit cette obstruction lorsqu'elle existe. Nous verrons que lorsque le graphe est entièrement décomposable, notre algorithme peut également construire un arbre correspondant à la décomposition en singletons.

Nous noterons $\tilde{\mathcal{P}}$ une famille de partitions quelconque d'un univers $\mathcal{U}$. Supposons que l'on dispose d'une partition $\mu = (A, B, C, \ldots, Z)$ de l'univers $\mathcal{U}$. Nous utiliserons la notation (α, β, A) pour décrire de manière concise cette partition de l'univers $\mathcal{U}$, où une lettre latine (telle que $A, B, \ldots$) représente un élément de la partition, et une lettre grecque (telle que $\alpha, \beta, \ldots$) représente un ensemble d'éléments de cette partition que nous appellerons parfois « sous-partition ».

Un *pétale* (*flap*) d'une famille de partitions quelconque $\tilde{\mathcal{P}}$ de l'univers $\mathcal{U}$ est un sous-ensemble F de $\mathcal{U}$ apparaissant comme élément d'une partition $\mu \in \tilde{\mathcal{P}}$.

Arbre partitionnant. De nombreuses largeurs de graphes sont définies à partir d'une décomposition des éléments (sommets ou arêtes) composant le graphe. L'univers correspond alors à l'ensemble des sommets ou des arêtes du graphe. Celui-ci est partitionné, et les différents éléments de cette partition sont rattachés à un arbre dont les propriétés sont définies en fonction de la largeur de graphes considérée.

Afin d'uniformiser les définitions de ces arbres, AMINI *et al.* [AMNT09] ont introduit la notion « d'arbre partitionnant ».

DÉFINITION 2.3.1 (arbre partitionnant)
Un arbre partitionnant *d'un univers $\mathcal{U}$ est un arbre T tel que les feuilles de l'arbre sont en bijection avec les éléments de $\mathcal{U}$.*

Cet arbre partitionnant est donc un découpage (complet) de l'univers $\mathcal{U}$, les branches de l'arbre décrivant une manière de le découper. De plus, chaque noeud v de l'arbre T peut être vu comme une partition de l'univers $\mathcal{U}$: les éléments de cette partition correspondent alors aux ensembles de feuilles des sous-arbres obtenus en supprimant le noeud v. La FIGURE 2.2 schématise un arbre partitionnant de l'univers $\mathcal{U} = \{a, b, c, d, e, f, g, h, i, j, k\}$.

L'utilisation d'un arbre partitionnant suppose que l'univers $\mathcal{U}$ peut être décomposé entièrement, puisque chaque feuille de l'arbre correspond à un unique élément de cet univers. Toutefois, et notamment dans le cas des décompositions liées aux largeurs de graphes, il peut être intéressant de considérer des arbres qui ne décomposent que partiellement l'univers $\mathcal{U}$. Nous utilisons pour cela la notion « d'arbre partitionnant partiel » introduite par LYAUDET *et al.* [LMT10].

DÉFINITION 2.3.2 (arbre partitionnant partiel)
Un arbre partitionnant partiel *d'un univers $\mathcal{U}$ est un arbre T tel que les feuilles sont en bijection avec les parties d'une partition μ de $\mathcal{U}$.*

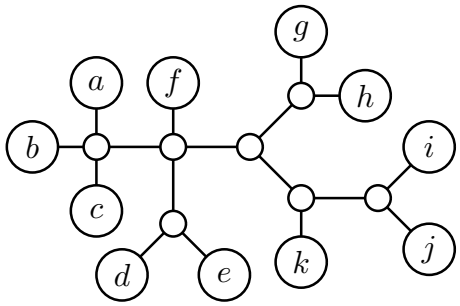


FIGURE 2.2 — Schéma d'un arbre partitionnant de l'univers $\mathcal{U} = \{a, b, c, d, e, f, g, h, i, j, k\}$.

Un arbre partitionnant partiel peut donc être vu comme une partition de l'univers $\mathcal{U}$ en ensembles d'éléments n'étant pas nécessairement des singletons. Cette notion généralise la notion d'arbre partitionnant introduite précédemment, puisqu'un arbre partitionnant *normal* est un arbre partitionnant partiel pour lequel toutes les feuilles sont des singletons. La FIGURE 2.3 schématise un arbre partitionnant partiel de l'univers $\mathcal{U} = \{a, b, c, d, e, f, g, h, i, j, k\}$.

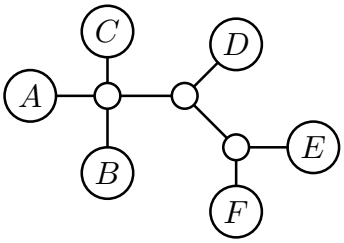


FIGURE 2.3 — Schéma d'un arbre partitionnant partiel de l'univers $\mathcal{U} = \{a, b, c, d, e, f, g, h, i, j, k\}$. Dans cet arbre, $A = \{a, b, c\}$, $B = \{d, e\}$, $C = \{f\}$, $D = \{g, h\}$, $E = \{i, j\}$, et $F = \{k\}$.

Étant donné un arbre partitionnant partiel T d'un univers $\mathcal{U}$, la partition correspondant à l'ensemble des feuilles de cet arbre est appelée *partition affichée* par l'arbre T.

Famille de partitions. S'autoriser n'importe quelle manière de partitionner l'univers $\mathcal{U}$ peut être dénué d'intérêt. Nous souhaitons écrire un algorithme utilisant ces partitions pour construire les obstructions associées à diverses largeurs de graphes. Or, si ce partitionnement est trop général et engendre un grand nombre de partitions, notre algorithme ne pourra pas les manipuler.

De ce fait, l'utilisation de la notion d'arbre partitionnant partiel, notamment dans le cas des décompositions liées aux largeurs de graphes, devient intéressante lorsque seules certaines partitions de l'univers $\mathcal{U}$ sont autorisées, en limitant leur nombre afin que notre algorithme puisse les manipuler. Cette restriction se traduit par la définition

d'une « famille de partitions » $\mathcal{P}$ *de base*, qui sera de taille raisonnable dans les cas qui nous intéressent, et d'une « famille étendue de partitions » $\hat{\mathcal{P}}$, définissant les partitions pouvant être affichées par des arbres partitionnants partiels de l'univers $\mathcal{U}$.

DÉFINITION 2.3.3 (famille de partitions)
Une famille de partitions $\mathcal{P}$ *d'un univers* $\mathcal{U}$ *est une collection de partitions de cet univers. Elle correspond à l'ensemble des partitions affichées par des arbres* $\mathcal{P}$*-partitionnants partiels ne contenant qu'un seul noeud interne.*

Lorsqu'un arbre $\mathcal{P}$-partitionnant partiel n'est composé que d'un seul noeud interne, nous l'appellerons également étoile $\mathcal{P}$-partitionnante partielle.

Par exemple, dans le cas de la largeur arborescente au plus k d'un graphe G, la famille de partitions $\mathcal{P}_k^{\text{tw}}$ est définie comme suit (voir également la SECTION 2.4) : une partition $\mu = (E_1, E_2, \ldots, E_p)$ de l'ensemble d'arêtes du graphe G appartient à la famille $\mathcal{P}_k^{\text{tw}}$ si :

- la « frontière » $\delta(\mu)$, c'est-à-dire l'ensemble des sommets incidents à au moins deux parties de μ, est de cardinalité au plus $k+1$;
- chaque partie E_i de μ correspond soit à une arête ayant ses deux extrémités dans $\delta(\mu)$, soit aux arêtes incidentes à une composante connexe de $G - \delta(\mu)$.

Pour *affiner* les partitions de la famille $\mathcal{P}$, nous introduisons la notion de « grammaire de recollement d'arbres » sur $\mathcal{P}$.

DÉFINITION 2.3.4 (grammaire de recollement d'arbres)
Soit $\hat{T}$ *un arbre* $\mathcal{P}$*-partitionnant partiel affichant la partition* $\mu_1 = (A_1, \ldots, A_p)$*,* T *une étoile* $\mathcal{P}$*-partitionnante partielle affichant la partition* $\mu_2 = (B_1, \ldots, B_q, B_{q+1}, \ldots, B_r)$*, et supposons que* $A_p = B_{q+1} \cup \ldots \cup B_r$*.*

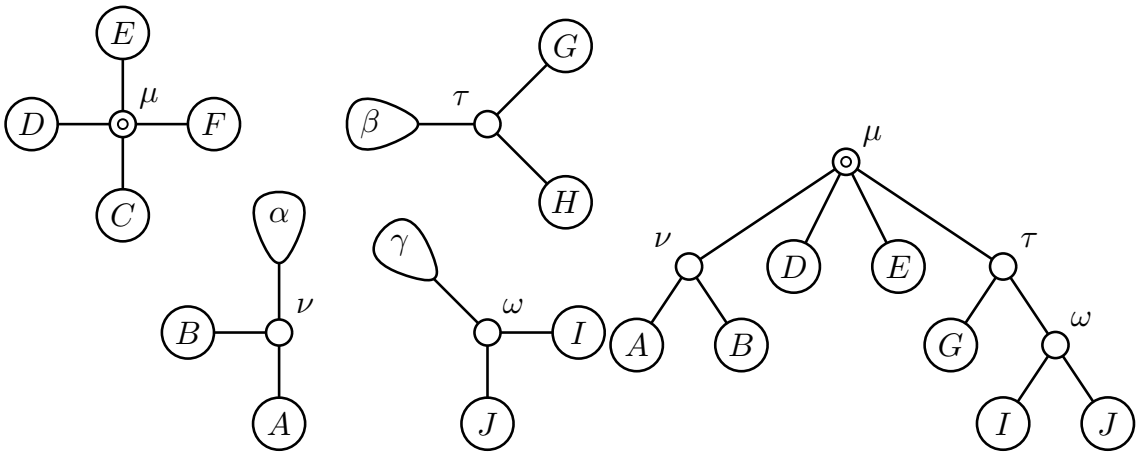
Alors nous pouvons construire un nouvel arbre $\mathcal{P}$*-partitionnant partiel* T'*, affichant la partition* $\nu = (A_1, \ldots, A_{p-1}, B_{q+1}, \ldots, B_r)$*, en identifiant la feuille correspondant à* A_1 *dans l'arbre* $\hat{T}$ *avec l'unique noeud interne de l'étoile* T*, puis en supprimant les feuilles correspondant à* $B_1, \ldots, B_q$*.*

Un arbre $\mathcal{P}$-partitionnant partiel, construit *via* ce processus de recollement d'arbres, est enraciné. La racine de cet arbre correspond à l'unique noeud interne de l'étoile à partir de laquelle cet arbre a été construit. Pour un noeud x d'un arbre $\mathcal{P}$-partitionnant partiel T, nous noterons $T(x)$ le sous-arbre de T enraciné en x.

Remarquons que ce processus de recollement d'arbres peut être réitéré plusieurs fois, à partir d'étoiles $\mathcal{P}$-partitionnantes partielles, nous permettant de construire des arbres $\mathcal{P}$-partitionnants partiels de l'univers $\mathcal{U}$ contenant plusieurs noeuds internes. De ce fait, les partitions affichées par ces arbres n'appartiennent plus nécessairement à la famille de partitions $\mathcal{P}$, et c'est pourquoi nous introduisons la notion de « famille étendue de partitions ».

DÉFINITION 2.3.5 (famille étendue de partitions)
Soit $\mathcal{P}$ *une famille de partitions d'un univers* $\mathcal{U}$*. L'ensemble des partitions affichées par les arbres* $\mathcal{P}$*-partitionnants partiels, construit à l'aide de la grammaire de recollement d'arbres sur* $\mathcal{P}$*, forme la* famille étendue de partitions $\hat{\mathcal{P}}$*.*

Dans la FIGURE 2.4, nous schématisons un exemple de tel recollement d'étoiles $\mathcal{P}$-partitionnantes partielles, permettant d'obtenir un arbre $\mathcal{P}$-partitionnant partiel affichant une partition de $\hat{\mathcal{P}}$.



(a) Collection d'étoiles $\mathcal{P}$-partitionnantes partielles (b) Arbre $\mathcal{P}$-partitionnant partiel

FIGURE 2.4 — Collection d'étoiles $\mathcal{P}$-partitionnantes partielles permettant d'obtenir, par recollements successifs, un arbre $\mathcal{P}$-partitionnant partiel affichant la partition $(A, B, C, D, E, F, G, H, I, J) \in \hat{\mathcal{P}}$. Dans ces étoiles, $C = A \cup B$, $F = G \cup H$, et $H = I \cup J$. L'étoile marquée d'un cercle est celle à partir de laquelle le recollement est effectué, et correspond donc à la racine de l'arbre $\mathcal{P}$-partitionnant partiel obtenu.

Par exemple, dans le cas de la largeur arborescente, la famille étendue de partitions $\hat{\mathcal{P}}_k^{\mathrm{tw}}$ contient la partition en singletons si et seulement si le graphe donné en entrée est de largeur au plus $k \in \mathbb{N}$. Un arbre $\mathcal{P}_k^{\mathrm{tw}}$-partitionnant partiel affichant la partition en singletons correspond à une décomposition arborescente de largeur au plus k du graphe.

Obstruction. Les arbres $\mathcal{P}$-partitionnants *non partiels* correspondent à la décomposition complète de l'univers $\mathcal{U}$ en singletons. Nous avons défini la notion d'arbre $\mathcal{P}$-partitionnant partiel pour représenter des décompositions partielles de l'univers $\mathcal{U}$ en sous-ensembles n'étant pas nécessairement des singletons. Lorsque l'univers ne peut pas être décomposé complètement à l'aide de partitions de la famille de partitions considérée, comment s'assurer qu'aucun arbre $\mathcal{P}$-partitionnant non partiel n'existe, et que seuls des arbres $\mathcal{P}$-partitionnants partiels peuvent être obtenus à partir d'une certaine famille de partitions $\mathcal{P}$?

Nous introduisons pour cela la notion de « $\mathcal{P}$-obstruction », objet dual aux arbres $\mathcal{P}$-partitionnants (non partiels). À l'instar des arbres $\mathcal{P}$-partitionnants, introduits pour généraliser les notions de décompositions pour la plupart des largeurs de graphes, la notion de $\mathcal{P}$-obstruction généralise celles des obstructions à ces largeurs de graphes.

Deux sous-ensembles $U_1, U_2 \subseteq \mathcal{U}$ *s'intersectent* si $U_1 \cap U_2 \neq \emptyset$. Nous appelons « trivial » un élément singleton dans une partition, c'est-à-dire un élément de cardinalité 1.

DÉFINITION 2.3.6 ($\tilde{\mathcal{P}}$-obstruction)
Soit $\tilde{\mathcal{P}}$ une famille de partitions de l'univers $\mathcal{U}$. Une $\tilde{\mathcal{P}}$-obstruction est un ensemble non vide de sous-ensembles de $\mathcal{U}$, s'intersectant deux-à-deux, et contenant au moins un élément non trivial de chaque partition appartenant à $\tilde{\mathcal{P}}$.

Il ne peut exister à la fois un arbre $\mathcal{P}$-partitionnant *non partiel*, et une $\hat{\mathcal{P}}$-obstruction, puisque cette dernière définit justement un ensemble de sous-ensembles de l'univers $\mathcal{U}$ n'étant décomposé entièrement en singletons par aucun arbre $\mathcal{P}$-partitionnant. Toutefois, sur la base des définitions introduites pour le moment, il est possible qu'aucun de ces deux objets n'existe. En fait, nous verrons qu'une condition supplémentaire sur la famille de partitions $\hat{\mathcal{P}}$ est nécessaire afin de garantir que l'un de ces deux objets mathématiques existe toujours.

Notons que les pétales d'une famille étendue de partitions $\hat{\mathcal{P}}$ sont les mêmes que ceux de la famille de partitions $\mathcal{P}$ correspondante, puisque les partitions de $\hat{\mathcal{P}}$ sont construites à partir des partitions de $\mathcal{P}$. La $\hat{\mathcal{P}}$-obstruction que nous construirons sera également une $\mathcal{P}$-obstruction.

2.3.2 Théorème de dualité généralisée

Propriété nécessaire et suffisante pour la dualité. Nous avons défini de manière générale les notions d'arbre $\mathcal{P}$-partitionnant et de $\hat{\mathcal{P}}$-obstruction pour toute famille de partitions $\mathcal{P}$, telles qu'introduites par LYAUDET *et al.* [LMT10], afin d'uniformiser les différents théorèmes de dualité pour la plupart des largeurs de graphes existantes. Toute famille de partitions $\mathcal{P}$ ne permet pas nécessairement d'obtenir une dualité entre arbre $\mathcal{P}$-partitionnant et $\hat{\mathcal{P}}$-obstruction, c'est-à-dire de garantir que l'un de ces deux objets mathématiques existe toujours. Nous avons toutefois vu que ces deux objets ne peuvent coexister.

LYAUDET *et al.* [LMT10] ont introduit une propriété nécessaire et suffisante d'une famille de partitions $\hat{\mathcal{P}}$, dite famille « raffinante », pour que celle-ci implique une dualité entre arbre $\mathcal{P}$-partitionnant et $\hat{\mathcal{P}}$-obstruction. Pour définir cette propriété sur une famille de partitions, il est nécessaire d'introduire d'abord une propriété sur les partitions elles-mêmes, permettant de *comparer* leur précision de découpage d'un univers $\mathcal{U}$.

Étant donnée une sous-partition β de l'univers $\mathcal{U}$, deux opérations peuvent être effectuées sur cette sous-partition :

- Suppression d'un élément de $\mathcal{U}$ apparaissant dans une partie de β. Par exemple, si $\beta = (B, \gamma)$, et $b \in B$, alors le résultat de l'opération de suppression est la sous-partition $(B \setminus \{b\}, \gamma)$.
- Partitionnement d'un élément de β. Par exemple, si $\beta = (B, \gamma)$, et δ est une partition de B, alors le résultat de l'opération de partitionnement est la sous-partition (δ, γ).

DÉFINITION 2.3.7 (partition plus fine)
Soient α, β deux sous-partitions de l'univers $\mathcal{U}$. La sous-partition α est plus fine *que la sous-partition β si elle peut être obtenue à partir de β par une séquence de suppressions et de partitionnements.*

La propriété d'une famille de partitions « raffinante », s'appliquant à toutes les partitions de la famille, peut maintenant être définie (cette propriété est schématisée dans la FIGURE 2.5).

DÉFINITION 2.3.8 (famille de partitions raffinante)
Une famille de partitions $\tilde{\mathcal{P}}$ est raffinante *si pour toutes partitions $(\alpha, A), (B, \beta) \in \tilde{\mathcal{P}}$, telles que $A \cap B = \emptyset$, il existe une partition $\mu \in \tilde{\mathcal{P}}$ plus fine que (α, β).*

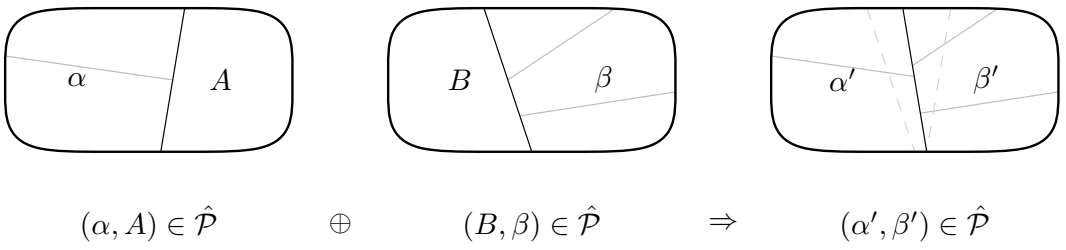


FIGURE 2.5 — Schéma de deux partitions $(\alpha, A), (B, \beta) \in \hat{\mathcal{P}}$, et d'une partition $\mu = (\alpha', \beta') \in \hat{\mathcal{P}}$ plus fine que (α, β).

Nous montrerons dans la SECTION 2.4 que les familles de partitions qui nous intéressent, liées à certaines décompositions de graphes, sont raffinantes.

Théorème de dualité généralisée. Nous pouvons maintenant présenter le théorème de dualité généralisée, montré par LYAUDET *et al.* [LMT10].

THÉORÈME 2.3.1 ([LMT10])
Soit $\tilde{\mathcal{P}}$ une famille de partitions raffinante sur un univers $\mathcal{U}$. Alors soit la partition $\tilde{\mathcal{P}}$ contient la partition en singletons, soit il existe une $\tilde{\mathcal{P}}$-obstruction.

En d'autres termes, si l'on considère une famille de partitions $\mathcal{P}$ pour laquelle la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante, alors l'univers $\mathcal{U}$ admet soit un arbre $\mathcal{P}$-partitionnant, soit une $\hat{\mathcal{P}}$-obstruction.

Lorsque l'univers $\mathcal{U}$ correspond à l'ensemble des arêtes d'un graphe, par exemple dans le cas de la largeur arborescente, ce résultat implique que pour toute largeur de graphe admettant une famille de partitions $\tilde{\mathcal{P}}$ raffinante, un graphe est de largeur au moins k si et seulement s'il admet une $\tilde{\mathcal{P}}$-obstruction d'ordre k.

Partant de ce théorème, notre but est d'écrire un algorithme générique construisant de manière explicite une $\hat{\mathcal{P}}$-obstruction (si elle existe), à partir d'une famille de partitions $\mathcal{P}$ d'un graphe G. Pour cela, nous allons utiliser cet autre résultat également montré par LYAUDET *et al.* [LMT10], définissant les conditions minimales nécessaires pour obtenir une $\tilde{\mathcal{P}}$-obstruction pour une famille de partitions quelconque $\tilde{\mathcal{P}}$.

THÉORÈME 2.3.2 ([LMT10])
Soit $\tilde{\mathcal{P}}$ une famille de partitions sur un univers $\mathcal{U}$, raffinante et ne contenant pas la partition en singletons. Soit $\mathcal{B}$ un ensemble de sous-ensembles de $\mathcal{U}$ tel que :

(i) chaque élément de $\mathcal{B}$ est de cardinalité au moins 2, et apparaît comme partie dans au moins une partition de $\mathcal{P}$;

(ii) pour toute partition $(A_1, \ldots, A_p) \in \tilde{\mathcal{P}}$, il existe une partie $A_i \in \mathcal{B}$;

(iii) $\mathcal{B}$ est clos par sur-ensembles ;

(iv) $\mathcal{B}$ est minimal par inclusion sous les trois conditions précédentes.

Alors l'ensemble $\mathcal{B}$ est une $\tilde{\mathcal{P}}$-obstruction.

Par soucis de complétion, nous donnons ici la preuve de ce théorème, telle que donnée par LYAUDET *et al.* [LMT10].

Démonstration. Notons tout d'abord que $\mathcal{B}$ existe toujours : en effet, prenons l'ensemble $\mathcal{B}_0$ correspondant à toutes les parties des partitions de $\tilde{\mathcal{P}}$ qui ne sont pas des singletons. Cet ensemble satisfait les conditions du théorème, à l'exception de la minimalité. Il suffit alors d'extraire le sous-ensemble minimal $\mathcal{B} \subseteq \mathcal{B}_0$ satisfaisant les conditions du théorème. Nous allons montrer que ce sous-ensemble minimal $\mathcal{B}$ est une $\tilde{\mathcal{P}}$-obstruction.

Supposons par contradiction que $\mathcal{B}$ n'est pas une $\tilde{\mathcal{P}}$-obstruction. Soient A, B deux ensembles minimaux disjoints appartenant à $\mathcal{B}$. Puisque $\mathcal{B}$ est minimal, il existe deux partitions $(A', A_2, \ldots, A_p), (B', B_2, \ldots, B_q) \in \tilde{\mathcal{P}}$, où seules les parties A' et B' appartiennent à $\mathcal{B}$, et telles que $A' \subseteq A$ et $B' \subseteq B$. Comme $\tilde{\mathcal{P}}$ est raffinant, il existe une autre partition $(C_1, \ldots, C_r) \in \tilde{\mathcal{P}}$ plus fine que $(A_2, \ldots, A_p, B_2, \ldots, B_q)$, et comme $\mathcal{B}$ contient une partie de toute partition appartenant à $\tilde{\mathcal{P}}$, $\mathcal{B}$ contient nécessairement l'une des parties C_i, qui est en particulier un sous-ensemble d'une certaine partie A_j ou B_k. Enfin, $\mathcal{B}$ étant clos par sur-ensembles, $A_j \in \mathcal{B}$ ou $B_k \in \mathcal{B}$, ce qui est une contradiction avec la supposition que seules A' et B' appartiennent à $\mathcal{B}$. Donc $\mathcal{B}$ est une $\tilde{\mathcal{P}}$-obstruction. □

L'algorithme que nous allons décrire construira une $\hat{\mathcal{P}}$-obstruction en temps polynomial en la cardinalité de la famille $\mathcal{P}$ et de l'univers $\mathcal{U}$, donnant alors un algorithme XP pour les quelques largeurs de graphes que nous considérerons, car les familles correspondantes seront de cardinalité $\mathcal{O}(n^{f(k)})$. Notons que dans le cas de certaines largeurs, en particulier la largeur arborescente, nous avons vu dans la SECTION 2.2.2 qu'il est possible de montrer, d'un point de vue théorique, que le calcul d'une $\hat{\mathcal{P}}$-obstruction peut être effectuée en temps FPT, à l'aide du « théorème des mineurs de graphes » de ROBERTSON et SEYMOUR [RS04].

2.3.3 Algorithme

Nous allons dans cette section appliquer le Théorème 2.3.2, afin d'obtenir un algorithme permettant de calculer explicitement une $\mathcal{P}$-obstruction, à partir d'une famille de partitions $\mathcal{P}$ telle que la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante. Remarquons que comme la famille de partitions $\mathcal{P}$ n'est en général pas raffinante, nous ne pouvons pas l'utiliser directement.

Algorithme. Nous allons montrer le résultat générique suivant, pour toute famille de partitions $\mathcal{P}$ telle que la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante.

THÉORÈME 2.3.3
Soit $\mathcal{P}$ une famille de partitions d'un univers $\mathcal{U}$. Supposons que la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante, et ne contient pas la partition en singletons. Alors il existe un algorithme pour construire une $\hat{\mathcal{P}}$-obstruction, dont le temps d'exécution est polynomial en la cardinalité de l'univers $\mathcal{U}$ et de la famille de partitions $\mathcal{P}$.

Nous pouvons écrire un premier algorithme naïf, décrit dans lALGORITHME 2.2, en traduisant directement le THÉORÈME 2.3.2 appliqué à la famille de partitions $\mathcal{P}$. L'ensemble retourné par cet algorithme est donc une $\hat{\mathcal{P}}$-obstruction.

Données : Une famille de partitions $\mathcal{P}$
Résultat : Une $\hat{\mathcal{P}}$-obstruction (s'il en existe une)
Début
  $\mathcal{B} \leftarrow$ l'ensemble des pétales de $\mathcal{P}$
  $\mathcal{B}_f \leftarrow \emptyset$
  Pour Chaque $F \in \mathcal{B}$ tel que $|F| = 1$ **Faire**
    Retirer F de $\mathcal{B}$
  Pour Chaque $F \in \mathcal{B}$ par ordre d'inclusion **Faire**
    Si $\exists\ \mu \in \hat{\mathcal{P}}$ telle que F est l'unique pétale non retiré dans cette partition
    ou $\exists\ F' \in \mathcal{B}_f$ **tel que** $F' \subseteq F$ **Alors**
      Ajouter F à $\mathcal{B}_f$
    Sinon
      Retirer F de $\mathcal{B}$
  Retourner $\mathcal{B}_f$
Fin

ALGORITHME 2.2 — CONSTRUCTION NAÏVE D'UNE OBSTRUCTION

Cependant, la cardinalité de la famille étendue de partitions $\hat{\mathcal{P}}$ peut être exponentielle en la cardinalité de l'univers $\mathcal{U}$ et de la famille de partitions $\mathcal{P}$. Cette explosion combinatoire provient de la conditionnelle "**Si** $\exists\ \mu \in \hat{\mathcal{P}}$ telle que F est l'unique pétale non retiré dans cette partition", qui travaille sur la famille étendue de partitions $\hat{\mathcal{P}}$. Afin d'obtenir une meilleure complexité en temps, nous devons modifier cette conditionnelle afin d'utiliser la famille de partitions $\mathcal{P}$. Pour cela, nous allons la remplacer par un processus de marquage sur les parties des partitions de $\mathcal{P}$.

Processus de marquage. Lorsqu'un pétale F est retiré suite à l'instruction "Retirer F de $\mathcal{B}$", celui-ci induit un forçage sur la $\hat{\mathcal{P}}$-obstruction en cours de construction : certains pétales devront être ajoutés à cette $\hat{\mathcal{P}}$-obstruction, tandis que d'autres ne pourront pas être ajoutés. Par ailleurs, ce forçage est susceptible de se propager à d'autres pétales. De ce fait, dès qu'un pétale est retiré, nous appellerons l'algorithme MISE À JOUR DES MARQUAGES défini ci-dessous.

Pour ce processus, nous introduisons deux autres marquages des pétales : *interdit* et *forcé*. Un pétale marqué *interdit* ne pourra pas être ajouté à la $\hat{\mathcal{P}}$-obstruction en cours de construction, tandis qu'un pétale marqué *forcé* devra y être ajouté. Pour obtenir notre algorithme CONSTRUCTION D'UNE OBSTRUCTION, décrit ci-dessous, nous

remplaçons dans l'algorithme CONSTRUCTION NAÏVE D'UNE OBSTRUCTION la conditionnelle problématique, travaillant sur la famille étendue de partitions $\hat{\mathcal{P}}$, par une nouvelle conditionnelle, travaillant cette fois sur la famille de partitions $\mathcal{P}$ *de base*, et contrôlant le marquage des pétales. L'algorithme MISE À JOUR DES MARQUAGES sera alors appelé après chaque occurrence de l'instruction "Retirer F de $\mathcal{B}$", et nous retirons ensuite tous les pétales ayant été marqués *interdits* durant ce processus de marquage.

Début
  `// marquage des pétales interdits`
  Tant Que $\exists$ un pétale F et une partition $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$ **tel que** $F = \left(\cup_{i=1}^{p} F_i\right)$ **et** $\forall 1 \leq i \leq p$, F_i est retiré ou *interdit* **Faire**
    Marquer F comme *interdit*
  `// marquage des pétales forcés`
  Tant Que $\exists (F, F_2, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$, avec $F' = F \cup F_2 \cup \ldots \cup F_p$ **tel que** soit $F' = \mathcal{U}$, soit F' est marqué *forcé*, **et** $\forall 2 \leq i \leq p$, F_i est retiré ou *interdit* **Faire**
    Marquer F comme *forcé*
Fin

ALGORITHME 2.3 — MISE À JOUR DES MARQUAGES

Données : Une famille de partitions $\mathcal{P}$
Résultat : Une $\hat{\mathcal{P}}$-obstruction (s'il en existe une)
Début
  $\mathcal{B} \leftarrow$ l'ensemble des pétales de $\mathcal{P}$
  $\mathcal{B}_f \leftarrow \emptyset$
  Pour Chaque $F \in \mathcal{B}$ tel que $|F| = 1$ **Faire**
    Retirer F de $\mathcal{B}$
  Pour Chaque $F \in \mathcal{B}$ par ordre d'inclusion **Faire**
    Si F est marqué *forcé* **ou** $\exists\ F' \in \mathcal{B}_f$ **tel que** $F' \subseteq F$ **Alors**
      Ajouter F à $\mathcal{B}_f$
    Sinon
      Retirer F de $\mathcal{B}$
      MISE À JOUR DES MARQUAGES
    Retirer tous les pétales marqués *interdits*
  Retourner $\mathcal{B}_f$
Fin

ALGORITHME 2.4 — CONSTRUCTION D'UNE OBSTRUCTION

Remarquons que dans tout arbre $\mathcal{P}$-partitionnant partiel T, pour chaque noeud x de T, l'union des feuilles du sous-arbre $T(x)$ correspond à un pétale noté $E(x)$. Par ailleurs, s'il existe un autre arbre $\mathcal{P}$-partitionnant partiel T' ayant un noeud y tel que $T'(y)$ correspond à un pétale $E(y) = E(x)$, alors nous pouvons remplacer dans T le sous-arbre $T(x)$ par $T'(y)$, et nous obtenons ainsi un nouvel arbre $\mathcal{P}$-partitionnant partiel.

LEMME 2.3.4
Un pétale F est marqué interdit *si et seulement s'il existe un sous-arbre $T(x)$ d'un arbre $\mathcal{P}$-partitionnant partiel, affichant une sous-partition de $\hat{\mathcal{P}}$, et tel que :*
- *chaque pétale associé à une feuille de $T(x)$ est retiré ;*
- *l'union de ces pétales est exactement F.*

Démonstration. Supposons dans un premier temps que le sous-arbre $T(x)$ existe. Nous allons montrer que le pétale F correspondant aux feuilles de $T(x)$ est marqué *interdit*. Notons que par la construction de l'arbre T, effectuée à l'aide de la grammaire de recollement d'arbres sur $\mathcal{P}$, chaque noeud interne y de $T(x)$ correspond à une partition $\mu_y \in \mathcal{P}$.

Par construction de l'arbre T, l'union des feuilles $E(y)$ de $T(y)$ forment un pétale. En considérant les noeuds internes du sous-arbre $T(x)$ des feuilles vers la racine, nous montrons par induction que tous les pétales de ce type sont alors marqués *interdits*. Ainsi, lorsque le noeud y est considéré, pour chacun de ses fils $y_1, \ldots, y_p$, soit ce fils est une feuille de l'arbre et donc correspond à un pétale qui a été retiré, soit $E(y_i)$ a été précédemment marqué *interdit*. De ce fait, l'algorithme MISE À JOUR DES MARQUAGES marque *interdit* le pétale formé par $E(y)$, et le pétale formé par $E(x)$ est finalement marqué *interdit*.

Supposons maintenant que F est un pétale marqué *interdit*. Nous allons construire un arbre $\mathcal{P}$-partitionnant partiel T, contenant un noeud x tel que $T(x)$ correspond au LEMME.

Nous allons procéder par induction, en suivant l'ordre par inclusion, sur les pétales marqués *interdits*. Lorsqu'un pétale F est marqué *interdit*, il est l'union de plusieurs pétales $F_1, \ldots, F_p$, où chacun de ces pétales est retiré ou marqué *interdit*, et tel que $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$. Par hypothèse d'induction, pour chaque pétale F_i, avec $i \leq p$, nous pouvons lui associer un sous-arbre T_i (qui est un sous-arbre d'un arbre $\mathcal{P}$-partitionnant partiel) tel que les pétales associés avec les feuilles de T_i, qui forment une partition de F_i, sont tous retirés. Notons que, si F_i est un pétale retiré, alors le sous-arbre T_i est en fait une feuille de l'arbre T.

Considérons maintenant un sous-arbre $T(x)$ formé par une racine x, correspondant à la partition $(F_1, \ldots, F_p, F_{p+1}, \ldots, F_q) \in \mathcal{P}$, et reliée aux racines des sous-arbres $T_1, \ldots, T_p$. Considérons également une partition $(F, F'_1, \ldots, F'_r) \in \mathcal{P}$ (une telle partition existe, car F est un pétale). Notons que $\cup_j F'_j = \cup_{i \geq p+1} F_i$. L'arbre T est alors obtenu en choisissant une racine z, correspondant à la partition $(F, F'_1, \ldots, F'_r)$, et à laquelle nous relions le sous-arbre $T(x)$ en ajoutant une branche $\{x, z\}$, et tel que pour chaque pétale F'_j nous ajoutons une feuille adjacente à la racine z et associée à ce pétale. De ce fait, l'arbre final T est un recollement de l'arbre enraciné en x et de l'arbre enraciné en z. Par construction l'arbre T est un arbre $\mathcal{P}$-partitionnant partiel affichant un élément de la famille étendue de partitions $\hat{\mathcal{P}}$. Il est aisé de voir que toutes les feuilles de $T(x)$ sont alors associées à des pétales retirés. □

LEMME 2.3.5
Un pétale F est marqué forcé *si et seulement s'il existe une partition dans $\hat{\mathcal{P}}$ telle que F est l'unique pétale non retiré.*

Démonstration. Supposons tout d'abord que F est l'unique pétale non retiré d'une partition $\mu \in \hat{\mathcal{P}}$, et montrons que F sera marqué *forcé*. Soit T un arbre $\mathcal{P}$-partitionnant partiel affichant la partition μ. On note $r = x_0$ la racine de T, et $P = (x_0, x_1, \ldots, x_t = u)$ l'unique chemin dans T de r vers la feuille u correspondant à F. Par le LEMME 2.3.4, pour chaque noeud x_i (avec $0 \leq i \leq t-1$), et chaque fils y de x_i, différent de x_{i+1}, le sous-arbre $T(y)$ correspond à un pétale retiré ou marqué *interdit*. Par la condition de marquage *forcé* utilisée dans notre algorithme MISE À JOUR DES MARQUAGES, les pétales $E(x_1), E(x_2), \ldots, E(x_t)$ seront successivement marqués *forcés*. En particulier, $F = E(x_t)$ sera marqué *forcé*.

Réciproquement, montrons que si le pétale F est marqué *forcé*, alors il existe une partition $\mu \in \hat{\mathcal{P}}$ telle que F est l'unique pétale non retiré. Pour cela, nous effectuons une induction sur l'ordre du marquage *forcé*. Lorsque du tout premier marquage *forcé* d'un certain pétale F_0, nous sommes nécessairement dans la situation où il existe une partition $(F_0, F_2, \ldots, F_p) \in \mathcal{P}$ telle que chaque F_i est retiré ou marqué *interdit*. Pour ce premier pétale F_0, construisons un arbre $\mathcal{P}$-partitionnant partiel T. Pour chaque F_i, avec $2 \leq i \leq p$, il existe un sous-arbre $T_i(x_i)$ d'un certain arbre $\mathcal{P}$-partitionnant partiel T_i où $E(x_i) = F_i$, tel que toutes les feuilles de $T_i(x_i)$ correspondent à des pétales retirées. L'arbre T est formé d'une racine r, à laquelle est recollé chaque sous-arbre $T_i(x_i)$, et une feuille correspondant au pétale F_0. Nous obtenons ainsi un arbre $\mathcal{P}$-partitionnant partiel affichant la partition $(F_0, F_2, \ldots, F_p)$ et dont l'unique pétale non retiré est F_0. Ceci prouve le cas de base de notre induction, ainsi que tous les cas où le marquage *forcé* se fait sur la condition $F' = \mathcal{U}$ dans notre algorithme MISE À JOUR DES MARQUAGES.

Il nous reste le cas où le marquage *forcé* du pétale F se fait sur la condition où F' est marqué *forcé*. Par hypothèse d'induction, il existe un arbre $\mathcal{P}$-partitionnant partiel T', affichant une partition dont F' est l'unique pétale non retiré. Soit y la feuille de T' correspondant au pétale F'. Nous recollons au noeud u de T' les sous-arbres $T_i(x_i)$ définis comme auparavant, ainsi qu'une feuille correspondant au pétale F. Nnous obtenons un arbre $\mathcal{P}$-partitionnant partiel affichant la partition $(F, F_2, \ldots, F_p, F_{p+1}, \ldots, F_q)$ et dont l'unique pétale non retiré est F. □

Complexité de l'algorithme. Intéressons-nous maintenant à la complexité en temps de notre algorithme. Cette complexité au pire des cas est obtenue par :

- les appels à l'ALGORITHME MISE À JOUR DES MARQUAGES ;
- la complexité du test "$\exists F' \in \mathcal{B}_f : \ F' \subseteq F$" de l'ALGORITHME CONSTRUCTION D'UNE OBSTRUCTION.

Ces deux cas sont polynomiaux en le nombre total de pétales, la cardinalité de $\mathcal{P}$, et la cardinalité de l'univers $\mathcal{U}$. En fait, le nombre total de pétales est au plus $|\mathcal{U}| \cdot |\mathcal{P}|$, puisque chaque partition a au plus $|\mathcal{U}|$ parties.

La complexité globale en temps de notre ALGORITHME CONSTRUCTION D'UNE OBSTRUCTION est donc :

$$\mathcal{O}^*\big(|\mathcal{P}|^2 \cdot poly(|\mathcal{U}|)\big)$$

Il nous faut toutefois nous assurer que le test déterminant si un pétale F contient un autre pétale $F' \in \mathcal{B}_f$ peut également être effectué dans ce même temps d'exécution. Pour cela, nous construisons un graphe orienté acyclique $\mathcal{G}_{petales}$ dans lequel chaque noeud est un pétale (non trivial) de $\mathcal{P}$, et tel que la clôture transitive du graphe correspond à la relation d'inclusion entre les pétales (le graphe est donc de taille $\mathcal{O}^*(|\mathcal{P}|^2)$).

Ainsi, lorsqu'un pétale F' est ajouté à la solution partielle $\mathcal{B}_f$, tous les pétales F tel qu'il existe un chemin de F' à F dans $\mathcal{G}_{petales}$ sont marqués *forcés par inclusion*. En utilisant des techniques standards, ce marquage peut être effectué en temps linéaire en la taille du graphe $\mathcal{G}_{petales}$.

Nous verrons dans la SECTION 2.4 qu'il est possible, pour la largeur arborescente notamment, de réduire la complexité de cet algorithme. Il nous faudra alors nous assurer que la taille du graphe $\mathcal{G}_{petales}$ suivra cette réduction.

2.4 Exemples pour quelques largeurs de graphes

Dans cette section, nous allons définir des familles de partitions associées à quelques largeurs de graphes bien connues. Pour une largeur considérée, notre algorithme CONSTRUCTION D'UNE OBSTRUCTION nous permet de construire, pour un graphe donné, une obstruction optimale à cette largeur en temps polynomial en la taille de la famille de partitions correspondante et la taille du graphe.

Les familles étendues de partitions $\hat{\mathcal{P}}$ que nous allons définir pour ces quelques largeurs seront construites à l'aide de notre grammaire de recollement, ce qui diffère légèrement de la construction utilisée par LYAUDET *et al.* [LMT10] pour leurs familles étendues de partitions $\hat{\mathcal{P}}'$. De ce fait, nos familles $\hat{\mathcal{P}}$ contiendront potentiellement moins de partitions que leurs familles $\hat{\mathcal{P}}'$. Or, nous aurons besoin de montrer que nos familles $\hat{\mathcal{P}}$ sont raffinantes, de sorte que notre algorithme CONSTRUCTION D'UNE OBSTRUCTION puisse s'appliquer à ces familles, et construire une $\mathcal{P}$-obstruction satisfaisant le THÉORÈME 2.3.1 de dualité.

Nous verrons qu'en fait, dans le cas des largeurs de graphes que nous allons étudier, chacune de nos familles $\hat{\mathcal{P}}$ est contenue dans la famille $\hat{\mathcal{P}}'$ *classique* correspondante, et que ces familles classiques sont raffinantes. Nous utiliserons alors le lemme suivant pour montrer que nos familles $\hat{\mathcal{P}}$ sont également raffinantes.

LEMME 2.4.1
Soient $\tilde{\mathcal{P}}$ et $\tilde{\mathcal{P}}'$ deux familles de partitions, telles que $\tilde{\mathcal{P}} \subseteq \tilde{\mathcal{P}}'$. Supposons que $\tilde{\mathcal{P}}'$ est raffinante, et que pour toute partition $\mu' \in \tilde{\mathcal{P}}'$, il existe une partition $\mu \in \tilde{\mathcal{P}}$ plus fine que μ'. Alors $\tilde{\mathcal{P}}$ est raffinante.

Démonstration. Considérons deux familles de partitions $\tilde{\mathcal{P}}$ et $\tilde{\mathcal{P}}'$ telles que définies dans ce lemme. Prenons deux partitions $(\alpha, A), (B, \beta) \in \tilde{\mathcal{P}}$, telles que $A \cap B = \emptyset$. Nous allons montrer que $\tilde{\mathcal{P}}$ contient une partition plus fine que (α, β), impliquant donc que la famille $\tilde{\mathcal{P}}$ est raffinante.

Puisque $\tilde{\mathcal{P}} \subseteq \tilde{\mathcal{P}}'$, les deux partitions $(\alpha, A), (B, \beta)$ appartiennent également à $\tilde{\mathcal{P}}'$. Comme la famille $\tilde{\mathcal{P}}'$ est raffinante, il existe une partition $\mu' \in \tilde{\mathcal{P}}'$ plus fine que la partition (α, β), et par hypothèse, la famille $\tilde{\mathcal{P}}$ contient une partition μ plus fine que la partition μ', qui est de surcroît plus fine que la partition (α, β). La famille $\tilde{\mathcal{P}}$ est donc raffinante. □

2.4.1 Largeur arborescente (*tree-width*)

Décomposition classique. Donnons une définition alternative de la décomposition arborescente *classique*, plus proche de nos outils de partitionnement. Soit T un arbre partitionnant partiel sur l'ensemble d'arêtes du graphe $G = (V, E)$, et considérons un noeud interne u de T. En supprimant le noeud u, T est décomposé en sous-arbres $T_1, \ldots, T_p$. Pour chaque sous-arbre T_i, l'union des sous-ensembles d'arêtes associés aux feuilles de T_i forme un ensemble E_i. Notons $\mu_T(u) = (E_1, \ldots, E_p)$, et observons que $\mu_T(u)$ est une partition de $E(G)$.

Rappelons que l'*arête-frontière* (*edge-border*) d'une partition μ de $E(G)$, notée $\delta(\mu)$, est l'ensemble des sommets de G incidents à au moins deux arêtes contenues dans des parties distinctes de μ. Nous dirons qu'un arbre partitionnant partiel T sur $E(G)$ est de largeur arborescente au plus $k \geq 1$ si, pour tout noeud interne u, $\delta(\mu_T(u))$ est de cardinalité au plus $k + 1$. Par AMINI *et al.* [AMNT09], un graphe connexe et non réduit à une arête est de largeur arborescente au plus k, s'il existe un arbre partitionnant T *non partiel* de largeur arborescente au plus k. Sans entrer dans les détails de leur démonstration, nous pouvons remarquer qu'un arbre partitionnant T *non partiel* de largeur arborescente au plus k peut être facilement transformé en une décomposition arborescente, au sens de la définition *classique* (voir SECTION 1.3). La décomposition arborescente utilise le même arbre T, en mettant comme sac de chaque noeud interne u l'arête-frontière $\delta(\mu_T(u))$, et comme sac de chaque feuille v les deux sommets correspondants à l'arête associée à v.

Notons qu'AMINI *et al.* [AMNT09] ont montré que la famille de partitions $\hat{\mathcal{P}}'_{\text{tw}}$, correspondant aux arbres partitionnants partiels de largeur arborescente au plus k d'un graphe, est raffinante.

Nouvelle famille de partitions. Pour notre objectif de construction explicite d'une obstruction pour la largeur arborescente, nous devons définir une famille de partitions $\mathcal{P}_k^{\text{tw}}$ *de base*, telle que la famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$ soit raffinante, qu'un graphe soit de largeur arborescente au plus k si et seulement si une partition en singletons appartient à $\hat{\mathcal{P}}_k^{\text{tw}}$, et que $\mathcal{P}_k^{\text{tw}}$ soit de cardinalité *raisonnable*.

Nous pourrions être tentés d'utiliser, comme le font AMINI *et al.* [AMNT09], la famille des partitions de $E(G)$ pour lesquelles l'arête-frontière est de cardinalité au plus $k + 1$. Cependant, en considérant comme graphe G une étoile ayant $n - 1$ feuilles, et $k = 1$, toute partition des arêtes satisfait cette propriété, et la famille de partitions contient alors un nombre exponentiel de partitions *de base*.

C'est pourquoi nous définissons la famille de partitions $\mathcal{P}_k^{\text{tw}}$ comme étant l'ensemble des partitions μ de $E(G)$ telles que $|\delta(\mu)| \leq k + 1$, et les pétales de μ correspondent soit à une arête ayant ses deux extrémités dans $\delta(\mu)$, soit aux arêtes incidentes à une composante connexe de $G - \delta(\mu)$. Cette seconde condition nous permet d'obtenir une famille de partitions $\mathcal{P}_k^{\text{tw}}$ dont la cardinalité est de l'ordre de $\binom{n}{k+1} = \mathcal{O}(n^{k+1})$.

La famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$ est alors construite à l'aide de notre grammaire de recollement d'arbres sur $\mathcal{P}_k^{\text{tw}}$ (voir SECTION 2.3.1). Clairement, notre famille $\hat{\mathcal{P}}_k^{\text{tw}}$ est incluse dans cette famille $\hat{\mathcal{P}}'_{\text{tw}}$. En fait, les arbres $\mathcal{P}_k^{\text{tw}}$-partitionnants partiels, obtenus par notre grammaire, correspondent aux décompositions arborescentes dites « connexes », et il est bien connu que toute décomposition arborescente peut être transformée en une décomposition arborescente « connexe ».

Pour que la famille de partitions $\mathcal{P}_k^{\text{tw}}$ puisse être utilisée par notre algorithme CONSTRUCTION D'UNE OBSTRUCTION, afin de construire une $\hat{\mathcal{P}}_k^{\text{tw}}$-obstruction (c'est-à-dire un *bramble*) du graphe donné en entrée, il nous reste à montrer que la famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$ est raffinante. Nous allons pour cela utiliser le LEMME 2.4.1.

LEMME 2.4.2
La famille $\hat{\mathcal{P}}_k^{\text{tw}}$ est raffinante.

Démonstration. Considérons un arbre partitionnant partiel T' de largeur au plus k sur la famille de partitions *classique* $\mathcal{P}'_{\text{tw}}$, enraciné en un noeud r et affichant une partition $\mu' \in \mathcal{P}'_{\text{tw}}$, et supposons qu'il existe un noeud x de cet arbre, tel que le sous-arbre $T'(x)$ contient des feuilles n'induisant pas une composante connexe de G. De ce fait, la partition μ' n'appartient pas à notre famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$.

Nous allons transformer cet arbre partitionnant partiel T', sur la famille de partitions *classique* $\mathcal{P}_k^{\text{tw}}$, en un arbre $\mathcal{P}_k^{\text{tw}}$-partitionnant partiel T, de telle sorte que la partition μ affichée par notre arbre T appartienne à notre famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$, et soit plus fine que la partition $\mu' \in \mathcal{P}'_{\text{tw}}$ affichée par l'arbre T'. Par le LEMMA 2.4.1, nous en déduirons alors que notre famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$ est raffinante.

Sans perte de généralité, supposons que x est l'un des noeuds les plus proches de la racine r de T'. Soit $\nu' = (\alpha', A'_1, \ldots, A'_p)$ la partition correspondant au noeud x de T', où $A'_1 \cup \ldots \cup A'_p$ correspond à l'union $E'(x)$ des feuilles du sous-arbre $T'(x)$, et supposons que A'_p ne correspond ni à une arête ayant ses deux extrémités dans $\delta(\nu')$, ni aux arêtes incidentes à une composante connexe de $G - \delta(\nu')$. Comme T' est de largeur au plus k, nous avons $|\delta(\nu')| \leq k$.

Sans augmenter la largeur de l'arbre T', le pétale A'_p peut être découpé en plusieurs pétales $B_1, \ldots, B_q$, de sorte que $B_1 \cup \ldots \cup \mathcal{B}_q = A'_p$, et telles que chacun de ces pétales correspond soit à une arête ayant ses deux extrémités dans $\delta(\nu')$, soit aux arêtes incidentes à une composante connexe de $G - \delta(\nu')$. De plus, il existe une étoile $\mathcal{P}_k^{\text{tw}}$-partitionnante partielle, affichant une partition $(B_1, \ldots, B_q, \beta) \in \hat{\mathcal{P}}_k^{\text{tw}}$ telle que $\beta = A'_1 \cup \ldots \cup A'_{p-1}$. Nous pouvons alors construire un nouvel arbre partitionnant partiel T'', en remplaçant la feuille de l'arbre T' correspondant au pétale A'_p par cette étoile $\mathcal{P}_k^{\text{tw}}$-partitionnante partielle. Par construction, la partition $\mu'' = (A'_1, \ldots, A'_{p-1}, B_1, \ldots, B_q)$ affichée par ce nouvel arbre T'' est plus fine que la partition μ' affichée par l'arbre T'.

En procédant ainsi pour chaque pétale A'_i ayant les mêmes propriétés, nous obtenons un arbre $\mathcal{P}_k^{\text{tw}}$-partitionnant partiel T, affichant une partition μ plus fine que μ', de largeur $|\delta(\mu)| = |\delta(\mu')| \leq k$, et appartenant à notre famille $\hat{\mathcal{P}}_k^{\text{tw}}$. En répétant cette construction pour toutes les arbres partitionnants partiels affichant une partition de la famille $\hat{\mathcal{P}}'_{\text{tw}}$, et par le LEMME 2.4.1, nous obtenons l'ensemble des éléments de la famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$, et en déduisons que notre famille $\hat{\mathcal{P}}_k^{\text{tw}}$ est raffinante. □

Le THÉORÈME 2.3.1 peut donc s'appliquer à notre famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{tw}}$ pour la largeur arborescente, et notre algorithme CONSTRUCTION D'UNE OBSTRUCTION construira une $\hat{\mathcal{P}}_k^{\text{tw}}$-obstruction du graphe donné en entrée si et seulement s'il n'admet pas d'arbre $\mathcal{P}_k^{\text{tw}}$-partitionnant.

Obstruction classique. La notion usuelle d'obstruction à la largeur arborescente est appelée « *bramble* ». Elle a été initialement introduite sous deux notions différentes mais équivalentes, appelées « écran » (*screen*) et « havre » (*haven*), par SEYMOUR et THOMAS [ST93]. Son nom actuel a été donné par REED [Ree97].

Un bramble d'ordre k, dans un graphe $G = (V, E)$, est un ensemble $\mathcal{B}$ de sous-ensembles de $V(G)$ tel que :

- chaque élément de $\mathcal{B}$ induit un sous-graphe connexe de G ;
- pour toute paire d'éléments de $\mathcal{B}$, ces deux éléments se touchent, c'est-à-dire que leur union induit un sous-graphe connexe de G ;
- il existe un « ensemble intersectant » d'éléments de $\mathcal{B}$ de cardinalité au plus k, c'est-à-dire un sous-ensemble de sommets $C \subseteq V(G)$ de taille au plus k tel que chaque élément de $\mathcal{B}$ contient au moins un sommet de C.

Construction d'une obstruction. Nous pouvons obtenir un *bramble* du graphe G, à partir de la $\hat{\mathcal{P}}_k^{\mathrm{tw}}$-obstruction construite par notre algorithme. Pour cela, à partir de la $\hat{\mathcal{P}}_k^{\mathrm{tw}}$-obstruction $\mathcal{B}$ obtenue, et pour chaque pétale $F \in \mathcal{B}$, nous mettons dans le *bramble* l'ensemble des sommets de G incidents uniquement aux arêtes contenues dans F.

Pour un graphe G donné, si la famille étendue de partitions $\hat{\mathcal{P}}_k^{\mathrm{tw}}$ définie plus haut ne contient pas la partition de $E(G)$ en singletons, alors notre algorithme CONSTRUCTION D'UNE OBSTRUCTION trouvera une $\hat{\mathcal{P}}_k^{\mathrm{tw}}$-obstruction de G en temps $\mathcal{O}\big(|\mathcal{P}_k^{\mathrm{tw}}|^2 \cdot \mathrm{poly}(n)\big)$. Comme notre famille de partitions $\mathcal{P}_k^{\mathrm{tw}}$ est de cardinalité au plus $\mathcal{O}(n^{k+1})$, nous obtenons un temps d'exécution de $\mathcal{O}(n^{2k+2} \cdot \mathrm{poly}(n))$.

Nous pouvons cependant améliorer l'estimation du temps d'exécution de notre algorithme CONSTRUCTION D'UNE OBSTRUCTION, en étudiant plus en finesse la structure de la famille de partitions $\mathcal{P}_k^{\mathrm{tw}}$.

LEMME 2.4.3
L'algorithme CONSTRUCTION D'UNE OBSTRUCTION *peut être modifié pour construire une $\hat{\mathcal{P}}_k^{\mathrm{tw}}$-obstruction pour la largeur arborescente en temps $\mathcal{O}(n^{k+4})$.*

Démonstration. Nous donnons ici les grandes lignes de la démonstration, et utilisons notamment certaines idées de ARNBORG *et al.* [ACP87] sur les décompositions arborescentes.

Soit $G = (V, E)$ un graphe. Étant donné un arbre $\mathcal{P}_k^{\mathrm{tw}}$-partitionnant partiel T de G, enraciné en r, et un noeud interne i de T, nous notons $E_T(i)$ le sous-ensemble formé par l'union des pétales associées aux feuilles du sous-arbre $T(i)$ enraciné en i. L'arbre T est alors transformé en un nouvel arbre $\mathcal{P}_k^{\mathrm{tw}}$-partitionnant partiel T' enraciné en r', tel que :

- les racines r et r' des deux arbres T et T' correspondent à la même partition de $\mathcal{P}$;
- pour chaque noeud interne i de T, différent de la racine r, il existe un noeud interne i' de T' tel que $E_T(i) = E_{T'}(i')$, et tel que les pétales associés aux feuilles de $T(i)$ sont exactement les pétales associés aux feuilles de $T'(i')$;
- pour chaque noeud interne j de T, différent de la racine r, dont μ_j est la partition correspondante, l'arête-frontière de la partition μ_j dans T' est exactement l'arête-frontière de la partition $\big(E_{T'}(j), E(G) \setminus E_{T'}(j)\big)$ dans T' complétée d'un sommet supplémentaire.

De ce fait, dans notre algorithme MISE À JOUR DES MARQUAGES, il nous suffit de considérer les « bons couples » de la forme $\big(E_{T'}(j), E(G) \setminus E_{T'}(j)\big)$, ce qui signifie que nous pouvons nous restreindre aux « bons couples » (F, μ) tels que $\delta(\mu)$ correspond à $\delta\big((F, E(G) \setminus F)\big)$ complété d'un sommet supplémentaire. De plus, nous pouvons supposer que le pétale F n'est pas un singleton (c'est-à-dire n'est pas un pétale de cardinalité 1), puisque ces singletons sont retirés dans une première étape de notre algorithme CONSTRUCTION D'UNE OBSTRUCTION.

Nous allons montrer que le nombre de ces pétales *utiles* est d'au plus $\mathcal{O}(n^{k+2})$, et que calculer la liste de ces pétales peut être effectuée en temps $\mathcal{O}(n^{k+3})$. Le nombre d'éléments de la famille de partitions $\mathcal{P}_k^{\text{tw}}$ est d'au plus $\mathcal{O}(n^{k+1})$, puisque nous considérons uniquement les partitions de la forme $\mu = (F_1, \ldots, F_p)$ où $|\delta(\mu)| \leq k+1$, par définition de la famille $\mathcal{P}_k^{\text{tw}}$. Pour chaque partition μ, le nombre maximum de pétales n'étant pas un singleton est au plus n, donc le nombre total de pétales *utiles* est d'au plus $n \cdot \mathcal{O}(n^{k+1}) = \mathcal{O}(n^{k+2})$. Et pour chaque pétale F n'étant pas un singleton, nous associons au plus n « bons couples » de la forme $\delta\big((F, E(G) \setminus F)\big)$ complété d'un sommet supplémentaire, ce qui nous donne au plus $\mathcal{O}(n^{k+3})$ « bons couples ». La liste de ces « bons couples » peut donc être calculée en temps $\mathcal{O}(n^{k+3})$.

Enfin, pour chaque « bon couple » (F, μ), nous associons au plus m pétales de μ contenus dans F, que nous pouvons réduire à au plus n pétales puisque nous ne conservons que les pétales n'étant pas des singletons. Ce traitement effectué pour la mise à jour des marquages nous donne finalement un temps d'exécution global de $\mathcal{O}(n^{k+4})$.

Enfin, pour que notre algorithme MISE À JOUR DES MARQUAGES puisse s'exécuter dans le même temps, nous devons nous assurer que le graphe $\mathcal{G}^{\text{tw}}_{petales}$ peut être également construit en temps $\mathcal{O}(n^{k+4})$. Là encore, en réutilisant certaines idées de ARNBORG *et al.* [ACP87], il peut être montré que pour tout pétales F et F' tels que $F' \subsetneq F$, il existe un autre pétale F'' tel que $F' \subseteq F'' \subseteq F$, et que le pétale F est associé à une partition μ où $\delta(\mu)$ correspond à $\delta\big((F, E(G) \setminus F)\big)$ complété d'un sommet supplémentaire. De ce fait, il suffit de n'ajouter au graphe $\mathcal{G}^{\text{tw}}_{petales}$ que les arcs reliant les pétales F et F'' et respectant cette propriété. Ainsi, pour un pétale F, nous considérons au plus n partitions μ, chacune d'elles nous donnant au plus n pétales F', et donc le pétale F a au plus n^2 arcs entrants dans le graphe $\mathcal{G}^{\text{tw}}_{petales}$. La taille globale du graphe $\mathcal{G}^{\text{tw}}_{petales}$ est donc au plus $\mathcal{O}(n^{k+3})$, et peut être construit en temps $\mathcal{O}(n^{k+4})$. □

Classes de graphes avec un nombre polynomial de séparateurs minimaux. Grâce à la structure de la famille de partitions $\mathcal{P}_k^{\text{tw}}$, nous avons pu obtenir un temps d'exécution linéaire en la taille de cette famille. Toutefois, le facteur exponentiel dans le temps d'exécution de notre algorithme CONSTRUCTION D'UNE OBSTRUCTION dépend de la largeur arborescente du graphe donné en entrée.

Lorsque notre algorithme CONSTRUCTION D'UNE OBSTRUCTION est appliqué à une classe de graphes ayant un nombre polynomial de séparateurs minimaux, il est possible de s'affranchir de cette dépendance en se concentrant sur la construction d'une « obstruction compacte » (*compact bramble*), et en ne conservant que les partitions μ pour lesquelles $\delta(\mu)$ est une « clique maximale potentielle ». Nous obtenons alors un temps d'exécution réellement polynomial de $\mathcal{O}(n^4 r^2)$, ne dépendant pas de k, où r est le nombre de séparateurs minimaux du graphe donné en entrée (voir [BT01, FKTV08]).

Pour plus de détails sur ce résultat, nous invitons le lecteur à se référer à la version longue de notre article [CMT09].

2.4.2 Largeur de branches (*branch-width*)

Décomposition classique. La notion classique de décomposition en branches (*branch-decomposition*) de largeur k d'un graphe $G = (V, E)$ est définie comme étant un arbre T *ternaire*, tel que les feuilles de T sont en bijection avec les arêtes de G, et pour toute branche $e \in T$, la partition $(A, \bar{A})$ de E engendrée par la suppression de la branche e est telle que $|\delta\big((A, \bar{A})\big)| \leq k$.

En réutilisant notre terminologie, les décompositions en branches *partielles* de E de largeur au plus k, où les feuilles de la décomposition peuvent être associées à des sous-ensembles d'arêtes, correspondent à des arbres $\mathcal{P}'_{\text{bw}}$-partitionnants partiels qui sont dits de largeur au plus k, et la famille *classique* de partitions $\hat{\mathcal{P}}'_{\text{bw}}$ pour la largeur de branches est composée de l'ensemble des partitions affichées par ces arbres $\mathcal{P}'_{\text{bw}}$-partitionnants partiels. Toutefois, à l'instar de la famille engendrée par les décompositions arborescentes *classiques*, les pétales des partitions μ de $\hat{\mathcal{P}}'_{\text{bw}}$ ne sont pas forcément des composantes connexes de $G - \delta(\mu)$, et la cardinalité de cette famille est donc potentiellement trop grande pour notre algorithme.

Nouvelle famille de partitions. Pour y remédier, nous allons définir une nouvelle famille de partitions pour la largeur de branches. Remarquons que FOMIN *et al.* [FMT05] ont introduit indépendamment une notion très similaire appelée « bloc », et que nous aurions pu également utiliser ici.

Soit $G = (V, E)$ un graphe. Une *k-troïka* d'un sous-ensemble $X \subseteq V(G)$ est un triplet (A, B, C) de sous-ensembles de $V(G)$, où $|A| \leq k$, $|B| \leq k$, $|C| \leq k$, et tels que $A \cup B = B \cup C = C \cup A = X$. Une partition $\mu = (E_1, \ldots, E_p)$ de $E(G)$ est *compatible* avec une k-troïka (A, B, C) si $\delta(\mu) = X$, et pour chaque élément E_i de la partition, $\delta\big((E_i, \bar{E}_i)\big)$ appartient à l'un des trois sous-ensembles A, B ou C. En d'autres termes, chaque pétale de la partition μ est rattaché à l'un des trois éléments A, B ou C de la k-troïka.

La famille $\mathcal{P}_k^{\text{bw}}$ est définie comme étant l'ensemble des partitions $\mu = (E_1, E_2, \ldots, E_p)$ de $E(G)$ compatibles avec des k-troïkas de $V(G)$, et telles que chaque élément E_i correspond soit à une arête ayant ses deux extrémités dans $\delta(\mu)$, soit aux arêtes incidentes à une composante connexe de $G - \delta(\mu)$. La famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{bw}}$ est alors construite à l'aide de notre grammaire de recollement d'arbres sur $\mathcal{P}_k^{\text{bw}}$ (voir SECTION 2.3.1), et correspond aux arbres $\mathcal{P}_k^{\text{bw}}$-partitionnants partiels de largeur au plus k.

Il est important, pour notre algorithme, que la famille de partitions $\mathcal{P}_k^{\text{bw}}$ soit de taille *raisonnable*. Nous allons donc décrire la construction de cette famille, afin d'en déduire sa cardinalité.

Observation ([PT05]). Si une partition μ est compatible avec une k-troïka de G, alors $|\delta(\mu)| \leq \lfloor 3k/2 \rfloor$.

Tout d'abord, nous énumérons tous les sous-ensembles connexes de $V(G)$ de taille au plus $\lfloor 3k/2 \rfloor$, en temps $\mathcal{O}(n^{\lfloor 3k/2 \rfloor})$. Pour chaque sous-ensemble connexe $X \subseteq V(G)$, nous ajoutons à $\mathcal{P}_k^{\text{bw}}$ toute partition non triviale (c'est-à-dire contenant certains pétales de cardinalité au moins 2) résultant d'une k-troïka (A, B, C) de X respectant les conditions définies ci-dessus pour la famille $\mathcal{P}_k^{\text{bw}}$. La famille de partitions $\mathcal{P}_k^{\text{bw}}$ obtenue contient alors au plus $\mathcal{O}(3^{\lfloor 3k/2 \rfloor} \cdot n^{\lfloor 3k/2 \rfloor})$ partitions, chacune d'entre elles contenant au plus n pétales non triviaux.

Remarquons que notre algorithme générique doit être légèrement adapté pour cette famille de partitions, en ne considérant que le recollement d'arbres (et donc de partitions de $\mathcal{P}_k^{\mathrm{bw}}$) compatibles avec des k-troïkas.

Pour que la famille de partitions $\mathcal{P}_k^{\mathrm{bw}}$ puisse être utilisée par notre algorithme CONSTRUCTION D'UNE OBSTRUCTION, afin de construire une $\mathcal{P}_k^{\mathrm{bw}}$-obstruction (c'est-à-dire un enchevêtrement) du graphe donné en entrée, il nous reste à montrer que la famille étendue de partitions $\hat{\mathcal{P}}_k^{\mathrm{bw}}$ est raffinante.

LYAUDET *et al.* [LMT10] ont montré que la famille *classique* de partitions $\hat{\mathcal{P}}'_{\mathrm{bw}}$ pour la largeur de branches est raffinante. Par ailleurs, toute décomposition en branches T peut être transformée en une décomposition en branches T' *connexe* (voir notamment [PT05, FMT05, BFF+10]), c'est-à-dire que les feuilles de T' correspondent à des composantes connexes, la partition affichée par T' est plus fine que la partition affichée par T, et pour tout noeud interne u' de T' correspondant à une partition μ', $\delta(\mu')$ correspond à une troïka. Les arbres $\mathcal{P}_k^{\mathrm{bw}}$-partitionnants partiels, que nous obtenons à l'aide des k-troïkas, correspondent à ces décompositions *connexes*. Donc pour toute partition μ' de la famille *classique* $\hat{\mathcal{P}}'_{\mathrm{bw}}$, notre famille $\hat{\mathcal{P}}_k^{\mathrm{bw}}$ contient une partition plus fine que μ'. De ce fait, comme la famille *classique* $\hat{\mathcal{P}}'_{\mathrm{bw}}$ est raffinante, et par le LEMME 2.4.1, nous en déduisons que notre famille $\hat{\mathcal{P}}_k^{\mathrm{bw}}$ pour la largeur des branches est raffinante.

Le THÉORÈME 2.3.1 peut donc s'appliquer à notre famille étendue de partitions $\hat{\mathcal{P}}_k^{\mathrm{bw}}$ pour la largeur de branches, et notre algorithme CONSTRUCTION D'UNE OBSTRUCTION construira une $\hat{\mathcal{P}}_k^{\mathrm{bw}}$-obstruction du graphe donné en entrée si et seulement s'il n'admet pas d'arbre $\mathcal{P}_k^{\mathrm{bw}}$-partitionnant (non partiel).

Obstruction classique. La notion usuelle d'obstruction à la largeur de branches est appelée « enchevêtrement » (*tangle*). Elle a été initialement introduite par ROBERTSON et SEYMOUR [RS91].

Un *enchevêtrement* (*tangle*) d'ordre k, pour une fonction de partitionnement Ψ dans un graphe $G = (V, E)$, est un ensemble $\mathcal{T}$ de sous-ensembles de $E(G)$ tel que :

- pour toute partition $(A, \bar{A})$ de $E(G)$ telle que $\Psi_{\mathrm{bw}}\big((A, \bar{A})\big) \leq k$, soit $A \in \mathcal{T}$, soit $\bar{A} \in \mathcal{T}$;
- si $A, B, C \in \mathcal{T}$, alors $A \cup B \cup C \neq E(G)$;
- pour toute arête $e \in E(G)$, $E(G) \setminus \{e\} \notin \mathcal{T}$.

Construction d'une obstruction. En utilisant notre algorithme CONSTRUCTION D'UNE OBSTRUCTION appliqué à la famille de partitions $\mathcal{P}_k^{\mathrm{bw}}$, nous pouvons obtenir une $\hat{\mathcal{P}}_k^{\mathrm{bw}}$-obstruction $\mathcal{B}$ du graphe donné en entrée en temps $\mathcal{O}\big(|\mathcal{P}_k^{\mathrm{bw}}|^2 \cdot \mathrm{poly}(n)\big)$. Comme notre famille de partitions $\mathcal{P}_k^{\mathrm{bw}}$ est de cardinalité au plus $\mathcal{O}(n^{\lfloor 3k/2 \rfloor})$, nous obtenons un temps d'exécution de $\mathcal{O}\big(n^{2 \cdot \lfloor 3k/2 \rfloor} \cdot \mathrm{poly}(n)\big) = \mathcal{O}^*(n^{3k})$.

Rappelons que la famille de partitions $\mathcal{P}_k^{\mathrm{bw}}$ est construite à partir de l'ensemble des arêtes $E(G)$ d'un graphe G. À partir de la $\hat{\mathcal{P}}_k^{\mathrm{bw}}$-obstruction obtenue, nous pouvons construire un enchevêtrement $\mathcal{T}$ d'ordre k du graphe G comme suit : $\mathcal{T} = \{\bar{A} \mid A \in \mathcal{B}\}$. Cela implique que le graphe G considéré est de largeur de branches au moins k. Nous pouvons remarquer que l'enchevêtrement que nous construisons est *compact*, c'est-à-dire de taille réduite par rapport à un enchevêtrement classique, car nous ne considérons que certaines sous-partitions provenant de décompositions partielles *connexes*.

Notons que HICKS [Hic04] a donné un algorithme spécifique pour la largeur de branches, construisant un enchevêtrement en temps $\mathcal{O}(m \cdot n^{2\mathrm{bw}-2})$, légèrement plus rapide que notre algorithme (mais toujours XP) en tirant partie de propriétés structurelles spécifiques de la largeur de branches.

2.4.3 Largeur de découpe (*carving-width*)

Décomposition classique. Classiquement, la notion de largeur de découpe est définie en terme de « découpage » (*carving*). Deux sous-ensembles de sommets $A, B \subseteq V(G)$ d'un graphe $G = (V, E)$ se *croisent* si $A \cap B$, $A - B$, $B - A$ et $V(G) - (A \cup B)$ sont tous non vides.

DÉFINITION 2.4.1 (découpage)
Un découpage *de l'ensemble $V(G)$ de sommets d'un graphe G est un ensemble $\mathcal{C}$ de sous-ensembles de V tel que :*

1. *$\emptyset, V(G) \notin \mathcal{C}$;*
2. *les éléments de $\mathcal{C}$ ne se croisent pas ;*
3. *$\mathcal{C}$ est maximal sous les deux conditions précédentes.*

En fait, un tel découpage peut être vu comme provenant d'une décomposition de l'ensemble $V(G)$, ce qui nous donne une notion de décomposition pour la largeur de découpe [ST94]. Prenons un arbre T *ternaire*, dans lequel les feuilles sont en bijection avec les sommets de G, et pour chaque branche $e \in T$, notons $T_1(e)$ et $T_2(e)$ les deux sous-arbres obtenus en supprimant la branche e dans l'arbre T, et $\mu_i(e) = \big(V_1(e), V_2(e)\big)$ la partition de $V(G)$ obtenue (où $V_2(e) = V(G) \setminus V_1(e)$). Alors l'ensemble $\mathcal{C}$ des partitions qu'il est possible d'obtenir en supprimant une branche de l'arbre T est un découpage de l'ensemble $V(G)$ [ST94]. Remarquons que cette notion de décomposition pour la largeur de découpe est très similaire à la décomposition en branches, en échangeant en quelque sorte le rôle des sommets et des arêtes.

La *largeur* d'une telle décomposition du graphe G est définie à partir de la notion de *sommet-frontière* (*vertex-border*) d'une partition μ, notée $\partial(\mu)$, qui contient l'ensemble des arêtes de G incidentes à deux sommets contenus dans des parties distinctes de la partition μ. Une décomposition de découpe est de largeur au plus k si pour toute partition μ de cette décomposition, $|\partial(\mu)| \leq k$.

Dans le cadre de notre terminologie, une *décomposition de découpe partielle* de $V(G)$ de largeur au plus k, où les feuilles peuvent correspondre à des sous-ensembles de sommets, correspondent à des arbres $\mathcal{P}'_{\mathrm{carw}}$-partitionnants partiels qui sont dits de largeur au plus k, et la famille *classique* de partitions $\hat{\mathcal{P}}'_{\mathrm{carw}}$ pour la largeur de découpe est composée de l'ensemble des partitions affichées par ces arbres $\mathcal{P}'_{\mathrm{carw}}$-partitionnants partiels. Là encore, cette famille est potentiellement de trop grande cardinalité pour notre algorithme.

Nouvelle famille de partitions. Nous définissons une nouvelle famille de partitions pour la largeur de découpe de manière similaire à la famille pour la largeur de branches, en introduisant la notion de « k-arête-troïka ». Soit $G = (V, E)$ un graphe. Une *k-arête-troïka* d'un sous-ensemble $X \subseteq E(G)$ est un triplet (A, B, C) de sous-ensembles de

$E(G)$, où $|A| \leq k$, $|B| \leq k$, $|C| \leq k$, et tels que $A \cup B = B \cup C = C \cup A = X$. Une partition $\mu = (V_1, \ldots, V_p)$ de $V(G)$ est *compatible* avec une k-arête-troïka (A, B, C) si $\delta(\mu) = X$, et pour chaque élément V_i de la partition, $V_i \cap X$ appartient à l'un des trois sous-ensembles A, B ou C.

La famille $\mathcal{P}_k^{\text{carw}}$ est définie comme étant l'ensemble des partitions $\mu = (V_1, V_2, \ldots, V_p)$ de $V(G)$ compatibles avec des k-arêtes-troïkas de $E(G)$, et telles que chaque élément V_i correspond à une composante connexe de $G - \partial(\mu)$. La famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{carw}}$ est quant à elle construite à l'aide de notre grammaire de recollement d'arbres sur $\mathcal{P}_k^{\text{carw}}$ (voir SECTION 2.3.1), et correspond aux arbres $\mathcal{P}_k^{\text{carw}}$-partitionnants partiels de largeur au plus k.

Par des arguments similaires à ceux utilisés dans le cas de la famille de partitions $\mathcal{P}_k^{\text{bw}}$ pour la largeur de branches, et en utilisant notamment des résultats de SEYMOUR et THOMAS [ST94] relatifs à la connexité d'un découpage, nous pouvons montrer que la famille $\mathcal{P}_k^{\text{carw}}$ est de taille *raisonnable*, et que la famille étendue de partitions $\hat{\mathcal{P}}_k^{\text{carw}}$ est raffinante.

Nous pouvons donc utiliser notre algorithme CONSTRUCTION D'UNE OBSTRUCTION pour construire $\hat{\mathcal{P}}_k^{\text{carw}}$-obstruction du graphe donné en entrée si et seulement s'il n'admet pas d'arbre $\mathcal{P}_k^{\text{carw}}$-partitionnant (non partiel).

Obstruction classique. La notion usuelle d'obstruction à la largeur de découpe est appelée « inclinaison » (*tilt*). Elle a été introduite par SEYMOUR et THOMAS [ST94].

Une *inclinaison* (*tilt*) d'ordre k, pour une fonction de partitionnement Ψ dans un graphe $G = (V, E)$, est un ensemble $\mathcal{T}$ de sous-ensembles de $V(G)$ tel que :

- pour toute partition $(A, \bar{A})$ de $V(G)$ tel que $\Psi\big((A, \bar{A})\big)$, soit $A \in \mathcal{B}$, soit $\bar{A} \in \mathcal{T}$;
- si $A, B, C \in \mathcal{T}$, alors $A \cup B \cup C \neq V(G)$;
- pour tout $v \in V(G)$, $\{v\} \in \mathcal{T}$.

Construction d'une obstruction. En utilisant notre algorithme CONSTRUCTION D'UNE OBSTRUCTION appliqué à la famille de partitions $\mathcal{P}_k^{\text{carw}}$, nous pouvons obtenir une $\hat{\mathcal{P}}_k^{\text{carw}}$-obstruction $\mathcal{B}$ du graphe donné en entrée en temps $\mathcal{O}\big(|\mathcal{P}_k^{\text{carw}}|^2 \cdot \text{poly}(n)\big)$. À partir de cette $\hat{\mathcal{P}}_k^{\text{carw}}$-obstruction, nous pouvons construire une inclinaison $\mathcal{T}$ d'ordre k comme suit : $\mathcal{T} = \bigcup_{v \in V(G)} \{v\} \cup \{\bar{A} \mid A \in \mathcal{B}\}$. Cela implique que le graphe donné en entrée est de largeur de découpe au moins k.

2.4.4 Discussion sur cette méthode, et limites

Construction d'un arbre $\mathcal{P}$-partitionnant. Pour une famille de partitions $\mathcal{P}$ d'une largeur de graphes donnée, nous avons obtenu un algorithme construisant une $\hat{\mathcal{P}}$-obstruction pour diverses largeurs de graphes, et permettant ainsi de certifier qu'un graphe est de grande largeur.

Toutefois, rien ne nous assure que le graphe donné en entrée est effectivement de grande largeur. S'il est de petite largeur, une $\hat{\mathcal{P}}$-obstruction ne peut pas être obtenue puisque le graphe peut alors être décomposé entièrement, c'est-à-dire qu'il existe un arbre $\mathcal{P}$-partitionnant *non partiel* pour ce graphe.

Nous pouvons aisément adapter notre algorithme pour construire un tel arbre $\mathcal{P}$-partitionnant, lorsqu'il existe. En effet, s'il existe, alors la famille $\mathcal{P}$ contient une partition en singletons de l'univers $\mathcal{U}$, et tous les pétales de cette partition seront marqués *interdits* par le premier appel de notre processus de marquage. Il existe alors un arbre $\mathcal{P}$-partitionnant dont la racine correspond à cette partition en singletons. En retraçant l'ordre dans lequel ces marquages sont effectués, et la façon dont ils sont décomposables, nous pouvons reconstruire cet arbre $\mathcal{P}$-partitionnant permettant d'obtenir la partition en singletons, et obtenir ainsi un certificat montrant que le graphe est de petite largeur.

Nécessité d'une famille de partitions de cardinalité $\mathcal{O}\big(n^{f(k)}\big)$. Une traduction naïve du « théorème de dualité généralisée » nous donne un algorithme dont la complexité dépend de la famille étendue de partitions $\hat{\mathcal{P}}$. Toutefois, cette famille peut être de cardinalité exponentielle en la cardinalité de la famille de partitions $\mathcal{P}$.

En utilisant un processus de marquage, nous sommes parvenus à rendre la complexité de notre algorithme dépendante uniquement de cette famille de partitions $\mathcal{P}$ *de base*, et avons obtenu un algorithme en temps $\mathcal{O}^*(|\mathcal{P}|^2)$. Cette amélioration est importante, car elle nous permet notamment d'obtenir un algorithme en temps $\mathcal{O}(n^{\mathrm{tw}+4})$ pour la largeur arborescente.

Toutefois, pour que notre algorithme appartienne effectivement à la classe XP, la famille de partitions $\mathcal{P}$ doit être de cardinalité raisonnable, c'est-à-dire de l'ordre de $\mathcal{O}(n^k)$. Dans le cas contraire, notre algorithme ne nous permet pas d'obtenir en temps XP une $\hat{\mathcal{P}}$-obstruction.

C'est le cas par exemple de la « largeur de rang » (*rank-width*), introduite par OUM et SEYMOUR [Oum05, OS06], qui a les bonnes propriétés de « sous-modularité » (voir [LMT10]), et dispose donc d'une notion combinatoire d'obstruction. Cependant, nous n'avons pas pu trouver de famille de partitions *de base*, autre que la famille naturelle dont la cardinalité est au moins de l'ordre de $\mathcal{O}\big(2^n\big)$ (même pour $k = 1$).

Nécessité d'une famille étendue de partitions $\hat{\mathcal{P}}$ raffinante. La validité de notre algorithme CONSTRUCTION D'UNE OBSTRUCTION repose principalement sur le théorème de dualité généralisée montré par LYAUDET *et al.* [LMT10], montrant une dualité entre arbre $\mathcal{P}$-partitionnant et $\hat{\mathcal{P}}$-obstruction.

Toutefois, pour que le théorème puisse s'appliquer, la famille étendue de partitions que nous considérons doit être raffinante. De ce fait, nous ne pouvons pas utiliser notre algorithme pour construire une obstruction à une largeur de graphes, lorsque la famille étendue de partitions n'est pas raffinante.

C'est le cas par exemple de la « largeur booléenne » (*boolean-width*), introduite récemment par BUI-XUAN *et al.* [BXTV09, BXTV11]. En réutilisant le formalisme de LYAUDET *et al.* [LMT10], nous utiliserions la *fonction de partitionnement* de la largeur booléenne pour construire une famille de partitions $\mathcal{P}$. Or, cette fonction n'étant pas *faiblement sous-modulaire* [Kim09], la famille étendue de partitions obtenue ne sera pas raffinante (voir [LMT10] pour les raisons de cette implication), et notre algorithme ne pourra donc pas s'appliquer à cette famille.

2.5 Conclusion

Se restreindre aux graphes de petite largeur permet, dans de nombreux cas, de résoudre efficacement des problèmes étant NP-difficiles pour des graphes quelconques. Il est dès lors important de pouvoir déterminer si un graphe est de petite largeur, ou certifier que celui-ci est de grande largeur.

La mise en évidence d'une obstruction à une largeur dans un graphe est l'une des principales méthodes permettant de certifier que ce graphe est de grande largeur. Toutefois, il n'existait jusqu'à ce jour aucun algorithme pour construire une telle obstruction.

Dans ce chapitre, nous avons défini un algorithme s'exécutant en temps XP, pour toute famille de partitions respectant certaines propriétés, et unifiant la construction explicite d'obstructions pour différentes largeurs de graphes connues. En particulier, nous avons obtenu le premier algorithme construisant de manière exacte un « bramble », obstruction à la largeur arborescente, en temps $\mathcal{O}(n^{\mathrm{tw}+4})$. Notre algorithme permet également d'obtenir une obstruction à la largeur de branches, en temps $\mathcal{O}^*(n^{3\mathrm{bw}})$. Dans ce dernier cas, il existait déjà un algorithmede HICKS [Hic04], construisant (notamment) une obstruction à la largeur de branches en temps $\mathcal{O}(m \cdot n^{2\mathrm{bw}-2})$ (donc légèrement plus rapide que notre algorithme, mais tout de même XP), en utilisant certaines propriétés structurelles spécifiques de la largeur de branches.

Même si nous avons pu éviter l'explosion combinatoire qui découlerait de l'utilisation de familles de partitions quelconques, les familles que nous considérons restent de grande cardinalité, et cette cardinalité semble ne pas pouvoir être réduite davantage pour obtenir un meilleur temps d'exécution pour la construction des obstructions correspondantes. Les obstructions aux différentes largeurs de graphes ne sont cependant pas uniques, et il est possible de considérer, pour certaines largeurs de graphes, des obstructions de plus petite taille.

Citons en particulier la notion de « mineur pseudo-grille » (*grid-like minor*), introduite récemment par REED et WOOD [RW08]. À l'aide de cette notion, REED et WOOD [RW08] ont montré qu'un graphe de largeur arborescente au moins $c \cdot k^4 \cdot \sqrt{\log k}$ contient une pseudo-grille d'ordre k comme mineur, et KREUTZER et TAZARI [KT10] ont donné un algorithme FPT (paramétré par l'ordre de la pseudo-grille) permettant de construire un mineur pseudo-grille dans un graphe quelconque.

Il est donc d'importance de trouver des notions d'obstructions efficaces pour les différentes largeurs de graphes étudiés, permettant de certifier qu'un graphe est de grande largeur.

Enfin, rappelons que l'approximation à facteur constant et en temps polynomial de la largeur arborescente est encore à l'heure actuelle un problème ouvert, la meilleure approximation étant à facteur $\mathcal{O}(\mathrm{tw}\sqrt{\log \mathrm{tw}})$ par un résultat de FEIGE *et al.* [FHL08]. Naturellement, le problème de déterminer la largeur arborescente exacte d'un graphe étant un problème NP-complet, il semble difficile (voire impossible) de trouver un algorithme permettant de construire en temps polynomial une obstruction d'ordre optimale, c'est-à-dire de cardinalité minimale (ou plus précisément, dont l'*ensemble tranversal* est minimal), puisque par le « théorème de dualité » de SEYMOUR et THOMAS [ST94], celle-ci permettrait d'en déduire directement la largeur arborescente exacte du graphe.

L'étude des obstructions à la largeur arborescente reste tout de même une piste intéressante, et notre algorithme est un résultat supplémentaire en ce sens.

Problèmes de domination 3

Sommaire

3.1 Introduction

Il est généralement intéressant d'obtenir une généralisation de notions partagées par de nombreux problèmes, notamment en théorie des graphes. Dans ce chapitre, nous allons nous intéresser à une généralisation des problèmes de domination, introduite par TELLE et PROSKUROSWKI [Tel94a, Tel94b, TP97].

Nous étudions dans ce chapitre la complexité de cette généralisation dans le cas de graphes de largeur arborescente bornée.

Les résultats de ce chapitre sont le fruit effectués seuls, et font l'objet d'un article actuellement soumis. Je tiens à remercier DIETER KRATSCH, MATHIEU LIEDLOFF, IOAN TODINCA, ainsi que plusieurs rapporteurs anonymes, pour leurs commentaires, remarques, corrections et suggestions sur diverses parties de ce travail.

3.1.1 Quelques problèmes de domination

En théorie des graphes, il existe de nombreux problèmes de domination, tels que les classiques ENSEMBLE DOMINANT ou CODE PARFAIT. Tous ces problèmes ont pour caractéristique commune la recherche d'un sous-ensemble de sommets d'un graphe G, tel que le voisinage de tout sommet de G respecte certaines contraintes par rapport aux éléments du sous-ensemble trouvé.

À l'origine, chacun de ces problèmes a été étudié indépendamment, et l'introduction d'un nouveau problème de ce type impliquait d'effectuer de nouveaux travaux pour en déterminer la complexité, écrire des algorithmes pour le résoudre,... Partant de ce constat, et dans le but de généraliser cette notion, TELLE et PROSKUROWSKI [Tel94a, Tel94b, TP97] ont introduit le problème de DOMINATION GÉNÉRALISÉE, aussi appelé ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Étant donné deux ensembles fixés d'entiers naturels $\sigma, \rho \in \mathbb{N}$, un *ensemble* $[\sigma, \rho]$-*dominant* dans un graphe $G = (V, E)$ est un sous-ensemble $S \subseteq V(G)$ de sommets de G tel que pour tout sommet $v \in S$, $|N(v) \cap S| \in \sigma$, et pour tout sommet $v \notin S$, $|N(v) \cap S| \in \rho$.

Ainsi, les deux ensembles σ et ρ sont en quelque sorte des contraintes imposées sur les sommets du graphe G : l'ensemble σ contraint le voisinage des sommets qui sont dans l'ensemble $[\sigma, \rho]$-dominant, tandis que l'ensemble ρ contraint le voisinage des sommets qui ne sont pas dans l'ensemble $[\sigma, \rho]$-dominant.

Lorsque le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est étudié comme un problème de décision, c'est-à-dire que nous souhaitons seulement à déterminer l'existence d'un ensemble $[\sigma, \rho]$-dominant existe dans un graphe, alors il est généralement supposé que $0 \notin \rho$. En effet, si $0 \in \rho$, alors l'ensemble vide $S = \emptyset$ est un ensemble $[\sigma, \rho]$-dominant *trivial*, puisque pour tout sommet v du graphe, $v \notin S$ et $|N(v) \cap S| = 0 \in \rho$.

La notion d'ensemble $[\sigma, \rho]$-dominant a été introduite afin de généraliser de nombreux problèmes de domination. Les contraintes de voisinage fixées par chacun de ces problèmes sont alors définies via les ensembles σ et ρ. Dans le TABLEAU 3.1, nous donnons en exemple les ensembles σ et ρ pour quelques problèmes de domination connus.

En fait, le pouvoir expressif de la notion d'ensemble $[\sigma, \rho]$-dominant permet également de redéfinir d'autres problèmes de recherche de sous-ensemble dans un graphe.

Problème	Ensemble σ	Ensemble ρ
ENSEMBLE DOMINANT	$\mathbb{N}$	$\mathbb{N}^*$
CODE PARFAIT	$\{0\}$	$\{1\}$
ENSEMBLE STABLE DOMINANT	$\{0\}$	$\mathbb{N}^*$
ENSEMBLE DOMINANT PARFAIT	$\mathbb{N}$	$\{1\}$
ENSEMBLE DOMINANT TOTAL	$\mathbb{N}^*$	$\mathbb{N}^*$
ENSEMBLE DOMINANT TOTAL PARFAIT	$\{1\}$	$\{1\}$

TABLEAU 3.1 — Quelques problèmes de domination définis sous la forme d'un problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Cette possibilité tient dans la plupart des cas de la possibilité d'intégrer l'entier 0 dans l'un ou l'autre des ensembles σ ou ρ. Nous donnons dans le TABLEAU 3.2 quelques exemples de ces problèmes.

Problème	Ensemble σ	Ensemble ρ
ENSEMBLE STABLE	$\{0\}$	$\mathbb{N}$
COUPLAGE INDUIT	$\{1\}$	$\mathbb{N}$
SOUS-GRAPHE q-RÉGULIER INDUIT	$\{q\}$	$\{0\}$
SOUS-GRAPHE INDUIT DE DEGRÉ BORNÉ PAR q	$\{0, 1, \ldots, q\}$	$\{0\}$

TABLEAU 3.2 — Quelques problèmes de recherche de sous-ensemble définis sous la forme d'un problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

La plupart des problèmes de recherche de sous-ensemble dans un graphe sont avant tout étudiés d'un point de vue optimisation, c'est-à-dire la recherche d'un sous-ensemble de plus petite ou plus grande cardinalité respectant les contraintes du problème. Ainsi, nous pourrons, selon le cas, rechercher un ensemble $[\sigma, \rho]$-dominant de plus petite ou de plus grande cardinalité, problèmes que nous nommerons respectivement ENSEMBLE $[\sigma, \rho]$-DOMINANT MINIMUM et ENSEMBLE $[\sigma, \rho]$-DOMINANT MAXIMUM.

Dans le cas du problème de décision ENSEMBLE $[\sigma, \rho]$-DOMINANT pour des ensembles σ et ρ arbitraires, les contraintes imposées par les ensembles σ et ρ peuvent être très fortes. De ce fait, il est déjà intéressant d'étudier le problème de déterminer l'existence d'un ensemble $[\sigma, \rho]$-dominant de cardinalité quelconque. C'est ce que nous allons faire dans ce chapitre.

3.1.2 État de l'art

Depuis son introduction par TELLE et PROSKUROWSKI [Tel94b, TP97], le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT a fait l'objet de nombreux travaux de recherche. Nous allons présenter ici un état de l'art des principaux résultats concernant la complexité classique et paramétrée de ce problème vu comme un problème de décision, c'est-à-dire que nous cherchons simplement à déterminer si un ensemble $[\sigma, \rho]$-dominant existe dans le graphe.

En complexité classique. Au moment de l'introduction de cette généralisation des problèmes de domination, TELLE et PROSKUROWSKI [Tel94b, TP97] ont effectué une première étude de la complexité classique du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT pour certains cas d'ensembles d'entiers σ et ρ.

Ainsi, ils ont montré que ce problème est NP-complet, pour des graphes quelconques, lorsque σ et ρ sont tous deux finis, lorsque $\sigma = \{0\}$ et $\rho = [r, +\infty[$ avec $r \geq 2$, correspondant au problème ENSEMBLE STABLE r-DOMINANT (*independent r-dominating set*), et lorsque $\sigma = \{1\}$ et $\rho = \mathbb{N}^*$, correspondant au problème ENSEMBLE D'ASSOCIATION INDUITE DOMINANT (*dominating induced matching*). Ils ont également montré que ce problème est déjà NP-complet pour les graphes bipartis planaires lorsque $\sigma = \{1\}$ et $\rho = \{1\}$, correspondant au problème ENSEMBLE DOMINANT TOTAL PARFAIT (*total perfect dominating set*).

Puisque le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est une généralisation de problèmes de domination, les résultats déjà connus pour des problèmes particuliers de domination se traduisent naturellement dans le formalisme du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Par exemple, le problème CODE PARFAIT est NP-complet pour les graphes planaires 3-réguliers [CN99], et correspond au cas où $\sigma = \{0\}$ et $\rho = \{1\}$.

Les résultats cités jusqu'à maintenant montrent divers cas où le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est NP-complet. Dans l'étude de la complexité d'un problème, il est toutefois intéressant de trouver des cas où le problème peut être résolu en temps polynomial.

Ainsi, TELLE et PROSKUROWSKI [Tel94b, TP97] ont montré que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps polynomial lorsque $\sigma = [a, +\infty[$ et $\rho = [b, +\infty[$ (avec $a, b \in \mathbb{N}$), ou lorsque $\sigma = \{0, 1\}$ et $\rho = [1, +\infty[$. Notons que dans ce dernier cas, il est NP-complet de trouver un ensemble $[\sigma, \rho]$-dominant de plus grande ou de plus petite cardinalité.

Dans de nombreux cas, il est nécessaire de considérer des classes de graphes particulières pour obtenir un algorithme polynomial permettant de résoudre un problème, celui-ci étant obtenu en tirant avantage des propriétés structurelles spécifiques des graphes de cette classe. Dans le cas des graphes de largeur arborescente bornée, TELLE et PROSKUROWSKI [Tel94b, TP97] ont montré que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps polynomial sur ces graphes lorsque σ et ρ sont deux ensembles d'entiers finis ou cofinis (c'est-à-dire le complément d'un ensemble fini). KRATOCHVÍL *et al.* [KMM95] ont quant à eux montré que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps polynomial sur les « graphes d'intervalles », lorsque σ et ρ sont deux ensembles d'entiers finis. Plus récemment, GOLOVACH et KRATOCHVÍL ont étudié le problème restreint aux graphes cordaux, et obtenu un résultat étonnant : lorsque σ et ρ sont deux ensembles finis, le problème peut être résolu en temps polyno-

mial si et seulement si le graphe contient au plus un ensemble $[\sigma, \rho]$-dominant ; sinon, le problème est NP-complet.

En complexité paramétrée. L'étude de la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, paramétrée selon divers paramètres, est plus récente. Celle-ci fait suite à l'introduction de la théorie de la complexité paramétrée par DOWNEY et FELLOWS [DF95a, DF95b], et est motivé par la volonté de généraliser les résultats connus sur la complexité paramétrée de nombreux problèmes de domination.

Il est important de noter que les deux ensembles d'entiers σ et ρ font partie de la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT ; ce ne sont pas des paramètres de ce problème.

Plusieurs paramètres naturels peuvent tout de même être considérés pour l'étude de la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, et notamment la cardinalité maximale $k \in \mathbb{N}$ de l'ensemble $[\sigma, \rho]$-dominant recherché. Le problème est alors noté k-ENSEMBLE $[\sigma, \rho]$-DOMINANT.

La plupart des problèmes classiques de domination sont connus pour être $\mathrm{W}[1]$-complets ou $\mathrm{W}[2]$-complets (voir notamment [DF95a, DF95b, DF99, FG06]). Puisque le problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT est une généralisation de ces problèmes de domination, nous pouvons immédiatement en déduire la complexité paramétrée pour les cas d'ensembles σ et ρ correspondants. Par exemple, le problème est $\mathrm{W}[2]$-complet lorsque $\sigma = \mathbb{N}$ et $\rho = \mathbb{N}^*$, ce qui correspond au problème k-ENSEMBLE DOMINANT.

Très récemment, GOLOVACH *et al.* [GKS10] ont généralisé ces résultats à tous les cas où les ensembles σ et ρ sont finis, et montré que le problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT est alors $\mathrm{W}[1]$-complet. RAMAN *et al.* [RSS08] ont quant à eux généralisé ces résultats aux cas où $\sigma = \mathbb{N}$ et ρ est cofini, montrant que certains sont FPT tandis que d'autres sont $\mathrm{W}[1]$-difficiles ou $\mathrm{W}[2]$-difficiles.

À l'instar de l'étude en complexité classique, il est habituel d'étudier la complexité paramétrée d'un problème lorsqu'il est restreint à certaines classes de graphes particulières.

Ainsi, ALON et GUTNER [AG09] ont étudié le problème de domination classique k-ENSEMBLE DOMINANT, paramétré par la cardinalité maximum $k \in \mathbb{N}$ de l'ensemble dominant recherché, et montré que ce problème est FPT lorsqu'il est restreint aux graphes de *dégénérescence* bornée. Pour ces mêmes graphes, GOLOVACH et VILLANGER [GV08] ont étudié d'autres problèmes particuliers de domination, montrant une complexité paramétrée différente selon le problème considéré.

La largeur arborescente est un autre paramètre naturel pouvant être considéré pour l'étude de la complexité du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Son importance s'explique en particulier par l'existence du résultat très général de COURCELLE *et al.* [Cou97, CMR01], que nous avons présenté dans la SECTION 1.4.3 des PRÉLIMINAIRES, et permettant de montrer qu'un problème est FPTlorsque, paramétré par la largeur arborescente, il peut être décrit par une formule MSO (logique monadique du second ordre).

Souhaitant résoudre efficacement le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, TELLE et PROSKUROSWKI [TP97] ont donné un algorithme pour résoudre le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT en temps FPT, paramétré par la largeur arborescente, lorsque σ et ρ sont finis ou cofinis. Plus précisément, leur algorithme s'exécute en temps $\mathcal{O}^*(c^{\mathrm{tw}})$, où c

est une constante dépendant linéairement uniquement des ensembles σ et ρ. Ce résultat a été récemment amélioré par VAN ROOIJ *et al.* [vRBR09], obtenant un algorithme FPT s'exécutant en temps $\mathcal{O}^*(s^{\mathrm{tw}})$, où s est le nombre minimum d'états nécessaires pour reconnaître les ensembles σ et ρ à l'aide d'un automate fini. Ce temps d'exécution est optimal, sous l'hypothèse « SETH » (*Strong Exponential Time Hypothesis*), supposant que le problème SAT ne peut pas être résolu en temps $c^n \cdot m^{c'}$ pour $c < 1$ et $c' > 0$ deux constantes fixées, où n est le nombre de *variables* et m le nombre de *clauses* de la formule (voir notamment [Wil10, PW10]).

3.1.3 Résultats de ce chapitre

La majorité des problèmes de graphe jusqu'à maintenant étudiés, appartiennent à la classe FPT lorsqu'ils sont paramétrés par la largeur arborescente du graphe donné en entrée. Tous les résultats obtenus concernant la complexité paramétré du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT suivent ce constat, puisque ce problème est FPT, paramétré par la largeur arborescente, pour tous les cas étudiés d'ensembles σ et ρ (finis ou cofinis).

Il est dès lors naturel de se demander si ce problème est FPT, paramétré par la largeur arborescente, pour n'importe quels ensembles σ et ρ, en supposant bien entendu que l'appartenance d'un entier à l'un de ces deux ensembles peut être calculée en temps polynomial.

Dans ce chapitre, nous nous intéressons à étendre les résultats connus sur la complexité paramétré du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT lorsque paramétré par la largeur arborescente.

Dans mon mémoire de Master [Cha08], les résultats connus d'appartenance à la classe FPT du problème ont été étendus aux ensembles σ et ρ ultimement périodiques, *via* l'écriture d'un algorithme FPT permettant de résoudre le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Nous rappelons succinctement ce résultat dans la SECTION 3.2, et donnons également un point de vue théorique de ce résultat à l'aide du « théorème de COURCELLE ».

Le principal apport de ce chapitre, présenté dans la SECTION 3.3, concerne la $\mathrm{W}[1]$-difficulté du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT lorsque paramétré par la largeur arborescente, pour un grand nombre de cas d'ensembles σ et ρ. Notre volonté étant d'effectuer une première étude des cas amenant ce problème à ne plus être FPT, nous nous focalisons principalement sur l'ensemble σ, fixant les contraintes sur les sommets appartenant à l'ensemble $[\sigma, \rho]$-dominant. Le résultat que nous obtenons est le suivant :

THÉORÈME
Supposons que ρ est un ensemble d'entiers cofini, et que σ est un ensemble d'entiers contenant des intervalles de longueur arbitraire entre deux éléments, sous une condition naturelle supplémentaire (voir SECTION *3.3.2). Alors le problème* ENSEMBLE $[\sigma, \rho]$-DOMINANT *est* $\mathrm{W}[1]$*-difficile lorsque paramétré par la largeur arborescente du graphe donné en entrée.*

De nombreux ensembles infinis d'entiers vérifient les conditions fixées sur l'ensemble σ. Nous pouvons citer, par exemple, l'ensemble des puissances de tout entier $\alpha > 1$ (pour $\alpha = 3$, nous obtenons l'ensemble $\sigma = \{0, 3, 9, 27, 81, \ldots\}$), l'ensemble de tous les nombres premiers (c'est-à-dire $\sigma = \{2, 3, 5, 7, 11, 13, \ldots\}$), ou encore l'ensemble des nombres de FIBONACCI (c'est-à-dire $\sigma = \{1, 2, 3, 5, 8, 13, \ldots\}$).

Toutefois, notre résultat ne s'applique pas aux ensembles infinis d'entiers ne contenant que des intervalles de longueur bornée entre deux éléments. Nous pouvons citer, par exemple, les ensembles ultimement périodiques, définis plus loin, pour lesquels le problème est en fait FPT (voir SECTION 3.2), ou encore l'ensemble des entiers pour lesquels les intervalles successifs correspondent aux décimales d'un nombre réel β (pour $\beta = 3.1415\ldots$, nous obtenons l'ensemble $\sigma = \{0, 4, 6, 11, 13, 18, \ldots\}$, où les intervalles entre deux éléments successifs de l'ensemble sont, dans l'ordre, $3, 1, 4, 1, 5, \ldots$). Nous ne connaissons pas à l'heure actuelle la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT pour un ensemble σ de ce type.

Enfin, dans la SECTION 3.4, nous nous intéressons à la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT dans le cas général, c'est-à-dire pour des ensembles σ et ρ quelconques, et montrons que ce problème appartient à la classe XP lorsqu'il est paramétré par la largeur arborescente uniquement, et peut être résolu en temps FPT lorsqu'il est paramétré à la fois par la largeur arborescente et par la cardinalité du sous-ensemble recherché. En effet, dès lors que nous souhaitons étudier la complexité paramétrée d'un problème, il est important de déterminer si cette étude est pertinente, et ainsi déterminer l'appartenance de ce problème à au moins l'une des classes de complexité paramétrée pertinentes et appartenant à la W-hiérarchie.

3.1.4 Quelques définitions

Domination généralisée Nous rappelons dans un premier temps la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, que nous avons déjà introduite précédemment dans la SECTION 3.1.1.

Afin de généraliser les notions communes aux nombreux problèmes de domination existants, TELLE et PROSKUROWSKI [Tel94a, Tel94b, TP97] ont introduit le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

DÉFINITION 3.1.1 (ensemble $[\sigma, \rho]$-dominant)
Soient σ et ρ deux ensembles d'entiers naturels, et soit $G = (V, E)$ un graphe. Un ensemble $[\sigma, \rho]$-dominant de G est un sous-ensemble $S \subseteq V(G)$ de sommets du graphe G tel que :
- *pour tout sommet $v \in S$, $|N(v) \cap S| \in \sigma$;*
- *pour tout sommet $v \notin S$, $|N(v) \cap S| \in \rho$.*

Le problème de décision associé est défini ainsi :

ENSEMBLE $[\sigma, \rho]$-DOMINANT

Entrée : Un graphe $G = (V, E)$, deux ensembles d'entiers $\sigma, \rho \subseteq \mathbb{N}$.
Question : Le graphe G admet-il un ensemble $[\sigma, \rho]$-dominant ?

Afin de simplifier certaines descriptions dans ce chapitre, nous utiliserons quelques notations spécifiques au problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Ainsi, lorsqu'un ensemble $[\sigma, \rho]$-dominant $S \subseteq V(G)$ respecte les conditions fixées par les ensembles σ et ρ pour un sommet $v \in V(G)$, nous dirons que le sommet v est $[\sigma, \rho]$-*dominé* par S. Un sommet v inclus dans l'ensemble $[\sigma, \rho]$-dominant S sera dit *selectionné*, tandis qu'un sommet non inclus dans cet ensemble sera dit *non sélectionné*.

Ensembles ultimement périodiques Dans mon mémoire de Master [Cha08], il a été montré que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est FPT, lorsque paramétré par la largeur arborescente, dès lors que les ensembles d'entiers σ et ρ sont ultimement périodiques (voir notamment [Mat94, BHMV94]). Nous rappellerons succinctement ce résultat dans la SECTION 3.2. Nous donnons ici quelques définitions utiles pour la compréhension de ce résultat.

DÉFINITION 3.1.2 (ensemble ultimement périodique)
Un ensemble $A \subseteq \mathbb{N}$ *est* ultimement périodique *s'il existe* k *entiers* $t_1, \ldots, t_k \in \mathbb{N}^*$ *(*$k \in \mathbb{N}$ *fixé), et* $n_0 \in A$ *tels que pour tout entier* $n \geq n_0$ *:*

$$n \in A \;\Rightarrow\; \forall \alpha \in \mathbb{N}, i \in \{1, \ldots, k\} \;:\; n + \alpha \cdot t_i \in A$$

En d'autres termes, un ensemble d'entiers ultimement périodique se décompose en :
- une ensemble fini d'entiers tous inférieurs strictement à n_0 ;
- une union de k ensembles infinis périodiques d'entiers supérieurs ou égaux à n_0, chacun de ces ensembles étant de période distincte t_i (avec $i \in \{1, \ldots, k\}$).

Les ensembles finis et cofinis d'entiers sont ultimement périodiques (avec respectivement $t = 0$ et $t = 1$). D'autres exemples d'ensembles ultimement périodiques sont les entiers multiples de 4, les entiers multiples de 3 ou 5 plus grands que 12, ou encore les années bissextiles.

La fonction caractéristique d'un ensemble d'entiers ultimement périodique peut être représentée sous la forme d'un automate fini déterministe à langage unaire [Mat94, BHMV94], en énumérant de manière itérative les éléments de cet ensemble. De plus, il existe un unique automate de taille minimum (en le nombre d'états) reconnaissant cet ensemble [Mat94, BHMV94].

L'algorithme polynomial que nous détaillons dans la SECTION 3.2 utilise un tel automate pour chacun des deux ensembles σ et ρ.

DÉFINITION 3.1.3
Un automate fini déterministe à langage unaire $\mathcal{A}$ *est la donnée d'un quadruplet* $\mathcal{A} = \big(\mathcal{Q}, \delta, (s_0), \mathcal{F}\big)$ *tel que :*
- $\mathcal{Q} = \big\{e_1, \ldots, e_p\big\}$ *est l'ensemble des états de l'automate ;*
- $\delta : \mathcal{Q} \to \mathcal{Q}$ *est la* fonction de transition *d'un état à un autre ;*
- s_0 *est l'état initial de l'automate ;*
- $\mathcal{F} \subseteq \mathcal{Q}$ *est l'ensemble des états finaux acceptants de l'automate.*

Un automate à langage unaire est un cas particulier des automates, dans lequel chaque état n'a qu'une seule transition sortante. C'est pourquoi la fonction de transition δ ne prend en paramètre qu'un état.

Pour exemple, nous schématisons dans la FIGURE 3.1 un automate fini déterministe à langage unaire permettant de reconnaître l'ensemble ultimement périodique $S = \{1\} \cup \{3 \cdot n + i \mid n \in \mathbb{N}^*,\ i = 0, 1\}$.

Nous pouvons remarquer qu'un automate fini déterministe à langage unaire, reconnaissant un ensemble d'entiers ultimement périodique, est composé de deux parties : un simple chemin depuis l'état initial de l'automate, puis une boucle « finale ». Puisque chaque état d'un tel automate a exactement une transition sortante, il ne peut exister au plus qu'une boucle.

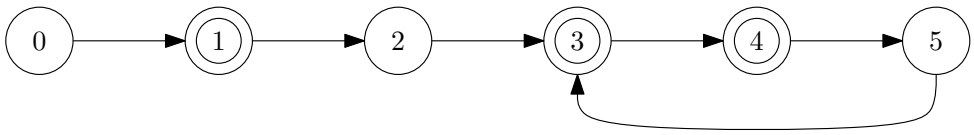


FIGURE 3.1 — Automate fini déterministe à langage unaire, permettant de reconnaître l'ensemble ultimement périodique $S = \{1\} \cup \{3 \cdot n + i \mid n \in \mathbb{N}^*,\ i = 0, 1\}$.

Étant donné un ensemble d'entiers ultimement périodique $\mathcal{N} \subseteq \mathbb{N}$, nous notons $\mathcal{A}_{\mathcal{N}} = \left(\mathcal{Q}_{\mathcal{N}}, \delta_{\mathcal{N}}, (s_0)_{\mathcal{N}}, \mathcal{F}_{\mathcal{N}}\right)$ l'automate fini déterministe à langage unaire de taille minimum reconnaissant les éléments de $\mathcal{N}$.

3.2 Cas FPT

Nous étudions dans cette section les cas d'ensembles d'entiers σ et ρ pour lesquels le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps FPT lorsque paramétré par la largeur arborescente du graphe donné en entrée. Nous montrons qu'un tel résultat peut être obtenu, dès lors que les ensembles σ et ρ sont ultimement périodiques, laissant transparaître une *régularité*.

Dans un premier temps, nous utilisons un résultat théorique de COURCELLE *et al.*, présenté dans la SECTION 1.4.3, pour confirmer la complexité paramétrée théorique du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT lorsque σ et ρ sont ultimement périodiques.

Dans un second temps, nous présentons les idées générales d'un algorithme FPT efficace permettant de résoudre explicitement le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, et développé lors de mon mémoire de Master [Cha08].

3.2.1 Résultat théorique *via* un théorème de COURCELLE *et al.*

COURCELLE *et al.* [CMR01] ont montré que tout problème pouvant être exprimé sous la forme d'une formule CMSO (logique monadique du second ordre avec comptage, voir par exemple [CMR01]) peut être résolu en temps FPT lorsqu'il est paramétré par la largeur arborescente du graphe donné en entrée (voir SECTION 1.4.3).

Nous allons montrer que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être exprimé sous la forme d'une formule CMSO, lorsque σ et ρ sont deux ensembles d'entiers ultimement périodiques, impliquant donc que ce problème est FPT lorsque paramétré par la largeur arborescente. Pour cela, nous donnons explicitement la formule CMSO exprimant le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Rappelons que comme σ et ρ sont deux ensembles d'entiers ultimement périodiques, il existe deux automates finis déterministes à langage unaire $\mathcal{A}_\sigma$ et $\mathcal{A}_\rho$, de taille minimum, permettant de reconnaître et énumérer les éléments de σ. Notons p le nombre total d'états de l'automate $\mathcal{A}_\sigma$ reconnaissant σ, et p_0 le nombre d'états contenu dans la boucle finale de cet automate. De même, notons q est le nombre total d'états de l'automate $\mathcal{A}_\rho$ reconnaissant ρ, et q_0 le nombre d'états contenu dans la boucle finale de cet automate.

Nous allons maintenant expliciter la formule CMSO exprimant le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Pour des raisons de lisibilité, nous la décrivons en plusieurs parties.

Le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être exprimé par une formule CMSO de la manière suivante, pour tout graphe $G = (V, E)$:

$$\begin{aligned}\phi \equiv \exists S, \bar{S} \ \forall v \in V : \quad & (v \in S \wedge v \notin \bar{S}) \vee (v \notin S \wedge v \in \bar{S}) \\ \wedge \quad & v \in S \Rightarrow |N(v) \cap S| \in \sigma \\ \wedge \quad & v \in \bar{S} \Rightarrow |N(v) \cap S| \in \rho\end{aligned}$$

Cette formule est une traduction directe de la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT (voir section 3.1.4). Il nous faut toutefois donner les sous-formules CMSO correspondant aux expressions $|N(v) \cap S| \in \sigma$ et $|N(v) \cap S| \in \rho$, lorsque les ensembles d'entiers σ et ρ sont ultimement périodiques.

Notons respectivement p_σ et p_ρ le nombre d'états des automates finis déterministes à langage unaire reconnaissant les deux ensembles σ et ρ. L'ensemble σ étant ultimement périodique, il correspond à l'union d'un ensemble fini d'entiers $X_\sigma \subseteq \{1, \ldots, p_\sigma\}$, et de k_σ ensembles infinis périodiques d'entiers, chacun de période distincte t_i^σ (pour $i \in \{1, \ldots, k_\sigma\}$). De même pour l'ensemble ρ, pour lequel nous utilisons des notations similaires. Nous pouvons alors représenter d'un point de vue logique ces ensembles, en utilisant le prédicat $\mathrm{Card}_p(S)$ fournit par la logique CMSO, qui est vrai si et seulement si la cardinalité de l'ensemble S est un multiple de p. Nous obtenons les expressions suivantes :

$$\begin{aligned}|N(v) \cap S| \in \sigma \equiv & \bigvee_{i \in \{1,\ldots,k_\sigma\}} \exists Y_S \left(\mathrm{Card}_{t_i^\sigma}(Y_S) \wedge \bigvee_{p \in \sigma, p \leq p_\sigma} \exists u_1, \ldots, u_p \ \zeta\right) \\ |N(v) \cap S| \in \rho \equiv & \bigvee_{i \in \{1,\ldots,k_\rho\}} \exists Y_S \left(\mathrm{Card}_{t_i^\rho}(Y_S) \wedge \bigvee_{q \in \rho, q \leq q_\rho} \exists u_1, \ldots, u_q \ \zeta\right)\end{aligned}$$

où

$$\zeta \equiv \left[\left(u_i \in (N(v) \cap S) \wedge u_i \notin Y_S\right) \wedge \forall u \ (u \neq u_i)\right] \Rightarrow \left(u \in Y_S \Leftrightarrow u \in (N(v) \cap S)\right).$$

Dans l'expression correspondant à $|N(v) \cap S| \in \sigma$, chaque ensemble Y_S correspond à l'un des k_σ ensembles infinis périodiques d'entiers composant l'ensemble σ, comptabilisant un nombre *périodique* de voisins du sommet v, tandis que l'expression ζ correspond à l'ensemble fini d'entiers $X_\sigma \subseteq \{1, \ldots, p_\sigma\}$ et comptabilise le reste (fini) des voisins de v. L'expression correspondant à $|N(v) \cap S| \in \rho$ est définie de manière équivalente pour l'ensemble ρ.

Pour que le résultat de COURCELLE *et al.* [CMR01] puisse s'appliquer, nous devons nous assurer que la longueur de la formule CMSO obtenue ne dépend pas de la taille du graphe. La taille de la formule obtenue dépend uniquement du nombre total d'états des deux automates minimum utilisés pour reconnaître les ensembles σ et ρ, et ces deux nombres ne dépendent que de σ et ρ. Puisque σ et ρ font partie de la définition du problème (ce ne sont pas des paramètres), ils peuvent être considérés comme des constantes, et notre formule est donc de longueur constante pour le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Nous en déduisons que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est FPT lorsque paramétré par la largeur arborescente, si les ensembles d'entiers σ et ρ sont ultimement périodiques.

Toutefois, le résultat de COURCELLE *et al.* [CMR01] est purement théorique, la complexité en temps de l'algorithme qu'ils fournissent étant déraisonnable (la dépendance au paramètre, la largeur arborescente, est une tour d'exponentielles dont la hauteur dépend de la *profondeur* de la formule CMSO).

Dans la suite de cette section, nous présentons les idées générales d'un algorithme efficace permettant de résoudre le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT en temps FPT paramétré par la largeur arborescente, et dont la dépendance au paramètre est simplement exponentielle.

3.2.2 Idée générale de l'algorithme

L'algorithme FPT, dont nous allons maintenant présenter les idées générales, a été développé lors de mon mémoire de Master [Cha08].

Pour rappel, nous considérons dans cette section que les ensembles d'entiers σ et ρ sont ultimement périodiques. De ce fait, il existe deux automates finis déterministes à langage unaire $\mathcal{A}_\sigma$ et $\mathcal{A}_\rho$, de taille minimum, permettant de reconnaître et énumérer les éléments de ces deux ensembles σ et ρ respectivement.

Soit $G = (V, E)$ un graphe, et (T, χ) une décomposition arborescente jolie de G. Comme le graphe donné en entrée est de largeur arborescente bornée, nous pouvons supposer qu'une décomposition arborescente jolie de ce graphe est également donnée, puisque cette décomposition peut être construite en temps FPT paramétré par la largeur arborescente (voir notamment [Bod96, Klo94]). Notre algorithme est de type « programmation dynamique », parcourant la décomposition arborescente jolie (T, χ) d'un graphe $G = (V, E)$ des feuilles vers la racine (voir SECTION 1.3). Récursivement, pour chacun des noeuds de cette décomposition, notre algorithme calcule des solutions partielles du problème. Et à la fin de l'algorithme, la solution complète du problème se trouve dans l'une des caractéristiques de la racine de cette décomposition. Dans le cas de notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, une solution partielle est définie ainsi :

DÉFINITION 3.2.1 (solution partielle)
Soit T_i le sous-arbre de T enraciné au noeud i, $X_i \in \chi$ le sac correspondant au noeud i, et G_i le sous-graphe de G induit par les seuls sommets apparaissant dans les sacs de T_i. Un sous-ensemble $S_i \subseteq V(G_i)$ associé à une caractéristique du noeud i est appelé solution partielle *si tout sommet $v \in V(G_i) \setminus X_i$ a un nombre valide de sommets sélectionnés, par rapport aux contraintes fixées par σ et ρ.*

Durant le calcul des solutions partielles, notre algorithme compte le nombre de voisins sélectionnés pour chaque sommet du graphe, en utilisant les deux automates $\mathcal{A}_\sigma$ et $\mathcal{A}_\rho$. Ainsi, n'est conservé dans les caractéristiques des noeuds de la décomposition que l'état courant de chaque sommet par rapport à l'automate correspondant : si le sommet est sélectionné, l'algorithme conserve un état de l'automate $\mathcal{A}_\sigma$, et s'il n'est pas sélectionné, l'algorithme conserve un état de l'automate $\mathcal{A}_\rho$. Plus formellement, l'état d'un sommet du graphe G, dans une caractéristique d'un noeud de la décomposition arborescente jolie, peut être défini ainsi :

DÉFINITION 3.2.2 (état d'un sommet)
*Soit $v \in V(G)$, i un noeud de la décomposition T, $X_i \in \chi$ le sac correspondant au noeud i, S_i une solution partielle, et Q_σ, Q_ρ les états des automates $\mathcal{A}_\sigma$ et $\mathcal{A}_\rho$ respectivement. On définit l'*état d'un sommet v de X_i, *noté* $\mathsf{e}_i(v)$, *par :*

$$\mathsf{e}_i(v) = \begin{cases} \sigma_j \in Q_\sigma & \textit{si } v \in S_i,\ 1 \leq j \leq p \\ \rho_k \in Q_\rho & \textit{si } v \notin S_i,\ 1 \leq k \leq q \end{cases}$$

Lors du parcours de la décomposition arborescente jolie, notre algorithme ne peut pas conserver toutes les informations relatives à une solution partielle, car celles-ci sont trop nombreuses, et leur calcul exploserait le temps d'exécution de notre algorithme. C'est pourquoi n'est associé à chaque noeud de la décomposition arborescente jolie qu'une collection de caractéristiques, stockant uniquement l'état des sommets contenus dans le sac correspondant à ce noeud.

DÉFINITION 3.2.3 (caractéristique d'un noeud)
Soit i un noeud de la décomposition T, S_i une solution partielle, et $n_i = |X_i| \leq \mathrm{tw}(G) + 1$. Une caractéristique *d'un noeud i est un n_i-uplet d'états $(\mathsf{e}_i(v_1), \ldots, \mathsf{e}_i(v_{n_i}))$, où l'état $\mathsf{e}_i(v_j)$ de v_j correspond indirectement, par comptage via l'automate correspondant, au nombre de voisins de v_j sélectionnés dans la solution partielle S_i. On note $c(i)$ la collection de caractéristiques du noeud i.*

Cela limite fortement la quantité d'informations pouvant être associée à un noeud de la décomposition arborescente jolie, puisque le nombre maximum de caractéristiques pour un noeud ne dépend que des deux automates reconnaissant les ensembles σ et ρ, dont les tailles ne dépendent que de ces deux ensembles.

Par définition d'une décomposition arborescente jolie (voir SECTION 1.3), une telle décomposition n'est composée que de quatre types de noeuds différents : noeud feuille, noeud introduction, noeud suppression, et noeud jonction. Notre algorithme est donc composé de quatre opérations, chacune associée à un type de noeud, et dont le rôle est de calculer les caractéristiques d'un noeud en fonction de ses fils.

Rappelons que dans une décomposition arborescente jolie, seuls les noeuds de type jonction ont deux fils, les autres noeuds ayant un ou zéro fils. Ainsi, dans le cas de notre algorithme, l'opération associée aux noeuds de type jonction est la plus complexe : pour être combinées, les caractéristiques associées aux deux fils d'un noeud jonction doivent être *compatibles*. D'une part, deux caractéristiques ne peuvent se combiner que si l'état associé à chaque sommet dans ces deux caractéristiques appartient au même automate (automate $\mathcal{A}_\sigma$ pour les sommets ajoutés à une solution, et automate $\mathcal{A}_\rho$ pour les sommets non ajoutés). D'autre part, la combinaison des états d'un sommet dans les deux caractéristiques doit être effectuée de sorte à correspondre correctement au nombre de ses voisins sélectionnés, et éviter de décompter deux fois certains de ces voisins.

Nous obtenons ainsi un algorithme FPT paramétré par la largeur arborescente, permettant de résoudre efficacement le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT lorsque les ensembles σ et ρ sont ultimement périodiques. Nous tirons notamment avantage de l'utilisation de la « convolution rapide de sous-ensembles » (*fast subset convolution*),

introduite BJÖRKLUND *et al.* [BHKK07], et permettant d'améliorer la combinaison de caractéristiques dans l'opération associée aux noeuds de type jonction.

THÉORÈME 3.2.1
Le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT *peut être résolu en temps* $\mathcal{O}^*(s^{\mathrm{tw}})$*, c'est-à-dire en temps* FPT *lorsque paramétré par la largeur arborescente du graphe donné en entrée, où* s *est un petit polynôme dépendant uniquement du nombre total d'états composant les deux automates* $\mathcal{A}_\sigma$ *et* $\mathcal{A}_\rho$ *énumérant respectivement les éléments des ensembles* σ *et* ρ*.*

Démonstration (idées principales). Pour obtenir ce temps d'exécution, nous utilisons le principe de « convolution rapide de sous-ensembles » (*fast subset convolution*), introduit par BJÖRKLUND *et al.* [BHKK07], et utilisée par VAN ROOIJ *et al.* [vRBR09] dans leur algorithme permettant de résoudre ENSEMBLE $[\sigma, \rho]$-DOMINANT en temps FPT, paramétré par la largeur arborescente, lorsque σ et ρ sont finis ou cofinis.

Supposons que $\mathcal{Q}_\sigma = \{\sigma_0, \ldots, \sigma_a, \ldots, \sigma_p\}$ et $\mathcal{Q}_\rho = \{\rho_0, \ldots, \rho_b, \ldots, \rho_q\}$, où $\{\sigma_0, \ldots, \sigma_a\}$ correspond aux premiers éléments de σ avant qu'il soit périodique, et $\{\rho_0, \ldots, \rho_b\}$ correspond aux premiers éléments de ρ avant qu'il soit périodique. Avant chaque opération sur un noeud i de type jonction, où $|X_i| = n_i$, l'ensemble des états $\{\sigma_0, \ldots, \sigma_a, \ldots, \sigma_p\}$ de $\mathcal{A}_\sigma$ est transformé en l'ensemble $\{\sigma_0, \sigma_{\leq 1}, \ldots, \sigma_{\leq a}, \sigma_{a+1}, \ldots, \sigma_p\}$, et l'ensemble des états $\{\rho_0, \ldots, \rho_b, \ldots, \rho_q\}$ de $\mathcal{A}_\rho$ est transformé en l'ensemble $\{\rho_0, \rho_{\leq 1}, \ldots, \rho_{\leq b}, \rho_{b+1}, \ldots, \rho_q\}$. Chaque opération sur un noeud de type jonction, composé de la transformation de l'ensemble d'états, du calcul de la caractéristique correspondante, et enfin de la transformation inverse, peut être effectuée en temps $\mathcal{O}^*\big(\big((p-a)^2 + (q-b)^2 + a + b\big)^{n_i}\big)$. Notons que dans le cas d'ensembles d'entiers σ et ρ finis ou cofinis, nous obtenons un temps d'exécution de $\mathcal{O}^*((p+q)^{n_i})$.

Le nombre de noeuds d'une décomposition arborescente jolie est au plus $\mathcal{O}(n)$, et $n_i \leq \mathrm{tw}$ pour tout noeud i de cette décomposition. Comme l'opération sur les noeuds de type jonction est la plus coûteuse en temps, le temps d'exécution global de notre algorithme est donc de $\mathcal{O}^*\big(\big((p-a)^2 + (q-b)^2 + a + b\big)^{n_i}\big)$. □

3.3 Quelques cas difficiles

3.3.1 Objectif et idées

Objectif. Nous avons pour le moment considéré les cas les plus simples pour les ensembles σ et ρ (finis, cofinis, ultimement périodiques), et montré que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps FPT lorsque paramétré par la largeur arborescente du graphe donné en entrée. Pour l'écriture d'un algorithme FPT, nous avons tiré partie de la régularité des ensembles σ et ρ.

La largeur arborescente permettant très souvent d'écrire des algorithmes efficaces pour des problèmes difficiles en général, il est naturel de se demander si le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est toujours FPT, paramétré par la largeur arborescente, et ce quelques soient les ensembles polynomialement décidables σ et ρ. Dans cette section, nous allons montrer que ce problème n'est pas toujours FPT, et qu'une part d'irrégularité dans l'ensemble σ suffit à rendre le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT W[1]-difficile lorsque paramétré par la largeur arborescente du graphe :

THÉORÈME 3.3.1
Supposons que ρ est un ensemble d'entiers cofini, et que σ est un ensemble d'entiers contenant des intervalles de longueur arbitraire entre deux éléments, à une distance polynomiale en la longueur de l'intervalle (voir SECTION *3.3.2). Alors le problème* ENSEMBLE $[\sigma, \rho]$-DOMINANT *est* $\mathrm{W}[1]$*-difficile lorsque paramétré par la largeur arborescente du graphe donné en entrée.*

Nous verrons dans la SECTION 3.4 que cette étude est pertinente, le problème étant XP lorsque σ et ρ sont deux ensembles d'entiers récursifs, dont l'appartenance d'un entier à ces ensembles peut être décidée en temps polynomial.

Comme nous l'expliquions dans les PRÉLIMINAIRES (voir SECTION 1.1.3), une méthode classique pour montrer qu'un problème est $\mathrm{W}[1]$-difficile consiste à effectuer une réduction à partir d'un problème dont la $\mathrm{W}[1]$-difficulté selon le même paramètre est déjà connue. Toutefois, il existe actuellement très peu de problèmes paramétrés par la largeur arborescente connus comme étant $\mathrm{W}[1]$-difficiles.

Idées de la réduction. Nous allons effectuer une réduction en deux étapes, à partir du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS, connu comme étant $\mathrm{W}[1]$-difficile paramétré par la largeur arborescente, par un résultat de DOM *et al.* [DLSV08]. [1]

La première étape de notre réduction va consister à réduire le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers une variante de notre problème, à savoir ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W. Cette variante est définie de la même manière que notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, mais l'entrée est également composée d'un sous-ensemble de sommets présélectionnés, chacun de degré 1, qui doivent nécessairement faire partie de l'ensemble $[\sigma, \rho]$-dominant et sont toujours satisfaits (c'est-à-dire que les contraintes de σ ne s'appliquent pas à ces sommets) :

ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W

Entrée : Un graphe $G = (V, E)$, deux ensembles d'entiers $\sigma, \rho \subseteq \mathbb{N}$, un sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés de degré 1.
Paramètre : $\mathrm{tw}(G)$.
Question : Le graphe G admet-il un ensemble $[\sigma, \rho]$-dominant D, c'est-à-dire que pour tout sommet $v \in V(G) \setminus W$, avec $W \subseteq D$, $v \in D \Rightarrow |N(v) \cap D| \in \sigma$ et $v \notin D \Rightarrow |N(v) \cap D| \in \rho$, et tel que chaque sommet de W soit toujours satisfait ?

La seconde étape de notre réduction va consister à montrer que la variante ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W peut se réduire à notre problème initial ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS est une version contrainte du problème classique ENSEMBLE DOMINANT, où chaque sommet appartenant à l'ensemble ne peut dominer qu'un nombre limité de ses voisins.

Formellement, étant donné un graphe $G = (V, E)$ et une *fonction de capacité* $\mathsf{cap} : V \to \mathbb{N}$ sur les sommets du graphe, un *ensemble dominant avec capacités* est un couple

1. Plus précisément, DOM *et al.* [DLSV08] montrent que le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS est $\mathrm{W}[1]$-complet lorsque paramétré par la cardinalité maximum k de l'ensemble dominant avec capacité, et la largeur arborescente du graphe donné en entrée.

(C, dom), où $C \subseteq V$ est un sous-ensemble de sommets, et $\mathsf{dom} : V \to \mathcal{P}(V)$ est une *fonction de domination* qui associe à chaque sommet $v \in C$ un sous-ensemble de sommets $\mathsf{dom}(v) \subseteq N(v) \setminus C$ de cardinalité au plus $\mathsf{cap}(v)$, et tels que chaque sommet $w \in V \setminus C$ appartient à un sous-ensemble $\mathsf{dom}(v)$ pour un certain $v \in C$. La fonction de capacité cap correspond donc au nombre maximum de voisins qu'un sommet sélectionné peut dominer. La FIGURE 3.2 schématise un tel ensemble dominant avec capacités.

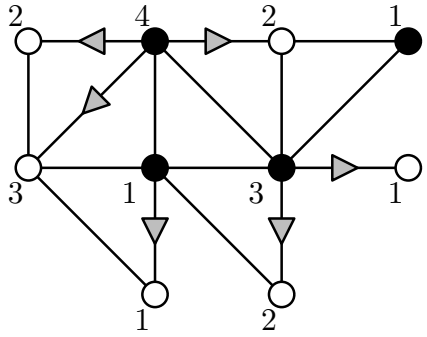


FIGURE 3.2 — Schéma d'un ensemble dominant avec capacités. La capacité de chaque sommet est indiquée par un entier, tandis que la fonction de domination dom est représentée par les flèches : une flèche allant d'un sommet u vers un sommet v signifie que u domine v.

Le problème paramétré correspondant est défini ainsi :

k-ENSEMBLE DOMINANT AVEC CAPACITÉS

Entrée : Un graphe $G = (V, E)$ de largeur arborescente tw, une fonction $\mathsf{cap} : V \to \mathbb{N}$, un entier $k \in \mathbb{N}$.
Paramètre : $k + \mathrm{tw}(G)$.
Question : Le graphe G admet-il un ensemble dominant avec capacités (C, dom) avec $|C| \leq k$?

Soit ρ un ensemble cofini d'entiers, et soit σ un ensemble d'entiers contenant des intervalles de longueur arbitraire entre deux éléments, à une distance polynomiale en la longueur de l'intervalle (voir SECTION 3.3.2). Ces deux ensembles peuvent être considérés comme étant des constantes, puisqu'ils font partie de la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT (ce ne sont pas des paramètres).

Nous définissons $q_0 = \min_{q \in \rho}\{q \mid \forall r \geq q, r \in \rho\}$, bien défini puisque ρ est cofini. Dans la suite de cette section, nous donnons en deux étapes une réduction FPT depuis le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, ce qui montre que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est $\mathrm{W}[1]$-difficile lorsque paramétré par la largeur arborescente du graphe donné en entrée.

Pour des raisons de lisibilité, nous supposons que $\min \rho \geq 2$ (et donc $q_0 \geq 2$), et que $\min \sigma \geq 1$. La réduction et les démonstrations peuvent être adaptées aux cas extrêmes où $\min \rho \geq 1$ ou $\min \sigma = 0$. [2]

2. Pour rappel, nous supposons que $0 \notin \rho$, car sinon $S = \emptyset$ serait une solution triviale au problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

3.3.2 Propriétés sur l'ensemble σ

Nous souhaitons montrer qu'une part d'irrégularité dans l'ensemble σ suffit à rendre le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT $\mathrm{W}[1]$-difficile lorsque paramétré par la largeur arborescente du graphe donné en entrée.

Pour cela, nous nous intéressons au cas où σ est un ensemble d'entiers contenant des intervalles de longueur arbitraire. Nous utiliserons ces intervalles, dans notre réduction du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, pour encoder les capacités des sommets du graphe.

Pour la description de notre réduction FPT, nous considérons que sont données les fonctions suivantes sur l'ensemble σ :

- $\Gamma_-(x, q) = \min_{p\in\sigma}\{p \ : \ p \geq q \ \wedge \ p - i \notin \sigma \text{ pour } 1 \leq i \leq x\}$;
- $\Gamma_+(x, q) = \min_{p\in\sigma}\{p \ : \ p \geq q \ \wedge \ p + i \notin \sigma \text{ pour } 1 \leq i \leq x\}$;
- $\Gamma_0(q) = \min_{p\in\sigma}\{p \ : \ p \geq q\}$.

En d'autres termes, la fonction $\Gamma_-(x, q)$ recherche le plus petit entier appartenant à σ, plus grand que q, pour lequel les x précédents entiers de $\mathbb{N}$ n'appartiennent pas à σ, formant alors un intervalle dans σ de longueur x avant cet entier. De même, $\Gamma_+(x, q)$ recherche le plus petit entier de σ avec un intervalle de longueur x après lui. Enfin, $\Gamma_0(q)$ recherche le premier entier de σ plus grand que q, et de manière équivalente correspond à $\Gamma_-(0, q)$ et $\Gamma_+(0, q)$.

Les ensembles σ et ρ font partie de la définition du problème. Ils peuvent donc être considérés comme des constantes, et de ce fait les fonctions Γ_-, Γ_+ et Γ_0 sont calculables en temps polynomial en les paramètres x et q.

Toutefois, nous devons ajouter une légère condition technique concernant les intervalles dans σ. En effet, la construction de certains gadgets de notre réduction (en particulier les gadgets $\mathcal{C}$ et $\mathcal{L}$) vont nécessiter des intervalles dont la longueur dépend de la taille du graphe initial. Pour que notre réduction soit effectivement FPT, nous devons nous assurer que la taille de ces gadgets reste polynomiale en la taille du graphe initial.

Nous ajoutons donc la condition suivante sur l'ensemble σ : pour tout entier $t \in \mathbb{N}$, un intervalle de longueur au moins t doit se trouver à une distance polynomiale en t dans σ, c'est-à-dire que $\Gamma_+(t, q)$ (pour $q \in \mathbb{N}$) est polynomiale en t. Plus formellement, la condition est la suivante :

$$\exists c \in \mathbb{N}, \ \forall t, q \in \mathbb{N} \ : \ \Gamma_+(t, q) = \mathcal{O}(q + t^c)$$

Considérons par exemple l'ensemble infini $S = \{n^2 \mid n \in \mathbb{N}\}$ des carrés de 2. On peut aisément vérifier que pour tout entier $t \in \mathbb{N}$, un intervalle de longueur au moins t débute à partir d'un entier inférieur ou égal à t^2 dans σ. Nous schématisons le début de cet ensemble, ainsi que les intervalles, dans la FIGURE 3.3.

Cette condition technique est également vraie pour la plupart des ensembles d'entiers que nous pouvons considérer, tels que l'ensemble des nombres de FIBONACCI, ou encore l'ensemble des puissances de $\alpha \geq 2$. Cette condition que nous imposons est donc relativement *naturelle* et peu restrictive.

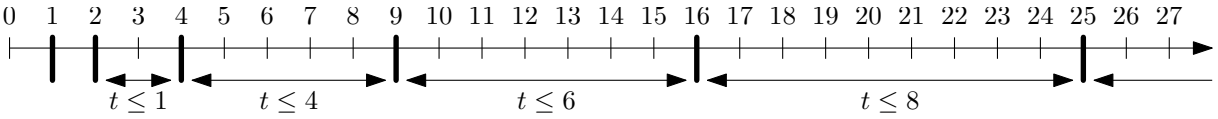


FIGURE 3.3 — Schéma représentant le début de l'ensemble infini $S = \{n^2 \mid n \in \mathbb{N}\}$ des carrés de 2, et les intervalles de longueur arbitraire.

3.3.3 Description de la première étape de réduction

Nous allons décrire dans cette section la première étape de réduction depuis le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Il s'agit de détailler les transformations effectuées sur une instance du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS, afin de la transformer en une instance du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W.

Réduction générale. Soit (G, cap, k) une instance du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS, paramétré par la cardinalité maximum k de l'ensemble dominant avec capacités, et par la largeur arborescente du graphe G, où cap est la fonction de capacités sur les sommets de G. Nous allons construire l'instance (H, W) du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W paramétré par la largeur arborescente du graphe H, telle que G admet un ensemble dominant avec capacités de cardinalité k si et seulement si H admet un ensemble $[\sigma, \rho]$-dominant contenant le sous-ensemble W de sommets présélectionnés toujours satisfaits et tous de degré 1. Notons que le paramètre k de l'instance initiale (G, cap, k) disparaît : il sera en fait encodé directement dans le graphe H.

L'idée de la réduction FPT est qu'en utilisant les fonctions Γ, nous pouvons trouver un intervalle dans σ de longueur plus grande que k, le paramètre du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS, et utiliser cet intervalle pour contrôler le nombre de voisins sélectionnés de chaque sommet du graphe G, et s'assurer que ce nombre ne dépasse pas la capacité du sommet s'il est également sélectionné.

Dans la suite de cette description, un sommet est dit *présélectionné* s'il appartient au sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés, chacun de degré 1, tel que défini dans le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W. Rappelons qu'un sommet *présélectionné* appartient nécessairement à l'ensemble $[\sigma, \rho]$-dominant D, et qu'il est toujours satisfait quelques soient les contraintes fixées par σ. Un sommet est dit *forcé* s'il doit nécessairement appartenir à l'ensemble $[\sigma, \rho]$-dominant, du fait de la construction du gadget (nous en expliquerons la raison dans la SECTION 3.3.4).

Nous commençons la réduction par le graphe d'incidence $I(G)$ de l'instance initiale, et ajoutons plusieurs gadgets reliés aux sommets-originels ou sommets-arêtes de $I(G)$. La FIGURE 3.4 schématise la manière dont ces gadgets sont reliés aux différents sommets du graphe d'incidence $I(G)$.

Pour chaque sommet-originel du graphe $I(G)$, nous ajoutons un gadget capacité $\mathcal{C}$, satisfiabilité $\mathcal{S}$ et domination $\mathcal{D}$, relié à ce sommet-originel de $I(G)$, et le gadget domination $\mathcal{D}$ est également relié à tous les sommets-arêtes voisins du sommet-originel correspon-

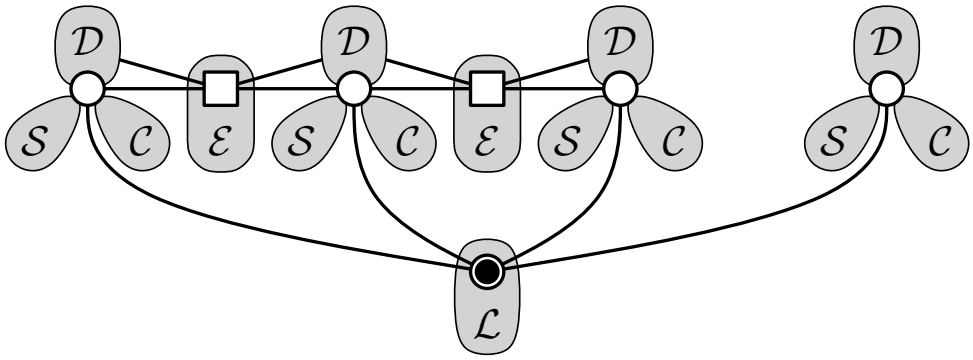


FIGURE 3.4 — Construction globale du graphe H, à partir du graphe d'incidence $I(G)$, pour la première étape de la réduction. Les sommets blancs sont les sommets du graphe d'incidence $I(G)$: les cercles correspondent aux sommets-originels, et les carrés correspondent aux sommets-arêtes.

dant. Pour chaque sommet-arête de $I(G)$, nous ajoutons un gadget arête-sélection $\mathcal{E}$ relié à ce sommet-arête. Enfin, un gadget global limitation $\mathcal{L}$ est ajouté au graphe, et est relié à tous les sommets-originels de $I(G)$ par le biais d'un sommet *central*.

Transposition d'une solution. Supposons que le graphe G admette un ensemble dominant avec capacités C de cardinalité k. Nous allons voir comment cet ensemble dans G va être encodé par un ensemble $[\sigma, \rho]$-dominant D dans le graphe H.

Si un sommet v de G est inclus dans C, alors le sommet-originel correspondant dans $I(G)$ sera inclus dans D. Pour chaque sommet $u \in V(G) \setminus C$, dominé par un sommet $v \in V(G)$, le sommet-arête e dans le graphe d'incidence $I(G)$, représentant l'arête entre u et v dans G, sera également inclus dans l'ensemble $[\sigma, \rho]$-dominant D. Cet encodage d'une solution du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS dans le graphe d'incidence est schématisé dans la FIGURE 3.5.

Le gadget limitation $\mathcal{L}$ permet de s'assurer qu'au plus k sommets-originels de $I(G)$ sont ajoutés à D, et permet ainsi d'encoder le paramètre k du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS. Les autres sommets du graphe H, qui n'appartiennent pas au graphe d'incidence $I(G)$, sont des sommets appartenant aux différents gadgets utilisés pour la réduction. Si une solution au problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W existe dans le graphe H, alors chaque gadget sera composé de deux types de sommets : les sommets « forcés » ou « présélectionnés », qui doivent nécessairement appartenir à l'ensemble $[\sigma, \rho]$-dominant, et les sommets « sélectionnables » qui sont toujours satisfaits, au regard des contraintes fixées par les ensembles σ et ρ, qu'ils soient sélectionnés ou non.

Détail des gadgets. Nous allons maintenant détailler la construction de chacun des gadgets, utilisés dans la première étape de réduction FPT, c'est-à-dire du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W. Pour plus de compréhension, nous donnons également un schéma

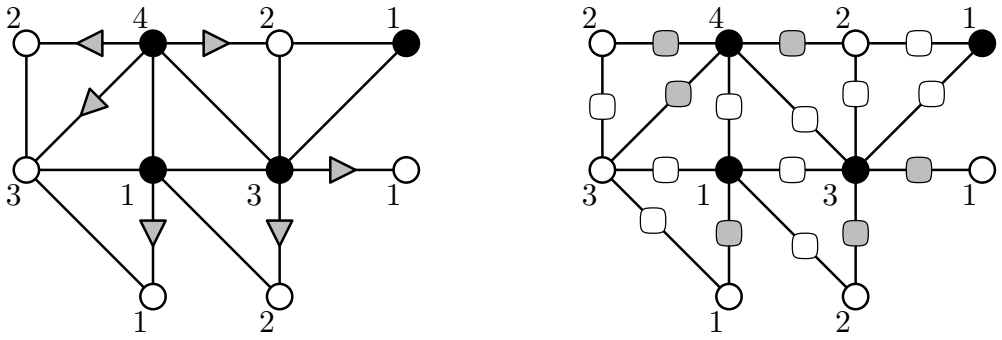


FIGURE 3.5 — Schéma d'une solution du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS dans le graphe G (figure de gauche), et encodage de cette solution dans le graphe d'incidence $I(G)$ (figure de droite).

pour chacun de ces gadgets. Dans ces schémas, les sommets triangulaires noirs sont *présélectionnés*, les autres sommets noirs (circulaires, cliques noires ou marqués d'un disque plein) sont *forcés*, les sommets marqués d'un cercle vide sont *sélectionnables*, et les sommets marqués d'une croix sont *non sélectionnables*.

gadget domination **($\mathcal{D}$).** Un gadget domination est ajouté à H pour chaque sommet-originel du graphe d'incidence $I(G)$. Ce gadget s'assure qu'un sommet-originel est soit sélectionné dans la solution, soit qu'il dispose d'au moins un voisin sélectionné dans $I(G)$, c'est-à-dire que les sommets sélectionnés dans $I(G)$ forment un ensemble dominant de $I(G)$.

Pour chaque sommet-originel $v \in I(G)$, nous ajoutons un sommet v' relié à v et à chaque sommet-arête $e \in N_I(v)$. Nous ajoutons également un ensemble stable de q_0-2 sommets *présélectionnés*, tous de degré 1 et reliés à v'. Enfin, nous ajoutons un sommet *forcé* v'', relié à v', à un ensemble stable de $\Gamma_+(1, \min\sigma) - \min\sigma$ sommets *présélectionnés* tous de degré 1, et à une clique contenant $\min\sigma$ sommets.

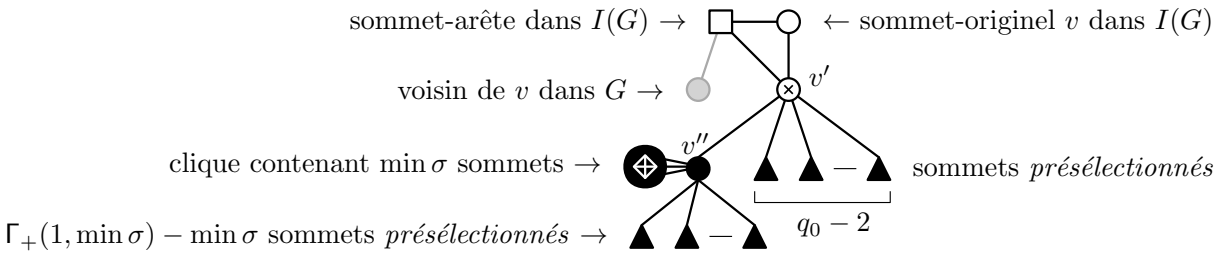


FIGURE 3.6 — Schéma représentant le gadget domination ($\mathcal{D}$).

gadget arête-sélection **($\mathcal{E}$).** Un gadget arête-sélection est ajouté à H pour chaque sommet-arête du graphe d'incidence $I(G)$. Les sommets-arêtes dans $I(G)$ qui seront sélectionnés vont encoder la *fonction de domination* dom de la solution au problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS depuis lequel nous faisons la réduction. Ce gadget s'assure donc qu'un sommet-arête n'est sélectionné que s'il a au moins un voisin, sommet-originel dans I(G), qui est également sélectionné dans la solution.

Pour chaque sommet-arête $e \in I(G)$, nous ajoutons au gadget un ensemble stable de $\Gamma_-(1, q_0+1) - 1$ sommets *présélectionnés*, tous de degré 1 et reliés à e.

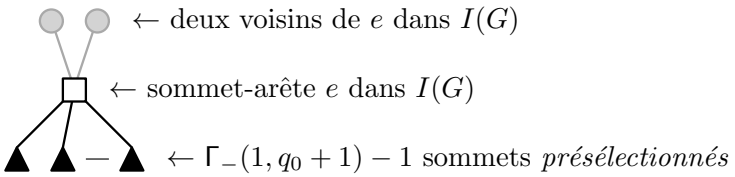


FIGURE 3.7 — Schéma représentant le gadget arête-sélection ($\mathcal{E}$).

gadget capacité **($\mathcal{C}$).** Un gadget capacité est ajouté à H pour chaque sommet-originel du graphe d'incidence $I(G)$. Ce gadget s'assure qu'un sommet-originel sélectionné a au plus $\mathsf{cap}(v)$ voisins de $I(G)$ également sélectionnés, c'est-à-dire que sa capacité dans l'instance initiale est respectée. De plus, ce gadget permet à un sommet-originel sélectionné d'avoir un nombre valide de voisins sélectionnés dans H, au regard des contraintes fixées par l'ensemble σ.

Pour chaque sommet-originel $v \in I(G)$, nous ajoutons un ensemble stable de $\Gamma_+\big(\mathsf{cap}(v), d_G(v) + q_0\big) - \mathsf{cap}(v) - 1$ sommets *présélectionnés*, tous de degré 1 et reliés à v. Nous ajoutons également un ensemble stable de $\mathsf{cap}(v)$ sommets *sélectionnables*, tous reliés à v, et pour chacun de ces sommets, nous ajoutons un ensemble stable de $\Gamma_0(q_0+1) - 1$ voisins *présélectionnés* tous de degré 1.

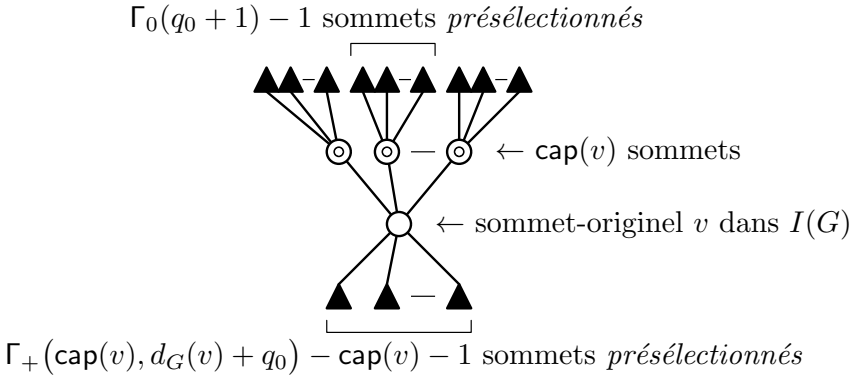


FIGURE 3.8 — Schéma représentant le gadget capacité ($\mathcal{C}$).

gadget satisfiabilité **($\mathcal{S}$).** Un gadget satisfiabilité est ajouté à H pour chaque sommet-originel de $I(G)$. Ce gadget s'assure qu'un sommet-originel de $I(G)$ non sélectionné a un nombre valide de voisins sélectionnés dans H, au regard des contraintes fixées par l'ensemble ρ.

Pour chaque sommet-originel $v \in I(G)$, nous ajoutons un ensemble stable de q_0 sommets *sélectionnables*, et pour chacun de ces sommets, nous ajoutons un ensemble stable de $\Gamma_0(q_0)$ voisins *présélectionnés* tous de degré 1.

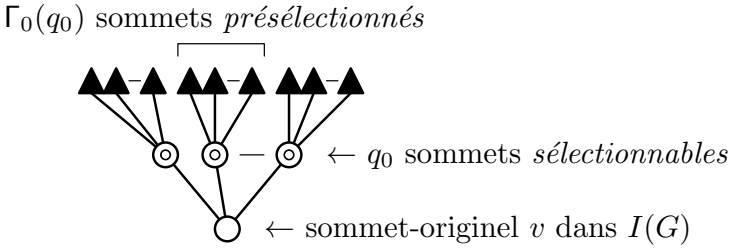


FIGURE 3.9 — Schéma représentant le gadget satisfiabilité ($\mathcal{S}$).

gadget limitation **($\mathcal{L}$).** Un seul gadget limitation global est ajouté au graphe H. Ce gadget limite le nombre de sommets-originels de $I(G)$ pouvant être sélectionnés dans une solution du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, encodant ainsi le paramètre k du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS depuis lequel nous effectuons la réduction.

Nous ajoutons un sommet *central* et *forcé* c, relié à tous les sommets-originels de $I(G)$ et à une clique contenant $\min\sigma$ sommets. Nous ajoutons également un ensemble stable de $\Gamma_+\big(|V(G)|+k, k+\min\sigma\big) - k - \min\sigma$ sommets *présélectionnés*, tous de degré 1 et reliés à c. Enfin, nous ajoutons un autre ensemble stable de k sommets *sélectionnables*, tous reliés à c, et pour chacun de ces sommets, nous ajoutons un ensemble stable de $\Gamma_0(q_0) - 1$ voisins *présélectionnés* tous de degré 1.

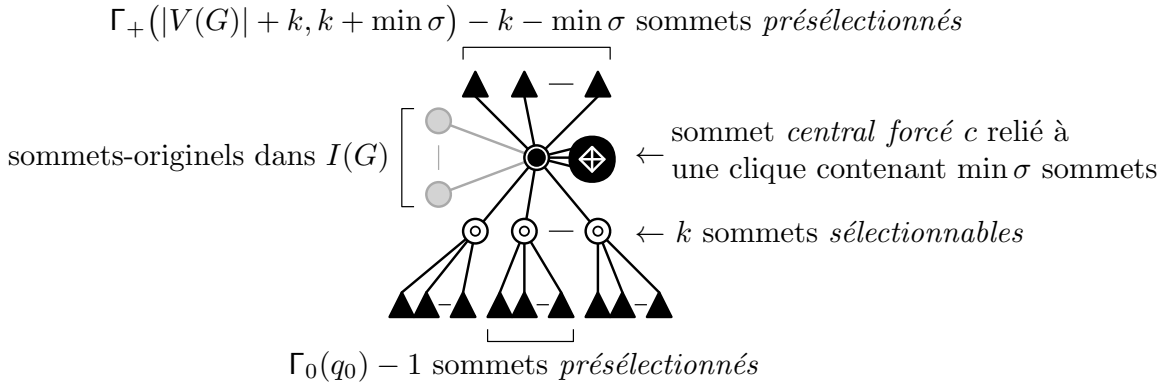


FIGURE 3.10 — Schéma représentant le gadget limitation ($\mathcal{L}$).

3.3.4 Validité de la première étape de réduction

Il nous faut maintenant montrer que cette première étape de réduction est correcte, c'est-à-dire :

- Dans chacun des gadgets, les sommets sont toujours satisfaits au regard des contraintes fixées par les ensembles σ et ρ.
- Chaque gadget joue correctement le rôle qui lui est attribué.
- Le graphe initial G admet un ensemble dominant avec capacités de cardinalité au plus k si et seulement si le graphe construit H admet un ensemble $[\sigma, \rho]$-dominant contenant le sous-ensemble de sommets présélectionnés toujours satisfaits.
- La réduction est FPT.

Validité des gadgets. Nous montrons tout d'abord que chaque gadget joue correctement le rôle qu'il lui est attribué (c'est-à-dire que les gadgets sont corrects), et que les sommets de tous les gadgets sont toujours satisfaits a regard des contraintes fixées par les ensembles σ et ρ. Ainsi, les gadgets n'interfèrent pas dans l'existence ou non d'une solution à la variante ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W, encodant une solution au problème initial k-ENSEMBLE DOMINANT AVEC CAPACITÉS.

Chaque gadget est composé de deux types de sommets : les sommets *forcés* ou *présélectionnés*, et les sommets *sélectionnables* (à l'exception du gadget domination $\mathcal{D}$, contenant également un sommet *non sélectionnable*). Par définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W, les sommets *présélectionnés*, tous de degré 1, doivent appartenir à la solution.

Notons que par la construction des différents gadgets, chaque sommet *sélectionnable* a un certain nombre de voisins lui permettant d'être toujours satisfait, qu'il soit sélectionné ou non dans la solution. En effet, chacun de ces sommets a au total $\Gamma_0(q_0)$ voisins sélectionnés, parce que présélectionnés ou forcés, et par définition $\Gamma_0(q_0) \in \sigma \cap \rho$.

Enfin, chaque sommet *forcé* v est forcé à être sélectionné dans la solution par le fait que ce sommet est relié à une clique contenant $\min \sigma$ sommets. En effet, puisque nous recherchons un ensemble $[\sigma, \rho]$-dominant, et que $0 \notin \rho$, au moins un sommet de la clique contenant $\min \sigma$ sommets doit être sélectionné, ou v doit être sélectionné. Si un sommet de la clique est sélectionné, alors il a besoin d'au moins $\min \sigma$ voisins sélectionnés, et de ce fait tous les sommets de la clique contenant $\min \sigma$ sommets, ainsi que le sommet v, doivent être sélectionnés pour que les sommets de la clique soient tous satisfaits. Si v est sélectionné, alors chaque sommet de la clique a un voisin sélectionné, et comme $1 \notin \rho$ (rappelons que pour des raisons de lisibilité des preuves, nous supposons que $\min \sigma \geq 1$ et $\min \rho \geq 2$), alors au moins un des sommets de la clique doit également être sélectionné. Donc v est dans tous les cas forcé à être sélectionné.

Nous montrons maintenant que chacun des gadgets est correct.

gadget satisfiabilité $\mathcal{S}$**.** Soit $v \in V(H)$ le sommet-originel non sélectionné à contrôler. Il a, dans le gadget satisfiabilité, un ensemble stable de q_0 voisins *sélectionnables*. De ce fait, le sommet v peut toujours avoir au moins q_0 voisins dans H sélectionnés, et sera donc satisfait s'il n'est pas sélectionné (car dans ce cas, le nombre de voisins sélectionnés qu'il doit avoir est contraint par l'ensemble cofini ρ).

gadget arête-sélection $\mathcal{E}$**.** Soit $e \in V(H)$ un sommet-arête à contrôler. Il a, dans le gadget arête-sélection, un ensemble de $\Gamma_-(1, q_0+1)-1$ voisins *présélectionnés* tous de degré

1. Si e n'est pas sélectionné, alors il aura $\Gamma_-(1, q_0+1)-1 \in \rho$ voisins *présélectionnés* dans le gadget, et sera donc satisfait. Si e est sélectionné, alors il aura besoin d'au moins un voisin sélectionné supplémentaire dans H, correspondant à un sommet-originel de $I(G)$, puisque $\Gamma_-(1, q_0+1)-1 \notin \sigma$ et $\Gamma_-(1, q_0+1) \in \sigma$.

gadget capacité $\mathcal{C}$**.** Soit $v \in V(H)$ un sommet-originel dont on veut contrôler la capacité s'il est sélectionné. Il a, dans le gadget capacité, un ensemble de $\Gamma_+\big(\mathsf{cap}(v), d_G(v)+q_0\big) - \mathsf{cap}(v)$ voisins *présélectionnés* ou *forcés*, incluant le sommet *central forcé* du gadget limitation, et peut donc avoir au plus $\mathsf{cap}(v)$ voisins sélectionnés supplémentaires dans H. Les seuls autres voisins qu'il a dans H sont des sommets-arêtes de $I(G)$, et un ensemble stable de $\mathsf{cap}(v)+q_0$ sommets *sélectionnables* dans les gadgets capacité et satisfiabilité, qui peuvent être sélectionnés si v a moins de $\mathsf{cap}(v)$ voisins sélectionnés dans $I(G)$.

Les voisins *sélectionnables* qu'il a dans le gadget capacité ont toujours un nombre valide de voisins sélectionnés dans H. En effet, si l'un d'eux est sélectionné et que v est également sélectionné, alors il aura $\Gamma_0(q_0+1) \in \sigma$ voisins sélectionnés dans H. Si l'un d'eux n'est pas sélectionné, alors il aura $\Gamma_0(q_0+1) \in \rho$ voisins sélectionnés dans H.

Par ce gadget, lorsque le sommet v est sélectionné, il ne peut pas avoir plus de $\mathsf{cap}(v)$ voisins (sommets-arêtes) également sélectionnés dans H, ceux-ci correspondant aux voisins qu'il domine dans G. Au final, v peut avoir exactement $\Gamma_+\big(\mathsf{cap}(v), d_G(v)+q_0\big) \in \sigma \cap \rho$ voisins sélectionnés dans H, et sera donc satisfait.

gadget domination $\mathcal{D}$**.** Soit $v \in V(G)$ le sommet du graphe initial G dont on veut contrôler l'état de domination. Le sommet $v' \in V(H)$ correspondant dans le gadget domination a q_0-2 voisins *présélectionnés* dans ce gadget, tous de degré 1, et un voisin *forcé* supplémentaire avec $\Gamma_+(1, \min\sigma)$ voisins sélectionnés au total ($\min\sigma$ voisins *forcés* dans la clique, et $\Gamma_+(1, \min\sigma) - \min\sigma$ voisins *présélectionnés*), qui empêchent v' d'être sélectionné car sinon ce sommet supplémentaire aurait $\Gamma_+(1, \min\sigma)+1 \notin \sigma$ voisins sélectionnés. De ce fait, v' ne peut pas être sélectionné, et a exactement $q_0-1 \notin \rho$ voisins sélectionnés. Au moins l'un de ses voisins dans $I(G)$ doit être sélectionné pour qu'il puisse être satisfait.

gadget limitation $\mathcal{L}$**.** Soit $c \in V(H)$ le sommet *central forcé* du gadget limitation, dont le but est de limiter le nombre de sommets sélectionnés dans le graphe G à au plus k sommets.

Ce sommet *central* est forcé par une clique contenant $\min\sigma$ sommets. Il a, dans le gadget limitation, un ensemble de $\Gamma_+\big(|V(G)|+k, k+\min\sigma\big) - k$ voisins nécessairement sélectionnés, incluant les $\min\sigma$ sommets *forcés* de la clique, et un ensemble stable de k voisins *sélectionnables*, qui peuvent être sélectionnés si le sommet c a moins de k voisins sélectionnés dans $I(G)$.

De ce fait, c peut avoir au plus $\Gamma_+\big(|V(G)|+k, k+\min\sigma\big)$ voisins sélectionnés dans H, dont au plus k correspondent aux sommets-originels de $I(G)$, car le prochain entier appartenant à σ et plus grand que $\Gamma_+\big(|V(G)|+k, k+\min\sigma\big)$ est également plus grand que le nombre total de sommets-originels de $I(G)$. Si une solution au problème existe, alors le sommet c sera satisfait.

Validité de la réduction. Montrons maintenant que la réduction est correcte.

LEMME 3.3.2
Le graphe initial G admet un ensemble dominant avec capacités de cardinalité au plus k si et seulement si le graphe construit H admet un ensemble $[\sigma, \rho]$-dominant contenant le sous-ensemble $D \subseteq V(G)$ de sommets présélectionnés toujours satisfaits et tous de degré 1.

Démonstration. Supposons dans un premier temps que le graphe H admette un ensemble $[\sigma, \rho]$-dominant D. Soit H_0 l'ensemble des sommets de H correspondant aux sommets-originels de $I(G)$, et soient $D_1, D_2 \subseteq D$ les sommets sélectionnés de H correspondant respectivement aux sommets-originels et aux sommets-arêtes de $I(G)$. On a naturellement $D_1 \subseteq H_0$.

Par le gadget domination $\mathcal{D}$, chaque sommet de H_0 est soit sélectionné (c'est-à-dire qu'il appartient à l'ensemble D_1), soit il a un voisin sélectionné dans D_2, et donc l'ensemble $D_1 \cup D_2$ forme un ensemble dominant (classique) de $I(G)$. Par le gadget arête-sélection $\mathcal{E}$, chaque sommet-arête dans D_2 a au moins un voisin sélectionné dans D_1. Par le gadget capacité $\mathcal{C}$, chaque sommet-originel $v \in V(H_0)$ sélectionné (c'est-à-dire $v \in D_1$) a au plus $\mathsf{cap}(v)$ voisins sélectionnés dans D_2, avec $|N_H(v) \cap D| \in \sigma$. Pour un sommet sélectionné, nous fixons $\mathsf{dom}(v) = \{u \mid u \in N_H(v) \cap D_2\}$. Par le gadget satisfiabilité $\mathcal{S}$, chaque sommet-originel de H_0 non sélectionné (c'est-à-dire $v \in H_0 \setminus D_1$) a au moins q_0 voisins sélectionnés dans H, et de ce fait $|N_H(v) \cap D| \in \rho$. Enfin, par le gadget limitation $\mathcal{L}$, $|D_1| \leq k$.

Par cette construction, (D_1, dom) est un ensemble dominant avec capacités de G de cardinalité au plus k, où dom est la *fonction de domination* de la solution pour les sommets du graphe initial G.

Supposons maintenant que le graphe G admet un ensemble dominant avec capacités (C, dom) de cardinalité au plus k, où dom est la *fonction de domination* de la solution pour les sommets du graphe initial G. Nous allons construire un ensemble $[\sigma, \rho]$-dominant D pour le graphe H. Soit $I(G)$ le graphe d'incidence de G.

Pour chaque sommet $v \in C$, et chaque sommet dominé $u \in V(G)$ tel que $u \in \mathsf{dom}(v)$, nous ajoutons un sommet-arête $e \in V\big(I(G)\big)$ à l'ensemble D, où e correspond à l'arête $\{u, v\}$ de G. Par cette construction, les sommets du graphe d'incidence $I(G)$, ainsi que les sommets des gadgets domination $\mathcal{D}$ et arête-sélection $\mathcal{E}$ sont satisfaits, et comme chacun des autres gadgets contient uniquement des sommets *forcés*, *présélectionnés* ou *sélectionnables* qui sont toujours satisfaits, alors le graphe H admet un ensemble $[\sigma, \rho]$-dominant D. □

LEMME 3.3.3
*La réduction depuis le problème k-*ENSEMBLE DOMINANT AVEC CAPACITÉS *vers le problème* ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W *est* FPT. *Plus précisément, le graphe H est de taille* $\mathrm{poly}\big(|V(G)|\big)$, *et* $\mathrm{tw}(H) \leq 4\mathrm{tw}(G) + \min \sigma + 1$.

Démonstration. Soient σ et ρ deux ensembles d'entiers fixés, faisant partie de la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W (ce ne sont pas des paramètres du problème). Soient G le graphe initial du problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS, et H le graphe construit lors de la réduction vers le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W.

Montrons tout d'abord que la taille du graphe construit H est polynomiale en la taille du graphe initial G.

Un seul gadget limitation $\mathcal{L}$ est créé pour tout le graphe H, et un gadget capacité $\mathcal{C}$ est ajouté pour chaque sommet-originel de $I(G)$. Les cardinalités de ces deux gadgets dépendent des fonctions Γ, et d'un polynôme en la taille de G (ce qui explique notre condition technique sur l'ensemble σ, voir pour cela la SECTION 3.3.2). Les autres gadgets sont ajoutés au graphe H pour chaque sommet-originel ou sommet-arête de $I(G)$, et chacun d'entre eux a une cardinalité dépendant uniquement des fonctions Γ et des deux ensembles σ et ρ.

De ce fait, le nombre total de sommets du graphe H dépend uniquement des ensembles σ et ρ, qui sont fixés et peuvent donc être considérés comme des constantes, et d'un polynôme en le nombre de sommets et arêtes du graphe G. La réduction est donc polynomiale en la taille du graphe G donné en entrée, et le paramètre k.

Montrons maintenant que la largeur arborescente du graphe construit H est polynomiale en la largeur arborescente du graphe initial G. Soit $T\big(I(G)\big)$ une décomposition arborescente optimale du graphe d'incidence $I(G)$, dont la largeur arborescente est au plus celle du graphe G. Nous allons expliquer comment construire une décomposition arborescente du graphe H de largeur bornée par la largeur arborescente de $I(G)$.

Par construction, tous les gadgets sont pratiquement des arbres, exceptés pour leurs sommets *forcés*, qui sont reliés à une clique contenant $\min\sigma$ sommets. De plus, chacun de ces gadgets n'a qu'un seul sommet relié à des sommets-originels et/ou des sommets-arêtes de $I(G)$ (cet unique sommet est ici appelé *racine*) : la *racine* d'un gadget domination $\mathcal{D}$ est reliée à un sommet-originel de $I(G)$ et à chaque sommet-arête adjacent à ce sommet-originel, les *racines* des gadgets satisfiabilité $\mathcal{S}$ et capacité $\mathcal{C}$ sont reliées à un sommet-originel, la *racine* d'un gadget arête-sélection $\mathcal{E}$ est reliée à un sommet-arête, et la *racine* du gadget limitation (son sommet *central*) est reliée à chaque sommet-originel (voir la FIGURE 3.4). En partant de la décomposition arborescente $T\big(I(G)\big)$, nous relions la décomposition arborescente de chaque gadget à l'un des sacs contenant l'un des sommets-originels ou sommets-arêtes de $I(G)$ auquel le gadget est relié *via* sa *racine*. Puis nous ajoutons la *racine* du gadget à tous les sacs de la décomposition arborescente de $T\big(I(G)\big)$ contenant les voisins de cette *racine*. Pour un sommet de $I(G)$, nous ajoutons donc au plus 3 sommets aux sacs contenant ce sommet.

Finalement, chaque gadget est de largeur arborescente au plus $\min\sigma+1$, correspondant aux sommets *forcés* par une clique contenant $\min\sigma$ sommets. Comme pour un sommet donné de $I(G)$, au plus 3 de ses sommets sont ajoutés aux sacs contenant ce sommet, la décomposition arborescente du graphe H est de largeur $\mathrm{tw}(H) \leq 4\cdot\mathrm{tw}(G)+\min\sigma+1$, où la valeur $\min\sigma$ est une constante puisque σ est fixé et fait partie de la définition du problème ENSEMBLE $[\sigma,\rho]$-DOMINANT AVEC ENSEMBLE W.

La réduction est donc FPT lorsque paramétrée par la largeur arborescente de G. □

3.3.5 Seconde étape de réduction

Nous allons maintenant décrire la seconde étape de la réduction depuis le problème k-ENSEMBLE DOMINANT AVEC CAPACITÉS vers notre problème ENSEMBLE $[\sigma,\rho]$-DOMINANT. Il s'agit cette fois de détailler les transformations effectuées sur une instance du problème ENSEMBLE $[\sigma,\rho]$-DOMINANT AVEC ENSEMBLE W, pour obtenir une instance de notre problème ENSEMBLE $[\sigma,\rho]$-DOMINANT.

Réduction. Dans le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W, certains sommets sont présélectionnés dans l'ensemble $[\sigma, \rho]$-dominant, sont tous de degré 1, et sont supposés toujours satisfaits au regard des contraintes fixées par l'ensemble σ. Pour cette seconde étape de réduction, il nous suffit donc de montrer comment forcer ces sommets à être sélectionnés et satisfaits dans une instance de notre problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, en respectant cette fois les contraintes fixées par l'ensemble σ. Remarquons que si $\min \sigma \in \sigma$, alors cette réduction est simple : il suffit de relier chaque sommet que l'on souhaite forcer à une clique contenant $\min \sigma$, pour les mêmes arguments que dans le cas du gadget limitation $\mathcal{L}$ de la réduction précédente.

Étant donné une instance (G, W) du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W, où G est un graphe et $W \subseteq V(G)$ le sous-ensemble de sommets présélectionnés, tous de degré 1 dans le graphe G, nous allons construire un graphe H, instance du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, de telle sorte que les sommets dans H correspondant au sous-ensemble W de sommets de G soient tous *forcés* à être sélectionnés, tout en étant satisfaits au rapport des conditions fixées par σ. Les ensembles σ et ρ ne sont pas modifiés lors de la réduction.

Rappelons que notre objectif final est de montrer que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est $\mathrm{W}[1]$-difficile lorsque paramétré par la largeur arborescente. Nous devons donc, dans cette seconde étape de réduction, nous assurer que la largeur arborescente du graphe construit H soit polynomiale en la largeur arborescente du graphe initial G.

Notons α le plus petit entier commun aux deux ensembles σ et ρ, c'est-à-dire $\alpha = \min\{q \mid q \in \sigma \cap \rho\}$, et notons β l'intervalle entre les deux premiers entiers appartenant à l'ensemble σ, c'est-à-dire $\beta = \min\{p \mid \min \sigma + p \in \sigma\}$. Puisque l'ensemble ρ est cofini, et σ est infini, ces deux entiers sont bien définis.

Nous pouvons maintenant décrire la construction du graphe H. Initialement, le graphe H est composé de α copies du graphe G. Pour chaque copie v_i d'un sommet présélectionné $v \in W$ du graphe G, avec $1 \leq i \leq \alpha$, nous ajoutons au graphe H une clique C_i contenant $\min \sigma$ sommets et reliée au sommet v_i. Enfin, pour chaque sommet présélectionné v de G, nous ajoutons un ensemble stable de β sommets au graphe H, et nous relions chacun de ces sommets à toutes les copies $v_i \in V(H)$ du sommet v de G.

Cette construction est schématisée dans la FIGURE 3.11.

Validité de la réduction. Nous devons maintenant montrer que cette réduction est correcte, en montrant que :

- si H admet un ensemble $[\sigma, \rho]$-dominant D_H, alors chacune des copies $v_i \in V(H)$ d'un sommet présélectionné v de G est nécessairement sélectionné dans D_H ;
- le graphe initial G admet un ensemble $[\sigma, \rho]$-dominant D_G contenant le sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés toujours satisfaits si et seulement si le graphe construit H admet un ensemble $[\sigma, \rho]$-dominant D_H ;
- la réduction est FPT.

Notons que les sommets de chaque clique de cardinalité $\min \sigma$ sont nécessairement satisfaits, pour les mêmes arguments que ceux utilisés dans la SECTION 3.3.4, lors de la démonstration de la validité des différents gadgets.

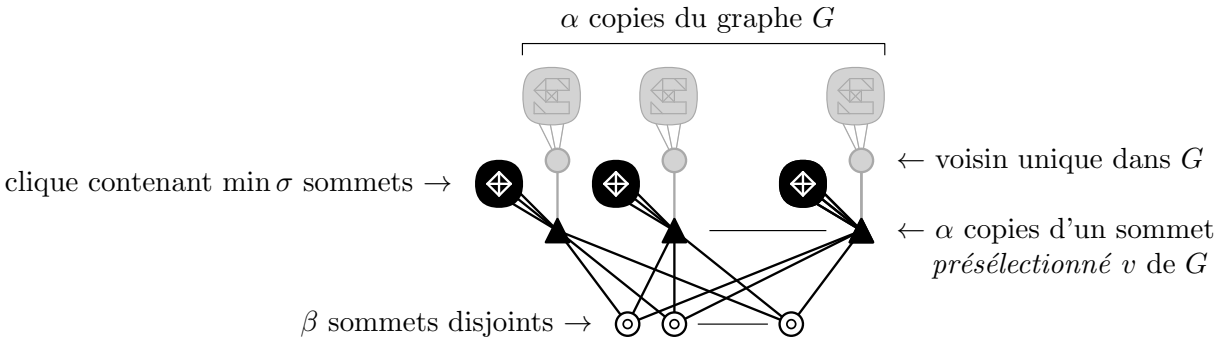


FIGURE 3.11 — Construction reliant, dans le graphe H, les α copies d'un sommet présélectionné v de G à l'aide d'un ensemble stable de β sommets dijsoints.

LEMME 3.3.4
Si le graphe H admet un ensemble $[\sigma,\rho]$-dominant D_H, alors chacune des copies $v_i \in V(H)$ d'un sommet présélectionné v de G est nécessairement sélectionné dans D_H.

Démonstration. Soit $v_i \in V(H)$ un sommet de H, copie d'un sommet présélectionné $v \in W$ de degré 1 dans G, avec $1 \leq i \leq \alpha$. Par construction, ce sommet v_i est relié dans H à une clique contenant $\min\sigma$ sommets. Pour les mêmes arguments que dans la SECTION 3.3.4, où nous montrions qu'un sommet *forcé* est nécessairement sélectionné car il était relié à une clique contenant $\min\sigma$ sommets, le sommet v_i est nécessairement sélectionné dans un ensemble $[\sigma,\rho]$-dominant D_H du graphe H. □

LEMME 3.3.5
Le graphe initial G admet un ensemble $[\sigma,\rho]$-dominant D_G contenant le sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés toujours satisfaits, et tous de degré 1, si et seulement si le graphe construit H admet un ensemble $[\sigma,\rho]$-dominant D_H.

Démonstration. Supposons dans un premier temps que le graphe G admette un ensemble $[\sigma,\rho]$-dominant D_G, contenant le sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés toujours satisfaits et tous de degré 1. Nous allons expliquer comment construire un ensemble $[\sigma,\rho]$-dominant D_H du graphe H.

Initialement, l'ensemble D_H ne contient aucun sommet du graphe H. Pour chaque sommet $u \in V(G)$ appartenant à l'ensemble $[\sigma,\rho]$-dominant D_G, toutes ses copies u_i dans H sont sélectionnées et ajoutées à l'ensemble D_H. De même, pour chaque sommet $u' \in V(G)$ n'appartenant pas à l'ensemble $[\sigma,\rho]$-dominant D_G, aucune de ses copies u'_i dans H n'est sélectionnée, et donc $u'_i \notin D_H$. Comme l'ensemble D_G est un ensemble $[\sigma,\rho]$-dominant de G, et que le voisinage sélectionné dans H de chaque copie u_i est le même, tous ces sommets sont satisfaits.

Considérons un sommet $v_i \in V(H)$, copie d'un sommet présélectionné $v \in W$ de G, et notons u_i la copie dans H du voisin u de v. Si le sommet u_i n'a pas été ajouté à l'ensemble D_H, alors aucun des voisins de v_i appartenant à l'ensemble stable de cardinalité β n'est ajouté à D_H. Si le sommet u_i a été ajouté à l'ensemble D_H, alors tous les voisins de v_i appartenant à l'ensemble stable de cardinalité β sont ajoutés à D_H.

Montrons que l'ensemble D_H est bien un ensemble $[\sigma, \rho]$-dominant de H. Rappelons que dans le graphe G, le sommet v est de degré 1, et n'a donc qu'un seul voisin que l'on notera u. Les seuls voisins du sommet v_i dans H sont donc la copie u_i du sommet u, les sommets de la clique contenant $\min \sigma$ sommets, et les sommets de l'ensemble stable de cardinalité β ajoutés lors de la construction du graphe H.

Considérons les deux cas possibles pour son voisin u_i, selon qu'il soit sélectionné ou non. Si u_i n'est pas sélectionné, alors le sommet v_i est satisfait lorsque seuls ses voisins appartenant à la clique de $\min \sigma$ sommets sont sélectionnés, puisqu'il a alors $\min \sigma \in \sigma$ voisins sélectionnés dans H. Si u_i est sélectionné, alors le sommet v_i est satisfait si tous ses autres voisins sont également sélectionnés, c'est-à-dire ses voisins appartenant à la clique de $\min \sigma$ sommets, et ses β autres voisins dans H, puisqu'il a alors $\min \sigma + \beta \in \sigma$ voisins sélectionnés dans H. Dans tous les cas, son voisin u_i est également satisfait, puisqu'il est la copie d'un sommet u de G supposé satisfait par l'ensemble $[\sigma, \rho]$-dominant D_G de G. Enfin, remarquons que les β sommets disjoints ont exactement $\alpha \in \sigma \cap \rho$ voisins sélectionnés, qui sont les sommets v_j pour $1 \leq j \leq \alpha$. De ce fait, chacun d'eux est toujours satisfait, qu'il soit sélectionné ou non dans l'ensemble D_H.

L'ensemble D_H que nous venons de construire est donc un ensemble $[\sigma, \rho]$-dominant du graphe H.

Supposons maintenant que le graphe H admette un ensemble $[\sigma, \rho]$-dominant D_H. Nous allons montrer comment obtenir un ensemble $[\sigma, \rho]$-dominant D_G du graphe G, contenant le sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés toujours satisfaits.

Par le LEMME 3.3.4, toutes les copies v_i dans H d'un sommet $v \in W$ du graphe G sont nécessairement sélectionnés, et comme D_H est un ensemble $[\sigma, \rho]$-dominant de H, tous ces sommets sont satisfaits. Notons H_1 l'une des copies du graphe G dans le graphe H. Prenons $D_G = D_H \cap V(D_{H_1})$, c'est-à-dire que l'ensemble D_G ne contient que les sommets de l'ensemble $[\sigma, \rho]$-dominant D_H appartenant à la copie H_1 du graphe G. Puisque D_H est en particulier un ensemble $[\sigma, \rho]$-dominant de la copie H_1 de G, alors D_G est bien un ensemble $[\sigma, \rho]$-dominant de G, contenant le sous-ensemble $W \subseteq V(G)$ de sommets présélectionnés toujours satisfaits. □

LEMME 3.3.6
La réduction est FPT. *Plus précisément, le graphe H est de taille polynomiale en celle du graphe G, et* $\mathrm{tw}(H) \leq (\min \sigma + \beta) \cdot \mathrm{tw}(G)$.

Démonstration. Soient σ et ρ deux ensembles d'entiers fixés, faisant partie de la définition du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT (ce ne sont pas des paramètres du problème). Soient (G, W) l'instance du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT AVEC ENSEMBLE W, et soit H l'instance du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT.

Soit $n = |V(G)|$. Le graphe H est construit à partir de α copies du graphe G, auxquelles sont ajoutées au plus n cliques de $\min \sigma$ sommets chacune (une clique pour chaque sommet présélectionné de chaque copie). Enfin, au plus n ensembles stables de β sommets chacuns sont ajoutés au graphe H, un ensemble pour chaque ensemble de copies d'un sommet présélectionné.

En résumé, nous obtenons $|V(H)| \leq |V(G)| \cdot \big((\alpha \cdot \min \sigma) + \beta\big)$.

Montrons maintenant que la largeur arborescente du graphe H est polynomiale en la largeur arborescente du graphe G. Pour cela, supposons que (T_G, χ_G) est une décompo-

sition arborescente du graphe G, de largeur $\mathrm{tw}(G)$, et notons (T_H, χ_H) la décomposition arborescente du graphe H que nous allons construire.

Tout d'abord, $T_H = T_G$. Pour chaque sac $X_i \in \chi_G$ de la décomposition arborescente du graphe G, nous construisons le sac correspondant $X'_i \in \chi_H$ de la décomposition arborescente du graphe H en ajoutant au sac X'_i toutes les copies $u_i \in V(H)$ de chaque sommet $u \in V(G)$ appartenant au sac X_i. De plus, si le sommet $u \in V(G)$ est un sommet présélectionné, alors nous ajoutons au sac X'_i :

- tous les sommets de chaque clique de cardinalité $\min \sigma$ reliée à chaque copie $u_i \in V(H)$ du sommet $u \in V(G)$;
- tous les sommets de l'ensemble stable de cardinalité β reliant toutes ces copies dans H.

Nous obtenons alors une décomposition arborescente du graphe H, et par construction, $\mathrm{tw}(H) \leq (\min \sigma + \beta) \cdot \mathrm{tw}(G)$.

La réduction est donc FPT lorsque paramétrée par la largeur arborescente du graphe initial G. □

La décomposition arborescente du graphe H que nous obtenons n'est peut-être pas optimale. Cela n'a cependant pas de conséquence sur notre résultat, puisque nous devons simplement nous assurer que la largeur arborescente du graphe construit H est polynomiale en la largeur arborescente du graphe initial G.

3.4 Complexité dans le cas d'ensembles quelconques

Nous nous sommes jusqu'à maintenant intéressé à la complexité du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT pour certains cas particuliers d'ensembles σ et ρ. Qu'en est-il de la complexité du problème dans le cas d'ensembles σ et ρ plus généraux ?

3.4.1 Paramétré par la largeur arborescente (*tree-width*)

Comme nous l'indiquions dans la SECTION 1.1.3, l'étude de la complexité paramétrée d'un problème n'est pertinente que si celui-ci est dans la classe XP. Nous montrons ici que c'est le cas pour notre problème, ENSEMBLE $[\sigma, \rho]$-DOMINANT paramétré par la largeur arborescente.

THÉORÈME 3.4.1
Soient σ et ρ deux ensembles d'entiers récursifs, pour lesquels l'appartenance d'un entier à ces ensembles peut être décidée en temps polynomial. Alors le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT*, ainsi que les versions d'optimisation* ENSEMBLE $[\sigma, \rho]$-DOMINANT MINIMUM *et* ENSEMBLE $[\sigma, \rho]$-DOMINANT MAXIMUM*, sont dans la classe* XP *lorsque paramétrés par la largeur arborescente du graphe donné en entrée.*

Démonstration. Soit $G = (V, E)$ le graphe donné en entrée au problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Un ensemble $[\sigma, \rho]$-dominant est bien entendu de cardinalité au plus $|V|$. De ce fait, nous pouvons considérer le problème ENSEMBLE $[\sigma', \rho']$-DOMINANT, où $\sigma' = \sigma \cap \{0, \ldots, |V|\}$ et $\rho' = \rho \cap \{0, \ldots, |V|\}$ sont tous deux finis. L'appartenance d'un entier t aux ensembles σ ou ρ pouvant être décidée en temps $\mathcal{O}(\mathrm{poly}(t))$, les deux ensembles σ' et ρ' peuvent être construits en temps polynomial.

En utilisant notre algorithme FPT présenté dans la SECTION 3.2, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps XP lorsque paramétré par la largeur arborescente de G, puisque dans ce cas la complexité paramétrée de notre algorithme est en $\mathcal{O}(n^{f(k)})$. Notons que cet algorithme peut être facilement adapté pour résoudre les problèmes ENSEMBLE $[\sigma, \rho]$-DOMINANT MINIMUM ou ENSEMBLE $[\sigma, \rho]$-DOMINANT MAXIMUM avec la même complexité paramétrée. □

3.4.2 Paramétré par la largeur arborescente (*tree-width*) et la taille de la solution

Pour ce type de problème sur les graphes, consistant à trouver un sous-ensemble respectant une certaine propriété, il est naturel de considérer également la taille de la solution comme paramètre. Nous allons voir ici que l'ajout de ce paramètre modifie sensiblement la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, celui-ci devenant FPT lorsqu'il est paramétré par la largeur arborescente du graphe et par la taille de la solution désirée.

THÉORÈME 3.4.2
*Soient σ et ρ deux ensembles d'entiers résursifs. Alors le problème k-*ENSEMBLE $[\sigma, \rho]$-DOMINANT *est* FPT *lorsque paramétré par la cardinalité maximum k de l'ensemble $[\sigma, \rho]$-dominant, et par la largeur arborescente du graphe donné en entrée.*

Démonstration. Le problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT consiste à trouver un ensemble $[\sigma, \rho]$-dominant de cardinalité au plus k. De ce fait, chaque sommet du graphe donné en entrée aura au plus k voisins appartenant à cet ensemble $[\sigma, \rho]$-dominant. Le problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT est donc équivalent au problème ENSEMBLE $[\sigma', \rho']$-DOMINANT, où $\sigma' = \sigma \cap \{0, \ldots, k\}$ et $\rho' = \rho \cap \{0, \ldots, k\}$ sont tous deux finis. Si $a_\sigma(k)$ représente le temps maximum nécessaire pour décider si $t \in \sigma$ avec $t \leq k$ un entier, alors l'ensemble d'entier σ' peut être construit en temps $\mathcal{O}(k \cdot a(k))$, ne dépendant pas de la taille du graphe donné en entrée. De même, ρ' peut être construit en un temps ne dépendant pas de la taille du graphe donné en entrée. Puisque les deux ensembles σ et ρ sont fixés, faisant partie de la définition du problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT (ce ne sont pas des paramètres), le temps de construction des ensembles σ' et ρ' est FPT.

En utilisant notre algorithme FPT présenté dans la SECTION 3.2, le problème k-ENSEMBLE $[\sigma, \rho]$-DOMINANT peut être résolu en temps FPT paramétré par la cardinalité maximum k de l'ensemble $[\sigma, \rho]$-dominant et par la largeur arborescente du graphe donné en entrée, puisque dans ce cas la complexité paramétrée de notre algorithme est en $\mathcal{O}\big(f\big(a_\sigma(k), a_\rho(k)\big) \cdot \mathrm{poly}(n)\big)$. □

3.5 Conclusion

Dans l'étude de nombreux problèmes en théorie des graphes, il est de coutume de considérer la largeur arborescente comme paramètre afin que le problème soit, dans ce cas, FPT. Il en va de même pour le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, dont la complexité paramétrée appartient à la classe FPT, lorsque paramétré par la largeur arborescente du graphe donné en entrée, et ce pour tous les cas jusqu'à maintenant étudiés dans la littérature pour les ensembles d'entiers σ et ρ.

La largeur arborescente comme paramètre amène-t-elle toujours le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT à être FPT, quelques soient les ensembles d'entiers σ et ρ ? Dans ce chapitre, nous avons étudié d'autres cas d'ensembles σ et ρ, et montré que cela n'est pas toujours le cas : il existe une infinité d'ensembles d'entiers σ et ρ pour lesquels le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT devient W[1]-difficile lorsque paramétré par la largeur arborescente (en intégrant toutefois une contrainte technique sur l'ensemble σ).

Il est à noter qu'auparavant, très peu de problèmes étaient connus pour ne pas être FPT lorsque paramétrés par la largeur arborescente (voir notamment [FFL+07, DLSV08, Lok09]), tout en étant pertinent du point de vue paramétré (c'est-à-dire au moins dans la classe XP). Le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, pour ces nouveaux cas d'ensembles σ et ρ, vient dès lors rejoindre cette liste.

Par ailleurs, les cas jusqu'à maintenant étudiés dans la littérature montrent que le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT appartient à la classe FPT lorsque σ et ρ sont des ensembles d'entiers finis ou cofinis. Nous avons dans ce chapitre étendu ces résultats, et montré *via* un résultat de COURCELLE *et al.* que c'est également le cas lorsque σ et ρ sont des ensembles d'entiers ultimement périodiques.

Nos différents résultats obtenus dans ce chapitre améliorent notre connaissance de la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT. Cette connaissance n'est toutefois pas complète, et diverses perspectives d'étude sont à envisager. Pour notre résultat de W[1]-difficulté, nous nous sommes focalisés sur l'ensemble σ ; il serait intéressant d'étudier également l'impact de différents cas pour l'ensemble ρ sur la complexité paramétrée du problème. Par ailleurs, les ensembles d'entiers σ que nous avons considéré contiennent des intervalles de longueur arbitraire ; que devient la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT lorsque σ ne contient que des intervalles de longueur bornée ? Nous pourrions également considérer d'autres paramètres, et étudier là encore la complexité paramétrée du problème.

Nous avons considéré ici une certaine *classe* de problèmes, que nous avons montré comme n'étant pas FPT lorsque paramétrés par la largeur arborescente. Nous pouvons envisager que d'autres *classes* de problèmes puissent l'être également.

Coloration additive

4

Sommaire

4.1 Introduction

Dans ce dernier chapitre, nous allons nous intéresser à la complexité d'un nouveau problème de coloration, combinant théorie des graphes et théorie des nombres, en introduisant une contrainte arithmétique sur les couleurs pouvant être affectées à des sommets *proches*.

Pour une grande partie de ce chapitre, nous nous écartons quelque peu du fil conducteur de cette thèse qu'est la largeur arborescente. Ce problème étant nouveau, il est en effet important d'en étudier dans un premier temps la complexité en général, et en particulier déterminer s'il est NP-complet, et dans quels cas il peut être approximé.

Les résultats de ce chapitre sont le fruit de travaux effectués avec NATACHA ASTROMUJOFF, MARTÍN MATAMALA, IOAN TODINCA et JOSÉ ZAMORA PONCE. Ils font l'objet d'un article actuellement soumis.

4.1.1 Colorations de graphe

Tandis que les graphes sont des objets mathématiques permettant de modéliser les situations rencontrées dans de nombreux domaines scientifiques, les colorations de graphe permettent de modéliser de nombreux problèmes survenant dans ces domaines, qu'ils soient théoriques ou appliqués.

D'un point de vue général, une coloration d'éléments d'un graphe (principalement sommets ou arêtes) consiste à affecter à chacun de ces éléments une couleur, ou de manière équivalente un entier naturel, de sorte que les couleurs affectées à des éléments *proches* respectent certaines contraintes. La notion formelle de cette *proximité* et des contraintes associées dépendent de la définition du problème de coloration.

Pour illustrer l'utilisation d'un problème de coloration pour résoudre un problème réaliste et modélisé par un graphe, considérons la situation suivante : un certain nombre d'antennes radio sont installées en différents endroits d'une région. Nous souhaitons allouer une fréquence à chacune de ces antennes, tout en évitant de possibles interférences. Deux antennes proches, au sens de la distance, devront donc chacune recevoir une fréquence distincte. Le coût de location d'une fréquence étant relativement élevé, nous souhaitons minimiser le nombre de fréquences différentes utilisées.

Cette situation peut se modéliser simplement sous la forme d'un problème de coloration : chaque antenne correspond à un sommet du graphe, et deux sommets sont reliés si les deux antennes correspondantes sont proches, c'est-à-dire que leurs zones de portée s'intersectent et que des interférences pourraient survenir. Pour résoudre ce problème, nous recherchons une affectation d'un nombre minimum de couleurs différentes aux sommets de ce graphe, de sorte que deux sommets adjacents reçoivent des couleurs distinctes. Chacune des couleurs utilisées correspond alors à une fréquence, et l'affectation des couleurs au sommets du graphe donne une allocation des fréquences aux antennes évitant les interférences.

Bien que cette situation soit relativement simple, il s'avère que la résolution optimale de ce problème des *difficile* (NP-difficile) : il n'existe pas d'algorithme polynomial permettant de trouver une affectation d'un nombre minimum de couleurs différentes aux sommets du graphe, sauf si $\mathsf{P} = \mathsf{NP}$. Plus encore, si l'on fixe le nombre maximum $K \in \mathbb{N}$

de couleurs différentes que l'on souhaite utiliser, il n'est pas possible de déterminer si une affectation correcte des couleurs existe (sauf lorsque $K \leq 2$).

De nombreux problèmes de coloration connus, tel que COLORATION (coloration classique des sommets), ARÊTES COLORATION (coloration classique des arêtes), ou encore COLORATION INJECTIVE (coloration à distance 2) partagent cette même propriété : il n'existe pas d'algorithme polynomial permettant de trouver une affectation optimale des couleurs, même lorsque le nombre maximum de couleurs différentes est fixé, sauf si $\mathsf{P} = \mathsf{NP}$.

MATAMALA et ZAMORA PONCE [MZP10] ont récemment introduit une nouvelle notion de coloration de graphes, appelée COLORATION ADDITIVE, et définie plus formellement dans la SECTION 4.1.4. Dans cette coloration, les couleurs sont affectées aux sommets du graphe, et chaque arête reçoit comme valeur la différence (en valeur absolue) des deux couleurs associées aux deux sommets incidents à cette arête. La coloration des sommets doit être telle que pour chaque arête du graphe, la valeur reçue par cette arête soit distincte de celles reçues par les arêtes adjacentes.

Ce problème combinant théorie des graphes et théorie des nombres, il est intéressant d'étudier les conséquences et interactions créées par l'association des problématiques provenant de ces deux domaines, notamment sur la complexité théorique du problème.

4.1.2 État de l'art

Étant fondamentaux en théorie des graphes, les différents problèmes de coloration, en particulier COLORATION et ARÊTES COLORATION, ont fait l'objet de très nombreux travaux de recherche. Il serait déraisonnable de vouloir les citer tous, c'est pourquoi nous allons nous focaliser sur les principaux résultats liés à l'étude du problème COLORATION ADDITIVE.

Le problème COLORATION est l'un des tout premiers problèmes dont la complexité classique a été étudiée. Il fait partie des vingt-et-un problèmes fondamentaux ayant été démontrés NP-complets par KARP [Kar72]. Comme nous l'expliquions dans la SECTION 1.1.3, KARP [Kar72] a en fait montré que ce problème est NP-complet même lorsque le nombre de couleurs $K \geq 3$ est fixé, et de ce fait le problème paramétré k-COLORATION est paraNP-complet.

Ce n'est que plus récemment que l'*approximabilité* de ce problème, c'est-à-dire la possibilité ou non de calculer en temps polynomial une valeur proche du nombre minimum de couleurs nécessaires, a été étudiée. Ainsi, LUND et YANNAKAKIS [LY94] ont montré que le problème COLORATION n'est pas approximable à facteur n^{ϵ}, pour tout $0 < \epsilon < 1$, sauf si $\mathsf{P} = \mathsf{NP}$. L'inapproximabilité de ce problème a également été étudiée par KHANNA *et al.* [KLS00], qui ont donné une preuve plus simple du résultat précédent, et apporté une meilleure compréhension des raisons de cette inapproximabilité. Finalement, FEIGE et KILLIAN [FK98] ont montré que le problème COLORATION ne peut pas être approximé à facteur $n^{1-\epsilon}$, pour tout $\epsilon > 0$, sauf si $\mathsf{NP} = \mathsf{ZPP}$. Cette condition peut être renforcée et remplacée par $\mathsf{P} = \mathsf{NP}$, par un résultat récent de ZUCKERMANN [Zuc06].

Le problème ARÊTES COLORATION est assez similaire au problème COLORATION, la différence provenant du fait qu'il s'agit cette fois d'affecter des couleurs aux arêtes.

Sans surprise, et à l'instar du problème COLORATION, le problème ARÊTES COLORATION est NP-complet pour tout nombre fixé $K \geq 3$ de couleurs, par un résultat de LEVEN et GALIL [LG83]. Remarquons que la NP-complétude de ce problème avait été précédemment montrée pour un nombre fixé $K = 3$ de couleurs, par HOLYER [Hol81].

Il est également intéressant de noter que LEVEN et GALIL [LG83] ont montré que pour tout graphe k-régulier, avec $k \geq 3$, il est NP-complet de décider si le nombre de couleurs nécessaires pour colorier les arêtes de ce graphe est k ou $k + 1$.

Le problème COLORATION INJECTIVE introduit récemment par HAHN *et al.* [HKvS02], est un autre problème de coloration lié à notre problème COLORATION ADDITIVE. Ce problème est similaire au problème COLORATION, mais cette fois ce sont les sommets à distance 2 qui doivent recevoir des couleurs différentes (l'affectation des couleurs à des sommets directement voisins est libre).

Dans leur article introductif, KAHN *et al.* [HKvS02] ont montré qu'à l'instar du problème classique COLORATION, le problème COLORATION INJECTIVE est NP-complet pour tout nombre fixé $K \geq 3$ de couleurs.

Plusieurs résultats relatifs au problème COLORATION INJECTIVE ont depuis été publiés, notamment sur la possibilité de résoudre efficacement ce problème sur les graphes planaires [BCRW09, LvT09], ou d'autres classes de graphes [CKY10, DHR10, CHRW11].

Le résultat qui nous intéresse le plus ici est celui de HELL *et al.* [HRS08], qui ont montré que le problème COLORATION INJECTIVE est inapproximable à facteur $n^{1/3-\epsilon}$, pour tout $\epsilon > 0$, sauf si NP $=$ ZPP, en utilisant le résultat de FEIGE et KILLIAN [FK98] concernant l'inapproximabilité du problème ARÊTES COLORATION. Par le résultat de ZUCKERMANN [Zuc06] cité précédemment, nous savons que cette condition peut être renforcée et remplacée par P $=$ NP.

Enfin, il est intéressant de citer certains des résultats de théorie des nombres, relatifs aux ensembles sans 3-progression arithmétique, c'est-à-dire ne contenant pas trois entiers $x, y, z \in \mathbb{N}$ tels que $|y - x| = |z - y|$. En effet, certaines des contraintes fixées par la définition du problème COLORATION ADDITIVE sont liées à ces ensembles, et les différents résultats de théorie des nombres ont donc une incidence sur l'étude de la complexité de ce problème.

Pour un entier n fixé, la problématique consiste à déterminer la cardinalité maximum d'un sous-ensemble de $\{1, \ldots, n\}$ ne contenant pas de 3-progression arithmétique (plusieurs formulations équivalentes existent ; nous donnons ici la formulation liée au problème COLORATION ADDITIVE).

L'étude des ensembles sans 3-progression arithmétique remonte à plus de soixante-dix ans, et a été initiée par ERDŐS et TURÁN [ET36]. Par la suite, plusieurs résultats ont suivi, afin de déterminer d'une part une borne inférieure sur cette cardinalité maximale, par les travaux de ROTH [Rot52, Rot53, Rot54], HEATH-BROWN [HB87], et SZEMEREDI [Sze90], et d'autre part une borne supérieure, par les travaux de SALEM et SPENCER [SS42], BEHREND [Beh46], BOURGAIN [Bou99, Bou08], et SANDERS [San11a]. Les résultats les plus récents, donnant respectivement une borne inférieure et une borne supérieure sur cette cardinalité, proviennent de ELKIN [Elk10] (voir aussi [GW10]) et de SANDERS [San11b].

Il est en effet étonnant de constater que depuis son introduction, la valeur exacte de cette cardinalité n'est toujours pas connue.

Ce problème de théorie des nombres est en lien direct avec le problème COLORATION ADDITIVE. En effet, résoudre le problème COLORATION ADDITIVE pour une clique de taille $K \in \mathbb{N}$ est équivalent à déterminer l'ensemble d'entiers $\{1, \ldots, l\}$ de plus grande cardinalité sans 3-progression arithmétique, et l le plus petit possible.

Sur cette base, GASARCH *et al.* [GGK08] ont donné la valeur exacte pour $K \leq 41$, et donné des bornes inférieures et supérieures pour $K \leq 100$.

4.1.3 Résultats de ce chapitre

Nous allons effectuer une première étude de la complexité théorique du problème COLORATION ADDITIVE, qu'il s'agisse de sa complexité en calcul exact (NP-difficulté et polynomialité), ou de la possibilité de l'approximer.

Dans un premier temps, nous nous intéressons à la complexité classique du problème COLORATION ADDITIVE.

Nous montrons dans la SECTION 4.2 que résoudre de manière exact le problème k-COLORATION ADDITIVE, c'est-à-dire déterminer la plus petite valeur maximale des couleurs à utiliser, est NP-complet pour les graphes de degré maximum $k \geq 4$ même lorsque k est fixé (le cas où $k = 3$ reste quant à lui ouvert). Pour cela, nous utilisons la méthode classique de preuve de NP-complétude, et effectuons une réduction depuis le problème k-ARÊTES COLORATION, connu comme étant NP-complet même lorsque le nombre de couleurs pour les arêtes est fixé à 3.

THÉORÈME
Pour tout entier $k \geq 4$, le problème k-COLORATION ADDITIVE est NP*-complet sur les graphes de degré maximum k.*

Nous étudions ensuite, dans la SECTION 4.3, la possibilité ou non d'approximer la valeur optimale du problème COLORATION ADDITIVE.

En utilisant un résultat similaire déjà connu pour le problème ARÊTES COLORATION, nous montrons que pour tout $\epsilon > 0$, ce problème ne peut pas être approximé à facteur $\mathcal{O}(n^{1/3-\epsilon})$ même pour un graphe séparant, mais qu'il peut être approximé à facteur $\mathcal{O}(n^{1/3+\epsilon})$. Pour rappel, un graphe séparant est composé d'un ensemble stable et d'une clique, reliés par des arêtes de façon quelconque.

THÉORÈME
Soit G un graphe, et $n = |V(G)|$.
Pour tout $0 < \epsilon < 1$, l'index chromatique additif $\chi'_a(G)$ de G est approximable à facteur $n^{1/3+\epsilon}$. Il n'est pas approximable à facteur $n^{1/3-\epsilon}$, même si G est un graphe séparant, sauf si P $=$ NP.

Enfin, dans la SECTION 4.4, nous étudions quelques cas pour lesquels le problème COLORATION ADDITIVE peut être résolu efficacement.

D'une part, à l'aide d'un résultat théorique de COURCELLE [Cou97], nous montrons que le problème k-COLORATION ADDITIVE peut être résolu en temps FPT, paramétré par la largeur arborescente du graphe donné en entrée, lorsque la valeur maximale k des couleurs est fixée.

THÉORÈME
*Le problème k-*COLORATION ADDITIVE *peut être résolu en temps* FPT*, paramétré par le nombre k de couleurs et la largeur arborescente du graphe donné en entrée.*

D'autre part, nous présentons un algorithme polynomial permettant de déterminer la valeur optimale du problème COLORATION ADDITIVE lorsqu'il est restreint aux arbres. Au vu du résultat précédent, l'étude de ce problème pour les arbres est pertinente, puisque ceux-ci correspondent aux graphes de largeur arborescente 1.

THÉORÈME
Le problème COLORATION ADDITIVE *peut être résolu en temps polynomial sur les arbres.*

4.1.4 Quelques colorations

Le problème COLORATION ADDITIVE est un problème de coloration de graphe.

Rappelons qu'une coloration d'éléments d'un graphe consiste à affecter à chacun de ces éléments une couleur, de sorte que celles-ci respectent certaines contraintes au regard des couleurs affectées aux éléments *proches*. Il convient dès lors de définir formellement la notion de *proximité*, et ce pour les différentes problèmes de coloration que nous allons rencontrer ici.

Nous parlerons de *nombre chromatique* lorsque les couleurs sont associées aux sommets d'un graphe, et d'*index chromatique* lorsqu'elles sont associées aux arêtes.

Nous allons dans cette section introduire les définitions de quelques problèmes de coloration, afin de situer le problème COLORATION ADDITIVE parmi ceux-ci. Il existe de nombreuses autres formes de coloration, mais nous nous contentons ici de définir les problèmes de coloration que nous utiliserons par la suite dans ce chapitre.

Coloration classique. Historiquement, la première notion de coloration de graphe associe une couleur à chaque sommet d'un graphe de sorte que deux sommets adjacents reçoivent deux couleurs distinctes.

Plus formellement, une « coloration » (classique) d'un graphe est définie ainsi :

DÉFINITION 4.1.1 (coloration)
Soit $G = (V, E)$ un graphe. Une coloration *(classique) du graphe G est une fonction $\varphi : V(G) \to \mathbb{N}^*$ qui, à chaque sommet de G, associe un entier positif appelé* couleur *du sommet tel que pour toute paire de sommets $u, v \in V(G)$ adjacents dans G, $\varphi(u) \neq \varphi(v)$.*

Le problème de coloration associé, appelé COLORATION, consiste à déterminer le « nombre chromatique » du graphe donné en entrée.

DÉFINITION 4.1.2 (nombre chromatique)
Pour un graphe $G = (V, E)$, le nombre chromatique $\chi(G)$ *est égal au plus petit entier k tel qu'il existe une coloration (classique) de G n'utilisant que les couleurs parmi $\{1, \ldots, k\}$.*

Ce problème est l'un des premiers à avoir été montré NP-complet, par KARP [Kar72].

En fait, il est déjà NP-complet de déterminer si un graphe k-régulier admet une coloration (classique) des sommets avec un nombre fixé $k \geq 3$ de couleurs [Kar72]. Comme

nous en discutions dans la SECTION 1.1.3, cela implique que le problème k-COLORATION est paraNP-complet lorsque paramétré par le nombre maximum k de couleurs à utiliser, et qu'il n'est donc pas pertinent d'étudier la complexité paramétrée de ce problème avec k comme paramètre.

Arêtes-coloration. La notion d'« arêtes-coloration » consiste à associer une couleur à chaque arête d'un graphe, de sorte que deux arêtes adjacentes reçoivent deux couleurs distinctes. Plus formellement, une « arêtes-coloration » d'un graphe est définie ainsi :

DÉFINITION 4.1.3 (arêtes-coloration)
Soit $G = (V, E)$ un graphe. Une arêtes-coloration *du graphe G est une fonction $\varphi' : E(G) \to \mathbb{N}^*$ qui, à chaque arête de G, associe un entier positif appelé* couleur *de l'arête tel que pour toute paire d'arêtes $e_1, e_2 \in E(G)$ adjacentes dans G, $\varphi'(e_1) \neq \varphi'(e_2)$.*

Le problème de coloration associé, appelé ARÊTES-COLORATION, consiste à déterminer l'« index chromatique » du graphe donné en entrée.

DÉFINITION 4.1.4 (index chromatique)
*Pour un graphe $G = (V, E)$, l'*index chromatique *$\chi'(G)$ est égal au plus petit entier k tel qu'il existe une arêtes-coloration de G n'utilisant que les couleurs parmi $\{1, \ldots, k\}$.*

À l'instar de la coloration classique, ce problème est NP-complet [Hol81, LG83], et il est également NP-complet de déterminer si un graphe k-régulier admet une coloration des arêtes avec un nombre fixé $k \geq 3$ de couleurs [Hol81, LG83]. Le problème k-ARÊTES-COLORATION est donc paraNP-complet lorsque paramétré par le nombre maximum k de couleurs à utiliser.

En fait, VIZING [Viz65] a montré que l'index chromatique d'un graphe dont le degré maximum est Δ est égal soit à Δ, soit à $\Delta + 1$. Il est donc NP-complet de décider entre ces deux valeurs pour l'index chromatique d'un graphe.

Coloration injective. Les deux problèmes de coloration définis ci-dessus fixent une contrainte sur l'adjacence directe des éléments considérés (sommets ou arêtes). Plus récemment, HAHN *et al.* [HKvS02] ont introduit la notion de « coloration injective », qui associe une couleur à chaque sommet d'un graphe de sorte que tous les voisins d'un sommet reçoivent une couleur distincte l'une de l'autre (deux sommets adjacents peuvent donc recevoir la même couleur).

Plus formellement, une « coloration injective » d'un graphe est définie ainsi :

DÉFINITION 4.1.5 (coloration injective)
Soit $G = (V, E)$ un graphe. Une coloration injective *du graphe G est une fonction $\varphi_i : V(G) \to \mathbb{N}^*$ qui, à chaque sommet de G, associe un entier positif appelé* couleur *du sommet tel que pour toute paire d'arêtes $\{u, v\}, \{v, w\} \in E(G)$: $\varphi_i(u) \neq \varphi_i(w)$.*

Le problème de coloration associé, appelé COLORATION INJECTIVE, consiste à déterminer le « nombre chromatique injectif » du graphe donné en entrée.

DÉFINITION 4.1.6 (nombre chromatique injectif)
Pour un graphe $G = (V, E)$, le nombre chromatique injectif $\chi_i(G)$ *est égal au plus petit entier k tel qu'il existe une coloration injective de G n'utilisant que les couleurs parmi $\{1, \ldots, k\}$.*

Ce problème a été montré NP-complet par un résultat de HAHN *et al.* [HKvS02], et il est également NP-complet de déterminer si un graphe k-régulier admet une coloration injective avec un nombre fixé $k \geq 3$ de couleurs [HKvS02], ce qui implique que le problème k-COLORATION INJECTIVE est paraNP-complet lorsque paramétré par le nombre maximum k de couleurs à utiliser.

Coloration additive. Très récemment, MATAMALA et ZAMORA PONCE [MZP10] ont eu l'idée de mélanger théorie des graphes et théorie des nombres, dans un nouveau problème de coloration appelé COLORATION ADDITIVE.

L'élément de théorie des nombres provient du problème des ensembles d'entiers sans 3-progression arithmétique, introduit par ERDŐS et TURÁN [ET36]. Un *ensemble d'entiers sans 3-progression arithmétique* est tel que pour tout triplet d'entiers $x, y, z \in \mathbb{N}$ appartenant à cet ensemble, nous avons $|y - x| \neq |z - y|$, équivalent à $y \neq \frac{x+z}{2}$.

Transposé en problème de coloration, cela donne la notion de « coloration additive » introduite par MATAMALA et ZAMORA PONCE [MZP10], qui associe une couleur à chaque sommet d'un graphe de sorte que les valeurs de deux arêtes adjacentes, correspondant à la différence absolue entre leur sommets incidents respectifs, soient distinctes.

Plus formellement, une « coloration additive » d'un graphe est définie ainsi :

DÉFINITION 4.1.7 (coloration additive)
Soit $G = (V, E)$ un graphe. Une coloration additive *du graphe G est une fonction $\varphi'_a : V(G) \to \mathbb{N}^*$ telle que, pour toute paire d'arêtes $\{u, v\}, \{v, w\} \in E(G)$:*

$$|\varphi'_a(u) - \varphi'_a(v)| \neq |\varphi'_a(v) - \varphi'_a(w)|.$$

De manière équivalente, la contrainte peut être définie ainsi :
$\forall u, v, w \in V(G) \ : \ \{u, v\}, \{v, w\} \in E(G) \ \Rightarrow \ \varphi'_a(u) \neq \varphi'_a(w) \ \wedge \ \varphi'_a(v) \neq \frac{|\varphi'_a(u)+\varphi'_a(w)|}{2}$.

Le problème de coloration associé, appelé COLORATION ADDITIVE, consiste à déterminer l'« index chromatique additif » du graphe donné en entrée.

DÉFINITION 4.1.8 (index chromatique additif)
*Pour un graphe $G = (V, E)$, l'*index chromatique additif $\chi'_a(G)$ *est égal au plus petit entier k tel qu'il existe une coloration additive de G n'utilisant que les couleurs parmi $\{1, \ldots, k\}$.*

Nous allons montrer dans ce chapitre qu'à l'instar des quelques autres problèmes de coloration connus et définis ci-dessus, notre problème COLORATION ADDITIVE est également NP-complet, même lorsque le nombre de couleurs $k \geq 4$ est fixé, pour les graphes de degré maximum k (le cas où $k = 3$ restant ouvert).

4.2 NP-complétude pour un nombre fixé de couleurs

Dans cette section, nous allons nous intéresser à la complexité de calcul du problème COLORATION ADDITIVE, et montrer que ce problème est NP-complet même lorsque le nombre de couleurs $k \geq 4$ est fixé.

4.2.1 Objectif et résultat général

Nous avons vu que de nombreux problèmes de coloration de graphe très étudiés, tels que NOMBRE CHROMATIQUE (coloration classique des sommets), INDEX CHROMATIQUE (coloration classique des arêtes) ou encore ÉTIQUETAGE $L(1,2)$ (voir notamment [GY92, CKK+00]) sont connus pour être NP-complets même lorsque le nombre de couleurs est fixé. Par exemple, il est NP-complet de déterminer si l'index chromatique d'un graphe 3-régulier vaut 3 ou non [Hol81].

Il est dès lors naturel de se demander s'il en est de même pour le problème COLORATION ADDITIVE, que nous étudions ici. La définition de notre problème est quelque peu différente à celles d'autres problèmes de coloration de graphe : certaines valeurs peuvent ne pas être utilisées dans une coloration optimale. C'est pourquoi nous nous intéresserons à la valeur de la plus grande couleur utilisée, plutôt qu'au nombre total de couleurs utilisées. Nous allons montrer que le problème k-COLORATION ADDITIVE est NP-complet, pour tout entier $k \geq 4$ fixé, où k est la valeur maximale des couleurs utilisées, impliquant que ce problème est paraNP-complet lorsqu'il est paramétré par k. Notre construction ne s'étend pas au cas où $k = 3$; la complexité du problème 3-COLORATION ADDITIVE reste donc ouverte.

Il est facile de montrer que le problème k-COLORATION ADDITIVE peut être résolu en temps polynomial lorsque $k \leq 2$: si $k = 1$, les seules instances positives correspondent aux graphes de degré maximum 1 ; lorsque $k = 2$, les seules instances positives correspondant aux graphes de degré maximal 2 dont la longueur de chaque cycle est un multiple de 4.

THÉORÈME 4.2.1
Pour tout entier $k \geq 4$, le problème k-COLORATION ADDITIVE est NP-complet sur les graphes de degré maximum k.

Nous pouvons aisément montrer que le problème est dans la classe NP. En effet, étant donnée une coloration (non nécessairement propre) des sommets d'un graphe, il peut être vérifié en temps polynomial s'il s'agit d'une coloration additive du graphe ou non.

Il nous reste à montrer que le problème k-COLORATION ADDITIVE est NP-difficile. Nous allons découper cette démonstration en deux parties : une preuve simple lorsque $k = 4$ (voir SECTION 4.2.2), et une preuve générique lorsque $k \geq 5$ (voir SECTION 4.2.3).

Nous allons montrer la NP-difficulté du problème k-COLORATION ADDITIVE par une réduction à partir d'un problème connu comme étant NP-complet. Pour chacune des deux parties de notre démonstration, la réduction se fera à partir du problème 3-ARÊTES COLORATION, connu comme étant NP-complet pour les graphes 3-réguliers par un résultat de LEVEN et GALIL [LG83]. [1]

1. Plus précisément, LEVEN et GALIL [LG83] ont montré que le problème k-ARÊTES COLORATION est

4.2.2 Pour $k = 4$ couleurs

Nous nous intéressons ici à la démonstration du THÉORÈME 4.2.1 lorsque $k = 4$. En fait, notre résultat est plus fort dans ce cas, puisque nous montrons la NP-complétude du problème 4-COLORATION ADDITIVE pour les graphes 3-réguliers (le THÉORÈME 4.2.1 montre seulement la NP-complétude pour les graphes de degré maximum 4 lorsque $k = 4$).

LEMME 4.2.2
Le problème 4-COLORATION ADDITIVE *est* NP*-complet, même lorsqu'il est restreint aux graphes* 3*-réguliers.*

Notre réduction se fait à partir du problème 3-ARÊTES COLORATION restreint aux graphes 3-réguliers, et est relativement simple dans le cas où $k = 4$. En effet, nous allons tirer avantage du fait qu'un sous-ensemble sans 3-progression arithmétique de $\{1, 2, 3, 4\}$ contient au plus 3 éléments. Nous allons utiliser ces 3 éléments pour encoder une 3-arête coloration du graphe d'origine.

Démonstration. Dans un premier temps, nous donnons la réduction du problème 3-ARÊTES COLORATION, restreint aux graphes 3-réguliers, vers le problème 4-COLORATION ADDITIVE, restreint également aux graphes 3-réguliers.

Soit G le graphe 3-régulier d'une instance du problème 3-ARÊTES COLORATION. Nous construisons un graphe 3-régulier H, à partir de G, en remplaçant chaque sommet $v \in V(G)$ par une clique $\mathcal{K}_3$, dénotée A_v (voir la FIGURE 4.1 pour une schématisation de cette transformation). Cette réduction est bien polynomiale : on a en effet $|V(H)| = 3|V(G)|$ et $|E(H)| = |E(G)| + 3|V(G)|$.

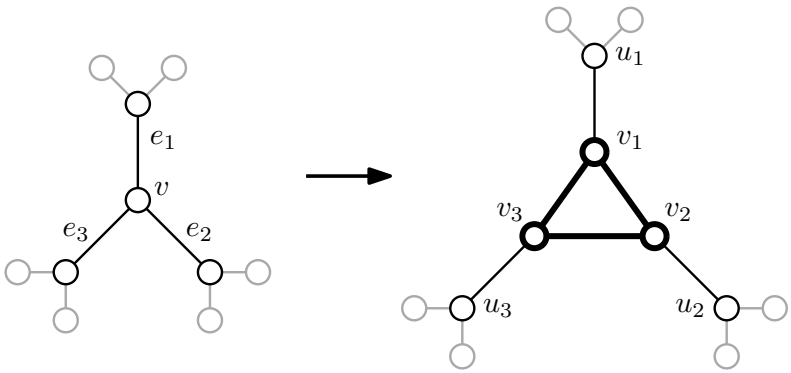


FIGURE 4.1 — Opération effectuée sur chaque sommet $v \in V(G)$ pour la réduction depuis un graphe 3-régulier G, instance du problème 3-ARÊTES COLORATION, vers un graphe 3-régulier H pour le problème 4-COLORATION ADDITIVE.

Nous allons maintenant montrer que notre réduction est valide, c'est-à-dire que le graphe d'origine G admet une 3-arête coloration si et seulement si le graphe construit H admet une 4-coloration additive. Pour toute arête $e \in E(G)$, nous noterons $\varphi'(G, e)$

NP-complet sur les graphes réguliers de degré k, pour tout entier $k \geq 3$.

la couleur associée à l'arête e dans une 3-arête coloration de G. De même, pour tout sommet $v \in V(H)$, nous noterons $\varphi'_a(H, v)$ la couleur associée au sommet v dans une 4-coloration additive de H.

Supposons tout d'abord que le graphe G admette une 3-arête coloration. Soient $v \in V(G)$ un sommet du graphe G, et $e_1, e_2, e_3 \in E(G)$ les trois arêtes adjacentes à v. Soient $v_1, v_2, v_3 \in V(H)$ les trois sommets du gadget A_v dans le graphe H correspondant au sommet v de G, et u_1, u_2, u_3 leurs voisins dans H en dehors du gadget A_v. Supposons, sans perte de généralité, que $\varphi'(G, e_1) = 1$, $\varphi'(G, e_2) = 2$ et $\varphi'(G, e_3) = 3$. Nous fixons alors $\varphi'_a(H, v_1) = \varphi'_a(H, u_1) = 1$, $\varphi'_a(H, v_2) = \varphi'_a(H, u_2) = 2$ et $\varphi'_a(H, v_3) = \varphi'_a(H, u_3) = 4$. Il est aisé de vérifier que cette coloration est bien une 4-coloration additive de H.

Supposons maintenant que le graphe H admette une 4-coloration additive.

Un gadget A_v, correspondant à une clique $\mathcal{K}_3$, admet deux 4-colorations additives possibles : $\{1, 2, 4\}$, ou $\{1, 3, 4\}$. Nous remarquons, d'une part que deux des trois sommets de ce gadget reçoivent nécessairement les deux couleurs de valeurs extrémales 1 et 4, et d'autre part que les couleurs 2 et 3 ne peuvent pas coexister.

Soient v_1, v_2, v_3 les trois sommets d'un gadget A_v, et soient u_1, u_2, u_3 leurs voisins respectives en dehors du gadget. Nous allons montrer que $\varphi'_a(H, u_i) = \varphi'_a(H, v_i)$ pour tout $i \in \{1, 2, 3\}$.

Si $\varphi'_a(H, v_1) = 1$, alors dans le gadget adjacent A_{u_1}, contenant le sommet u_1, seul le sommet u_1 peut recevoir la couleur 1, ses voisins dans A_{u_1} étant à distance deux du sommet v_1 et donc ne pouvant pas recevoir la même couleur que v_1. Avec les mêmes arguments, on déduit que si $\varphi'_a(H, v_1) = 4$, alors $\varphi'_a(H, u_1) = 4$. Si $\varphi'_a(H, v_1) = 2$, alors sans perte de généralité, $\varphi'_a(H, v_2) = 1$ et $\varphi'_a(H, v_3) = 4$, et de ce fait u_1 ne peut pas recevoir les couleurs 1 et 4. Il ne peut également pas recevoir la couleur 3, car sinon une 3-progression arithmétique serait créée entre v_2 (de couleur 1), v_1 (de couleur 2), et lui-même de couleur 3. Donc si $\varphi'_a(H, v_1) = 2$, alors $\varphi'_a(H, u_1) = 2$. Enfin, en utilisant les mêmes arguments, on déduit que si $\varphi'_a(H, v_1) = 3$, alors $\varphi'_a(H, u_1) = 3$. Pour les mêmes raisons, $\varphi'_a(H, u_2) = \varphi'_a(H, v_2)$ et $\varphi'_a(H, u_3) = \varphi'_a(H, v_3)$.

Pour chaque couple de sommets $\{v_i, u_i\}$ ($i \in \{1, 2, 3\}$) du graphe H, représentant une arête $e_i \in E(G)$ adjacente à un sommet $v \in V(G)$, nous disposons ainsi de trois classes de paires de couleurs : la paire $\{1, 1\}$, la paire $\{4, 4\}$, et les paires $\{2, 2\}$ et $\{3, 3\}$. Nous pouvons dès lors utiliser ces trois classes pour encoder une 3-arête coloration du graphe G : si les sommets v_i, u_i reçoivent la paire de couleurs $\{1, 1\}$, alors nous associons la couleur 1 à l'arête e_i dans G ; s'ils reçoivent la paire de couleurs $\{4, 4\}$, alors l'arête e_i reçoit la couleur 3 ; s'ils reçoivent la paire de couleurs $\{2, 2\}$ ou $\{3, 3\}$, alors l'arête e_i reçoit la couleur 2. Notons que comme les couleurs 2 et 3 ne peuvent coexister dans un même gadget A_v du graphe H, les paires de couleurs $\{2, 2\}$ et $\{3, 3\}$ ne peuvent pas apparaître simultanément pour encoder les couleurs de deux arêtes adjacentes du graphe G. Il est alors aisé de vérifier que cette coloration est bien une 3-arête coloration du graphe G. □

Notre réduction et sa démonstration comportent deux idées clés :
- deux des trois sommets du gadget A_v doivent nécessairement recevoir les deux couleurs de valeurs extrémales, ici 1 et 4 ;
- chaque sommet à l'extérieur du gadget A_v doit recevoir la même couleur que son voisin dans le gadget A_v.

Ainsi, pour deux sommets du graphe H représentant une arête du graphe G, nous disposons de trois classes de paires de couleurs : la paire $\{1,1\}$ pour encoder une arête de couleur 1, la paire $\{4,4\}$ pour encoder une arête de couleur 3, et les paires $\{j,j\}$ (avec $1 < j < 4$) pour encoder une arête de couleur 2. Ces trois classes nous permettent d'encoder une 3-arête coloration du graphe G sous la forme d'une 4-coloration additive du graphe H. Dans la suite, nous allons réutiliser ces mêmes idées pour la démonstration du THÉORÈME 4.2.1 lorsque $k \geq 5$.

4.2.3 Pour $k \geq 5$ couleurs

Nous allons maintenant finaliser notre démonstration du THÉORÈME 4.2.1, en montrant le cas où $k \geq 5$:

LEMME 4.2.3
*Le problème k-*COLORATION ADDITIVE *est* NP*-complet, pour tout entier $k \geq 5$, même lorsqu'il est restreint aux graphes de degré maximum k.*

Notre réduction va reprendre les mêmes idées clés que pour la démonstration du THÉORÈME 4.2.1 lorsque $k = 4$, en construisant notamment un gadget A_v tel que :
- deux des trois sommets du gadget A_v doivent nécessairement recevoir les deux couleurs de valeurs extrémales, à savoir 1 et k ;
- chaque sommet à l'extérieur du gadget A_v doit recevoir la même couleur que son voisin dans le gadget A_v.

Lorsque $k = 4$, le gadget A_v seul suffit à obtenir ces deux propriétés. Ce n'est toutefois plus le cas lorsque $k \geq 5$. C'est pourquoi nous devrons ajouter un second gadget, dont le seul but sera d'obtenir à nouveau ces propriétés.

Démonstration. Nous allons décrire la réduction pour $k \geq 7$. Les autres cas ($5 \leq k \leq 6$) peuvent être également démontrés en utilisant une réduction et des arguments similaires, et quelques cas-par-cas.

Dans un premier temps, nous décrivons la réduction du problème 3-ARÊTES COLORATION, restreint aux graphes 3-réguliers, vers notre problème k-COLORATION ADDITIVE restreint aux graphes de degré maximum k (pour $k \geq 5$).

Soit G le graphe 3-régulier d'une instance du problème 3-ARÊTES COLORATION. Notre réduction est composée de deux gadgets. Pour construire le graphe H de degré maximum k, nous partons tout d'abord du graphe G, et remplaçons chaque sommet $v \in V(G)$ par une clique $\mathcal{K}_3$, dénotée A_v, et composé des trois sommets v_1, v_2, v_3 (voir la FIGURE 4.2 pour une schématisation de cette transformation).

Nous ajoutons ensuite au graphe H un second gadget, noté C_v. Ce gadget est schématisé dans la FIGURE 4.3.

Il peut être facilement vérifié que la réduction est bien polynomiale, c'est-à-dire que la taille du graphe H est polynomiale en la taille du graphe G et le paramètre k.

Par la construction, les sommets v_1, v_2, v_3 sont chacun de degré $k-1$, les sommets v'_1, v'_2, v'_3 sont chacun de degré 2, l'ensemble stable I_v contient $k-5$ sommets chacun de degré 4, leurs voisins dans le second ensemble stable sont chacun de degré 3, et les sommets x, y sont chacun de degré k. De ce fait, le graphe H est bien de degré maximum k.

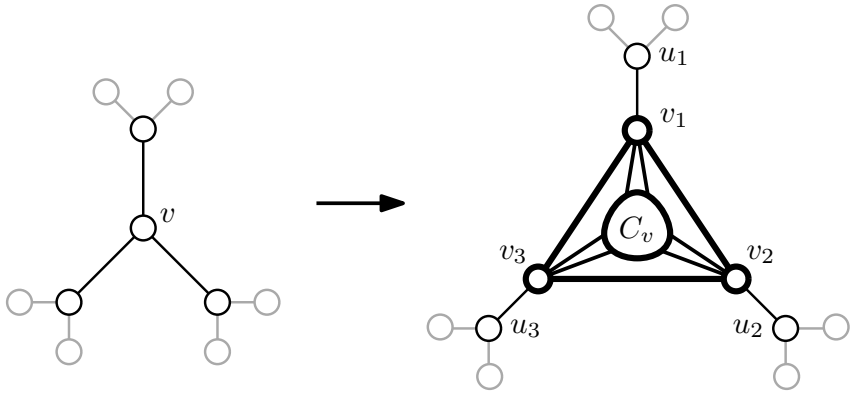


FIGURE 4.2 — Première opération appliquée à chaque sommet $v \in V(G)$, correspondant au gadget A_v, pour la réduction depuis un graphe 3-régulier G, instance du problème 3-ARÊTES COLORATION, vers un graphe H pour le problème k-COLORATION ADDITIVE (avec $k \geq 5$). L'élément noté C_v correspond au second gadget de notre réduction, schématisé sur la FIGURE 4.3.

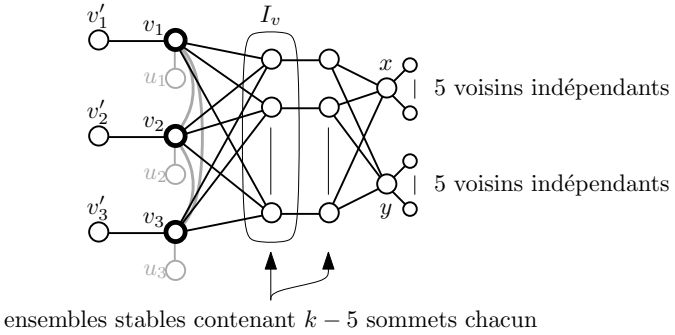


FIGURE 4.3 — Seconde opération appliquée à chaque sommet $v \in V(G)$ lorsque $k \geq 7$, correspondant au gadget C_v, pour la réduction depuis un graphe 3-régulier G, instance du problème 3-ARÊTES COLORATION, vers un graphe H de degré maximum k pour le problème k-COLORATION ADDITIVE (avec $k \geq 5$).

Nous pouvons également faire l'observation suivante :

Observation. Les sommets x, y dans le gadget C_v étant chacun de degré k, et à distance 2 l'un de l'autre, ils doivent tous deux recevoir les couleurs 1 et k. De ce fait, les sommets de l'ensemble stable I_v ne peuvent recevoir que des couleurs parmi $\{2, \ldots, k-1\}$.

Nous allons maintenant montrer que notre réduction est valide, c'est-à-dire que le graphe d'origine G admet une 3-arête coloration si et seulement si le graphe construit H admet une k-coloration additive. Pour tout sommet $v \in V(H)$, nous noterons $\varphi'_a(H, v)$ la couleur associée au sommet v dans une k-coloration additive de H.

Supposons tout d'abord que le graphe G admette une 3-arête coloration. Soit $v \in V(G)$ un sommet adjacent à trois autres sommets $u_1, u_2, u_3 \in V(G)$, tel que chaque

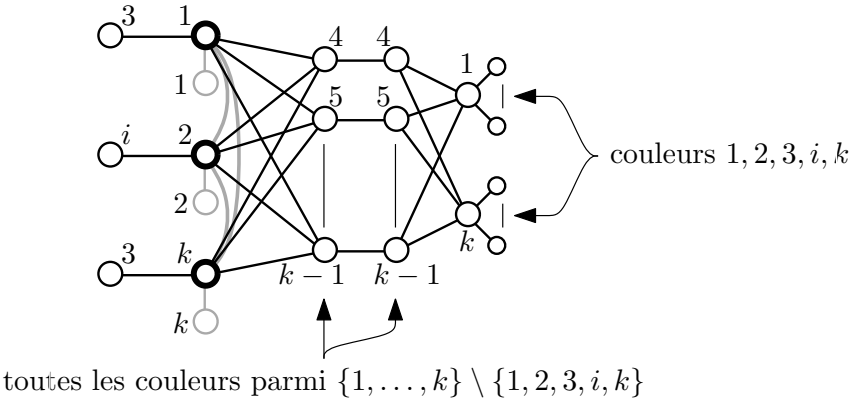


FIGURE 4.4 — Encodage d'une 3-arête coloration par une k-coloration additive, lorsque $k \geq 7$.

arête $\{v, u_i\}$ reçoit la couleur i avec $1 \leq i \leq 3$. Soient v_1, v_2, v_3 les trois sommets correspondants à v dans H, et supposons sans perte de généralité que v_i est adjacent au sommet u_i dans H, pour $1 \leq i \leq 3$.

Alors dans H nous fixons $\varphi'_a(H, v_1) = \varphi'_a(H, u_1) = 1$, $\varphi'_a(H, v_2) = \varphi'_a(H, u_2) = 2$, et $\varphi'_a(H, v_3) = \varphi'_a(H, u_3) = k$. Les sommets de l'ensemble stable I_v ne peuvent pas prendre les couleurs 1, k, mais également 3, $\frac{1+k}{2} = i$ et $\frac{2+k}{2} = j$, et seule l'une des deux valeurs i et j est un entier correspondant à une couleur. Supposons que i est une couleur. Nous fixons alors $\varphi'_a(H, v'_1) = 3$, $\varphi'_a(H, v'_2) = i$, et $\varphi'_a(H, v'_3) = 3$. Nous fixons également $\varphi'_a(H, x) = 1$ et $\varphi'_a(H, y) = k$, leurs voisins de degré 1 reçoivent toutes les couleurs restantes parmi $\{2, \ldots, k-1\}$, les sommets de l'ensemble stable I_v reçoivent toutes les couleurs parmi $\{1, \ldots, k\} \setminus \{1, 2, 3, i, k\}$, et chacun de leurs voisins respectifs reçoit la même couleur que son voisin dans I_v. Cette coloration est schématisée dans la FIGURE 4.4. Il est aisé de vérifier que cette coloration est bien une k-coloration additive du graphe H.

Supposons maintenant que le graphe H admette une k-coloration additive. Soient $v_1, v_2, v_3, v'_1, v'_2, v'_3, x, y, u_1, u_2, u_3$ certains sommets de H tels que décrit lors de la réduction.

Affirmation. Sans perte de généralité, $1 = \varphi'_a(H, v_1) < \varphi'_a(H, v_2) < \varphi'_a(H, v_3) = k$.

Pour montrer cette affirmation, supposons tout d'abord, et sans perte de généralité, que $1 < \varphi'_a(H, v_1) < \varphi'_a(H, v_2) < \varphi'_a(H, v_3) < k$. Alors $3 \leq \varphi'_a(H, v_2) \leq k-2$, et seules $k-2$ couleurs parmi $\{1, \ldots, k\}$ peuvent être utilisées pour les voisins de v_2. Or v_2 a $k-1$ voisins, et donc l'un d'eux ne pourra pas recevoir de couleur valide, ce qui est une contradiction avec le fait que H admette une k-coloration additive.

Supposons maintenant, sans perte de généralité, que $1 = \varphi'_a(H, v_1) < \varphi'_a(H, v_2) < \varphi'_a(H, v_3) < k$. Alors $\varphi'_a(H, v_2) = 2$ et $\varphi'_a(H, v_3) = k-1$, car sinon le précédent cas s'applique. Dans ce cas, les sommets de l'ensemble stable I_v ne peuvent pas recevoir les couleurs $1, 2, 3, i, k-1, k$, avec $i = \frac{k}{2}$ (si k est pair) ou $i = \frac{k+1}{2}$ (si k est impair), et seules $k-6$ couleurs sont encore disponibles pour colorier ces sommets. Or l'ensemble

I_v contient $k-5$ sommets, et donc l'un d'eux ne pourra pas recevoir de couleur valide, ce qui est là encore une contradiction avec le fait que H admet une k-coloration additive.

Donc, sans perte de généralité, $1 = \varphi'_a(H, v_1) < \varphi'_a(H, v_2) < \varphi'_a(H, v_3) = k$.

Affirmation. Si $\varphi'_a(H, v_1) = 1$ et $\varphi'_a(H, v_3) = k$, alors $\varphi'_a(H, u_1) = \varphi'_a(H, v_1) = 1$ et $\varphi'_a(H, u_3) = \varphi'_a(H, v_3) = k$.

Sans perte de généralité, supposons que $1 = \varphi'_a(H, v_1) < \varphi'_a(H, v_2) < \varphi'_a(H, v_3) = k$. Lorsqu'un sommet $v' \in H$ est de degré $k-1$, nous noterons $\bar{c}(v')$ la seule couleur non affectée aux voisins de v'. Puisque le sommet v_1 est de degré $k-1$, la couleur $\bar{c}(v_1)$ est bien définie. Si $\bar{c}(v_1) = 1$, alors $\varphi'_a(H, u_1) \neq 1$, mais dans ce cas aucun des trois sommets du gadget adjacent A_{u_1} ne pourra recevoir la couleur 1, ce qui est en contradiction avec l'affirmation précédente qui prouve que chaque gadget A_z contient un sommet de couleur 1. Si $\bar{c}(v_1) \neq 1$, alors $\varphi'_a(H, u_1) = 1$, car sinon nous retrouvons la contradiction précédente. Donc $\varphi'_a(H, u_1) = \varphi'_a(H, v_1) = 1$. En utilisant les mêmes arguments, il peut être montré que $\varphi'_a(H, u_3) = \varphi'_a(H, v_3) = k$.

À l'instar du cas où $k = 4$, pour deux sommets du graphe H représentant une arête du graphe G, nous disposons maintenant de trois classes de paires de couleurs : la paire $\{1, 1\}$ pour encoder une arête de couleur 1, la paire $\{k, k\}$ (k étant la valeur maximale) pour encoder une arête de couleur 3, et les paires $\{j, j\}$ (avec $1 < j < k$) pour encoder une arête de couleur 2. Ces trois classes nous permettent d'encoder une 3-arête coloration du graphe G sous la forme d'une k-coloration additive du graphe H. □

4.3 Approximabilité

Dans cette section, nous nous intéressons à la possibilité d'approximer avec un algorithme polynomial, pour un graphe $G = (V, E)$, la valeur de l'index chromatique additif $\chi'_a(G)$. Nous allons montrer que dans le cas d'un graphe « séparant » (*split*), avec $n = |V(G)|$, cette valeur ne peut pas être approximée à un facteur $n^{1/3-\epsilon}$ avec un algorithme polynomial, sauf si $\mathsf{P} = \mathsf{NP}$, tandis qu'elle peut l'être à un facteur $n^{1/3+\epsilon}$.

Nous allons pour cela nous appuyer sur un résultat de HELL *et al.* [HRS08], montrant que le nombre chromatique injectif $\chi_i(G)$ n'est pas approximable à un facteur $n^{1/3-\epsilon}$ avec un algorithme polynomial, sauf si $\mathsf{ZPP} = \mathsf{NP}$, et ce même pour les graphes séparants. Ce résultat utilise la démonstration d'inapproximabilité du nombre chromatique *classique* $\chi(G)$, obtenue par FEIGE et KILLIAN [FK98], et sur un résultat de ZUCKERMAN [Zuc06] montrant que la condition $\mathsf{ZPP} = \mathsf{NP}$ peut être remplacée par $\mathsf{P} = \mathsf{NP}$.

4.3.1 Inapproximabilité à facteur $n^{1/3-\epsilon}$

THÉORÈME 4.3.1
Pour tout $0 < \epsilon < 1$, l'index chromatique additif $\chi'_a(G)$ d'un graphe G n'est pas approximable à facteur $n^{1/3-\epsilon}$, même si G est un graphe séparant, sauf si $\mathsf{P} = \mathsf{NP}$.

Démonstration. Soit $0 < \epsilon < 1$, et supposons que l'on dispose d'un algorithme polynomial d'approximation pour $\chi'_a(G)$ qui, étant donné un graphe séparant G, calcule une

approximation α de l'index chromatique additif $\chi'_a(G)$ de G telle que $\chi'_a(G) \leq \alpha \leq n^{1/3-\epsilon}\chi'_a(G)$.

Remarquons qu'une coloration additive est de surcroît une coloration injective : en effet, deux voisins d'un même sommet du graphe ne peuvent recevoir la même couleur, car sinon les valeurs associées aux deux arêtes incidentes sont les mêmes. Par ailleurs, G étant un graphe séparant, il est dans le pire des cas réduit à une clique. Dans ce cas, une coloration additive de G doit non seulement être une coloration injective, mais de plus la coloration doit éviter toute 3-progression arithmétique entre tout triplet de sommets du graphe. Nous avons donc $\chi'_a(G) \leq \chi'_a\big(\mathcal{K}_{\chi_i(G)}\big)$.

Le problème de la coloration additive d'une clique est en fait lié à un problème en théorie des nombres, toujours ouvert à ce jour. Ce problème consiste à déterminer le plus petit entier p, tel qu'il existe un ensemble de n entiers parmi $\{1, \ldots, p\}$ ne contenant pas de 3-progression arithmétique. De manière équivalente, en terme de graphe, il s'agit de déterminer la valeur optimale d'une coloration additive d'une clique de cardinalité n.

Actuellement, seules une borne inférieure et une borne supérieure sont connues pour cet entier p. Pour montrer l'inapproximabilité du problème COLORATION ADDITIVE, nous allons utiliser les résultats de SALEM et SPENCER [SS42], et BEHREND [Beh46], réétudiés et améliorés très récemment par ELKIN [Elk10], qui donnent une borne supérieure pour cet entier p, c'est-à-dire une fonction $g : \mathbb{N} \to \mathbb{N}$ telle que :

$$\chi'_a(G) \leq \chi'_a\big(\mathcal{K}_{\chi_i(G)}\big) \leq \chi_i(G) \cdot g\big(\chi_i(G)\big)$$

et pour laquelle, pour tout $a > 0$, $\lim_{n\to+\infty} \frac{g(n)}{n^a} = 0$. Nous avons alors :

$$\chi_i(G) \leq \alpha \leq n^{1/3-\epsilon} \cdot \chi_i(G) \cdot g(n)$$

et comme $g(n) = \mathcal{O}(n^{\epsilon/2})$, nous en déduisons que pour des valeurs de n suffisamment grandes, on a $\alpha \leq n^{1/3-\epsilon/2} \cdot \chi_i(G)$.

Ceci implique que pour tout $\epsilon > 0$, il n'existe pas d'algorithme polynomial d'approximation à facteur $n^{1/3-\epsilon}$ pour l'indice chromatique additif $\chi'_a(G)$, même lorsque G est un graphe séparant. □

4.3.2 Approximation à facteur $n^{1/3+\epsilon}$

THÉORÈME 4.3.2
Pour tout $\epsilon > 0$, l'index chromatique additif $\chi'_a(G)$ d'un graphe séparant G est approximable à facteur $n^{1/3+\epsilon}$.

Démonstration. Supposons que l'on dispose d'une approximation α du nombre chromatique injectif d'un graphe séparant G, telle que $\chi_i(G) \leq \alpha \leq \chi_i(G) \cdot n^{1/3}$. D'après les résultats cités dans la démonstration du THÉORÈME 4.3.1, nous savons qu'il existe une fonction croissante $g : \mathbb{N} \to \mathbb{N}$ telle que :

$$\chi'_a(G) \leq \chi'_a\big(\mathcal{K}_{\chi_i(G)}\big) \leq \chi_i(G) \cdot g\big(\chi_i(G)\big) \leq \alpha \cdot g(\alpha) \leq \chi_i(G) \cdot n^{1/3} \cdot g\big(\chi_i(G) \cdot n^{1/3}\big).$$

Or, $g\big(\chi_i(G)\big) \leq g\big(n^{4/3}\big)$, et pour n suffisamment grand, $g\big(n^{4/3}\big) \leq n^{\epsilon}$ pour tout $\epsilon > 0$. Nous en déduisons l'inéquation suivante :

$$\chi'_a(G) \leq \alpha \cdot g(\alpha) \leq \chi'_a(G) \cdot n^{1/3+\epsilon}$$

impliquant que l'index chromatique additif $\chi'_a(G)$ d'un graphe séparant G peut être approximé à facteur $n^{1/3+\epsilon}$ pour tout $\epsilon > 0$. □

4.4 Cas polynomiaux et FPT

Nous avons vu dans la SECTION 4.2 que le problème COLORATION ADDITIVE est NP-complet, pour les graphes de degré maximum k, et ce même lorsque la valeur maximale $k \geq 4$ des couleurs est fixée.

Dans cette section, nous allons montrer que ce problème peut être résolu efficacement et en temps FPT sur les graphes de largeur arborescente bornée, si la valeur maximale des couleurs est fixée. Nous présentons également un algorithme polynomial permettant de calculer la valeur optimale du problème COLORATION ADDITIVE restreint aux arbres, et donc résoudre ce problème même lorsque la valeur maximale des couleurs n'est pas fixée.

4.4.1 Résultat théorique *via* le théorème de COURCELLE

Dans les PRÉLIMINAIRES (voir SECTION 1.4.3), nous avons présenté un résultat de COURCELLE [Cou97] montrant que tout problème pouvant être exprimé sous la forme d'une formule MSO (logique monadique du second ordre), peut être résolu en temps FPT paramétré par la largeur arborescente du graphe donné en entrée.

Lorsque la valeur maximale des couleurs k est fixée, il est possible d'écrire une formule MSO, dont la taille ne dépend que de k, et exprimant le problème k-COLORATION ADDITIVE. En explicitant cette formule MSO, et par le résultat de COURCELLE [Cou97], nous allons montrer que le problème k-COLORATION ADDITIVE, avec k fixé, peut être résolu en temps FPT paramétré par la largeur arborescente du graphe donné en entrée.

Le problème k-COLORATION ADDITIVE, avec $k \in \mathbb{N}$ fixé, peut être exprimé par la formule MSO suivante :

$$\exists X_1, X_2, \ldots, X_k \subseteq V(G) \text{ s.t. } \forall x, y, z \in V(G) : \big(\{x, y\} \in E(G) \wedge \{y, z\} \in E(G)\big)$$

$$\Rightarrow \quad \Big[\big(\bigvee_{i\in[k]} x \in X_i\big) \wedge \big(\bigvee_{i\in[k]} y \in X_i\big) \wedge \big(\bigvee_{i\in[k]} z \in X_k\big)\Big] \tag{4.1}$$

$$\wedge \quad \neg\Big[\bigvee_{i\in[k]} \big(x \in X_i \wedge z \in X_i\big)\Big] \tag{4.2}$$

$$\wedge \quad \neg\Big[\bigvee_{i,j\in[k]} \big(x \in X_i \wedge y \in X_{i+j} \wedge z \in X_{i+2j}\big)\Big] \tag{4.3}$$

Dans cette formule, les sous-ensembles $X_1, X_2, \ldots, X_k \in V(G)$ représentent la coloration des sommets (un ensemble par couleur), et les trois sommets x, y, z représentent tout chemin de longueur 2 dans le graphe G. L'ÉQUATION 4.1 indique que les sommets x, y, z doivent recevoir une couleur parmi $\{1, 2, \ldots, k\}$. L'ÉQUATION 4.2 indique que les sommets x, z, qui sont à distance 2 l'un de l'autre, doivent recevoir des couleurs différentes (il s'agit de la même condition que pour une coloration injective, voir SECTION 4.1.4). Enfin, l'ÉQUATION 4.3 indique que les couleurs associées aux sommets x, y, z ne doivent pas créer une 3-progression arithmétique (voir SECTION 4.1.4).

Par le théorème de COURCELLE [Cou97], nous en déduisons que le problème k-COLORATION ADDITIVE peut être résolu en temps FPT, paramétré par la largeur arborescente du graphe donné en entrée, si le nombre de couleurs k est fixé. En fait, le problème peut également être résolu en temps FPT lorsqu'il est paramétré par la « largeur de clique » (*clique-width*, une autre largeur de graphe que nous ne définissons pas ici), par ce même théorème de COURCELLE [Cou97].

4.4.2 Algorithme polynomial pour les arbres

Nous avons vu que lorsque la valeur maximale des couleurs est fixée, le problème k-COLORATION ADDITIVE peut être résolu efficacement sur les arbres, et plus généralement sur les graphes de largeur arborescente bornée. Nous allons maintenant nous intéresser à la complexité du problème COLORATION ADDITIVE lorsque le nombre de couleurs n'est pas fixé. La première classe de graphes non triviaux qu'il convient de considérer est celle des arbres, correspondant aux graphes de largeur arborescente 1.

Nous allons montrer que l'on peut déterminer en temps polynomial l'index chromatique additif d'un arbre, en explicitant un algorithme polynomial calculant cet index. Pour cela, nous utiliserons notamment un résultat récent de MATAMALA et ZAMORA PONCE [MZP10], montrant qu'un arbre T est d'index chromatique additif au plus $5/3 \cdot \Delta(T)$, où Δ est le plus grand degré des sommets dans l'arbre T.

Soit $T = (X, I)$ un arbre. Notre algorithme polynomial suit un schéma de « programmation dynamique », des feuilles vers la racine de l'arbre T. À chaque noeud $v \in X(T)$ de l'arbre T est associé un ensemble de valeurs de vérité $OPT[v, c, f]$, où :

- v est le noeud considéré ;
- c est la couleur associée au noeud v ;
- f est la couleur associée à son père w.

Lors du déroulement de notre algorithme, les valeurs de vérité d'un noeud sont calculées à partir des valeurs de vérité de ses fils. Une valeur de vérité $OPT[v, c, f]$ sera mise à vraie si :

- le sous-arbre T_v admet une coloration additive, avec $\varphi'_a(v) = c$;
- pour chaque fils u_i de v, il existe une couleur c_i telle que $OPT[u_i, c_i, c]$ vaut vraie, c'est-à-dire que le noeud u_i peut recevoir une couleur compatible avec la couleur f associée à son père v, et telle que le sous-arbre T_{u_i} admette une coloration additive.

La FIGURE 4.5 schématise la partie d'un arbre T contenant un noeud v, de père w et ayant d fils $u_1, u_2, \ldots, u_d$.

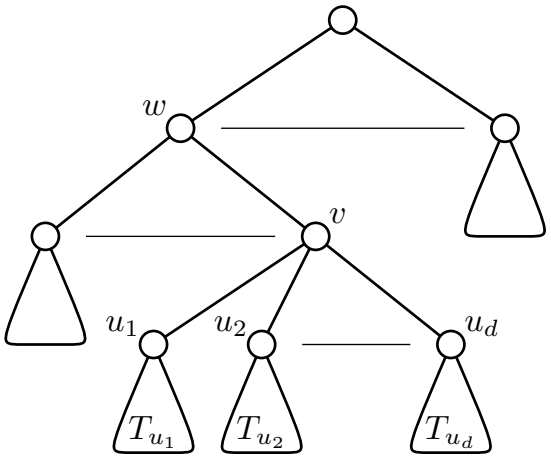


FIGURE 4.5 — Arbre T contenant un noeud v, de père w et avec d fils $u_1, u_2, \ldots, u_d$.

Au début de l'algorithme, chaque feuille de l'arbre T reçoit un ensemble de valeurs de vérité $OPT[v, c, f]$, toutes assignées à vraie, correspondant à chacune des couleurs $\{1, 2, \ldots, 5/3 \cdot \Delta(T)\}$ pouvant être associées à cette feuille et à son père. À la fin de l'algorithme, la racine ne disposant pas de père, il peut être supposé la valeur ∞ pour la couleur f, ne pouvant ainsi pas créer de 3-progression arithmétique.

L'étape cruciale et non triviale de notre algorithme consiste à déterminer, en temps polynomial, si tous les fils u_i d'un noeud v peuvent recevoir une couleur compatible. Puisque le degré des noeuds de l'arbre n'est pas borné, il n'est pas possible d'effectuer une recherche directe de couleurs compatibles, le nombre de possibilités différentes étant alors exponentiel. Pour effectuer cette étape, nous allons utiliser un algorithme polynomial de calcul de flot maximum, sur un réseau construit spécifiquement.

Tous les fils du noeud v sont à distance 2 les uns des autres. De ce fait, et par définition d'une coloration additive (voir SECTION 4.1.4), ils doivent tous recevoir une couleur différente. Par ailleurs, il est nécessaire de s'assurer que la couleur associée à un fils u_i du noeud v ne crée pas de 3-progression arithmétique :

– avec les couleurs associées au noeud v et à son père w ;
– avec les couleurs associées au noeud v et à un autre de ses fils u_j.

Ces conditions sur les couleurs possibles pour les fils du noeud v peuvent être représentées sous la forme d'un réseau de taille polynomiale en le nombre de fils du noeud v et le nombre de couleurs possibles. Un algorithme polynomial de calcul de flot maximum va nous permettre de déterminer, pour chaque couleur c associée au noeud v et chaque couleur f associée à son père w, une affectation correcte de couleurs pour les fils du noeud v.

Le réseau est construit de la manière suivante. Supposons que le noeud v ait d fils, notés $u_1, u_2, \ldots, u_d$, et un père noté w. Nous notons c la couleur associée au noeud v, et f la couleur associée à son père w. La construction s'effectue à partir d'un réseau contenant uniquement une source s et un puits p. Nous ajoutons à ce réseau un premier ensemble stable S de d sommets, notés $s_1, s_2, \ldots, s_d$, et représentant les d fils du noeud v. Nous ajoutons ensuite un deuxième ensemble stable C de sommets représentant les couleurs $\{1, 2, \ldots, 5/3\Delta(T)\} \setminus \{f\}$.

La source s est reliée à chacun des sommets de l'ensemble S par un arc. Chaque sommet de l'ensemble S, correspondant à un fils u_i du noeud v, est relié par un arc aux sommets de l'ensemble C correspondant aux couleurs j pour lesquelles $OPT[u_i, j, c]$ est vraie, signifiant que le sous-arbre T_{u_i} admet une coloration additive dans laquelle le noeud u_i reçoit la couleur j. Lorsque deux couleurs j_1, j_2 ne peuvent être associées simultanément à deux fils u_1, u_2 de v (car cette affectation créerait une 3-progression arithmétique entre ces deux fils et le noeud v), un sommet $c_{1,2}$ est ajouté au réseau, et un arc est ajouté depuis chacun des deux sommets correspondant aux couleurs j_1, j_2 vers ce nouveau sommet. Enfin, le puits p est relié par un arc à chaque sommet de l'ensemble C non relié à un sommet $c_{i,j}$, et à chaque sommet $c_{i,j}$.

Tous les arcs de ce réseau reçoivent une capacité de 1. Cette construction du réseau est schématisée sur la FIGURE 4.6.

Toutes les affectations possibles sont représentées dans ce réseau. Ainsi, s'il existe une affectation correcte de couleurs pour les fils du noeud v, correspondant à une coloration additive, alors il existe un flot de valeur exactement d (donc maximum) dans ce réseau, où l'unique arc sortant du noeud s_i dans le flot indique (par le sommet vers lequel cet arc pointe dans le réseau) la couleur affectée au fils u_i du noeud v dans l'ar-

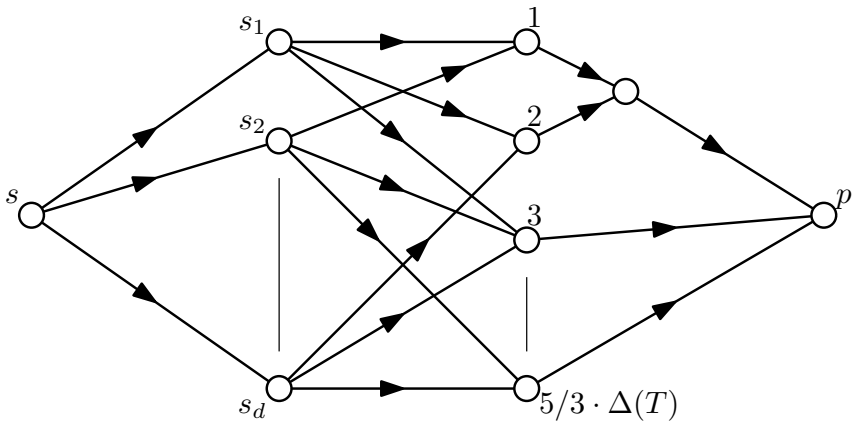


FIGURE 4.6 — Réseau permettant de déterminer s'il existe une affectation correcte de couleurs pour les fils $s_1, s_2, \ldots, s_d$ de v, parmi les couleurs $\{1, 2, \ldots, 5/3 \cdot \Delta(T)\}$.

bre. Inversement, si notre algorithme trouve un flot de valeur d, alors ce flot nous donne une affectation correcte de couleurs pour les fils du noeud v.

Notre algorithme de calcul de flot maximum est polynomial par rapport au degré maximum des noeuds de l'arbre T, et le nombre de couleurs possibles. Rappelons que le nombre maximum de couleurs possibles est borné par $5/3 \cdot \Delta(T)$, d'après un résultat de MATAMALA et ZAMORA PONCE [MZP10]. Notre algorithme global, permettant de déterminer l'index chromatique additif d'un arbre quelconque, est donc bien polynomial en la taille de l'arbre T.

4.5 Conclusion

Il existe de nombreuses variantes de problèmes de coloration, fondamentaux en théorie des graphes, qui s'avèrent être NP-complets et *difficilement* approximables. De par leur importance, une bonne connaissance de leurs caractéristiques théoriques est primordiale.

Le problème COLORATION ADDITIVE est un nouveau problème de coloration, introduit récemment par MATAMALA et ZAMORA PONCE [MZP10]. À l'instar des autres problèmes de coloration, il est important d'en connaître les caractéristiques théoriques.

Dans ce chapitre, nous avons effectué une première étude de la complexité de ce problème : nous avons montré que ce problème est NP-complet pour tout nombre fixé $k \geq 4$ de couleurs (il est donc paraNP-complet), qu'il ne peut être approximé à facteur $n^{1/3-\epsilon}$ mais peut l'être à facteur $n^{1/3+\epsilon}$, et qu'il peut être résolu en temps polynomial lorsque restreint aux arbres.

Nos résultats sont un premier aperçu des différentes caractéristiques théoriques du problème COLORATION ADDITIVE, combinant théorie des graphes et théorie des nombres. De nombreuses problématiques concernant ce problème restent ouvertes.

Tout d'abord, nous n'avons pu déterminer la complexité du problème k-COLORATION ADDITIVE, dans le cas des graphes de degré maximum k, que lorsque $k \geq 4$, en montrant que le problème est NP-complet dans ces cas, ou lorsque $k \leq 2$, où le problème peut être résolu en temps polynomial. Les réductions que nous avons utilisées ne s'adaptent pas au cas où $k = 3$, et de ce fait la complexité du problème 3-COLORATION ADDITIVE reste actuellement ouverte.

Une perspective naturelle de notre travail serait de compléter cette étude, en déterminant la complexité du problème k-COLORATION ADDITIVE lorsque $k = 3$.

Par ailleurs, et bien que cela puisse paraître au premier abord assez simple, la complexité du problème COLORATION ADDITIVE restreint aux cliques reste à ce jour inconnue : nous ne savons pas si l'index chromatique additif d'une clique de taille quelconque peut être calculé en temps polynomial ou non ! Plus surprenant encore, la valeur exacte de cet index n'est connu que pour des cliques contenant au plus 41 de sommets [GGK08].

En fait, le cas du problème COLORATION ADDITIVE sur une clique de cardinalité $K \in \mathbb{N}$ est directement lié à un problème très connu en théorie des nombres qui, à partir de l'ensemble des K premiers entiers, consiste à trouver le plus grand sous-ensemble d'entiers (en terme de cardinalité) ne contenant pas de 3-progression arithmétique. Ce problème, introduit par ERDŐS et TURÁN en 1936 [ET36], n'a jusqu'à ce jour pas trouvé de solution exacte, les meilleures bornes actuelles provenant des résultats de ELKIN [Elk10] et SANDERS [San11a, San11b].

Une autre problématique concerne la complexité de ce problème lorsqu'il est restreint aux graphes de largeur arborescente bornée. Nous avons en effet montré que ce problème peut être résolu en temps polynomial lorsqu'il est restreint aux arbres, c'est-à-dire aux graphes de largeur arborescente égale à 1. Mais qu'en est-il des graphes de largeur arborescente plus grande ?

Cette problématique n'est pas dénuée de sens : en effet, il a été montré que le problème ÉTIQUETAGE $L(2,1)$ [CK96], similaire au problème COLORATION ADDITIVE en certains aspects (des contraintes sont imposées sur les couleurs de sommets à distance 1 et distance 2), est polynomial pour les arbres [CK96], c'est-à-dire les graphes de largeur arborescente égale à 1, tandis qu'il est NP-complet pour les graphes de largeur arborescente supérieure ou égale à 2 [FGK05]. Nous pouvons dès lors nous demander si ce changement de complexité se produit également dans le cas du problème COLORATION ADDITIVE. Toutefois, ces deux problèmes ne sont pas identiques : le problème COLORATION ADDITIVE impose des contraintes arithmétiques fortes sur les couleurs associées aux sommets.

Conclusion et perspectives

Dans ce travail de thèse, nous nous sommes intéressés à quelques limites liées aux décompositions de graphes. Ce travail a été motivé par la faible quantité de travaux s'intéressant à ces limites, et en particulier celles relatives aux décompositions arborescentes (à l'exception de récentes études des mineurs « grille » et « pseudo-grille »), comparé aux nombreux articles étudiant les diverses conséquences positives de leur utilisation.

La première partie de notre travail, présentée dans le CHAPITRE 2, a porté sur la construction exacte d'obstructions pour certaines largeurs de graphes. À l'origine de ce travail, nous avons souhaité obtenir le premier algorithme permettant de construire une obstruction optimale pour la largeur arborescente, certifiant qu'un graphe donné est de grande largeur arborescente.

À partir de récents résultats de AMINI *et al.* [AMNT09] et de LYAUDET *et al.* [LMT10], que nous avons adaptés et étendus, nous avons obtenu un algorithme permettant de construire, en temps XP, une obstruction pour diverses largeurs de graphes. En fait, cet algorithme générique s'applique à toute largeur pour laquelle nous pouvons obtenir une famille de partitions $\mathcal{P}$ de taille raisonnable, et pour laquelle la famille étendue de partitions $\hat{\mathcal{P}}$ est raffinante. Nous obtenons ainsi le premier algorithme permettant de construire une obstruction à la largeur arborescente, en temps $\mathcal{O}(n^{\mathrm{tw}+4})$ où n est la taille du graphe et tw sa largeur arborescente.

De plus, notre algorithme peut être aisément adapté pour retourner une décomposition de petite largeur du graphe, lorsque cela est possible. En conséquence, notre algorithme certifie dans tous les cas la réponse qu'il retourne, qu'il s'agisse d'une obstruction certifiant la grande largeur du graphe, ou d'une décomposition certifiant la petite largeur du graphe. En d'autres termes, notre algorithme généralise et unifie à la fois la construction d'obstructions de différentes largeurs, notamment l'algorithme de HICKS [Hic04] pour la largeur de branches, mais également la construction d'une décomposition tel que l'algorithme de ARNBORG *et al.* [ACP87], en temps XP (et donc polynomial lorsque le paramètre est fixé).

Dans la deuxième partie de notre travail, présentée dans le CHAPITRE 3, nous avons étudié la complexité paramétrée du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT pour de nouveaux cas d'ensembles σ et ρ. Deux aspects ont motivé ce travail : d'une part, il existe très peu de problèmes paramétrés ne devenant pas FPT lorsque paramétrés par la largeur arborescente du graphe donné en entrée, tout en appartenant à la classe XP (c'est-à-dire dont l'étude du point de vue paramétré est pertinente) ; d'autre part, dans tous les cas d'ensembles σ et ρ précédemment étudiés (finis ou cofinis), ainsi que dans le cas d'ensembles ultimement périodiques étudié dans mon mémoire de Master [Cha08], le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT a été montré FPT.

Nous avons montré que pour un nombre infini de nouveaux cas pour les ensembles σ et ρ, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT n'est pas FPT. Plus précisément, le problème devient $\mathrm{W}[1]$-difficile lorsque l'ensemble σ contient des intervalles de longueur arbitraire entre deux éléments consécutifs (avec une condition supplémentaire et nécessaire sur cet ensemble). Par ce résultat, nous avons obtenu une nouvelle collection de problèmes n'étant pas FPT lorsque paramétrés par la largeur arborescente.

Enfin, nous avons présenté dans le CHAPITRE 4 les résultats d'une collaboration effectuée dans le cadre d'un projet franco-chilien, et portant sur l'étude d'un nouveau problème de coloration appelé COLORATION ADDITIVE. Dans ce problème, les couleurs affectées aux sommets ne doivent pas créer de 3-progression arithmétique, faisant de ce fait intervenir un problème depuis longtemps ouvert en théorie des nombres.

Nous avons montré qu'à l'instar des problèmes classiques de coloration, ce problème est NP-complet pour tout nombre fixé $k \geq 4$ de couleurs, impliquant que ce problème est paraNP-complet, et qu'il n'est donc pas pertinent de l'étudier du point de vue paramétré si le paramètre est le nombre maximum de couleurs utilisées. Nous avons également montré que ce problème peut être résolu en temps polynomial sur les arbres, c'est-à-dire les graphes de largeur arborescente 1, apportant une première étape dans l'étude de ce problème lorsque paramétré par la largeur arborescente.

Perspectives

De nombreuses perspectives apparaissent à la suite de ce travail de thèse. Nous souhaitons discuter ici de certaines de ces perspectives, relatives aux limites des décompositions de graphes et aux problèmes étudiés durant cette thèse.

Largeurs de graphes. Le calcul de la largeur arborescente d'un graphe quelconque est un problème NP-complet ; l'existence d'un algorithme polynomial pour trouver la valeur optimale de cette largeur pour un graphe donné est donc peu probable. Toutefois, par un algorithme de BODLAENDER [Bod96], il est possible de déterminer en temps FPT paramétré par $k \in \mathbb{N}$, plus précisément en temps $\mathcal{O}\big(f(k) \cdot n\big)$, si un graphe est de largeur arborescente au plus k, et le cas échéant construire une décomposition arborescente de largeur au plus k. En conséquence, pour tout entier k fixé, nous disposons d'un algorithme déterminant en temps linéaire si un graphe est de largeur arborescente au plus k. La fonction f est cependant de la forme $2^{\mathcal{O}(k^3)}$, et de ce fait l'exposant dépendant du paramètre k n'est pas linéaire.

Par ailleurs, le « théorème des mineurs de graphes » de ROBERTSON et SEYMOUR (voir notamment [RS86a, RS04]), une obstruction à la largeur arborescente (*bramble*) peut également être construite en temps FPT, mais le temps d'exécution de l'algorithme obtenu est déraisonnable (il inclut une tour d'exponentielles). Quant à l'algorithme générique que nous avons décrit, il est de complexité XP.

Question. Peut-on écrire un algorithme FPT s'exécutant en temps $\mathcal{O}^*\big(2^{\mathcal{O}(\mathrm{tw})}\big)$ pour le calcul de la largeur arborescente d'un graphe quelconque, par la construction d'une décomposition de largeur minimale ou d'une obstruction d'ordre maximal ?

Notre algorithme généralise et unifie la construction d'obstructions, et nous avons défini de *bonnes* familles de partitions pour quelques largeurs de graphes connus. Néanmoins, il ne nous a pas été possible d'obtenir une telle famille de partitions pour la

largeur de rang (*rank-width*) à partir de la famille *naturelle*, celle-ci étant de taille exponentielle.

Question. Peut-on obtenir une *bonne* famille de partitions pour la largeur de rang (*rank-width*), et ainsi utiliser notre algorithme pour la construction d'une obstruction à la largeur de rang ?

La meilleure approximation connue à ce jour pour la largeur arborescente d'un graphe quelconque est à facteur $\mathcal{O}(\mathrm{tw}\sqrt{\log \mathrm{tw}})$, par un résultat de FEIGE *et al.* [FHL08], qui donnent toutefois une approximation à facteur constant pour les graphes excluant un certain mineur donné. Nous ne savons pas si une approximation à facteur constant de la largeur arborescente est possible pour les graphes quelconques. L'une des pistes possibles pour obtenir une telle approximation est la construction d'une *bonne* obstruction à la largeur arborescente, de taille polynomiale, et notamment la recherche d'une pseudo-grille comme mineur [RW08]. Notons que par un résultat de GROHE et MARX [GM09], montrant que certains graphes de largeur k n'admettent que des obstructions optimales de cardinalité exponentielle en k, la notion classique d'obstruction pour la largeur arborescente (*bramble*) semble ne pas convenir.

Question. Peut-on approximer à facteur constant la largeur arborescente d'un graphe quelconque, notamment par la construction d'une *bonne* obstruction ?

Problèmes paramétrés par la largeur arborescente. Il a été montré précédemment que les cas les plus *structurés* des ensembles σ et ρ, c'est-à-dire lorsqu'ils sont ultimement périodiques, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT est FPT. Nous avons vu que pour lorsque l'ensemble σ est moins structuré, et plus précisément lorsqu'il contient des intervalles de longueur arbitraire, le problème ENSEMBLE $[\sigma, \rho]$-DOMINANT n'est plus FPT et devient $\mathrm{W}[1]$-difficile. Pour parfaire la connaissance de la complexité paramétrée de ce problème, certains cas pour les ensembles σ et ρ restent à étudier, notamment lorsque ρ est moins structuré, ou lorsque ces deux ensembles ne contiennent que des intervalles de longueur bornée.

Question. Quelle est la complexité du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, paramétré par la largeur arborescente, pour d'autres cas *naturels* des ensembles σ et ρ ?

Très peu de problèmes sont connus pour ne pas être FPT lorsqu'ils sont paramétrés par la largeur arborescente, et nous avons en ce sens présenté dans le CHAPITRE 3 une nouvelle collection de problèmes étant $\mathrm{W}[1]$-difficiles avec ce paramètre. Trouver d'autres problèmes dans le même cas pourrait nous apporter une meilleure compréhension des liens unissant la largeur arborescente à la *difficulté* des problèmes étudiés sous ce paramètre.

Question. Existe-t-il d'autres problèmes n'étant pas FPT lorsqu'ils sont paramétrés par la largeur arborescente du graphe donné en entrée ?

La *vraie* complexité des problèmes paramétrés par la largeur arborescente est encore mal connue. De récents résultats tentent de répondre à cette question, en étudiant les aspects et conséquences théoriques de toute amélioration de leur complexité paramétrée. D'un côté, LOKSHTANOV *et al.* [LMS10] ont montré que, sous l'hypothèse ETH (*Exponential-Time Hypothesis*), la complexité connue de certains des problèmes fondamentaux paramétrés par la largeur arborescente est optimale, donnant ainsi une borne inférieure sur les possibilités d'amélioration. De l'autre, CYGAN *et al.* [CNP+11]

ont amélioré drastiquement la complexité connue de nombreux problèmes de connectivité paramétrés par la largeur arborescente, en donnant un algorithme dont la dépendance exponentielle au paramètre est linéaire.

Question. Quelle est la *vraie* complexité des problèmes paramétrés par la largeur arborescente ?

Coloration additive. Nous avons vu que le problème de coloration k-COLORATION ADDITIVE est NP-complet pour tout $k \geq 4$ fixé, tandis que la plupart des problèmes classiques de coloration sont connus pour être NP-complet même lorsque $k = 3$. Notre réduction générale ne s'applique cependant pas au cas où $k = 3$, car celui-ci est relativement différent compte tenu des contraintes arithmétiques définissant le problème k-COLORATION ADDITIVE, et de ce fait nous ne connaissons pas encore la complexité du problème lorsque $k = 3$.

Question. Quelle est la complexité du problème 3-COLORATION ADDITIVE ?

Nous avons montré que le problème COLORATION ADDITIVE peut être résolu en temps polynomial sur les arbres, c'est-à-dire les graphes de largeur arborescente égale à 1. Il serait intéressant de déterminer la complexité de ce problème pour de plus grandes largeurs arborescentes, et en premier lieu dans le cas où la largeur arborescente est au plus égale à 2. En effet, certains problèmes de coloration, tel que le problème ÉTIQUETAGE $L(2,1)$, sont connus pour être polynomiaux sur les arbres mais NP-complets sur les graphes de largeur arborescente au plus 2.

Question. Quelle est la complexité du problème COLORATION ADDITIVE lorsqu'il est restreint aux graphes de largeur arborescente bornée ?

Enfin, le cas de la clique de taille n est une situation assez particulière du problème COLORATION ADDITIVE, car il correspond alors à un problème de théorie des nombres encore ouvert aujourd'hui, consistant à trouver le plus grand sous-ensemble d'entiers parmi $\{1, \ldots, n\}$ ne contenant pas de 3-progression arithmétique. Résoudre le problème COLORATION ADDITIVE dans ce cas semble donc difficile, mais peut être une piste pour tenter d'attaquer ce problème de théorie des nombres.

Question. Peut-on résoudre le problème COLORATION ADDITIVE sur les cliques en temps polynomial ?

Bibliographie

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

[AG09] Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54:544–556, 2009.

[AH76] Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. *Bulletin of the American Mathematical Society*, 82(5):711–712, 1976.

[AM92] Samson Adamsky and Tom S.E. Maibaum. *Handbook of logic in computer science: background: mathematical structures*, volume 1. Oxford University Press, 1992.

[AMNT09] Omid Amini, Frédéric Mazoit, Nicolas Nisse, and Stephan Thomassé. Submodular partition functions. *Discrete Mathematics*, 309(30):6000–6008, 2009.

[BCRW09] Yuehua Bu, Dong Chen, André Raspaud, and Weifan Wang. Injective coloring of planar graphs. *Discrete Applied Mathematics*, 157(4):663–672, 2009.

[BD02] Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Combinatoric, Probability, and Computing*, 11:1–7, 2002.

[Beh46] Felix A. Behrend. On sets of integers which contain no three terms in arithmetical progression. In *Proceedings of the National Academy Science, USA*, volume 32, pages 331–332, 1946.

[Ber70] Claude Berge. *Graphes et hypergraphes*. Dunod, 1970.

[BFF+10] Lali Barrière, Paola Flocchini, Fedor V. Fomin, Pierre Fraigniaud, Nicolas Nisse, Nicola Santoro, and Dimitrios M. Thilikos. Connected graph searching. Technical Report RR-7363, INRIA, August 2010.

[BGK08] Hans L. Bodlaender, Alexander Grigoriev, and Arie M.C.A. Koster. Treewidth lower bounds with brambles. *Algorithmica*, 51(1):81–98, 2008.

[BHKK07] Andreas Björklund, Thore Husfledt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *STOC'07, Proceedings of the 39th Annual ACM Symposium on the Theory of Computing*, pages 67–74, San Diego, California, June 2007.

[BHMV94] Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p-recognizable sets of integers. *Bulletin of the Belgian Mathematical Society*, 1(2):191–238, 1994.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[Bou99] Jean Bourgain. On triples in arithmetic progression. *Geometric and Functional Analysis*, 9(5):968–984, 1999.

[Bou08] Jean Bourgain. Roth's theorem on progressions revisited. *Journal d'Analyse Mathématique*, 104:155–192, 2008.

[BT01] Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: grouping the minimal separators. *SIAM Journal on Computing*, 31(1):212–232, 2001.

[BT02] Vincent Bouchitté and Ioan Todinca. Listing all potential maximal cliques of a graph. *Theoretical Computer Science*, 276(1–2):17–32, 2002.

[BXTV09] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. In *IWPEC'09, Proceedings of the 4th International Workshop on Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2009.

[BXTV11] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. *Theoretical Computer Science*, 412(39):5187–5204, 2011.

[CE11] Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*. Cambridge University Press, 2011.

[Cha08] Mathieu Chapelle. Domination généralisée sur quelques classes de graphes. Master's thesis, Université Paul-Verlaine, Metz, 2008.

[CHRW11] Min Chen, Geňa Hahn, André Raspaud, and Weifan Wang. Some results on the injective chromatic number of graphs. *Journal of Combinatorial Optimization*, 2011.

[CK96] Gerard J. Chang and David Kuo. The $L(2,1)$ labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996.

[CKK+00] Gerard J. Chang, Wen-Tsai Ke, David Kuo, Daphne D.-F. Liu, and Roger K. Yeh. On $L(d,1)$-labelings of graphs. *Discrete Mathematics*, 220:57–66, 2000.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CKY10] Daniel W. Cranston, Seog-Jin Kim, and Gexin Yu. Injective colorings of sparse graphs. *Discrete Mathematics*, 310(21):2965–2973, 2010.

[CMR01] Bruno Courcelle, Johann A. Makowsky, and Udi 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.

[CMT09] Mathieu Chapelle, Frédéric Mazoit, and Ioan Todinca. Constructing brambles. In Ratislav Královič and Damian Niwiński, editors, *MFCS'09, Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science*, volume 5734 of *Lecture Notes in Computer Science*, pages 223–234, Nový Smokovec, High Tatras, Slovakia, August 2009. Springer.

[CN99] Paul Cull and Ingrid Nelson. Error-correcting codes on the tower of Hanoi graphs. *Discrete Mathematics*, 208/209:157–175, 1999.

[CNP+11] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M.M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS'11, Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, 2011.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC'71, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, 1971.

[Cou97] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 313–400, River Edge, NJ, 1997. World Scientific.

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness. I. basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[DF95b] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness. II. on completeness for W[1]. *Theoretical Computer Science*, 141:109–131, 1995.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer, 1999.

[DHR10] Alain Doyon, Geňa Hahn, and André Raspaud. Some bounds on the injective chromatic number of graphs. *Discrete Mathematics*, 310(3):585–590, 2010.

[Die10] Reinhard Diestel. *Graph theory*, volume 173. Springer-Verlag, New York, 4 edition, 2010.

[Dik59] Edsger W. Dikjstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[DJGT99] Reinhard Diestel, Tommy R. Jensen, Konstantin Yu. Gorbunov, and Carsten Thomassen. Highly connected sets and the excluded grid theorem. *Journal of Combinatorial Theory, Series B*, 75(1):61–73, 1999.

[DLSV08] Michaël Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: a parameterized perspective. In *IWPEC'08, Proceedings of the 3rd International Workshop on Parameterized and Exact Computation*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90, Victoria, Canada, May 2008. Springer.

[Elk10] Michael Elkin. An improved construction of progression-free sets. In *SODA'10, Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 886–905. ACM-SIAM, 2010.

[ET36] Paul Erdős and Paul Turán. On some sequences of integers. *Journal of the London Mathematical Society*, 11:261–264, 1936.

[FFL+07] Michaël Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, and Stefan Thomassen. On the complexity of some colorful problems parameterized by treewidth. In *COCOA'07, Proceedings of the 1st International Conference on Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 366–377, Xi'an, China, August 2007. Springer.

[FG03] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[FG06] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer, 2006.

[FGK05] Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In *ICALP'05, Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 3650–372, Lisboa, Portugal, July 2005. Springer.

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *ACM Journal on Computing*, 38(2):629–657, 2008.

[FK98] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

[FK10] Fedor V. Fomin and Dieter Kratsch. *Exponential time algorithms*. Springer, 2010.

[FKTV08] Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, and Yngve Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM Journal on Computing*, 38(3):1058–1079, 2008.

[FMT05] Fedor V. Fomin, Frédéric Mazoit, and Ioan Todinca. Computing branchwidth via efficient triangulations and blocks. In *WG'05, Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 3787 of *Lecture Notes in Computer Science*, pages 374–384, Metz, France, June 2005. Springer.

[GGK08] William Gasarch, James Glenn, and Clyde P. Kruskal. Finding large 3-free sets I: The small n case. *Journal of Computer and System Sciences*, 74(4):628–655, 2008.

[GGRW06] Jim Geelen, Bert Gerards, Neil Robertson, and Geoff Whittle. Obstructions to branch-decomposition of matroids. *Journal of Combinatorial Theory Series B*, 96:560–570, 2006.

[GGW09] Jim Geelen, Bert Gerards, and Geoff Whittle. Tangles, tree-decompositions and grids in matroids. *Journal of Combinatorial Theory Series B*, 99:657–667, 2009.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[GKS99] Arvind Gupta, Damon Kaller, and Thomas Shermer. On the complements of partial k-trees. In *ICALP'99, Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 382–391, Prague, Czech Republic, July 1999. Springer.

[GKS10] Petr A. Golovach, Jan Kratochvíl, and Ondřej Suchý. Parameterized complexity of generalized domination problems. In *WG'09, Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 5911 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2010.

[GM09] Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal on Combinatorial Theory Series B*, 99(1):218–228, 2009.

[GV08] Petr A. Golovach and Yngve Villanger. Parameterized complexity for domination problems on degenerate graphs. In *WG'08, Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 5344 of *Lecture Notes in Computer Science*, pages 195–205. Springer, 2008.

[GW10] Ben Green and Julia Wolf. A note on Elkin's improvement of Behrend's construction. In *Additive Number Theory: Festschrift in Honor of the Sixtieth Birthday of Melvyn B. Nathanson*, pages 141–144, 2010.

[GY92] Jerrold R. Griggs and Roger K. Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992.

[Hal76] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1–2):171–186, 1976.

[Har89] Juris V. Hartmanis. Gödel, von Neumann and the $p =?np$ problem. *Current Trends in Theoretical Computer Science: Essays and Tutorials*, 40:445–450, 1989.

[HB87] David Rodney Heath-Brown. Integer sets containing no arithmetic progressions. *Journal of the London Mathematical Society*, 35:385–394, 1987.

[Hic04] Illya V. Hicks. Graphs, branchwidth, and tangles! oh my! *Networks*, 45(2):55–60, 2004.

[HKvS02] Geňa Hahn, Jan Kratochvíl, Jozef Širáň, and Dominique Sotteau. On the injective chromatic number of graphs. *Discrete Mathematics*, 256(1–2):179–192, 2002.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 2001.

[Hol81] Ian Holyer. The NP-completeness of edge-colouring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[HRS08] Pavol Hell, André Raspaud, and Juraj Stacho. On injective colourings of chordal graphs. In *LATIN'08, Proceedings of the 8th Latin American Symposium on Theoretical Informatics*, volume 4957 of *Lecture Notes in Computer Science*, pages 520–530, Búzios, Brazil, 2008. Springer.

[HS65] Juris Varlejs Hartmanis and Richard Edwin Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] Fred C. Hennie and Richard Edwin Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM*, 13(4):533–546, 1966.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Tatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum.

[Kim09] K.H. Kim. private communication, 2009.

[Klo94] Ton Kloks. *Treewidth. Computations and approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[KLS00] Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3):393–415, 2000.

[KMM95] Jan Kratochvíl, Paul D. Manuel, and Mirka Miller. Generalized domination in chordal graphs. *Nordic Journal on Computing*, 2:41–50, 1995.

[Knu76] Donald E. Knuth. Big Omicron and big Omega and big Theta. *ACM SIGACT News*, 8(2):18–24, 1976.

[KT10] Stephan Kreutzer and Siamak Tazari. On brambles, grid-like minors, and parametrized intractability of monadic second-order logic. In *SODA'10, Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, Austin, Texas, January 2010.

[Lag98] Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *Journal of Combinatorial Theory Series B*, 73:7–40, 1998.

[Lev73] Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):265–266, 1973. in Russian.

[LG83] David Leven and Zvi Galil. NP-completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44, 1983.

[LMS10] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. July 2010.

[LMT10] Laurent Lyaudet, Frédéric Mazoit, and Stephan Thomassé. Partitions versus sets: a case of duality. *European Journal of Combinatorics*, 31(3):681–687, 2010.

[Lok09] Daniel Lokshtanov. *New methods in parametrized algorithms and complexity*. PhD thesis, University of Bergen, April 2009.

[Luc07] Brian Lucena. Achievable sets, brambles, and sparse treewidth obstructions. *Discrete Applied Mathematics*, 155(8):1055–1065, 2007.

[LvT09] Borut Lužar, Riste Škrekvoski, and Martin Tancer. Injective colorings of planar graphs with few colors. *Discrete Mathematics*, 309(18):5636–5649, 2009.

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[Mat94] Armando B. Matos. Periodic sets of integers. *Theoretical Computer Science*, 127:287–312, 1994.

[MZP10] Martin Matamala and José Zamora Ponce. Additive coloring. manuscript, 2010.

[Nie06] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.

[Odi89] Piergiorgio Odifreddi. *Classical recursion theory. The theory of functions and sets of natural numbers*, volume 125 of *Studies in logic, and foundations of mathematics*. North-Holland, 1989.

[OS06] Sang-Il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory Series B*, 96:514–528, 2006.

[Oum05] Sang-Il Oum. *Graphs of bounded rank-width*. PhD thesis, Faculty of Princeton University, May 2005.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[PT05] Christophe Paul and Jan Arne Telle. New tools and simpler algorithms for branchwidth. In *ESA 2005, 13th Annual European Symposium*, volume 3669 of *Lecture Notes in Computer Science*, pages 379–390, Palma de Mallorca, Spain, October 2005. Springer-Verlag.

[PW10] Mihai Pătrascu and Ryan Williams. On the possibility of faster sat algorithms. In *SODA'10, Proceedings of the 21th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075, Hyatt Regency Austin, Austin, Texas, January 2010. ACM-SIAM.

[Ram97] Siddharthan Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM Journal on Discrete Mathematics*, 10(1):146–157, 1997.

[Ree97] Bruce Reed. Tree width and tangles: a new connectivity measure and some applications. In *Surveys in Combinatorics*, 1997.

[Rot52] Klaus F. Roth. Sur quelques ensembles d'entiers. *C.R. Académie des Sciences de Paris*, 234:388–390, 1952.

[Rot53] Klaus F. Roth. On certain sets of integers. *Journal of the London Mathematical Society*, 28:104–109, 1953.

[Rot54] Klaus F. Roth. On certain sets of integers (II). *Journal of the London Mathematical Society*, 29:20–26, 1954.

[RS83] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory Series B*, 35(1):39–61, 1983.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory Series B*, 36(1):49–64, 1984.

[RS86a] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[RS86b] Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory Series B*, 41(1):92–114, 1986.

[RS91] Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree decompositions. *Journal of Combinatorial Theory Series B*, 52(2):153–190, 1991.

[RS04] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory Series B*, 92(2):325–357, 2004.

[RSS08] Venkatesh Raman, Saket Saurabh, and Sriganesh Srihari. Parameterized algorithms for generalized domination. In *COCOA'08, Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications*, volume 5165 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.

[RSST97] Neil Robertson, Tom Sanders, Paul D. Seymour, and Robin Thomas. The four-color theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.

[RST94] Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.

[RW08] Bruce A. Reed and David R. Wood. Polynomial treewidth forces a large grid-like-minor. 2008.

[San11a] Tom Sanders. On certain other sets of integers. *Journal d'Analyse Mathématique*, 2011. To appear.

[San11b] Tom Sanders. On Roth's theorem on progressions. *Annals of Mathematics*, 174:619–636, 2011.

[Sav98] John E. Savage. *Models of computation: exploring the power of computing*. Addison-Wesley, 1998.

[Sip06] Michael Sipser. *Introduction to the theory of computation*. Thomson Course Technology, 2nd edition, 2006.

[SS42] R. Salem and Donald C. Spencer. On sets of integers which contain no three terms in arithmetical progression. In *Proceedings of the National Academy Science, USA*, volume 28, pages 561–563, 1942.

[ST93] Paul D. Seymour and Robin Thomas. Graph searching, and a min-max theorem for tree-width. *Journal of Combinatorial Theory Series B*, 58:22–33, 1993.

[ST94] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[Sze90] Endre Szemerédi. Integer sets containing no arithmetic progressions. *Acta Mathematica Hungarica*, 56(1–2):155–158, 1990.

[Tel94a] Jan Arne Telle. Complexity of domination-type problems in graphs. *Nordic Journal of Computing*, 1:157–171, 1994.

[Tel94b] Jan Arne Telle. *Vertex partitioning problems: characterization, complexity and algorithms on partial k-trees*. PhD thesis, University of Oregon, 1994.

[Tho98] Robin Thomas. An update on the four-color theorem. *Notices of the AMS*, 45(7):848–859, 1998.

[TP97] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

[Tra84] Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Viz65] Vadim G. Vizing. The chromatic class of a multigraph. *Cybernetics*, 1:32–41, 1965.

[vRBR09] Johan M.M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *ESA'09, Proceedings of the 17th Annual European Symposium on Algorithms*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577, Copenhagen, Denmark, September 2009. Springer.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *STOC'10, Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 231–240. ACM, 2010.

[Woe03] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: a survey. In *Papers dedicated to Jack Edmonds, 5th International Workshop*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207, Aussois, France, March 2003. Springer.

[Zuc06] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *STOC'06, Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, Seattle, Washington, USA, May 2006.

Mathieu CHAPELLE

Décompositions de graphes : quelques limites et obstructions

Résumé :
Les décompositions de graphes, lorsqu'elles sont de *petite largeur*, sont souvent utilisées pour résoudre plus *efficacement* des problèmes étant *difficiles* dans le cas de graphes quelconques. Dans ce travail de thèse, nous nous intéressons aux limites liées à ces décompositions, et à la construction d'obstructions certifiant leur grande largeur.
Dans une première partie, nous donnons un algorithme généralisant et unifiant la construction d'obstructions pour différentes largeurs de graphes, en temps XP lorsque paramétré par la largeur considérée. Nous obtenons en particulier le premier algorithme permettant de construire *efficacement* une obstruction à la largeur arborescente en temps $\mathcal{O}(n^{\mathrm{tw}+4})$.
La seconde partie de notre travail porte sur l'étude du problème ENSEMBLE $[\sigma, \rho]$-DOMINANT, une généralisation des problèmes de domination sur les graphes et caractérisée par deux ensembles d'entiers σ et ρ. Les diverses études de ce problème apparaissant dans la littérature concernent uniquement les cas où le problème est FPT, lorsque paramétré par la largeur arborescente. Nous montrons que ce problème ne l'est pas toujours, et que pour certains cas d'ensembles σ et ρ, il devient $\mathrm{W}[1]$-difficile lorsque paramétré par la largeur arborescente.
Dans la dernière partie, nous étudions la complexité d'un nouveau problème de coloration appelé k-COLORATION ADDITIVE, combinant théorie des graphes et théorie des nombres. Nous montrons que ce nouveau problème est NP-complet pour tout $k \geq 4$ fixé, tandis qu'il peut être résolu en temps polynomial sur les arbres pour k quelconque et non fixé.

Mots clés : théorie des graphes, décomposition de graphes, complexité de calcul, complexité paramétrée.

Graphs decompositions: some limits and obstructions

Abstract:
Graphs decompositions of *small width* are usually used to solve *efficiently* problems which are *difficult* in general. In this thesis, we focus on some limits of these decompositions, and the construction of some obstructions certifying a large width.
First, we give a generic algorithm unifying obstructions' construction for several graph widths, in XP time when parameterized by the considered width. In particular, it gives the first algorithm computing *efficiently* an obstruction to tree-width in time $\mathcal{O}(n^{\mathrm{tw}+4})$.
Secondly, we study the parameterized complexity of $[\sigma, \rho]$-DOMINATING SET, a generalization of some domination problems characterized by two sets of integers σ and ρ. All known studies focused only on cases where this problem is FPT when parameterized by tree-width. In this work, we show that there are some cases where the problem is no longer FPT, and become $\mathrm{W}[1]$-hard instead.
Finally, we study the computational complexity of a new coloration problem, named k-ADDITIVE COLORING, which combines both graph theory and number theory. We show that this new problem is NP-complete for any fixed number $k \geq 4$, while it can be solved in polynomial time on trees for any k.

Keywords: graph theory, graph decomposition, computational and parameterized complexity.

Laboratoire d'Informatique Fondamentale d'Orléans
Bâtiment 3IA, rue Léonard de Vinci, B.P. 6759
45067 ORLEANS cedex 2, FRANCE

Version du 22/11/2011 pour jury