Algorithmes de noyau pour des problèmes d'édition de graphes et autres structures

Anthony Perez

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ **M**ONTPELLIER **II**
Sciences et **T**echniques du **L**anguedoc

THÈSE

présentée au Laboratoire d'Informatique de Robotique
et de Microélectronique de Montpellier pour
obtenir le diplôme de doctorat

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Information, Structures, Systèmes**

Algorithmes de noyau pour des problèmes d'édition de graphes et autres structures

par

Anthony PEREZ

Soutenue le 14 Novembre 2011, devant le jury composé de :

Directeur de thèse
M. Christophe PAUL ..... Directeur de Recherche CNRS, LIRMM, Université Montpellier II

Co-Directeur de thèse
M. Stéphane BESSY ..... Maître de Conférences, LIRMM, Université Montpellier II

Rapporteurs
M. Rolf NIEDERMEIER ..... Professor, Technische Universität, Berlin
M. Ioan TODINCA ..... Professeur, LIFO, Université d'Orléans

Président du jury
M. Bruno DURAND ..... Professeur, LIRMM, Université Montpellier II

Examinateur
M. Frédéric HAVET ..... Chargé de recherche CNRS, INRIA Sophia-Antipolis

Table des matières

Table des matières **i**

Remerciements **1**

Introduction **3**
Théorie de la complexité . . . 3
Formalisation de la complexité paramétrée . . . 7
Travaux réalisés durant la thèse . . . 9

1 Préliminaires et notations **13**
1.1 Graphes non-orientés . . . 13
1.1.1 Quelques familles de graphes connues . . . 14
1.1.2 Décomposition modulaire . . . 16
1.1.3 Edition de graphes . . . 18
1.2 Autres structures : collection de relations . . . 18

2 Complexité paramétrée et algorithmes de noyau **21**
2.1 Complexité et algorithmes paramétrés . . . 23
2.1.1 Définitions et hiérarchie de classes . . . 23
2.1.2 Applications et exemples . . . 24
2.1.3 Quel paramètre choisir ? . . . 26
2.2 Algorithmes de noyau . . . 29
2.2.1 Définition et exemples . . . 29
2.3 Techniques pour prouver la non-existence de noyaux polynomiaux . . . 34
2.3.1 Compositions . . . 35
2.3.2 Transformations polynomiales en temps et paramètre . . . 37
2.3.3 Couleurs et identifiants . . . 38
2.3.4 Cross-composition . . . 39

2.3.5 Bornes inférieures pour des problèmes d'édition de graphes 40

Partie I. Edition de graphes - réduire les branches **47**
Règles de réduction génériques pour le problème $\mathcal{G}$-EDITION 49
Réduction via la notion de branches 52
Lien avec les protrusions 54

3 Noyau cubique pour CLOSEST 3-LEAF POWER **57**
3.1 Définition et caractérisation des 3-leaf powers 59
3.1.1 Un outil combinatoire approprié : les cliques critiques 60
3.1.2 Branches 62
3.2 Règles de réduction via la notion de *branches* 63
3.2.1 Couper les 1-branches 64
3.2.2 Couper les 2-branches 67
3.3 Borner la taille d'une instance réduite 71
3.4 Noyaux cubiques pour 3-LEAF POWER COMPLETION et EDGE-DELETION 73

4 Noyau polynomial pour PROPER INTERVAL COMPLETION **77**
4.1 Définition et caractérisation des graphes d'intervalles propres 79
4.1.1 Branches et K-joins 81
4.2 Règles de base 84
4.3 Règles de réduction via la notion de *branches* 85
4.3.1 Borner la taille d'un K-join propre 85
4.3.2 Extraire un K-join propre depuis un K-join 91
4.3.3 Borner la taille d'un K-join 92
4.3.4 Couper les 1-branches 93
4.3.5 Couper les 2-branches 96
4.4 Détecter les branches en temps polynomial 98
4.5 Borner la taille d'une instance réduite 101
4.6 Un cas particulier : BIPARTITE CHAIN DELETION 102

5 Noyau cubique pour COGRAPH EDITION **105**
5.1 Cographes et décomposition modulaire 107
5.2 Noyaux cubiques pour COGRAPH EDGE-DELETION et COMPLETION 109
5.2.1 Règles de réduction basées sur la décomposition modulaire 109
5.2.2 Borner la taille d'une instance réduite 111
5.3 Noyau cubique pour COGRAPH EDITION 113

Partie II. Edition d'autres structures - Conflict Packing **115**

6 Conflict Packing : un outil de conception de noyaux **117**
6.1 Principe général et exemples 119
6.1.1 Partition Sûre et Conflict Packing 119
6.1.2 TRIANGLE EDGE DELETION 122
6.1.3 Algorithme de noyau via ajustement 123

6.2 Noyau quadratique pour CLUSTER VERTEX DELETION 124

7 Applications de la méthode Conflict Packing 127
7.1 Noyau linéaire pour FEEDBACK ARC SET IN TOURNAMENTS 129
7.1.1 Tournois et acyclicité 129
7.1.2 Conflict Packing et Partition Sûre 130
7.2 Noyau linéaire pour DENSE ROOTED TRIPLET INCONSISTENCY 134
7.2.1 Adaptation de la notion de Partition Sûre 136
7.2.2 Définition et conséquences du Conflict Packing 139
7.2.3 Algorithme de noyau 141
7.3 DENSE BETWEENNESS et DENSE CIRCULAR ORDERING 143
7.3.1 DENSE BETWEENNESS 144
7.3.2 DENSE CIRCULAR ORDERING 149

Partie III. Techniques alternatives et Conclusion 151

8 Techniques alternatives 153
8.1 Règles de branchement 155
8.1.1 Sur quels problèmes brancher ? 155
8.1.2 Questions ouvertes 156
8.2 Réduction à treewidth bornée 157
8.2.1 Dichotomie petite/grande treewidth 157
8.2.2 Application au problème MULTICUT 157
8.3 Compression itérative 159
8.3.1 Description générale 159
8.3.2 Application sur FEEDBACK VERTEX SET 161

9 Conclusion et perspectives de recherche 165
9.1 État de l'art 167
9.2 Techniques introduites et résultats obtenus 168
9.3 Perspectives de recherche et problèmes ouverts 169
9.3.1 Edition de graphes 169
9.3.2 Edition de relations 171

Bibliographie 173

Index 185

Table des figures 187

A Bornes inférieures pour C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION 191

B Noyaux pour FEEDBACK ARC SET IN TOURNAMENTS 201

C Réduction du problème MULTICUT à treewidth bornée 215

Remerciements

Bien que m'ayant apporté une expérience professionnelle unique et particulièrement enrichissante, cette thèse a surtout été importante sur le plan humain. Il est évident que je n'aurai pas pu la mener à terme sans l'aide de Stéphane et Christophe, dont la patience, la disponibilité, le soutien, la culture scientifique et la bonne humeur permanente m'ont permis de mener ces trois années de recherche dans des conditions plus que parfaites. En ajoutant le fait de travailler avec du dEUS en fond ou avec L'équipe ouverte en permanence, l'environnement était vraiment idéal. Un grand merci également à Stéphan, avec qui travailler fut un réel bonheur (surtout que tu avais tes doctorants sous la main pour passer tes nerfs, et que j'ai ainsi pu échapper à ton courroux à de nombreuses reprises). Je tiens également à remercier tous les membres de l'équipe AlGCo (Philippe, Ignasi, Alex, Daniel, Benjamin, …), pour leur gentillesse et leur dynamisme, ainsi que le personnel administratif pour m'avoir aidé dans toutes ces démarches.

Durant ces trois années de thèse, j'ai eu l'opportunité de rencontrer de nombreuses personnes, avec lesquelles j'ai eu la chance de pouvoir travailler. Je remercie donc tous mes co-auteurs pour leur collaboration précieuse. Leur contact m'a permis d'apprécier la recherche de différentes manières, et je ne les en remercierai jamais assez. Un immense merci également à Ioan Todinca et Rolf Niedermeier pour avoir accepté de rapporter ce manuscrit, ainsi qu'à Bruno Durand pour avoir accepté de présider le jury de ma soutenance, et à Frédéric Havet pour son rôle d'examinateur.

Je n'oublie bien sûr pas mes co-bureaux : Jean, qui m'a supporté toutes ces années, même durant ma période cheveux longs (bien qu'il ait préféré effacer cette période de sa mémoire). Merci d'avoir eu le bon goût de me faire découvrir La Horde du Contrevent (qui a -presque- compensé ton sacrilège concernant Arctic Monkeys), j'espère que tu sauras apprécier la petite dédicace. Nicolas B., Sandrine, Kevin, Sarah (et Marc-Oliver), merci également pour votre gentillesse, qui vous a permis de supporter sans broncher mes impressionnants retards. Nicolas T., merci pour tous nos moments geeks, nos moult sandwichs partagés et ton coup de main inestimable pour le déménagement ! Merci enfin aux membres du LIFO (notamment Jérôme, Ali, Ioan, Mathieu L. et Mathieu C.), dont l'accueil chaleureux m'a permis de préparer ma soutenance dans les meilleures conditions. Quant

Ω
Δ >
¬) π

à la clique de doctorants/docteurs, merci d'avoir rendu mon intégration si rapide et agréable. Anthony Della Rocca vous salue bien !

Je tiens évidemment à remercier ma famille, et notamment mes parents -qui m'ont forcément poussé à arriver jusque là. Faire une liste exhaustive serait beaucoup trop long, donc je me contenterai d'un message général : merci à tous ! Pour toute votre aide, et pour tous les souvenirs accumulés. Un remerciement particulier au frère (et à Angélique), pour tous nos bons moments passés ensemble, que ce soit à se geler devant un fantastique Nîmes - Croix-de-Savoie ou à se prendre un déluge pendant un concert mythique d'Arcade Fire. Merci aussi pour les innombrables kilomètres que vous m'avez aidé à parcourir. Angélique, merci de ne pas avoir (trop) râlé face à la masse de foot que nous t'avons imposée, et pour ta gentillesse. Enfin, je tiens à remercier ma belle-famille, pour son accueil incroyable et son aide plus que précieuse (mais si Kévin tu nous aideras à déménager dans l'autre sens, c'était pas si dur !). Mathieu, merci de me rappeler chaque fois qu'on se voit (même indirectement) qu'il existe des docteurs qui sauvent des vies, eux.

Pour terminer, un immense merci à tous mes amis de longue date, sans qui ces trois années n'auraient pas été les mêmes. Olivier, merci pour ton côté cartésien qui m'a aidé à relativiser plus de situations que tu ne l'imagines, et pour toutes nos sessions Playstation qui m'auront vidé la tête et fait le plus grand bien (même si tu campes). Je vous souhaite tout le bonheur du monde à toi, Stéphanie et au petit Lucas qui vient tout juste de débarquer. Mattias, merci pour tes innombrables coups de main, pour ta bonne humeur hautement communicative, pour ta concentration plus qu'exceptionnelle à Fifa, et pour tous nos fous rires ("passe ! centre ! frappeeee ! ! oooooohh non la barre !"). Coni, enfin, même si on ne se voit plus des masses, chacune de nos rencontres -aussi brèves soient-elles- est un réel plaisir, et j'espère qu'on aura plus l'occasion de se croiser à l'avenir.

Tous ces jolis remerciements n'auraient jamais vu le jour sans le soutien et la bonne humeur constante de Virginie, qui a (presque !) réussi à effacer mon côté gronchon invétéré, et à me faire sortir de ma tanière plus que je ne l'aurai jamais imaginé. Je ne serai jamais allé au bout si tu ne m'avais pas transformé en amont, et si je n'avais pas eu ton soutien. Je ne suis pas très fort pour exprimer les sentiments, alors je vais conclure avec une phrase qui n'aura guère de sens pour la plupart des gens, mais qui en a énormément pour nous.

> *Let's go for a drive, and see the town tonight. There's nothing to do, but I don't mind, when I'm with you.*

Introduction

Théorie de la complexité

Contexte historique. La **théorie de la complexité** a été introduite afin de permettre de mesurer la difficulté d'un problème algorithmique en fonction de la **taille** de ses instances. De manière plus générale, la théorie de la complexité mesure la **difficulté d'un algorithme** résolvant le problème considéré. Il existe deux principaux types de problèmes : les problèmes d'**optimisation**, où l'objectif est de minimiser ou de maximiser une certaine propriété, et les problèmes de **décision**, demandant une réponse de type **oui/non**. Dans la suite de cette introduction ainsi que de cette thèse, nous nous intéressons principalement à des problèmes de **décision**. De plus, nous considérons uniquement l'aspect **temporel** de la théorie de la complexité, et ne considérons donc pas la notion d'**espace**. Un problème peut se résoudre de manière **efficace** lorsqu'il existe un algorithme le résolvant avec un temps d'exécution **polynomial en la taille de l'instance considérée**. Pendant de nombreuses années, la recherche d'algorithmes polynomiaux pour des problèmes donnés a constitué un axe de recherche majeur [51, 58, 69, 113]. Cependant, de nombreux problèmes classiques (tels que le problème du VOYAGEUR DE COMMERCE [46]) n'arrivaient pas à être résolus en temps polynomial, laissant supposer que tous les problèmes n'avaient pas une difficulté équivalente. Dans un papier fondateur, intitulé **Paths, Trees and Flowers** et paru en **1965** dans le **Canadian Journal of Mathematics**, **Jack Edmonds** a été l'un des premiers à pressentir que certains problèmes pouvaient effectivement être plus **difficiles** que d'autres. Cela se traduit en particulier par la citation suivante :

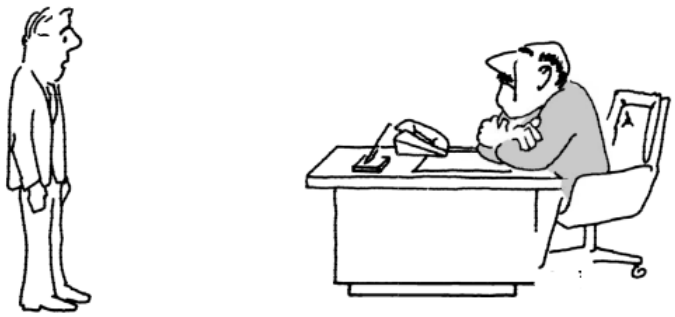
> [...] *There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether or not there exists an algorithm whose difficulty increases only algebraically with the size of the graph.*

L'existence systématique d'un algorithme polynomial pour un problème donné était donc remise en question, et permettait d'ouvrir de nouvelles perspectives de recherche.

> *Does there or does there not exist an algorithm of given order of difficulty for a given class of problems ?*

Quelques années après la parution de cet article, la théorie de la NP-**Complétude** a été formalisée [40, 72], permettant de donner un cadre théorique aux idées informelles de **Jack Edmonds**.

Théorie de la NP-Complétude. L'intuition de la **théorie de la** NP-**Complétude** est la suivante, et est décrite comme telle dans le livre **A guide to the theory of** NP-**Completeness**, par **Garey et Johnson** [72]. Supposons qu'une personne essaie de résoudre de manière **efficace** un problème L et, qu'après plusieurs mois de travail, elle ne soit pas capable de concevoir un tel algorithme. En l'état, cette personne doit donc se résoudre à admettre :



© Garey, M. & Johnson, D. - W. H. Freeman and Co., 1979

Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial.

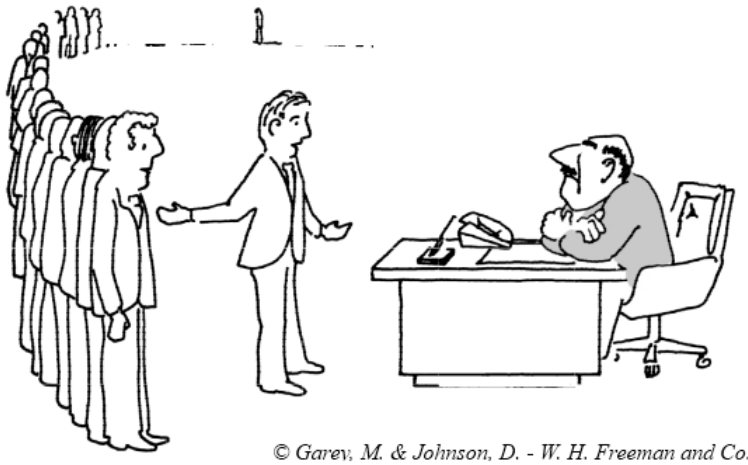
En suivant l'idée de **Jack Edmonds** stipulant que certains problèmes sont probablement plus difficiles que d'autres, il serait alors idéal de trouver un argument permettant de prouver la non-existence d'un tel algorithme.

© Garey, M. & Johnson, D. - W. H. Freeman and Co., 1979

Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial car il n'existe pas de tel algorithme.

Malheureusement, déterminer la non-existence d'un algorithme polynomial pour un problème est **au moins aussi difficile** que d'en démontrer l'existence. Ainsi, cette piste de recherche ne semble pas être concluante. La solution proposée par la **théorie de la** NP-**Complétude** est d'étudier les **relations d'équivalence** existant entre les problèmes. Plus exactement, cette théorie se base

sur le principe de **réductions polynomiales**. Soient L et L' deux problèmes. Le problème L se **réduit en temps polynomial vers** L' s'il existe un algorithme qui prend en entrée une instance I de L, s'exécute en temps polynomial en $|I|$ et retourne une instance **équivalente** I' du problème L'. Ainsi, tout algorithme résolvant L' en temps polynomial résoudra L en temps polynomial. En effet, l'algorithme consistant à réduire L vers L' en temps polynomial puis à utiliser l'algorithme résolvant L' en temps polynomial permet de résoudre L en temps polynomial. De ce fait, le problème L' est **au moins aussi difficile** que le problème L. Ainsi, si l'on peut réduire le problème étudié L **depuis** un problème **classique** L' considéré comme **difficile**, cela permettra d'établir que L est **aussi difficile** que L', qui a été largement étudié et pour lequel aucun algorithme polynomial n'est connu.



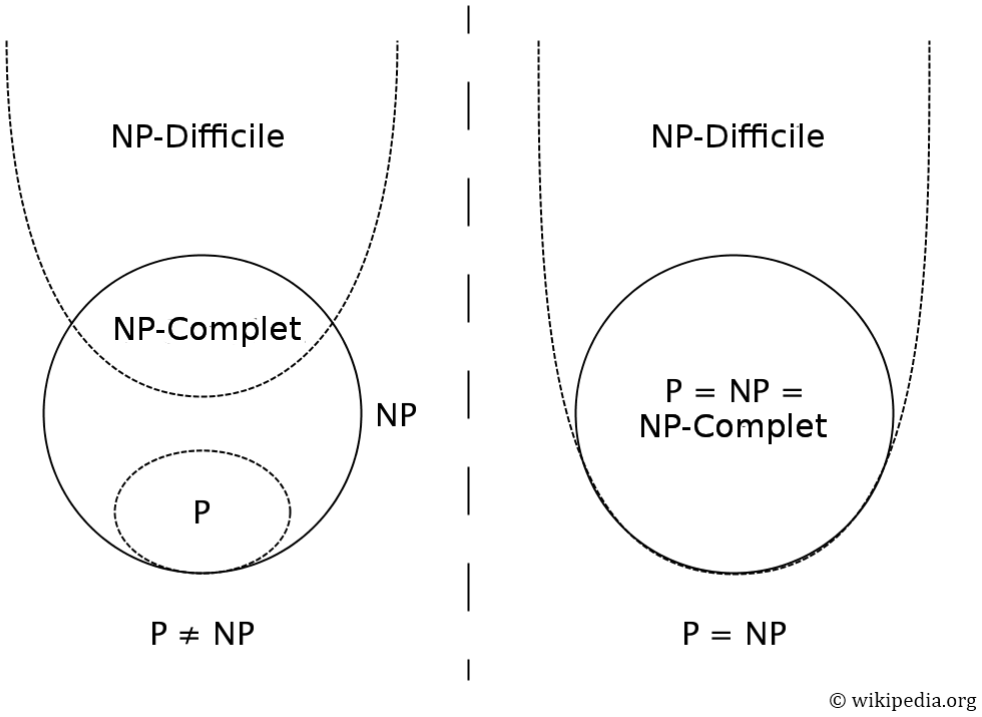
Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial, mais je peux vous garantir qu'aucune de ces personnes n'y est arrivée.

Il reste donc à déterminer formellement ce que sont les **problèmes difficiles**. Ce concept peut être établi via la **théorie de la** NP**-Complétude**, qui s'appuie principalement sur les deux classes de problèmes suivantes :

- P : classe de problèmes pouvant se résoudre de manière **déterministe** en temps **polynomial**.
- NP : classe de problèmes pouvant se résoudre de manière **non-déterministe** en temps **polynomial**.

Une autre formulation de la classe NP est la suivante : un problème appartient à NP s'il est possible de **vérifier en temps polynomial** un **certificat** pour le problème, *i.e.* de vérifier en temps polynomial si une solution proposée est une solution du problème considéré. De manière non formelle, les problèmes de la classe P sont ceux que l'on peut **résoudre efficacement**, ceux de la classe NP ceux que l'on peut **vérifier efficacement**. Considérée comme l'une des **conjectures du millénaire** par le **Clay Mathematics Institute**, la conjecture suivante est l'hypothèse de travail supposée tout le long de cette thèse et a été formulée indépendamment par **Stephen COOK** [40] et **Leonid LEVIN** [115] en **1971**.

Conjecture 1 ($P \neq NP$)**.** *Il existe un problème de la classe NP qui n'appartient pas à la classe P.*



Une illustration de la hiérarchie de complexité sous les hypothèses $P \neq NP$ et $P = NP$.

Nous donnons maintenant la définition centrale de cette théorie, qui permet de définir formellement des problèmes **difficiles**.

Définition (Problème NP-Complet)**.** *Un problème L est NP-*Complet *si $L \in NP$ et si pour tout problème $L' \in NP$ il existe une réduction polynomiale de L' vers L.*

Remarque. Un problème vérifiant la définition précédente mais n'appartenant pas à NP est dit NP-**Difficile**.

Intuitivement, un problème NP-Complet est **au moins aussi difficile** que **tout problème appartenant à** NP, dans la mesure où un algorithme polynomial pour un problème NP-Complet permettrait de résoudre **tous** les problèmes de la classe NP. L'avancée majeure de cette théorie est représentée par le **Théorème de Cook** [40], datant de **1971** et démontrant qu'il existe un problème NP-Complet.

Théorème (Cook [40])**.** *Le problème* SAT *est NP-Complet.*

Ce théorème a ainsi permis d'établir une hiérarchie de problèmes NP-Complets de manière rigoureuse, en utilisant la notion de réduction polynomiale. En effet, puisque le problème SAT est NP-Complet, tout problème de NP se réduit à SAT en temps polynomial. Considérons maintenant un problème $L \in NP$: s'il existe une réduction polynomiale de SAT vers L, alors L est NP-Complet. En effet, tout problème de NP se réduit à SAT, et donc à L par transitivité, en temps polynomial.

Ainsi, pour démontrer la NP-Complétude d'un problème L, il est suffisant de construire une réduction polynomiale d'un problème NP-Complet L' vers L. Utilisant cette remarque, **Karp** a établi, dès **1972**, une collection de **21 problèmes NP-Complets**, parmi lesquels CLIQUE, VERTEX COVER et HITTING SET [102].

Cette théorie énonce donc le fait suivant : il existe des problèmes que l'on **ne peut pas résoudre en temps polynomial**, *i.e.* pour lesquels **une complexité super-polynomiale semble inévitable**.

Formalisation de la complexité paramétrée

Partant de ce constat, de nombreuses approches ont depuis été proposées pour considérer la résolution de problèmes dits **difficiles**. En particulier, des **heuristiques** ont été développées [133], et les théories des **algorithmes exponentiels exacts** [66] et des **algorithmes d'approximation** [154] ont reçu une attention considérable durant les dernières décennies. Dans le premier cas, il s'agit de résoudre le problème **de manière exacte** en essayant de limiter au maximum l'explosion combinatoire de la complexité, mesurée par rapport à la taille de l'instance. Dans le second cas, qui concerne principalement les problèmes d'optimisation, l'algorithme a un temps d'exécution **polynomial en la taille de l'instance**, mais retourne une solution approchée, en garantissant un **ratio d'approximation**. De nombreuses classes de complexité ont été dérivées des algorithmes d'approximation, comme par exemple $PTAS$ [154]. Les problèmes L appartenant à cette classe se formalisent comme suit : étant donnés une instance I de L et un entier $\epsilon > 0$, il existe un algorithme **polynomial en** $|I|$ retournant une solution approchant la solution optimale à un facteur $(1+\epsilon)$ pour les problèmes de minimisation, $(1-\epsilon)$ pour les problèmes de maximisation. En d'autres termes, il est possible d'approcher en **temps polynomial** la solution optimale aussi près qu'on le désire. De manière évidente, la complexité de tels algorithmes dépend fortement du ratio d'approximation désiré. Ainsi, des algorithmes ayant une complexité $O(|I|^{\frac{1}{\epsilon}})$ voire $O(|I|^{exp(\frac{1}{\epsilon})})$ sont considérés comme des PTAS. Deux raffinements de ces classes ont été proposés :

- EPTAS : problèmes admettant un PTAS ayant une complexité $f(1/\epsilon)\cdot|I|^{O(1)}$, f étant une fonction calculable **quelconque**.
- FPTAS : problèmes admettant un PTAS ayant une complexité $f(1/\epsilon)\cdot|I|^{O(1)}$, f étant une fonction calculable **polynomiale**.

Dans tous les cas, que ce soit pour les algorithmes exponentiels exacts ou pour les algorithmes d'approximation, la complexité est principalement mesurée en fonction de la **taille de l'instance** $|I|$. En particulier, l'information structurelle apportée par un **paramètre** associé à l'instance -comme par exemple la taille de la solution recherchée- n'est pas prise en compte.

Algorithmes FPT. La notion de **complexité paramétrée** s'est donc construite autour du constat suivant : quelles informations peut apporter l'ajout d'un **paramètre spécifique au problème** pour mesurer sa complexité ? Dans ce cadre particulier, toute instance I d'un problème L est accompagnée d'un **paramètre** k. Le problème L est appelé **problème paramétré**. Deux questions naturelles se posent alors, rejoignant la dichotomie présentée pour les classes $PTAS$ et $FPTAS$:

- étant donnée une instance (I,k) de L, peut-on résoudre le problème L avec une complexité de la forme $O(|I|^k)$?

– peut-on le résoudre avec une complexité de la forme $f(k) \cdot |I|^{O(1)}$ pour une certaine fonction calculable f ?

Dans le premier cas, le problème appartient à la classe XP. Dans le second cas, le problème est dit FPT. Un **algorithme FPT** pour un problème paramétré L est ainsi un algorithme **résolvant une instance** (I,k) **du problème de décision considéré avec une complexité** $f(k) \cdot |I|^{O(1)}$ pour une certaine fonction calculable f. En d'autres termes, l'objectif de la complexité paramétrée est de déterminer si l'explosion combinatoire pour résoudre un problème NP-Complet (inévitable si $P \neq NP$) peut être **confinée au paramètre** k. La notion d'algorithmes FPT trouve ainsi son principal intérêt lorsque le paramètre associé au problème est *petit* comparé à la taille de l'instance, ce qui est souvent le cas dans des applications pratiques. Cette idée est particulièrement bien illustrée par les citations suivantes, issues respectivement de **Parameterized complexity** de **Flum et Grohe** [64] et **Invitation to fixed-parameterized algorithms** de **Rolf Niedermeier** [129].

> *However, measuring complexity only in terms of the input size means ignoring any structural information about the input instances in the resulting complexity theory. Sometimes, this makes problems appear harder than they typically are.*
>
> *The fundamental idea is to restrict the corresponding, seemingly unavoidable, "combinatorial explosion" that causes the exponential growth in the running time of certain problem-specific parameters.*

A titre d'exemple, le problème classique VERTEX COVER (consistant à décider s'il existe un ensemble d'au plus k sommets **couvrant** toutes les arêtes d'un graphe) admet un algorithme FPT ([33, 131], Section 2.1.2). Il est également intéressant de noter que si un problème d'optimisation NP-Difficile appartient à la classe $FPTAS$, alors sa version paramétrée classique (demandant s'il existe une solution de taille au plus k) admet un algorithme FPT [32].

Une hiérarchie de classes, appelée W-hiérarchie [57, 64], a été proposée et permet d'établir (sous certaines hypothèses de travail, dont $P \neq NP$) que **tout problème paramétré n'admet pas un algorithme FPT**. En effet, le problème CLIQUE (ou encore INDEPENDENT SET), demandant l'existence d'une clique de taille k dans un graphe, n'admet pas d'algorithme FPT. Ce problème est alors dit $W[1]$**-difficile** [56]. Un autre exemple majeur est le problème DOMINATING SET, qui est quant à lui $W[2]$**-difficile** [57]. La W-hiérarchie permet ainsi d'organiser la complexité des problèmes paramétrés comme suit :

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[t] \subseteq XP$$

où XP représente la classe des problèmes pouvant se résoudre en temps $|I|^{f(k)}$.

Bien que moins puissante que la Conjecture 3, la conjecture suivante semble être vérifiée.

Conjecture 2 ($FPT \neq W[1]$)**.** *Il existe un problème de la classe* $W[1]$ *qui n'appartient pas à la classe* FPT.

Durant les dernières décennies, de nombreuses techniques permettant de concevoir des algorithmes FPT ont été introduites, comme par exemple le **Color Coding** [6], la **réduction à treewidth bornée** [123, 143] ou encore la **Compression Itérative** [141]. Dans le cadre de cette thèse, nous nous intéressons principalement à une des techniques les plus puissantes permettant de concevoir des algorithmes FPT [14], à savoir les **algorithmes de noyau**.

Algorithme de Noyau. Lorsque l'on rencontre un problème NP-Complet, une approche naturelle est d'essayer de **réduire** au maximum l'instance (en temps polynomial) afin de n'en préserver que la partie réellement difficile. Ce sont ce que l'on appelle les *heuristiques* pour un problème. Observons cependant que, sous l'hypothèse $P \neq NP$, il ne peut pas exister d'algorithme polynomial qui étant donnée une instance I d'un problème NP-Complet L, retourne une **instance équivalente** I' vérifiant $|I'| < |I|$. Une question naturelle se pose alors : comment mesurer **l'efficacité** d'une heuristique réduisant un problème en temps polynomial ?

Une des contributions majeures de la complexité paramétrée est d'offrir un **cadre formel** permettant de mesurer les **performances** des heuristiques pour un problème paramétré, avec la notion d'**algorithme de noyau**. Étant donnée une instance (I,k) d'un problème paramétré L, un **(algorithme de) noyau** pour L est un algorithme **polynomial en** $|I|+k$ retournant une instance **équivalente** (I',k') de L vérifiant $|I'| \leqslant f(k)$ et $k' \leqslant f(k)$ pour une fonction calculable f. Cette fonction est appelée la **taille** du noyau. Un résultat classique en théorie de la complexité établit qu'**un problème paramétré admet un algorithme FPT si et seulement si il admet un algorithme de noyau** [129]. Par défaut, lorsque le problème considéré est NP-Difficile, la taille de noyau obtenue via ce résultat est super-polynomiale en k. Déterminer l'existence d'un noyau ayant la plus petite taille possible (*i.e.* polynomiale -voire linéaire- en k) constitue donc un enjeu important sur le plan pratique [87], mais également sur le plan théorique. En effet, obtenir des noyaux polynomiaux permet d'améliorer la complexité des algorithmes FPT en utilisant des techniques dites d'**interleaving** [130]. De plus, il a récemment été démontré que, à moins que la hiérarchie polynomiale ne s'effondre au troisième niveau, **il existe des problèmes paramétrés n'admettant pas de noyaux polynomiaux** [16, 70].

Travaux réalisés durant la thèse

Problèmes étudiés et plan de la thèse. Dans le cadre de cette thèse, nous nous intéressons à des problèmes paramétrés d'**édition de graphes et autres structures**. Ces problèmes demandent s'il est possible de **modifier** (ou d'**éditer**) au plus k éléments d'une structure donnée afin de lui permettre de satisfaire une certaine propriété. Nous nous intéressons plus particulièrement à **l'existence d'algorithmes de noyaux polynomiaux** pour de tels problèmes. Dans les Chapitres 1 et 2, nous posons les bases nécessaires aux travaux expliqués par la suite. En particulier, dans le **Chapitre 2**, nous définissons formellement la notion de **complexité paramétrée**, en donnant plusieurs exemples basés sur les problèmes VERTEX COVER [33, 131] et CLUSTER EDITING [37, 136] (Section 2.1.2). Par la suite, nous développons la technique dite **d'algorithme de noyau**, constituant la principale méthode étudiée dans le cadre de cette thèse. Nous démontrons et énonçons plusieurs résultats fondamentaux en théorie des noyaux, toujours en nous basant sur les problèmes VERTEX COVER et CLUSTER EDITING (Section 2.2). Finalement, nous détaillons des méthodes récentes permettant d'établir la **non-existence de noyaux polynomiaux** pour certains problèmes paramétrés (Section 2.3). Nous démontrons notamment que les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION n'admettent pas de noyaux polynomiaux pour $l \geqslant 7$. Ces résultats sont basés sur des travaux réalisés avec Sylvain GUILLEMOT, Christophe PAUL et Frédéric HAVET.

Dans la **Partie I**, nous nous consacrons essentiellement aux problèmes d'éditions de graphes. Ces derniers constituent pour la plupart des problèmes NP-Complets, et ont été particulièrement étudiés dans la littérature [35, 73, 100, 102, 147], notamment en raison des nombreuses applications existant pour ces derniers. A titre d'exemple, les problèmes NP-Complets VERTEX COVER [33, 131], FEEDBACK VERTEX SET [36, 139, 151] ou bien encore MULTICUT [22, 126, 139] constituent des problèmes d'édition de graphes majeurs en théorie de la complexité classique et paramétrée. Dans les **Chapitres 3 et 4**, nous étudions respectivement les problèmes CLOSEST 3-LEAF POWER [11, 12] et PROPER INTERVAL COMPLETION [13], ayant des applications en bioinformatique [100, 101, 148]. Dans les deux cas, nous établissons des **noyaux polynomiaux**, qui constituent les premiers noyaux polynomiaux pour ces problèmes. Le premier résultat répond à une question laissée ouverte par **Dom et al.** [52], alors que le deuxième répond à un problème ouvert depuis un article de **Kaplan et al.** paru à **FOCS'94** [100, 101]. Les règles de réduction introduites sont basées sur la notion de *branches* d'un graphe, que nous définissons dans l'introduction de la **Partie I**. Finalement, dans le **Chapitre 5**, nous nous intéressons au problème COGRAPH EDITION, fournissant encore une fois le premier noyau polynomial connu pour le problème. Les **cographes** étant les graphes **totalement décomposables pour la décomposition modulaire** [92], nous utilisons cet outil afin d'élaborer nos règles de réduction.

Dans la **Partie II**, nous considérons des problèmes d'édition pour des structures différentes des graphes non orientés, qui peuvent être visualisées comme une collection de **relations**. Étant donnés un ensemble de sommets V et une collection $\mathscr{R}$ de relations de taille $p \geqslant 2$ définies sur V, l'objectif est d'**éditer** au plus k relations afin que l'ensemble $\mathscr{R}$ ainsi modifié respecte une certaine propriété Π. Un des exemples les plus classiques de problèmes pouvant se formaliser de la sorte est le problème du FEEDBACK ARC SET IN TOURNAMENTS [35, 105]. Dans le **Chapitre 6**, nous développons et améliorons une technique utilisée pour quelques problèmes paramétrés [27, 63, 155], que nous appelons **Conflict Packing** [134]. Cette technique est particulièrement adaptée lorsque la propriété cible Π peut être caractérisée par un ensemble de *conflits finis* (*i.e.* d'ensembles de sommets de taille minimum ne respectant pas la propriété Π). En utilisant cette dernière, nous retrouvons des résultats connus pour les problèmes TRIANGLE EDGE DELETION [27] et CLUSTER VERTEX DELETION [152] (Sections 6.1 et 6.2, Chapitre 6). Finalement, nous utilisons cette technique pour obtenir des noyaux linéaires pour les problèmes DENSE ROOTED TRIPLET INCONSISTENCY et DENSE BETWEENNESS (**Chapitre 7**), établissant de nouveaux résultats dans les deux cas.

Pour conclure cette thèse, nous évoquons dans la **Partie III** des techniques d'algorithmique paramétrée qui peuvent être utilisées lorsque les algorithmes de noyau semblent difficiles à concevoir. Nous évoquons en particulier les techniques de **compression itérative** [141] et de **réduction à treewidth bornée** [42, 123], deux méthodes puissantes qui ont permis d'établir la complexité paramétrée de nombreux problèmes. En particulier, nous mentionnons un travail réalisé avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ [44] dans lequel nous démontrons que le problème MULTICUT peut se réduire en temps FPT à des instances de treewidth bornée en fonction du paramètre k. Ce résultat a servi de base à Bousquet et al. [22] dans l'élaboration d'un algorithme FPT pour MULTICUT, ce qui a permis de répondre à l'un des problèmes ouverts majeurs en théorie de la complexité paramétrée [49].

Publications. Les travaux réalisés dans cette thèse ont donné lieu à plusieurs publications dans des conférences internationales avec comité de lecture, ainsi que des revues. Nous donnons une liste de ces résultats.

Conférences Internationales.

- [11] **Polynomial kernels for 3-LEAF POWER GRAPH MODIFICATION problems**. Avec Stéphane BESSY et Christophe PAUL. **IWOCA 2009**, number 5874 dans *Lecture Notes in Computer Science*, pages 72-80, 2009.
- [9] **Kernels for FEEDBACK ARC SET IN TOURNAMENTS**. Avec Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH et Stéphan THOMASSÉ. **FSTTCS 2009**, number 4 dans *Leibnitz International Proceedings in Informatics*, pages 37-47, 2009.
- [83] **On the (non-)existence of polynomial kernels for PL-FREE EDGE MODIFICATION problems**. Avec Sylvain GUILLEMOT et Christophe PAUL. **IPEC 2010**, number 6478 dans *Lecture Notes in Computer Science*, pages 147-157, 2010.
- [134] **Conflict Packing yields linear vertex-kernels for k-FAST, k-DENSE RTI and a related problem**. Avec Christophe PAUL et Stéphan THOMASSÉ. **MFCS 2011**, number 6907 dans *Lecture Notes in Computer Science*, pages 497-507, 2011.
- [13] **Polynomial kernels for PROPER INTERVAL COMPLETION and related problems**. Avec Stéphane BESSY. **FCT 2011**, number 6914 dans *Lecture Notes in Computer Science*, pages 229-239, 2011.

Revues.

- [12] **Polynomial kernels for 3-LEAF POWER GRAPH MODIFICATION problems**. Avec Stéphane BESSY et Christophe PAUL. *Discrete Applied Mathematics 158* (2010) pages 1732-1744.
- [10] **Kernels for FEEDBACK ARC SET IN TOURNAMENTS**. Avec Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH et Stéphan THOMASSÉ. Accepté pour publication à *Journal of Computer and System Sciences*, 2010.

Divers.

- [44] **Treewidth reduction for the parameterized MULTICUT problem**. Avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ. *Preprint*, 2010.

Réalisation d'un compendium en ligne de problèmes paramétrés. Au cours des dix dernières années, le nombre de résultats relatifs à la complexité paramétrée et la communauté s'intéressant à ce type de problèmes n'ont cessé de croître. De ce fait, il semblait nécessaire d'établir un **compendium de problèmes paramétrés**, permettant de regrouper dans un même endroit un maximum de résultats concernant ce domaine. Un tel compendium a été créé avec succès par **Mario Cesati**, sous la forme d'un PDF regroupant plus de **300 résultats de complexité paramétrée** (http ://bravo.ce.uniroma2.it/home/cesati/research/compendium/compendium.pdf). Cependant, ce dernier n'a pas été mis à jour depuis quelques années, et ne reflète donc plus la réalité actuelle du domaine. En ce sens, et en s'inspirant du **compendium en ligne de problèmes *NP*-Complets** (http ://www.csc.kth.se/~viggo/wwwcompendium/), un **compendium en ligne de problèmes paramétrés** a été créé. Ce dernier est encore en phase de développe-

ment, mais regroupe d'ores et déjà de nombreux résultats paramétrés. A terme, il serait intéressant d'y ajouter des résultats d'**algorithmes exponentiels exacts** ainsi que les bornes des **algorithmes de noyau**, afin de donner un éventail aussi complet que possible en ce qui concerne la complexité (paramétrée) des problèmes. Le compendium est accessible via l'adresse suivante : **http ://www.lirmm.fr/~perez/compendium**.

CHAPITRE

1 Préliminaires et notations

1.1 Graphes non-orientés

Notations. Un *graphe non-orienté* G est une paire (V,E), où V est un ensemble fini de *sommets*, et E un ensemble fini d'*arêtes*, *i.e.* de paires de sommets (*non-ordonnées*). Lorsque le contexte n'est pas clairement défini, nous utilisons $V(G)$ et $E(G)$ pour dénoter les ensembles de sommets et d'arêtes de G, respectivement. Dans la suite de cette thèse, nous posons $n := |V|$ et $m := |E|$. Dans un souci de simplicité, si deux sommets u et v sont reliés par une arête, nous écrivons uv au lieu de $\{u,v\}$ pour désigner cette arête. Étant donnés deux sommets u et v tels que $uv \in E$, nous disons que u et v sont *adjacents*. De manière similaire, étant donnés $u \in V$ et $S \subseteq V$, nous disons que u *est adjacent à* S (ou u *domine* S) si pour tout sommet $s \in S$, l'arête us appartient à E. Étant donnée $e := uv \in E$, les sommets u et v sont dits *incidents* à e. De plus, étant donnés $A, B \subseteq V$, $E(A,B)$ dénote l'ensemble des arêtes uv telles que $u \in A$ et $v \in B$. Finalement, étant donné $F \subseteq (V \times V)$, l'ensemble $E \triangle F := (E \cup F) \setminus (E \cap F)$ dénote la *différence symétrique* de E et F.

graphe non orienté

$E \triangle F$

Voisinages et sous-graphes. Soient $G := (V,E)$ un graphe et v un sommet de V. Le *voisinage ouvert* de v est défini comme $N_G(v) := \{u \in V \ : \ uv \in E\}$, et son *voisinage clos* comme $N_G[v] := N_G(v) \cup \{v\}$. Comme précédemment, lorsque le contexte est clair, nous écrivons simplement $N(v)$ et $N[v]$. Le *degré* d'un sommet v correspond à $|N(v)|$ et est dénoté $d(v)$. Deux sommets u et v sont *vrais* (resp. *faux*) *jumeaux* si $N[u] = N[v]$ (resp. $N(u) = N(v)$ et $uv \notin E$). Le graphe obtenu à partir de G en *ajoutant un vrai* (resp. *faux*) *jumeau à un sommet* u est le graphe $G' := (V \cup \{u'\}, E')$, où u' est un sommet adjacent (resp. non-adjacent) à u ayant le même voisinage que u. Étant donnés un ensemble de sommets $S \subseteq V$ et $v \in V$, nous posons $N_S(v) := N_G(v) \cap S$, et $N_G(S)$ dénote l'ensemble $\cup_{s \in S} N_{V \setminus S}(s)$. Le *sous-graphe induit par* S est le graphe $G[S] := (S, E_S)$, où $E_S := \{uv \in E \ : \ (u \in S) \text{ et } (v \in S)\}$. Dans un abus de notation, nous dénotons par $G \setminus S$ le graphe $G[V \setminus S]$. Ainsi, étant donné $v \in V$, l'ensemble $N_{V \setminus S}(v)$ est également dénoté $N_{G \setminus S}(v)$. Nous définissons la *frontière* de S comme l'ensemble $\delta(S) := \{s \in S \ : \ N_{G \setminus S}(s) \neq \emptyset\}$. De manière similaire, étant donné un ensemble $F \subseteq (V \times V)$, $F[S]$ dénote l'ensemble des paires de F ayant leurs deux extrémités dans S. De plus, si $S' \subseteq V$, $F[S \times S']$ dénote toutes les paires de F ayant une extrémité dans S et une extrémité dans S'.

$N_G(v)$

$N_G[v]$

degré

jumeaux

$N_G(S)$

$G[S]$

frontière

Finalement, un *join* dans un graphe est une partition (V_1, V_2) de V et un ordre $u_1 \dots u_{|V_1|}$ sur U tel que pour tout $i \in [|V_1| - 1]$ (*i.e.* $1 \leqslant i \leqslant |V_1| - 1$), $N_V(u_i) \subseteq N_V(u_{i+1})$.

chemin
cycle
connexe
$d_G(u,v)$
composante connexe

Connexité. Soient $G := (V, E)$ un graphe et u et v deux sommets de G. Un *chemin* reliant u à v dans G est une suite de sommets $P_l := \{u_1, \dots, u_i, u_{i+1}, \dots u_l\}$ telle que $u_1 = u$, $u_l = v$ et $u_i u_{i+1} \in E$ pour tout $i \in [l-1]$. La *longueur* l d'un chemin P_l est égale au nombre de sommets qu'il comporte. Un chemin de longueur l reliant un sommet u à lui-même est appelé un *cycle* de G, et est dénoté C_l. Nous disons que G est *connexe* lorsqu'il existe un chemin de u à v pour tous sommets u et v de V, $u \neq v$. De plus, nous dénotons par $d_G(u,v)$ la distance d'un *plus court chemin* de u à v dans G. Finalement, un ensemble maximal de sommets $C \subseteq V$ tel que le sous-graphe induit $G[C]$ est un graphe *connexe* est appelé une *composante connexe* de G.

$u <_\sigma v$
$u \leqslant_\sigma v$
σ_S

Ordre sur les sommets. Soient $G := (V, E)$ un graphe, et $\sigma := v_1 \dots v_n$ un ordre défini sur les sommets V de G. Pour tous sommets u et v, nous dénotons par $u <_\sigma v$ le fait que $\sigma(u) < \sigma(v)$ (*i.e.* le fait que u apparaisse *avant* v dans σ). Dans le cas où u et v peuvent être égaux, nous utilisons la notation $u \leqslant_\sigma v$. De manière similaire, étant donnés $S \subseteq V$ et $u \in V$, nous posons $u <_\sigma S$ si pour tout sommet $s \in S$, $u <_\sigma s$ est vérifié. Finalement, nous définissons l'*ordre induit par* S comme l'ordre σ restreint aux sommets de S, dénoté σ_S .

héréditaire
$G_1 \oplus G_2$
$\mathscr{H}$-free

Famille de graphes. Soit $\mathscr{G}$ une classe de graphes. Nous disons que $\mathscr{G}$ est *héréditaire* si pour tout graphe $G := (V, E)$ appartenant à $\mathscr{G}$, tout sous-graphe induit de G appartient à $\mathscr{G}$. De plus, nous disons que $\mathscr{G}$ est *close par ajout de vrai* (resp. *faux*) *jumeau* si tout graphe obtenu à partir de G en ajoutant un vrai (resp. faux) jumeau à un sommet de G appartient à $\mathscr{G}$. Finalement, nous disons que $\mathscr{G}$ est *stable par union disjointe* si, étant donnés deux graphes $G_1 := (V_1, E_1)$ et $G_2 := (V_2, E_2)$ de $\mathscr{G}$, le graphe $G_1 \oplus G_2 := (V_1 \cup V_2, E_1 \cup E_2)$ appartient à $\mathscr{G}$. Soit $\mathscr{H}$ une famille de graphes. Un graphe $G := (V, E)$ est *$\mathscr{H}$-free* si et seulement si G ne contient aucun graphe de la famille $\mathscr{H}$ comme sous-graphe induit. Une classe de graphes $\mathscr{G}$ est *caractérisée par un ensemble de sous-graphes induits interdits* s'il existe une famille de graphes $\mathscr{H}$ telle que $G \in \mathscr{G}$ si et seulement si G est $\mathscr{H}$-free.

1.1.1 Quelques familles de graphes connues

Soit $G := (V, E)$ un graphe. Dans la suite cette section, nous décrivons quelques familles de graphes connues et largement utilisées en théorie des graphes. Une illustration de chacun de ces graphes est donnée Figureřeffig :graphesconnus.

Cliques. Un ensemble de sommets $C \subseteq V$ est une *clique* de G si $uv \in E(G)$ pour toute paire de sommets $\{u, v\} \in C \times C$, $u \neq v$.

Ensembles indépendants. Un ensemble de sommets $I \subseteq V$ est un *ensemble indépendant* (ou *stable*) de G si $uv \notin E(G)$ pour toute paire de sommets $\{u, v\} \in I \times I$. En d'autres termes, un ensemble indépendant est le *complémentaire* d'une clique.

Bipartis. Le graphe G est *biparti* si son ensemble de sommets peut être partitionné en deux ensembles indépendants.

Arbres. Le graphe G est un arbre si et seulement si G est connexe et ne contient pas de cycle. De manière équivalente, G est un arbre si et seulement si G est connexe et comporte $n-1$ arêtes. Il existe d'autres caractérisations des arbres, mais les deux énoncées précédemment sont suffisantes pour la suite de cette thèse.

Forêts. Le graphe G est une *forêt* si chacune de ses composantes connexes est un arbre.

Couverture de sommets. Un ensemble de sommets $C \subseteq V$ est un *vertex cover* de G si pour toute arête $uv \in E(G)$ $u \in C$ ou $v \in C$ est vérifié. Il suit alors par définition que le graphe $G \setminus C$ est un ensemble indépendant.

Couplage. Un *couplage* M de G est un ensemble d'arêtes deux-à-deux sommet-disjointes. Un sommet $v \in V(G)$ est *couvert* par une arête du couplage s'il existe $e \in M$ incidente à v. Étant donné $S \subseteq V$, M *sature* S si tout sommet de S est couvert par une arête du couplage.

Graphes triangulés. Le graphe G est *triangulé* (ou *chordal*) s'il ne contient aucun cycle induit de longueur supérieure ou égale à 4. Les graphes triangulés constituent une des familles de graphes les plus étudiées dans la littérature [74, 109, 122, 123, 150, 158], et ont notamment de nombreuses relations avec la notion de *largeur arborescente* [142, 143]. Nous verrons par la suite qu'une grande majorité des propriétés de graphes que nous allons considérer sont des sous-classes des graphes triangulés [53, 100].

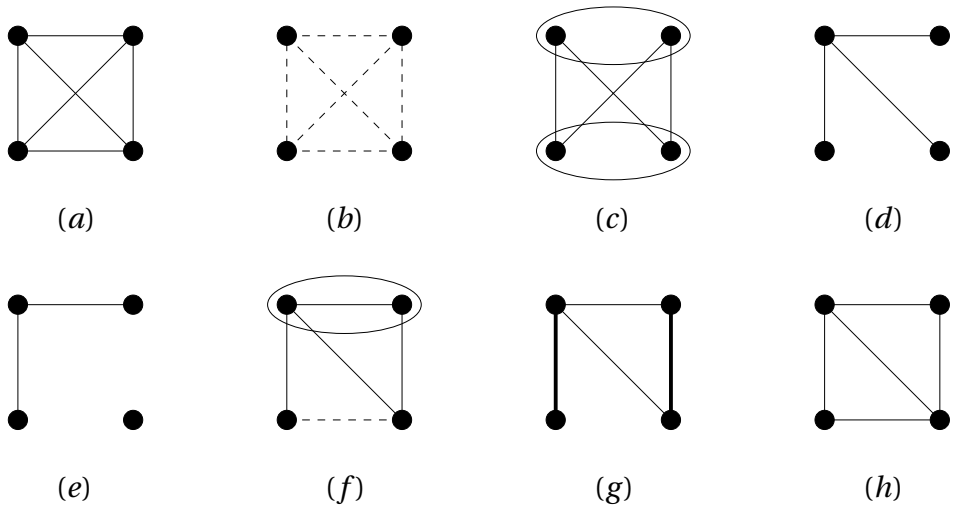


FIGURE 1.1 : Illustration des différentes classes de graphes présentées : (a) une clique ; (b) un ensemble indépendant ; (c) un graphe biparti ; (d) un arbre ; (e) une forêt ; (f) les sommets encerclés représentent un vertex cover ; (g) les arêtes épaisses représentent un couplage ; (h) un graphe triangulé.

1.1.2 Décomposition modulaire

Une notion communément utilisée en théorie des graphes est celle de *décomposition*, permettant d'obtenir certaines propriétés sur un graphe donné. Dans cette thèse, nous évoquons à plusieurs reprises la notion de *décomposition modulaire* d'un graphe [92].

Définition 1.1 (Module)**.** *Soient $G := (V, E)$ un graphe et $M \subseteq V$. L'ensemble M est un* module *de G si pour tout sommet $u \in V \setminus M$, $M \subseteq N_G(u)$ ou $M \cap N_G(u) = \emptyset$.*

En d'autres termes, un ensemble de sommets M est un module si ses sommets ont le même comportement vis-à-vis de l'extérieur du graphe. Observons que pour tous modules M et M', les ensembles M et M' sont soit complètement adjacents, soit non-adjacents. Tout graphe $G := (V, E)$ admet des ensembles de modules particuliers : l'ensemble V et les singletons de V sont des modules de G, dits *triviaux* . Remarquons également que toute composante connexe d'un graphe est module de ce graphe. Les graphes connexes n'admettant que des modules triviaux sont appelés *graphes premiers pour la décomposition modulaire*. Une partition $\mathscr{P} := \{M_1, \ldots, M_l\}$ des sommets d'un graphe $G := (V, E)$ dont les parties sont des modules est appelée *partition modulaire*. Un *graphe quotient* $G_{\mathscr{P}}$ est associé à toute partition modulaire $\mathscr{P}$: ses sommets sont les parties de $\mathscr{P}$ et il existe une arête entre M_i et M_j si M_i et M_j sont adjacents.

modules triviaux

partition modulaire

Définition 1.2 (Module fort)**.** *Soient $G := (V, E)$ un graphe et $M \subseteq V$ un module de G. Le module M est* fort *si pour tout module $M' \subseteq V$, alors $M \cap M' = \emptyset$ ou $M \subseteq M'$ ou $M' \subseteq M$.*

Il suit par la Définition 1.2 que l'ensemble des modules forts d'un graphe s'arrange dans un arbre d'inclusion, appelé *arbre de décomposition modulaire* . Chaque noeud x de cet arbre est associé à un graphe quotient G_x dont les sommets correspondent aux fils $\{x_1, \ldots, x_l\}$ de x (voir Figure 1.2). Nous disons qu'un noeud x de l'arbre est *Série* si G_x est une clique, *Parallèle* si G_x est un ensemble indépendant et *Premier* sinon. Finalement, observons que calculer l'arbre de décomposition modulaire d'un graphe est un problème largement étudié dans la littérature, et peut se faire en temps $O(n+m)$ [92]. Pour conclure, nous définissons deux opérations que nous utiliserons dans le Chapitre 5.

arbre de décomposition modulaire

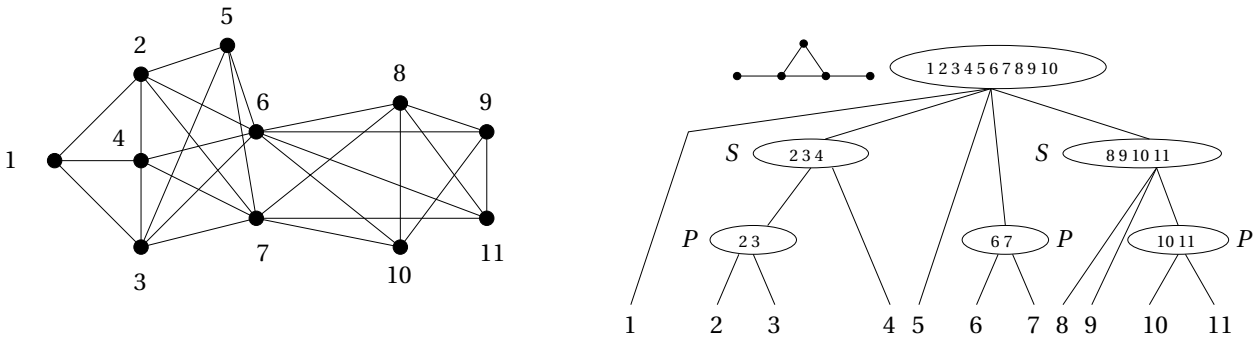


FIGURE 1.2 : Un graphe et son arbre de décomposition modulaire associé.

Définition 1.3 (Composition Série et Parallèle)**.** *Soient* $G_1 := (V_1, E_1)$ *et* $G_2 := (V_2, E_2)$ *deux graphes sommets-disjoints. La* composition Série *de* G_1 *et* G_2 *est le graphe* $G_1 \otimes G_2 := (V_1 \cup V_2, E_1 \cup E_2 \cup V_1 \times V_2)$. *La* composition Parallèle *de* G_1 *et* G_2 *est le graphe* $G_1 \oplus G_2 := (V_1 \cup V_2, E_1 \cup E_2)$.

Remarque. Les noeuds Parallèle et Série de l'arbre de décomposition modulaire correspondent respectivement à des compositions Parallèle et Série de leurs fils.

Cliques critiques

Nous introduisons à présent la notion de *cliques critiques*, qui sont des modules particuliers. Ce concept a été introduit par Lin et al. [116] dans le cadre de l'étude de problèmes phylogénétiques, et nous sera utile dans l'élaboration de plusieurs de nos résultats (Chapitres 3 et 4).

Définition 1.4 (Clique critique [116])**.** *Soient* $G := (V, E)$ *un graphe et* $C \subseteq V$. *L'ensemble* C *est une* clique critique *de* G *si* (i) $G[C]$ *est une clique,* (ii) C *est un module et* (iii) C *est maximal sous ces propriétés.* clique critique

Observation 1.5 (Classique)**.** *Soient* $G := (V, E)$ *un graphe et* $\mathscr{C}_G$ *l'ensemble des cliques critiques de* G. *La famille* $\mathscr{C}_G$ *définit une partition de* V.

De ce fait, nous pouvons définir un graphe quotient utilisant la la partition en cliques critiques du graphe.

Définition 1.6 (Graphe des cliques critiques)**.** *Soient* $G = (V, E)$ *un graphe et* $\mathscr{C}_G$ *la partition de* G *en cliques critiques. Le* graphe des cliques critiques *(Figure 1.3) est le graphe* $\mathscr{C}(G) := (\mathscr{C}_G, E_{\mathscr{C}})$, *où :* graphe des cliques critiques

$$CC' \in E_{\mathscr{C}} \Leftrightarrow \forall\ v \in C,\ \forall v' \in C'\ vv' \in E(G)$$

Remarquons que le graphe des cliques critiques $\mathscr{C}(G)$ d'un graphe $G := (V, E)$ peut-être vu comme un sous-graphe de G induit en conservant un *unique sommet* par clique critique de G.

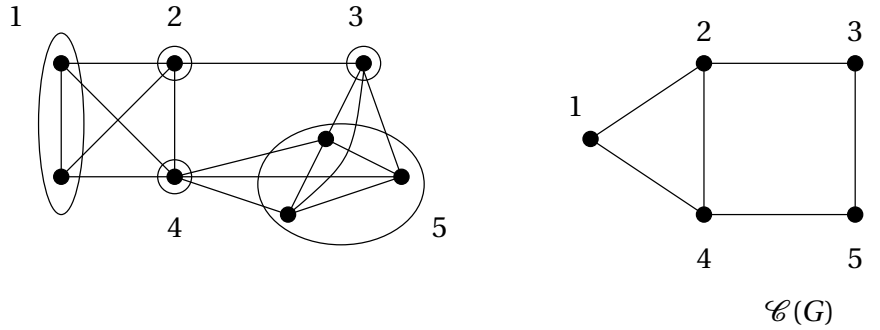


FIGURE 1.3 : Un graphe et son graphe des cliques critiques $\mathscr{C}(G)$.

Observation 1.7 ([136])**.** *Soit* $G := (V, E)$ *un graphe. Le* graphe des cliques critiques $\mathscr{C}(G)$ *peut être construit en temps* $O(n + m)$.

1.1.3 Edition de graphes

Graphes. Soit $\mathscr{G}$ une classe de graphes. Dans le cadre de cette thèse, nous nous intéressons principalement au problème suivant :

> $\mathscr{G}$-EDITION :
> **Entrée** : Un graphe $G := (V, E)$, et un entier k.
> **Paramètre** : k.
> **Question** : Existe-t-il un ensemble $F \subseteq (V \times V)$ de taille au plus k tel que le graphe $H := (V, E \triangle F)$ appartient à la classe $\mathscr{G}$?

édition

Soit (G, k), avec $G := (V, E)$ une instance du problème $\mathscr{G}$-EDITION [1]. Une *édition* (de G) est un ensemble $F \subseteq (V \times V)$ tel que le graphe $H := (V, E \triangle F)$ appartient à la classe $\mathscr{G}$. Nous disons que F est une *k-édition* lorsque $|F| \leqslant k$, et que F est *optimale* lorsque $|F|$ est minimum. Dans un abus de notation, nous utiliserons $G \triangle F$ pour représenter le graphe H. Nous définissons un sommet comme *affecté* lorsqu'il appartient à une paire de F.

k-édition

$G \triangle F$

sommet affecté

Nous définissons de manière similaire les problèmes $\mathscr{G}$-COMPLETION et $\mathscr{G}$-EDGE DELETION, devant respectivement vérifier $F \subseteq (V \times V) \setminus E$ et $F \subseteq E$. Dans le premier cas, il s'agit donc de trouver un ensemble F de *non-arêtes* de G tel que le graphe $H := (V, E \cup F)$ (aussi noté $G + F$) appartient à $\mathscr{G}$. Dans le second cas, le problème consiste à trouver un ensemble F d'*arêtes* de G tel que le graphe $H := (V, E \setminus F)$ (aussi noté $G - F$) appartient à $\mathscr{G}$. Nous considérons également la version de ce problème où seules des suppressions de sommets sont autorisées :

$G + F$

$G - F$

> $\mathscr{G}$-VERTEX DELETION :
> **Entrée** : Un graphe $G := (V, E)$, un entier k.
> **Paramètre** : k.
> **Question** : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ appartient à $\mathscr{G}$?

1.2 Autres structures : collection de relations

Collection de relations. Nous considérons également des problèmes d'édition pour des structures différentes des graphes non-orientés. De manière informelle, ces structures peuvent être considérées comme des collections $\mathscr{R}$ de *relations de taille $p \geqslant 2$*, définies sur un univers V. Nous étudions plus particulièrement des collections *denses, i.e.* contenant *exactement* une relation pour tout sous-ensemble de V de taille p. Nous préciserons pour chaque problème considéré en quoi consistent de telles relations.

Edition de relations. Nous nous intéressons principalement au problème suivant :

1. Dans la suite de cette thèse, nous dénotons de telles instances par (G, k) uniquement, le fait que le graphe G soit défini par $G := (V, E)$ étant sous-entendu.

Π-EDITION :
Entrée : Une collection dense $\mathscr{R}$ de relations de taille $p \geqslant 2$ définies sur un univers V, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $\mathscr{F} \subseteq \mathscr{R}$ de taille au plus k dont l'édition dans $\mathscr{R}$ permet d'obtenir une collection satisfaisant la propriété Π ?

Soit $R := (V, \mathscr{R})$ une instance du problème Π-EDITION. Une *édition* (de $\mathscr{R}$) est un ensemble $F \subseteq \mathscr{R}$ dont la modification dans $\mathscr{R}$ permet d'obtenir une collection satisfaisant la propriété Π. Nous disons que F est une k-édition lorsque $|F| \leqslant k$, et que F est optimale lorsque $|F|$ est minimum.

CHAPITRE

2 Complexité paramétrée et algorithmes de noyau

Dans ce chapitre, nous donnons les définitions permettant de poser les bases de la complexité paramétrée et des algorithmes de noyau.

Complexité paramétrée. Dans un premier temps, nous définissons les **problèmes** et **algorithmes FPT** (**Section 2.1**). Nous évoquons également la notion de W-**hiérarchie** (**Section 2.1.1**), introduite pour permettre de déterminer la difficulté d'un problème paramétré, et donnons deux exemples simples d'algorithmes FPT (**Section 2.1.2**). Nous introduisons en particulier la méthode d'**algorithme de branchement**, et décrirons d'autres techniques permettant d'obtenir des algorithmes FPT dans le **Chapitre 8**. Finalement, nous présentons différents paramètres possibles pour un même problème, en précisant l'intérêt de chacun de ces paramètres (**Section 2.1.3**).

Algorithmes de noyau. La principale partie de ce chapitre est consacrée à la notion d'**algorithme de noyau**. Nous donnons tout d'abord une définition formelle de ce principe, et énonçons ses principales conséquences en théorie de la complexité paramétrée (**Section 2.2.1**). Nous donnons par la suite des exemples de noyau pour les problèmes VERTEX COVER [2, 3, 149] et CLUSTER EDITING [37, 61, 85, 136]. Pour conclure, nous décrivons des techniques récentes permettant de prouver la **non-existence de noyaux polynomiaux** pour certains problèmes paramétrés (**Section 2.3**). Nous illustrons ces techniques sur les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION, travaux réalisés en collaboration avec Frédéric HAVET, Sylvain GUILLEMOT et Christophe PAUL.

Théorème 2.1. *Les problèmes C_l-*FREE EDGE DELETION *et P_l-*FREE EDGE DELETION *n'admettent pas de noyau polynomial pour tout $l \geqslant 7$.*

2.1 Complexité et algorithmes paramétrés

2.1.1 Définitions et hiérarchie de classes

Définition 2.2 (Problème paramétré)**.** *Un* problème paramétré *est un langage $L \subseteq \Sigma^* \times \mathbb{N}$, où Σ dénote un alphabet fini. Le second composant est appelé le* paramètre *du problème.*

Étant donné un problème paramétré L, nous définissons un couple $(I,k) \in \Sigma^* \times \mathbb{N}$ comme une *instance* de L. Nous disons que l'instance (I,k) est *positive* si et seulement si (I,k) *appartient* à L. instance

Définition 2.3 (algorithme FPT)**.** *Un problème paramétré L* admet un algorithme FPT *s'il peut être décidé en temps $f(k)\cdot|I|^{O(1)}$ si une instance (I,k) de L est positive, où f dénote une fonction calculable ne dépendant que de k.*

La classe des problèmes paramétrés admettant un algorithme FPT est appelée FPT (pour *fixed-parameter tractable*). Nous disons de manière similaire qu'un problème paramétré peut être résolu en temps FPT s'il admet un algorithme FPT. FPT

Remarques :

- Formellement, un tel algorithme est appelé *uniformément FPT*. En effet, certains problèmes admettent des algorithmes *non-uniformément* FPT, *i.e.* pour tout entier k fixé, il existe *un algorithme polynomial particulier* résolvant le problème. En d'autres termes, l'algorithme peut différer suivant la valeur du paramètre, ce qui n'est pas le cas des algorithmes uniformément FPT. Par souci de simplicité, nous parlerons uniquement d'algorithmes FPT, le fait qu'ils soient uniformes étant sous-entendu.
- Lorsque nous évoquons la complexité d'un algorithme FPT, nous utilisons la notation O^* , qui fait abstraction de la partie polynomiale de la complexité de l' algorithme considéré. Ainsi, $O^*(2^k)$ représente la complexité $2^k \cdot n^{O(1)}$. $O^*()$

La notion d'algorithmes FPT trouve son principal intérêt lorsque le paramètre associé au problème est *petit* comparé à la taille de l'instance. Ainsi, arriver à contenir l'explosion combinatoire à ce seul paramètre permet d'obtenir des algorithmes efficaces en pratique [87]. Dans de nombreuses applications, et notamment en ce qui concerne les problèmes d'édition de graphes, le paramètre correspondra donc au *nombre d'éditions autorisé*, qui sera petit en comparaison de la taille de l'instance. Dans le cadre de cette thèse, nous nous intéresserons principalement à ce paramètre. Théoriquement parlant, déterminer l'existence d'un algorithme FPT pour un problème paramétré donné est également important, puisqu'il est conjecturé que certains problèmes paramétrés n'admettent *pas* d'algorithme FPT.

Hiérarchie de classes. Nous introduisons brièvement la *W-hiérarchie* [56, 57], qui permet de hiérarchiser la difficulté des problèmes paramétrés. Le premier niveau (mis à part P et FPT) de difficulté en théorie de la complexité paramétrée est représenté par la classe $W[1]$. En particulier, un analogue au Théorème de Cook [40] a été démontré, énonçant que le problème SHORT TURING MACHINE ACCEPTANCE est $W[1]$-Difficile [129]. En d'autres termes, si ce problème est résoluble en temps $O(n^{k+1})$, il serait surprenant qu'il admette un algorithme FPT. Cela a donc donné lieu à la conjecture suivante, qui est l'analogue de $P \neq NP$ en théorie de la complexité classique. W-hiérarchie

Conjecture 3 ($FPT \neq W[1]$)**.** *Il existe un problème de la classe $W[1]$ qui n'appartient pas à FPT.*

Une hiérarchie se développe donc autour de ces classes, et est matérialisée comme suit :

$$FPT \subseteq W[1] \subseteq ... \subseteq W[t] \subseteq XP$$

où XP dénote les problèmes paramétrés (par k) pouvant être résolus en temps $O(n^k)$.

La définition suivante est similaire aux réductions évoquées dans l'introduction dans le cadre de la théorie de la NP-Complétude. Elle sert de base pour établir l'équivalence entre différents problèmes paramétrés, et ainsi établir une hiérarchie de classes pour les problèmes paramétrés.

Définition 2.4 (Réductions paramétrées)**.** *Soient L_1 et $L_2 \subseteq \Sigma^* \times \mathbb{N}$ deux problèmes paramétrés et (I_1, k_1) une instance de L_1. Il existe une* réduction paramétrée de L_1 vers L_2 *s'il existe deux fonctions f et g et un algorithme $\mathscr{A} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ tels que :*

(i) l'algorithme $\mathscr{A}$ s'exécute en temps $f(k_1) \cdot |I_1|^{O(1)}$ et,

(ii) $(I_1, k_1) \in L_1$ si et seulement si $\mathscr{A}(I_1, k_1) := (I_2, k_2) \in L_2$, où $k_2 = g(k_1)$.

Le lien avec les réductions permettant de prouver la NP-Complétude d'un problème est le suivant.

Observation 2.5 (Classique)**.** *Soient L_1 et L_2 deux problèmes paramétrés tels que L_1 admet une réduction paramétrée vers L_2. Si le problème L_2 admet un algorithme FPT, alors L_1 admet un algorithme FPT.*

2.1.2 Applications et exemples

Dans cette sous-partie, nous présentons deux algorithmes FPT par *arbre de recherche* [129] permettant de résoudre en temps FPT les problèmes VERTEX COVER [129] et $\mathscr{G}$-EDITION [31] lorsque $\mathscr{G}$ est caractérisée par une famille *finie* de sous-graphes induits interdits. Nous verrons que ce deuxième algorithme est en fait une généralisation de l'algorithme présenté pour VERTEX COVER. Les algorithmes par arbre de recherche représentent *une* des *nombreuses* techniques permettant d'obtenir des algorithmes FPT. Dans la mesure où ce chapitre est essentiellement consacré à la notion d'*algorithme de noyau*, nous présentons uniquement la méthode d'arbre de recherche comme exemple d'algorithme FPT. Nous décrirons le principe de plusieurs autres techniques dans le Chapitre 8.

Résoudre le problème VERTEX COVER en temps FPT

Nous donnons un premier exemple d'algorithme FPT pour le problème VERTEX COVER. A l'heure actuelle, l'algorithme le plus efficace permettant de résoudre ce problème a une complexité $O^*(1.27^k)$ [33, 131]. Nous présentons un algorithme de branchement permettant d'obtenir un algorithme de complexité $O^*(2^k)$.

VERTEX COVER :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un vertex cover de G de taille au plus k ?

Cet algorithme repose sur l'observation suivante, qui est clairement vraie.

Observation 2.6 (Classique)**.** *Soient* (G, k) *une instance de* VERTEX COVER *et* $uv \in E$ *une arête de* G*. Tout vertex cover de* G *doit contenir le sommet* u *ou le sommet* v*.*

Algorithm 1: Algorithme $VC(G, C, k)$ pour le problème VERTEX COVER

Entrée: Une instance $G := (V, E)$ de VERTEX COVER, C et un entier $k \in \mathbb{N}$.
Sortie : Un vertex cover de G de taille au plus k, s'il en existe un.

```
si (E = ∅) alors
    Un vertex cover de taille au plus k existe;
sinon
    Soit uv une arête de G;
    si (k ⩾ 0) et (|C| ⩽ k) alors
        si (C couvre E(G)) alors
            Un vertex cover de taille au plus k existe;
        sinon
            C ← C ∪ {u} ;
            VC(G \ {u}, C, k − 1);
            C ← C \ {u};
            C ← C ∪ {v} ;
            VC(G \ {v}, C, k − 1);
```

L'Algorithme 1 représente bien un algorithme FPT pour le problème VERTEX COVER : en effet, observons que chaque étape de l'algorithme *branche* en exactement deux sous-cas. De plus, à chaque branchement, le paramètre est *décrémenté* de 1. Il suit que l'arbre de recherche résultant de cet algorithme a une profondeur bornée par k, et qu'il possède ainsi 2^k feuilles. Par construction, ces dernières correspondent à des solutions potentielles. Comme il est possible de vérifier en temps polynomial si un ensemble de sommets donné couvre les arêtes d'un graphe (*i.e.* le problème VERTEX COVER est dans la classe NP), l'Algorithme 1 a une complexité $2^k \cdot n^{O(1)}$ et est donc bien un algorithme FPT pour le problème VERTEX COVER.

Remarque. Lors de l'exécution de l'Algorithme 1, deux cas de figure peuvent se produire : soit *Un vertex cover de taille au plus k existe* est retourné une (unique) fois, et ainsi la réponse au problème est *oui*. Dans le cas contraire, rien n'a été retourné, impliquant que la taille du vertex cover est supérieure à k, et ainsi que la réponse est *non*.

Résoudre le problème $\mathcal{G}$-EDITION en temps FPT lorsque $\mathcal{G}$ est héréditaire

Nous présentons maintenant une généralisation de cette idée, qui permet de concevoir un algorithme FPT pour le problème $\mathcal{G}$-EDITION (ainsi que $\mathcal{G}$-VERTEX DELETION) dès lors que la classe $\mathcal{G}$ est *héréditaire* et caractérisée par une famille *finie* de sous-graphes induits interdits. Nous présentons ce résultat dans le cas de la suppression de sommets, les idées étant les mêmes pour l'édition d'arêtes. Ce résultat, dû à Cai [31], est un résultat majeur en ce qui concerne la complexité paramétrée des problèmes d'édition de graphes. Le principe de cet algorithme est le suivant. Nous supposons dans la suite que h dénote le nombre maximum de sommets contenus dans un sous-graphe induit interdit de $\mathcal{G}$.

(i) détecter un sous-graphe induit interdit H de G en temps $O(n^h)$ (par recherche exhaustive) ;

(ii) brancher sur les (au plus) h possibilités de *détruire* l'obstruction H, en décrémentant k de 1 à chaque branchement :

(iii) répéter les étapes précédentes tant que $k \geqslant 0$ et G n'appartient pas à $\mathcal{G}$.

Comme précédemment, cet arbre de recherche a une hauteur bornée par k et possède donc au plus h^k feuilles. Cela permet donc d'obtenir un algorithme FPT ayant une complexité $O^*(h^k)$. Observons que décider si un graphe appartient à la classe $\mathcal{G}$ peut se faire en temps $O(n^h)$ par définition de $\mathcal{G}$. Remarquons également que cela permet de retrouver l'algorithme présenté pour VERTEX COVER, puisque ce problème peut se formuler de la manière suivante : existe-t-il un ensemble d'au plus k sommets dont la suppression permet d'obtenir un ensemble indépendant ? La classe des ensembles indépendants admet un seul-sous graphe induit interdit, composé de deux sommets reliés par une arête. Ainsi, cet algorithme a bien une complexité de $2^k \cdot n^{O(1)}$.

2.1.3 Quel paramètre choisir ?

L'étude des problèmes paramétrés repose en grande partie sur le choix du paramètre par rapport auquel la complexité va être mesurée. Dans cette section, nous présentons quelques paramètres possibles pour divers problèmes. Dans un premier temps, nous évoquons un paramètre *standard*, que nous considérons tout le long de cette thèse et qui consiste en la taille de la solution recherchée. Ensuite, nous décrivons d'autres paramètres possibles, et notamment des paramètres dits *non-standards*, permettant d'étudier la complexité paramétrée de problèmes difficiles pour le paramètre standard. De manière similaire, nous verrons qu'il est intéressant de *raffiner* un paramètre pour lequel le problème considéré est FPT. La liste de paramètres évoquée dans cette section n'est pas exhaustive, et nous renvoyons à [62] pour un aperçu plus général de cette *écologie des paramètres*.

Taille de la solution

Dans de nombreux problèmes, l'objectif est de déterminer s'il existe une solution de taille au plus k vérifiant certaines propriétés. A titre d'exemple, le problème du VERTEX COVER demande s'il existe un ensemble d'au plus k sommets *couvrant* les arêtes d'un graphe, alors que le problème DOMINATING SET consiste lui à déterminer l'existence d'un ensemble d'au plus k sommets *dominant* les sommets du graphe. Un paramètre *naturel* pour ces problèmes est donc la *taille de la solution*

recherchée. En ce sens, ce sont ces paramètres qui sont étudiés en premier lieu, et sont donc définis comme *standards*. Comme évoqué précédemment, le problème VERTEX COVER paramétré par la taille de la solution admet un algorithme FPT [33, 131], alors que le problème DOMINATING SET avec le même paramètre est $W[2]$-difficile [57]. Dans de tels cas, il est alors intéressant de *relaxer* le paramètre du problème, en considérant des paramètres dits *non-standards*.

Paramètres structurels non-standards

Nous évoquons maintenant des travaux récents qui étudient des problèmes ayant des paramètres structurels *non-standards* [18, 19, 62, 98]. L'idée est la suivante : supposons qu'un problème paramétré L soit W-difficile pour un certain paramètre k. Dans ce cas, il semble naturel de regarder la complexité paramétrée de ce même problème paramétré par un certain $k' \geqslant k$. Nous donnons un exemple concret avec le problème DOMINATING SET : soit (G, k) une instance de ce problème, où le paramètre k est la taille de l'ensemble dominant recherché. Ce problème est $W[2]$-difficile. Maintenant, considérons le problème suivant :

DOMINATING SET PARAMÉTRÉ PAR VERTEX COVER :
Entrée : Un graphe $G := (V, E)$, un vertex cover C de G et un entier l.
Paramètre : $|C|$.
Question : Existe-t-il un ensemble dominant de G de taille l ?

L'intérêt de l'étude de ce paramètre se justifie par le fait que tout vertex cover d'un graphe $G := (V, E)$ (sans sommets isolés -*i.e.* de degré 0-, ce qui peut être supposé sans perte de généralité) est en particulier un ensemble dominant, et qu'ainsi $|C| \geqslant |D|$ pour tout vertex cover C et tout ensemble dominant *minimum* D de G. Ce paramètre *relaxe* donc le paramètre standard qu'est la taille de la solution recherchée. Ce problème admet un algorithme FPT, mais n'admet cependant pas de noyau polynomial [55].

Supposons maintenant qu'un problème paramétré L soit FPT pour un certain paramètre k. Dans ce cas, il est naturel de se demander si le même problème paramétré par un certain paramètre $k' \leqslant k$ admet un algorithme FPT ou non. Nous prenons cette fois comme exemple le problème VERTEX COVER, qui est FPT lorsque paramétré par la taille de la solution, et considérons le problème suivant :

VERTEX COVER PARAMÉTRÉ PAR FEEDBACK VERTEX SET :
Entrée : Un graphe $G := (V, E)$, F un feedback vertex set de G, un entier l.
Paramètre : $|F|$.
Question : Existe-t-il un ensemble dominant dans G de taille l ?

Un *feedback vertex set* correspond à un ensemble de sommets de taille minimum dont la suppression permet d'obtenir une forêt. Encore une fois, nous avons $|C| \geqslant |F|$ pour tout vertex cover C et tout feedback vertex set *minimum* F d'un graphe $G := (V, E)$. Ainsi, ce paramètre permet de *raffiner* le paramètre *taille de la solution* pour lequel le problème admet un algorithme FPT, et

d'avoir une meilleure compréhension sur la difficulté dudit problème. Le problème VERTEX COVER PARAMÉTRÉ PAR FEEDBACK VERTEX SET admet également un algorithme FPT [97]. La principale différence réside dans l'existence (ou non) d'un noyau polynomial pour la version *valuée* de ces problèmes.

Largeurs de graphe

Il existe de nombreux paramètres permettant de mesurer la *largeur* d'un graphe [28, 132, 142, 143]. Dans le cadre de cette thèse, nous évoquons uniquement l'un des plus classiques, la *treewidth* [143] (que nous utiliserons Chapitre 8). Introduite par Robertson et Seymour dans leur série d'articles sur les mineurs de graphes (voir [142], par exemple), cette largeur arborescente a été par la suite largement étudiée et utilisée [42, 44, 123]. Un des théorèmes les plus importants à ce sujet est le **Théorème de Courcelle** [42], s'énonçant de manière simplifiée comme suit :

Théorème 2.7 (Courcelle [42])**.** *Soit* (G,ω) *une instance d'un problème L paramétré par la treewidth* ω *du graphe G. Si l'appartenance au langage L peut s'exprimer en Logique Monadique du Second Ordre, alors décider si* (G,ω) *appartient à L est FPT.*

Nous ne rentrons pas ici dans les détails de la logique monadique du second ordre, ni dans le fonctionnement ou la complexité de cet algorithme. Cependant, il est important de connaître l'existence d'un tel théorème, notamment car il a permis d'établir la complexité paramétrée de nombreux problèmes d'édition de graphes [78, 123] (comme par exemple CHORDAL DELETION, voir Chapitre 8).

Au-dessus d'une garantie

Dans de nombreux problèmes, il est possible de calculer en temps polynomial une *borne inférieure* à la valeur de toute solution optimale. Considérons par exemple le problème VERTEX COVER. Soit (G,k) une instance de ce problème paramétré par la taille de la solution recherchée k. De plus, soit M un couplage maximum de G : remarquons que M peut être calculé en temps polynomial [58]. Par définition de M, toute arête de M doit être incidente à (au moins) un sommet de tout vertex cover C. Il suit alors que $|C| \geqslant |M|$. De ce fait, il est possible de *raffiner* le paramètre du problème, en ne considérant non plus la taille du vertex cover recherché, mais une valeur *au-dessus de la garantie*. En d'autres termes, le nouveau problème paramétré se formule comme suit :

VERTEX COVER ABOVE GUARANTEE :
Entrée : Un graphe $G := (V,E)$, μ la valeur d'un couplage maximum de G, un entier k.
Paramètre : k.
Question : Existe-t-il un vertex cover de G de taille $\mu + k$?

Ce problème admet un algorithme FPT [127], qui est conçu en utlisant un algorithme FPT pour le problème ALMOST 2-SAT [140]. De nombreux travaux ont été réalisés sur les problèmes paramétrés au-dessus une certaine garantie, et notamment en ce qui concerne les problèmes de *satisfiabilité* [89, 90]. Le problème BETWEENNESS PARAMETERIZED ABOVE LOWER BOUND admet ainsi

un algorithme FPT [88], tout comme le problème MULTIWAY CUT PARAMETERIZED ABOVE LOWER BOUNDS [43].

2.2 Algorithmes de noyau

2.2.1 Définition et exemples

Nous décrivons maintenant l'une des techniques les plus puissantes permettant d'obtenir des algorithmes FPT [14], à savoir les *algorithmes de noyau*. Ces derniers sont des *réductions polynomiales* de l'instance (appelées *règles de réduction*), permettant d'obtenir une instance équivalente, tout en garantissant une borne sur sa taille, ne dépendant que du paramètre.

Définition 2.8 ((Algorithme de) noyau)**.** *Soit* (I,k) *une instance d'un problème paramétré* L*. Un* algorithme de noyau *pour* L *est un algorithme polynomial en* $|I|+k$ *retournant une instance* réduite (I',k') *de* L *vérifiant :*

$$(I,k) \in L \Leftrightarrow (I',k') \in L$$

$$et\ |I'| \leqslant f(k)\ et\ k' \leqslant k$$

pour une certaine fonction calculable f *ne dépendant que de* k*.* [1]

L'instance réduite est appelée *noyau* du problème paramétré L. Nous disons que L *admet un noyau de taille* $f(k)$, et parlons de *noyau polynomial* lorsque la fonction f est un polynôme. Nous disons également que L admet un algorithme de noyau de taille $f(k)$.

Remarque. Dans la suite de cette thèse, nous considérons des problèmes définis sur un ensemble de sommets (ou d'éléments) V. Ainsi, lorsque nous mentionnons la taille d'un noyau, nous sous-entendons sauf mention contraire que cette dernière est relative au *nombre de sommets (ou d'éléments)* contenus dans l'instance réduite.

Règles de réduction. L'algorithme de noyau est obtenu via l'application de *règles de réduction*, qui sont des algorithmes polynomiaux en $|I|+k$. Étant donné un ensemble de règles de réduction $\{R_1,\ldots,R_r\}$, nous disons qu'une instance (I,k) d'un problème paramétré L est *réduite par les règles* R_1 *à* R_r lorsqu'aucune des règles contenues dans $\{R_1,\ldots,R_r\}$ ne peut être appliquée à l'instance (I,k). Remarquons qu'il existe deux types de règles de réduction : les règles *dépendantes du paramètre*, qui utilisent le paramètre k, et les règles *indépendantes du paramètre*, qui sont valides quel que soit le paramètre considéré. Nous verrons que la majorité des règles de réduction considérées sont dépendantes du paramètre.

règles de réduction

instance réduite

Équivalence des solutions. Par définition même d'un algorithme de noyau, chaque application d'une règle de réduction R devra être démontrée *valide*, *i.e.* l'instance originale est positive si et seulement si l'instance réduite par R est positive. Plus formellement, nous obtenons la définition suivante.

1. Certaines définitions énoncent $k' \leqslant f(k)$. Cependant, dans tous les exemples considérés dans cette thèse, $k' \leqslant k$ sera vérifié.

validité **Définition 2.9** (Validité)**.** *Soient* (I,k) *une instance d'un problème paramétré* L *et* R *une règle de réduction pour le problème* L*. La règle* R *est* valide *si l'instance* $(I',k') := R((I,k))$ *obtenue en appliquant la règle* R *sur* (I,k) *vérifie :*

$$(I,k) \in L \Leftrightarrow (I',k') \in L$$

Remarque. Nous verrons par la suite que, dans certains cas, il est possible de *décider* le problème considéré en temps polynomial. Pour être conforme à la notion d'algorithme de noyau, il faudrait donc retourner une instance équivalente et de taille bornée en fonction de k au lieu de *décider* le problème. Cela peut se faire de la manière suivante : s'il est possible de décider l'instance en temps polynomial, l'algorithme de noyau retourne une instance positive ou négative (suivant la décision), triviale et de taille constante.

Les problèmes paramétrés étant des problèmes de *décision*, nous ne cherchons pas à construire *toutes* les solutions du problème paramétré considéré, mais à *décider* si une solution existe. Cela nous permet en particulier de ne considérer que *certains types* de solutions afin de concevoir nos règles de réductions. Remarquons que la notion de *noyaux préservant l'ensemble des solutions minimales* a été étudiée par Damaschke [45] sous le nom *full kernel*. Cependant, nous ne considérons pas ce concept dans le cadre de cette thèse.

Taille de l'instance réduite. Afin de borner la taille des noyaux, nous démontrons que toute instance *positive* réduite par l'ensemble des règles de réduction considéré a une taille bornée par un polynôme p ne dépendant que de k. Cela constitue un des cas de décision énoncé dans la remarque précédente. En effet, il suivra de ce fait que toute instance de L réduite par les règles de réduction et possédant plus de $p(k)$ sommets constituera une instance *négative*. Comme les règles de réduction s'exécutent en temps polynomial, nous déciderons dans ce cas le problème en temps polynomial, et retournerons donc une instance négative triviale et de taille constante.

Importance théorique et pratique

L'un des principaux intérêts des algorithmes de noyau est que leur existence, combinée à *n'importe quel* algorithme résolvant le problème de manière exacte, permet d'obtenir un algorithme FPT pour le problème considéré. En fait, ces deux propriétés sont équivalentes, comme énoncé par le résultat suivant.

Théorème 2.10 (Classique)**.** *Un problème paramétré (décidable)* L *admet un algorithme FPT si et seulement si* L *admet un algorithme de noyau.*

Preuve. $\Leftarrow$ Soit (I,k) une instance du problème paramétré L. En utilisant l'algorithme de noyau de L, nous calculons l'instance réduite (I',k') vérifiant $|I'| \leqslant f(k)$ pour une certaine fonction calculable f, en temps $|I|^{O(1)}$. Ainsi, appliquer n'importe quel algorithme permettant de résoudre le problème avec une complexité $h(|I'|)$ permet d'obtenir un algorithme FPT pour L de complexité $O(n^c + h(f(k)))$.

$\Rightarrow$ Inversement, supposons que L admette un algorithme FPT ayant une complexité de la forme $f(k) \cdot O(|I|^c)$ pour une constante $c > 0$. Tout d'abord, si $|I| > f(k)$, alors cet algorithme s'exécute en temps $O(|I|^{c+1})$, ce qui est polynomial en la taille de I. Il est donc possible de décider le problème

L en temps polynomial, et l'algorithme de noyau consiste simplement à retourner une instance positive ou négative triviale et de taille constante. Dans le cas contraire, $|I| < f(k)$ et donc l'instance (I,k) est elle-même un noyau pour le problème L. □

Remarque. Les algorithmes de noyau obtenus via le Théorème 2.10 permettent uniquement de construire des noyaux de taille $f(k)$, où f dénote la complexité de l'algorithme FPT du problème considéré. Ainsi, lorsque f est super-polynomiale en k (ce qui est en particulier le cas lorsque le problème considéré est NP-Difficile), le Théorème 2.10 permet uniquement d'obtenir des noyaux de taille super-polynomiale en k.

Dans certains cas (comme par exemple pour le problème CLIQUE COVER [80]), le seul algorithme FPT connu dépend d'un algorithme de noyau exponentiel. De plus, utiliser des algorithmes de noyau polynomiaux à chaque étape d'un algorithme FPT via arbre de recherche peut également permettre d'améliorer la complexité d'un algorithme FPT existant. Cette technique, introduite par Niedermeier et Rossmanith [130], est connue sous le nom d'*interleaving*. Pour certains problèmes, la combinaison d'un algorithme de noyau polynomial et d'un algorithme exponentiel exact peut également permettre d'améliorer la complexité connue. Par exemple, le problème MAXIMUM INTERNAL SPANNING TREE admet un noyau contenant au plus $3k$ sommets [65] et un algorithme exponentiel exact ayant une complexité $O^*(2^n)$ [128], ce qui permet d'obtenir un algorithme FPT ayant une complexité $O^*(8^k)$ [65], améliorant l'algorithme en $O(49.4^k)$ [39].

Algorithmes de noyau pour le problème VERTEX COVER

Nous présentons maintenant plusieurs algorithmes de noyau pour le problème VERTEX COVER, permettant notamment d'obtenir un noyau contenant au plus $4k$ sommets [119]. Les algorithmes présentés dans cette partie ne sont pas optimaux, puisque le problème du VERTEX COVER admet un noyau avec au plus $2k-c$ sommets pour toute constante $c > 0$ [149]. Cependant, ces algorithmes donnent une vision d'ensemble de la dichotomie noyau quadratique/linéaire (en nombre de sommets) existant dans la littérature.

Nous présentons dans un premier temps des règles de réduction simples pour le problème, permettant d'obtenir un noyau avec $O(k^2)$ sommets et arêtes. Nous verrons dans la suite de cette thèse (Partie I) que les règles de réduction utilisées pour cet algorithme de noyau sont des règles de réduction *génériques* qui peuvent être utilisées dans de nombreux problèmes. Dans la suite de cette thèse, nous utiliserons ainsi de manière extensive les règles dites de *sommet isolé* et de *sunflower*.

Règle 2.1 (Sommet isolé)**.** *Soient* (G,k) *une instance de* VERTEX COVER *et* $u \in V$ *un sommet de degré* 0. *Supprimer* u *de* G.

Lemme 2.11. *La Règle 2.1 est valide et peut être appliquée en temps linéaire*[2].

Preuve. Soit $G' := G[V \setminus \{u\}]$ l'instance réduite par la Règle 2.1. Pour démontrer la validité de cette règle de réduction, nous devons prouver que G a une couverture par sommets de taille au plus k si et seulement si G' a une couverture par sommets de taille au plus k. Premièrement, si G admet

2. Dans la suite de cette thèse, nous disons qu'une règle de réduction peut être appliquée en temps polynomial s'il est possible de trouver une structure sur laquelle appliquer cette règle en temps polynomial.

une couverture par sommets C de taille au plus k, il est clair que l'ensemble C est une couverture par sommets de taille au plus k de G'. Inversement, si G' admet une couverture par sommets C de taille au plus k, alors C est également une couverture par sommets de G puisque le sommet u n'est incident à aucune arête de G. Pour conclure, remarquons qu'un tel sommet u peut être détecté en temps linéaire dans G. □

Nous énonçons maintenant une nouvelle règle de réduction indépendante du paramètre. Nous verrons dans le Théorème 2.15 que cette règle n'est pas nécessaire pour obtenir un noyau quadratique pour VERTEX COVER. Cependant, cette dernière est à l'origine du noyau linéaire pour le problème, qui sera décrit plus tard dans cette section.

Règle 2.2 (Degré 1)**.** *Soient* (G,k) *une instance de* VERTEX COVER *et* $u \in V$ *un sommet de* G *de degré* 1. *Supprimer* u *et* $N_G(u)$ *de* G*, ajouter* $N_G(u)$ *à la solution et décrémenter* k *de* 1.

Lemme 2.12. *La Règle 2.2 est valide et peut être appliquée en temps linéaire.*

Preuve. Soient $v := N_G(u)$ et $G' := G[V \setminus \{u,v\}]$ l'instance réduite (de paramètre $k-1$). Nous démontrons la validité de cette règle de réduction via l'affirmation suivante.

Affirmation 2.13. *Il existe toujours une couverture par sommets optimale de* G *contenant* v *et ne contenant pas* u.

Preuve. Soit C une couverture par sommets optimale de G. Si C contient v, alors C ne contient pas u : en effet, dans le cas contraire, $C \setminus \{u\}$ est une couverture par sommets, contredisant l'optimalité de C. Inversement, par l'Observation 2.6, C doit contenir u. Mais alors, l'ensemble $C' := (C \setminus \{u\}) \cup \{v\}$ est également une couverture par sommets de G, puisque le sommet v couvre l'arête uv (ainsi qu'éventuellement d'autres arêtes), qui est la seule arête de G incidente au sommet u. ◇

Cela justifie le fait d'ajouter v à la solution lors de la réduction de l'instance. Maintenant, si G' admet une couveture par sommets C' de taille $k-1$, alors l'ensemble $C := C' \cup \{v\}$ est une couverture par sommets de G de taille k. Inversement, si C est une couverture par sommets de G de taille k, alors, par l'Affirmation 2.13, nous pouvons supposer que C contient v, et ainsi $C' := C \setminus \{v\}$ est une couverture par sommets de G' de taille $k-1$. □

Règle 2.3 (Sunflower)**.** *Soient* (G,k) *une instance de* VERTEX COVER *et* u *un sommet de* G *de degré supérieur à* k*. Supprimer* u *de* G*, ajouter* u *à la solution et décrémenter* k *de* 1.

Lemme 2.14. *La Règle 2.3 est valide et peut être appliquée en temps* $O(m)$.

Preuve. Soient $u \in V$ un sommet de degré supérieur à k dans G et $G' := G[V \setminus \{u\}]$ l'instance réduite (de paramètre $k-1$). Supposons dans un premier temps que G' admette une couverture par sommets C de taille $k-1$: dans ce cas, l'ensemble $C \cup \{u\}$ est une couverture par sommets de G de taille k. Inversement, soit C une couverture par sommets de G de taille au plus k. Nous affirmons que C contient obligatoirement le sommet u. Par contradiction, supposons que ce ne soit pas le cas : il suit que C contient $N_G(u)$ afin de couvrir l'ensemble des arêtes incidentes à u. Comme $d(u) > k$, cela implique que $|C| > k$: contradiction. Ainsi l'ensemble $C \setminus \{u\}$ est une couverture par sommets de G' de taille $k-1$. De plus, il est clair qu'un tel sommet u peut être obtenu en temps $O(m)$. □

Théorème 2.15. *Le problème* VERTEX COVER *admet un noyau avec au plus $k^2 + k$ sommets et $O(k^2)$ arêtes.*

Preuve. Soient (G, k) une instance positive de VERTEX COVER réduite par les Règles 2.1 et 2.3, et C une couverture par sommets de G de taille k. Par définition, le graphe $G \setminus C$ est un ensemble indépendant. Puisque G est réduit par la Règle 2.3, chacun des k sommets de C possède au plus k voisins dans $G \setminus C$. Il suit que $|V| \leqslant k^2 + k$. Comme $G \setminus C$ est un ensemble indépendant, et vu que $G[C]$ contient au plus $\frac{k\cdot(k-1)}{2}$ arêtes, il suit que G contient au plus $O(k^2)$ arêtes. □

Remarque. Les règles de réduction de type *sommet isolé* et *sunflower*, qui traitent l'instance de manière *locale*, permettent souvent d'obtenir des noyaux polynomiaux pour des problèmes d'édition [5, 82]. Cependant, afin d'obtenir un noyau *linéaire* (en nombre de sommets), il est souvent nécessaire d'avoir recours à des règles de réduction plus *globales*, qui requièrent donc une analyse supplémentaire. Nous allons montrer comment obtenir de telles règles pour le problème VERTEX COVER, et exploiterons à nouveau cette idée dans le Chapitre 7.

Décomposition en couronne. Nous décrivons maintenant des règles de réduction dont l'application permet d'obtenir un noyau contenant au plus $4k$ sommets pour VERTEX COVER [119]. Ces règles de réduction sont basées sur le principe de *(décomposition en) couronne* d'un graphe. Ces dernières consistent en une *généralisation* de la Règle 2.2.

Définition 2.16 (Couronne)**.** *Soit $G := (V, E)$ un graphe. Une* couronne *de G est une couple (I, H) telle que $I \subseteq V$, $H \subseteq V$ et satisfaisant les propriétés suivantes :*

(i) $I \neq \emptyset$ *est un ensemble indépendant de G,*

(ii) $H = N_G(I)$ *et,*

(iii) il existe un couplage M dans $G[I \cup H]$ saturant les sommets de H.

L'ensemble H est appelé la *tête* de la couronne, et nous définissons la *taille* d'une couronne par $|H|$.

Règle 2.4. *Soient $G := (V, E)$ un graphe et (I, H) une décomposition en couronne de G. Supprimer $I \cup H$ de G, ajouter les sommets de H à la solution et décrémenter k de $|H|$.*

Lemme 2.17 ([119])**.** *La Règle 2.4 est valide.*

Preuve. Nous prouvons que le graphe G a une couverture par sommets de taille au plus k si et seulement si le graphe $G' := G[V \setminus (I \cup H)]$ a une couverture par sommets de taille au plus $k' := k - |H|$.

Affirmation 2.18. *Il existe toujours une couverture par sommets C_H de C contenant les sommets de H.*

Preuve. Pour voir cela, rappelons qu'il existe un couplage entre I et H saturant H dans G. Il suit que $|C \cap (I \cup H)| \geqslant |H|$, puisque toute couverture par sommets doit contenir au moins un sommet par arête du couplage. De plus, comme I est un ensemble indépendant et $H = N_G(I)$, l'ensemble

$C_H := C \setminus I \cup H$ est une couverture par sommets de G de taille au plus k. ⋄

Supposons maintenant que le graphe G' ait une couverture par sommets C' de taille k'. Dans ce cas, il est clair que l'ensemble $C' \cup H$ est une couverture par sommets de G (puisque $H = N_G(i)$ par définition) de taille au plus $k' + |H| = k$. Inversement, supposons que G admette une couverture par sommets C de taille au plus k. Par construction, il suit que $C[V \setminus (I \cup H)]$ est une couverture par sommets de $G[V \setminus (I \cup H)]$ de taille au plus k'. □

Pour obtenir un algorithme de noyau utilisant la Règle 2.4, il reste maintenant à prouver qu'une couronne d'un graphe peut être obtenue en temps polynomial. Pour ce faire, nous utilisons le résultat suivant, énonçant qu'il est possible de trouver une couronne dans un graphe sous certaines conditions de cardinalité, ce qui sera suffisant pour l'algorithme de noyau.

Lemme 2.19 ([38])**.** *Soient $G := (V, E)$ un graphe et $I \subseteq V$ un ensemble indépendant de G tel que $|I| > |N(I)|$. Une* décomposition en couronne *(I', H) de G telle que $I' \subseteq I$ peut être obtenue en temps $O(n+m)$.*

Nous allons voir que l'énoncé de ce résultat provient de la preuve permettant de borner le nombre de sommets contenus dans une instance positive du problème VERTEX COVER.

Théorème 2.20. *Le problème* VERTEX COVER *admet un noyau contenant au plus $4k$ sommets.*

Preuve. Soit (G, k) une instance positive de VERTEX COVER. Dans un premier temps, nous calculons en temps polynomial un couplage maximum M de G. Soit $V(M)$ l'ensemble des sommets correspondant aux extrémités des arêtes de M. Puisque toute couverture par sommets de G doit contenir au moins un sommet par arête de M, il suit que $|M| \leqslant k$ et donc $|V(M)| \leqslant 2k$. Supposons maintenant que $|V \setminus V(M)| > 2k$. Nous obtenons ainsi un ensemble indépendant $I := V \setminus V(M)$ vérifiant $|I| > |V(M)| \geqslant |N(I)|$. Le Lemme 2.19 peut donc s'appliquer pour trouver une décomposition en couronne de G, et ainsi nous pouvons réduire le graphe G en utilisant la Règle 2.4. Nous répétons les étapes précédentes sur l'instance réduite, jusqu'à ce que $k < 0$ ou $|V| \leqslant 4k$. Dans le premier cas, nous décidons le problème en temps polynomial, et retournons donc une instance négative triviale de taille constante. Dans le second cas, l'instance réduite possède bien au plus $4k$ sommets, ce qui conclut la preuve. □

2.3 Techniques pour prouver la non-existence de noyaux polynomiaux

Pour conclure cette introduction sur les algorithmes de noyau, nous présentons des résultats récents permettant d'établir la *non-existence de noyaux polynomiaux* pour des problèmes paramétrés. Ces résultats sont basés sur certaines hypothèses de théorie de la complexité, notamment $NP \not\subseteq coNP/poly$ [16, 70]. Rappelons que, par le Théorème 2.10, l'existence d'un algorithme de noyau est équivalente à l'existence d'un algorithme FPT pour un problème paramétré L. Cependant, lorsque le problème L est NP-Difficile, ce résultat ne permet d'obtenir que des noyaux de taille *super-polynomiale* en k. Une question naturelle découlant de cette observation peut donc se formuler comme suit.

Problème 1. *Existe-t-il des problèmes paramétrés n'admettant pas de noyaux polynomiaux ?*

En 2008, une réponse positive a été apportée à cette question par plusieurs équipes de recherches [16, 70], qui ont développé des méthodes similaires sous différents formalismes. Nous présentons ici principalement les travaux de Bodlaender et al. [16, 20], qui ont introduit la notion d'*algorithmes de composition* et de *transformation polynomiale en temps et paramètre* pour répondre à ce problème.

2.3.1 Compositions

Distillation et Composition. Nous présentons dans un premier temps la notion d'*algorithme de distillation* pour un problème classique L. Nous verrons plus tard que cette notion peut s'adapter pour les problèmes paramétrés (*algorithmes de composition*).

Définition 2.21 (Ou-distillation [16])**.** *Un algorithme de* ou-distillation *pour un problème L est un algorithme $\mathscr{A}$ recevant en entrée une séquence d'instances $I_1, \ldots, I_t$ de L, s'exécutant en temps polynomial en* $\max_{i\in[t]} |I_i| + t$ *et retournant une instance D de L telle que :*

$$D := \mathscr{A}(I_1, \ldots, I_t) \in L \Leftrightarrow \exists\, i \in [t],\ (I_i, k) \in L$$

Conjecture 4 (Ou-distillation [16])**.** *Il n'existe pas de problème NP-Complet admettant un algorithme de distillation.*

Théorème 2.22 ([70])**.** *Si la conjecture Ou-distillation n'est pas vérifiée, alors $NP \subseteq coNP/poly$.*

Remarque. La notion de *Et-distillation* peut être définie de la même manière que les algorithmes de Ou-distillation. La principale différence réside dans le fait qu'un algorithme de Et-distillation retourne une instance positive si et seulement si toutes les instances qu'il reçoit en entrée sont positives. La Conjecture de Et-Distillation est également similaire à la Conjecture 4. Cependant, il n'existe pas à l'heure actuelle d'équivalent au Théorème 2.22. De ce fait, cette hypothèse de travail est plus *faible* que la précédente, et nous ne considérons pas de tels algorithmes dans la suite de cette thèse.

Nous donnons maintenant la définition de *Ou-composition* pour un problème paramétré, qui a été dérivée de la définition de Ou-distillation pour un problème classique.

Définition 2.23 (Ou-composition [16])**.** *Un algorithme de* ou-composition *pour un problème paramétré L est un algorithme recevant en entrée une séquence d'instances $(I_1, k), \ldots, (I_t, k)$ de L pour $i \in [t]$, s'exécutant en temps polynomial en $\Sigma_{i=1}^{t} |I_i| + k$ et retournant une instance (C, k') de L telle que :* ou-composition

(i) $(C, k') \in L \Leftrightarrow (I_i, k) \in L$ *pour un certain $i \in [t]$ et,*

(ii) k' *est polynomial en k.*

Nous disons qu'un problème paramétré admettant un tel algorithme est *ou-composable*. De plus, étant donné un problème paramétré $L \subseteq \Sigma^* \times \mathbb{N}$, sa *version non-paramétrée* L^c est définie comme $\{I\#1^k : (I, k)$ est une instance de $L\}$, où $\#$ est un symbole n'appartenant pas à Σ.

Théorème 2.24 ([16])**.** *Soit L un problème paramétré ou-composable dont la version non-paramétrée L^c est NP-Complète. Le problème L n'admet pas de noyau polynomial, sauf si la Conjecture 4 n'est pas vérifiée, et donc si $NP \subseteq coNP/poly$.*

Remarque. Comme pour les algorithmes de Ou-distillation, il est possible de définir de manière similaire des algorithmes de Et-composition. Un analogue au Théorème 2.24 peut alors être établi, énonçant que la Conjecture Et-distillation n'est pas vérifiée si un problème paramétré admet un algorithme de Et-composition. Cependant, comme nous l'avons évoqué précédemment, cette hypothèse de travail est plus faible que la Conjecture 4, et nous ne considérons pas de tels algorithmes dans la suite de cette thèse.

Exemple. Nous donnons un exemple de construction d'un algorithme de composition pour le problème paramétré suivant :

k-PATH : **Entrée** : Un graphe $G := (V, E)$, un entier k. **Paramètre** : k. **Question** : Existe-t-il un chemin de longueur k dans G ?

La complexité paramétrée de ce problème a été établie par Alon et al. [6], en introduisant la technique COLOR CODING, qui a depuis été réutilisée à de nombreuses reprises [5]. Cependant, l'existence d'un noyau polynomial est restée ouverte de nombreuses années, avant qu'un algorithme de ou-composition ne soit construit par Bodlaender et al. [16].

Ou-Composition. Considérons une séquence $(G_1, k), \ldots, (G_t, k)$ d'instances de k-PATH. L'algorithme de Ou-composition retourne simplement l'instance (G', k), où $G' := \oplus_{i=1}^{t} G_i$ est obtenu en prenant l'union disjointe des graphes G_i, $i \in [t]$. Il est clair que cet algorithme s'exécute en temps polynomial en $\Sigma_{i=1}^{t} |G_i| + k$ et que k' est borné par un polynôme en k. Finalement, l'existence d'un chemin de longueur k dans un graphe quelconque étant équivalente à l'existence d'un tel chemin dans *une* de ses composantes connexes, il suit que l'instance (G', k) est positive si et seulement si il existe $i \in [t]$ tel que (G_i, k) est une instance positive. Maintenant, remarquons que la version non paramétrée du problème k-PATH est NP-Complète : en effet, ce problème contient le problème HAMILTONIAN PATH, qui est un problème NP-Complet classique [72]. Le Théorème 2.24 implique alors que le problème k-PATH n'admet pas de noyau polynomial, sauf si $NP \subseteq coNP/poly$.

Intuition. Nous donnons maintenant une idée permettant de justifier le fait que le problème k-PATH n'admette pas de noyau polynomial [14]. Supposons qu'il existe un algorithme de noyau pour le problème k-PATH, retournant une instance contenant au plus k^c sommets pour une certaine constante $c > 0$. Considérons maintenant un graphe $G := (V, E)$ contenant k^{2^c} composantes connexes. De manière évidente, il existe un chemin de longueur k dans G si et seulement si au moins une composante connexe de G contient un chemin de longueur k. Ainsi, puisque le nombre de composantes connexes du graphe est largement supérieur à la taille du noyau, il semble que certaines composantes connexes doivent être *résolues en temps polynomial* pour pouvoir obtenir une telle réduction. Comme le problème k-PATH est NP-Complet, il suit que cela ne peut pas être fait, sauf si $P = NP$.

2.3.2 Transformations polynomiales en temps et paramètre

Nous présentons maintenant une méthode introduite par Bodlaender et al. [20] permettant de *porter* des résultats de non-existence d'un noyau polynomial pour un certain problème paramétré L vers un *autre* problème paramétré L'. Ces réductions sont semblables aux réductions de NP-Complétude présentées dans l'introduction. Cependant, certaines hypothèses supplémentaires doivent être vérifiées afin de s'assurer que l'existence d'un noyau *polynomial* pour L' implique l'existence d'un noyau *polynomial* pour L, ce qui permettra d'établir les résultats de bornes inférieures mentionnés précédemment.

Définition 2.25 (Transformation polynomiale en temps et paramètre [20])**.** *Soient L et L' deux problèmes paramétrés. Une* transformation polynomiale en temps et paramètre *de L vers L', notée $L \leqslant_{tptp} L'$, est un algorithme polynomial prenant en entrée une instance (I,k) de L et retournant une instance (I',k') de L' telle que :* $\leqslant_{tptp}$

(i) $(I,k) \in L \Leftrightarrow (I',k') \in L'$ *et,*

(ii) $k' \leqslant p(k)$ *pour un certain polynôme* p*.*

Théorème 2.26 ([20])**.** *Soient L et L' deux problèmes paramétrés et L^c et L'^c leurs versions non-paramétrées. Supposons que L^c soit NP-Complet et que L'^c appartienne à NP. Si $L \leqslant_{tptp} L'$, et si L' admet un noyau polynomial, alors L admet également un noyau polynomial.*

En utilisant la contraposée de ce résultat, nous obtenons le corollaire suivant, largement utilisée dans l'obtention de bornes inférieures pour des problèmes paramétrés [20, 55, 83, 111]. Nous verrons un exemple d'utilisation de ce résultat dans la Section 2.3.5.

Corollaire 2.27 ([20])**.** *Soient L et L' deux problèmes paramétrés et L^c et L'^c leurs versions non-paramétrées. Supposons que L^c soit NP-Complet et ou-composable, et que L'^c appartienne à NP. Si $L \leqslant_{tptp} L'$, alors L' n'admet pas de noyau polynomial, sauf si $NP \subseteq coNP/poly$.*

Remarque. À nouveau, un résultat similaire peut être établi pour les problèmes admettant des algorithmes de Et-composition.

Le principal intérêt de cette méthode est qu'elle permet de se concentrer sur un *problème-jouet*, pour lequel il sera possible de concevoir un algorithme de composition ainsi qu'une transformation polynomiale en temps et paramètre vers le problème original. Nous donnons un exemple d'utilisation de ces transformations pour le problème DISJOINT CYCLES, défini comme suit :

DISJOINT CYCLES :
Entrée : Un graphe $G := (V,E)$, un entier k.
Paramètre : k.
Question : Existe-t-il k cycles sommets-disjoints dans G ?

Nous considérons le *problème-jouet* DISJOINT FACTORS. Soient Σ_k l'alphabet composé des lettres $\{1,2,\ldots,k\}$, et Σ_k^* l'ensemble des mots définis sur Σ_k. Un *facteur* d'un mot $M := m_1 \ldots m_r \in \Sigma_k^*$ est un sous-mot (*i.e.* consécutif dans M) $m_i \ldots m_j \in \Sigma_k^*$, avec $1 \leqslant i < j \leqslant r$, commençant et terminant par la même lettre. Un mot $M \in \Sigma_k^*$ a la *propriété des facteurs disjoints* s'il existe k facteurs

disjoints $F_1, \ldots, F_k$ de M tels que le facteur F_i commence et termine par la lettre i. Formellement, le problème DISJOINT FACTORS se définit comme suit :

DISJOINT FACTORS :
Entrée : Un mot $M \in \Sigma_k^*$ de longueur n, un entier k.
Paramètre : k.
Question : Existe-t-il k facteurs disjoints dans m ?

Proposition 2.28 ([20])**.** *Le problème* DISJOINT FACTORS *est NP-Complet et peut être résolu en temps* $O(nk \cdot 2^k)$.

Observons que le problème DISJOINT FACTORS n'admet pas d'algorithme de composition trivial (comme l'était celui pour le problème k-PATH). Néanmoins, il est possible de concevoir un algorithme de Ou-composition pour ce problème.

Proposition 2.29 ([20])**.** *Le problème* DISJOINT FACTORS *est ou-composable.*

Pour conclure sur le problème original DISJOINT CYCLES, il reste donc à démontrer que DISJOINT FACTORS $\leqslant_{tptp}$ DISJOINT CYCLES, ce qui est effectivement le cas.

Théorème 2.30 ([20])**.** *Le problème* DISJOINT CYCLES *n'admet pas de noyau polynomial, sauf si* $NP \subseteq coNP/poly$.

2.3.3 Couleurs et identifiants

En utilisant les deux méthodes précédemment introduites, Dom et al. [55] ont développé le concept de *couleurs et d'identifiants* pour des problèmes paramétrés, qui permet d'obtenir des bornes inférieures pour l'existence de noyaux polynomiaux. Nous donnons dans la suite un *schéma général* de leur méthode.

(i) Définir une version colorée (paramétrée) adéquate du problème. Cela permet d'obtenir plus de contrôle sur la structure des solutions.

(ii) Démontrer que la version non paramétrée du problème considéré est dans NP, et également que la version colorée (non paramétrée) est NP-Difficile.

(iii) Décrire une transformation polynomiale en temps et paramètre de la version colorée vers la version non colorée du problème considéré. Cela permettra de répercuter les bornes inférieures obtenues sur la version colorée vers la version non colorée (Corollaire 2.27).

(iv) Prouver qu'une instance (I, k) du problème coloré peut être résolue en temps $2^{k^c} \cdot |I|^{O(1)}$ pour une certaine constante $c > 0$.

(v) Pour conclure, démontrer que la version colorée est ou-composable. Pour ce faire :

  (a) si le nombre d'instances dans l'entrée est supérieur à 2^{k^c} alors il est possible de conclure.

  (b) sinon, chaque instance reçoit un identifiant unique, ce qui requiert $log(2^{k^c}) = k^c$ bits par instance.

  (c) Utiliser les informations de codage fournies par les couleurs et les identifiants pour obtenir un algorithme de composition pour la version colorée du problème.

En particulier, Dom et al. [55] ont démontré que le problème suivant n'admet pas de noyau polynomial, et ont ensuite proposé des transformations polynomiales en temps et paramètre depuis ce problème pour obtenir différents résultats (Théorème 2.31).

RED-BLUE DOMINATING SET :
Entrée : Un graphe biparti $G := (T \cup N, E)$, un entier k.
Paramètre : $|T|$, k.
Question : Existe-t-il $N' \subseteq N$ de taille au plus k tel que tout sommet de T ait *au moins* un voisin dans N' ?
Résultat : Pas de noyau en $O(k^{O(1)})$.
Version colorée : N est coloré avec des couleurs dans $\{1,\ldots,k\}$ et N' doit contenir un sommet de chaque couleur.

Théorème 2.31 ([55]). *Les problèmes* RED-BLUE DOMINATING SET *et* STEINER TREE *paramétrés par* $(|T|,k)$, *les problèmes* CONNECTED VERTEX COVER *et* CAPACITATED VERTEX COVER *paramétrés par la taille de la solution k, et le problème* SET COVER *paramétré par la taille de la solution k et la taille maximale des ensembles n'admettent pas de noyau polynomial, sauf si* $NP \subseteq coNP/poly$.

2.3.4 Cross-composition

Pour conclure cette section, nous évoquons une technique introduite récemment par Bodlaender et al. [18] et intitulée *cross-composition*. Cette dernière, basée sur la notion de *relation d'équivalence polynomiale* (Définition 2.32), permet de prouver la non-existence de noyaux polynomiaux en proposant des réductions depuis des problèmes NP-Difficiles, et non plus nécessairement paramétrés. En particulier, cela permet d'autoriser une plus grande croissance du paramètre pour l'instance réduite.

Définition 2.32 (Relation d'équivalence polynomiale [18]). *Une relation d'équivalence* $\mathcal{R}$ *sur* Σ^* *est une* relation d'équivalence polynomiale *si :*

(i) il existe un algorithme décidant si deux instances $I, I' \in \Sigma^*$ *appartiennent à la même classe d'équivalence en temps* $(|I|+|I'|)^{O(1)}$ *et,*

(ii) pour tout ensemble d'instances $\mathcal{I}$, *la relation d'équivalence* $\mathcal{R}$ *partitionne les éléments de* $\mathcal{I}$ *en au plus* $(\max_{I\in\mathcal{I}}|I|)^{O(1)}$ *classes d'équivalence.*

cross-composition

Définition 2.33 (Cross-composition [18]). *Soient* $L \subseteq \Sigma^*$ *un problème et* $L' \in \Sigma^* \times \mathbb{N}$ *un problème paramétré. Le problème L se* cross-compose *vers* L' *s'il existe une relation d'équivalence polynomiale et un algorithme qui, étant données t instances* $I_1,\ldots,I_t$ *de L appartenant à la même classe d'équivalence de* $\mathcal{R}$, *retourne une instance* (I',k') *de* L' *en temps polynomial en* $\Sigma_{i=1}^{t}|I_i|$ *telle que :*

- $(I',k') \in L' \Leftrightarrow I_i \in L$ *pour un certain* $i \in [t]$ *et,*
- k' *est borné par un polynôme en* $\max_{i\in[t]}|I_i| + \log t$.

Théorème 2.34 ([18]). *Soient L un problème NP-Difficile et* L' *un problème paramétré. Si L se cross-compose vers le problème* L', *et si* L' *a un noyau polynomial, alors il existe un algorithme de oudistillation pour la version non-paramétrée* L'^{c}.

Corollaire 2.35 ([18])**.** *Supposons qu'un problème NP-Difficile L se cross-compose vers un problème paramétré L'. Alors L' n'admet pas de noyau polynomial, à moins que $NP \subseteq coNP/poly$ soit vérifié.*

Remarque. La notion de cross-composition permet d'unifier les notions de Ou-composition et les transformations polynomiales en temps et paramètre vues précédemment. En effet, remarquons dans un premier temps que tout algorithme de Ou-composition pour un problème paramétré L est en fait une cross-composition du problème non paramétré L^c vers le problème L. De plus, l'association d'une Ou-composition pour L et d'une transformation polynomiale en temps et paramètre de L vers un problème paramétré L' constitue également une cross-composition : dans un premier temps, nous appliquons l'algorithme de Ou-composition sur des instances de L pour obtenir une instance *équivalente* de L, que nous transformons ensuite en instance de L' en utilisant la transformation polynomiale en temps et paramètre. Il suit que le problème (classique) L^c se cross-compose vers le problème paramétré L'.

Nous résumons dans la Figure 2.1 les principaux résultats obtenus via la notion de cross-composition. Les versions paramétrées de tous les problèmes énoncés ci-après sont connues FPT [18]. Observons que les paramètres considérés sont non-standards (Section 2.1.3).

Nom du problème	Paramètre	Taille du noyau
CLIQUE	|vertex cover|	non-polynomial
NOMBRE CHROMATIQUE	|vertex cover|	non-polynomial
FEEDBACK VERTEX SET	distance à cluster	non-polynomial
FEEDBACK VERTEX SET	distance à co-cluster	non-polynomial
FEEDBACK VERTEX SET VALUÉ	|vertex cover|	non-polynomial

FIGURE 2.1 : Bornes inférieures pour l'existence de noyaux polynomiaux obtenues via la notion de cross-composition.

2.3.5 Bornes inférieures pour des problèmes d'édition de graphes

Afin d'illustrer ces différentes techniques permettant de prouver la non-existence de noyaux polynomiaux pour des problèmes paramétrés, nous évoquons maintenant des travaux réalisés avec Sylvain GUILLEMOT, Frédéric HAVET et Christophe PAUL, dans lesquels nous obtenons des bornes inférieures sur l'existence de noyaux polynomiaux pour le problème Γ-FREE EDGE DELETION, où Γ dénote un graphe fini. Nous présentons uniquement les intuitions de nos algorithmes de composition et de nos transformations polynomiales en temps et paramètre, et présentons l'intégralité des preuves dans l'Annexe A.

Par le résultat de Cai [31] présenté dans la Section 2.1.2, nous savons que ce problème admet un algorithme FPT. Mais qu'en est-il de l'existence d'un algorithme de noyau polynomial ? De manière plus générale, le résultat de Cai a donné lieu à la conjecture suivante, également proposée par Cai [15].

Conjecture 5 ([15])**.** *Le problème $\mathcal{G}$-EDITION admet un noyau polynomial lorsque $\mathcal{G}$ est héréditaire et caractérisée par un ensemble fini de sous-graphes induits interdits.*

En 2009, Krastch et Walshtröm ont démontré par la négative la Conjecture 5. Pour ce faire, ils ont prouvé qu'une version particulière de SAT n'admettait pas de noyau polynomial, et ont ensuite réduit ce problème via une transformation polynomiale en temps et paramètre au problème Γ-FREE EDGE-DELETION, où Γ est le graphe représente Figure 2.2.

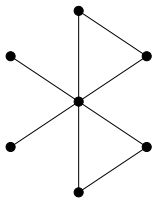


FIGURE 2.2 : Le graphe Γ pour lequel le problème Γ-FREE EDGE DELETION n'admet pas de noyau polynomial [111].

Avec Sylvain GUILLEMOT et Christophe PAUL, nous avons adapté l'algorithme de composition de Kratsch et Walshtröm au problème NOT-1-IN-3-EDGE-TRIANGLE, et utilisé ce résultat pour démontrer que les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION n'admettaient pas de noyau polynomial pour $l \geqslant 12$ et $l \geqslant 13$, respectivement. Avec l'aide de Frédéric HAVET, nous avons par la suite démontré que ce résultat pouvait être amélioré jusqu'à $l \geqslant 7$ pour les deux problèmes, résultats que nous présentons dans cette section. Les problèmes considérés sont définis comme suit :

C_l-FREE EDGE DELETION :
Entrée : Un graphe $G := (V, E)$, $S \subseteq V$, deux entiers k et l.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est C_l-free ?

P_l-FREE EDGE DELETION :
Entrée : Un graphe $G := (V, E)$, $S \subseteq V$, deux entiers k et l.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est P_l-free ?

Pour cela, nous proposons des transformations polynomiales en temps et paramètre depuis une version graphique du problème NOT-1-IN-3-SAT, défini et prouvé ou-composable par Kratsch et Wahlström [111]. Formellement, nous définissons le problème suivant. Étant donnés un graphe $G := (V, E)$ et une arête-bicoloration partielle $B : E' \to \{0, 1\}$, avec $E' \subseteq E$, une arête-bicoloration $B' : E \to \{0, 1\}$ *étend* B lorsque $B'(uv) = B(uv)$ pour toute arête $uv \in E(G)$ colorée par B. Le *poids* d'une arête-bicoloration B' est défini comme le nombre de 1-arêtes $\omega(B')$ de B'. Nous disons qu'une arête-bicoloration B' étendant B est *valide* lorsqu'aucun triangle $\{u, v, w\}$ de G possède *exactement* une arête colorée 1.

NOT-1-IN-3-EDGE-TRIANGLE :
Entrée : Un graphe $G := (V,E)$, une arête-bicoloration partielle $B : E' \to \{0,1\}$, avec $E' \subseteq E$, un entier k.
Paramètre : k
Question : Existe-t-il une arête-bicoloration *valide* B', de poids au plus k, qui étend B ?

Ce problème est NP-Complet et ou-composable. La preuve permettant d'obtenir l'algorithme de ou-composition est similaire à celle présentée par Kratsch et Wahlström [111] afin d'obtenir un algorithme de composition pour le problème NOT-1-IN-3-SAT. En particulier, nous utilisons des gadgets dits de *sélection* et de *propagation*, permettant d'imposer qu'une solution pour l'instance composée implique l'existence d'une solution pour une des instances données en entrée.

Lemme 2.36. *Le problème* NOT-1-IN-3-EDGE-TRIANGLE *est NP-Complet et ou-composable.*

Par le Corollaire 2.27, nous obtenons le résultat suivant.

Corollaire 2.37. *Le problème* NOT-1-IN-3-EDGE-TRIANGLE *n'admet pas de noyau polynomial, à moins que $NP \subseteq coNP/poly$.*

Le problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE est la restriction du problème NOT-1-IN-3-EDGE-TRIANGLE où le graphe $G := (V,E)$ est triparti, *i.e.* V peut être partitionné en 3 ensembles indépendants V_1, V_2, V_3. Les résultats de complexité obtenus pour le problème NOT-1-IN-3-EDGE-TRIANGLE peuvent être appliqués à cette restriction, comme l'indique le résultat suivant.

Proposition 2.38. *Le problème* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE *est NP-Complet et n'admet pas de noyau polynomial, sauf si $NP \subseteq coNP/poly$.*

Par la suite, TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE sera le point de départ de nos transformations polynomiales en temps et paramètre vers les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION.

Nous allons maintenant décrire les transformations polynomiales en temps et paramètre depuis le problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE. Pour des raisons techniques, le problème ne sera pas directement réduit vers les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION mais vers une version *annotée* des problèmes, définie comme suit :

ANNOTATED $\mathcal{G}$-EDGE-DELETION :
Entrée : Un graphe $G := (V,E)$, $S \subseteq V$, un entier k.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E \cap (S \times S)$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ appartient à $\mathcal{G}$?

Nous prouvons que la version annotée d'un problème se réduit vers sa version classique sous certaines hypothèses. Nous verrons plus tard que ces hypothèses sont vérifiées par les problèmes considérés, et ce résultat nous permettra donc de conclure nos preuves.

Observation 2.39. *Soit $\mathcal{G}$ une classe de graphes héréditaire et close par ajout de vrai (resp. faux) jumeau. Il existe une transformation polynomiale en temps et paramètre du problème* ANNOTATED $\mathcal{G}$-EDGE-DELETION *vers le problème* $\mathcal{G}$-EDGE-DELETION.

Preuve. Soit (G,k) une instance du problème ANNOTATED $\mathcal{G}$-EDGE-DELETION. Nous supposons dans un premier temps que $\mathcal{G}$ est close par ajout de vrai jumeau. Nous construisons une instance (G',k') du problème $\mathcal{G}$-EDGE-DELETION comme suit : tout sommet u de $V(G)\setminus S$ est remplacé par une clique de taille $k+1$ ayant le même voisinage que u. Soit F une k-suppression de G. Par définition, $F \subseteq E(G)\cap(S\times S)$, et ainsi le graphe $G'-F$ est obtenu à partir de $G-F$ en ajoutant des vrais jumeaux aux sommets de $V(G)\setminus S$. Il suit que $G'-F$ appartient à $\mathcal{G}$ et ainsi F est une k-suppression de G'. Inversement, soit F' une k-suppression de G'. Puisque $\mathcal{G}$ est close par ajout de vrai jumeau et héréditaire, nous pouvons supposer (Lemme 2.49) que F' supprime toute ou aucune arête entre deux cliques critiques. Il suit que les cliques de G' correspondant aux sommets de $V(G)\setminus S$ ne sont pas affectées par F', et ainsi $F' \subseteq E(G')\cap(S\times S)$. Cela implique que F' est une k-suppression de G.

Dans le cas où $\mathcal{G}$ est close par ajout de faux jumeau, la preuve est similaire et consiste à remplacer les sommets de $V\setminus S$ par un ensemble indépendant de taille $k+1$. □

Nous démontrons maintenant les principaux résultats de cette section.

Théorème 2.40. *Le problème* C_l-FREE EDGE DELETION *n'admet pas de noyau polynomial pour tout* $l \geqslant 7$, *sauf si* $NP \subseteq coNP/poly$.

Preuve. Nous énonçons tout d'abord un résultat technique similaire à un résultat démontré par Kratsch et Wahlström pour le problème NOT-1-IN-3-SAT.

Affirmation 2.41. *Soit* (G,B,k) *une instance de* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE. *Une instance équivalente* (G',B',k) *telle que toute arête* $uv \in E(G')$ *colorée par* B' *vérifie* $B'(uv)=1$ *peut être créée en temps polynomial.*

Nous décrivons une transformation polynomiale en temps et paramètre depuis la restriction du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE sans 0-arêtes vers le problème ANNOTATED C_l-FREE EDGE-DELETION. Le résultat suit alors du Corollaire 2.27 et du fait que ANNOTATED C_l-FREE EDGE-DELETION se réduise à C_l-FREE EDGE DELETION par l'Observation 2.39 (puisque les graphes sans C_l sont close par ajout de vrai jumeau pour $l \geqslant 4$).

Soit (G,B,k) une instance du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE, où V_1, V_2 et V_3 dénotent les trois ensembles indépendants de $G := (V,E)$. La construction de l'instance (H,S,k') de ANNOTATED C_l-FREE EDGE-DELETION est la suivante : premièrement, les ensembles V_1, V_2 et V_3 sont transformés en cliques et les 1-arêtes de G sont supprimées. En plus de V, le graphe H contient un ensemble U de nouveaux sommets. Pour toute paire $t := (e,v)$, avec $e := uw \in E(G)$ et $v \in V(G)$ tels que $\{u,v,w\}$ induit un triangle, nous créons dans H un chemin $P_t := \{p_t^1,\dots,p_t^{l-3}\}$ de longueur $l-3$. Les sommets du chemin P_t sont ajoutés à l'ensemble de sommets U. Remarquons que chaque triangle de G génère trois chemins de la sorte. Pour terminer la construction, nous ajoutons des *arêtes de sécurité* incidentes aux sommets de U.

- pour tous sommets x et y n'appartenant pas au même chemin, xy est une arête de H.
- Dans tout chemin P_t, le sommet p_t^1 (resp. p_t^{l-3}) est adjacent à $V\setminus\{v,w\}$ (resp. $V\setminus\{u,v\}$).

Nous dénotons par $H := (V_H, E_H)$ le graphe résultant de cette réduction (voir Figure A.2). Pour terminer la description de l'instance (H, S), nous posons $S := V$ et $k' := k - k_1$, où k_1 correspond au nombre de 1-arêtes de G. L'intuition de cette construction est que tout cycle de longueur l dans H correspondra à un triangle de G ne contenant qu'une seule 1-arête. Comme ces triangles ne peuvent plus exister dans une arête-bicoloration valide B' de G, une arête-suppression pour C_l pourra ainsi être déduite de B'. Formellement, nous obtenons les résultats suivants.

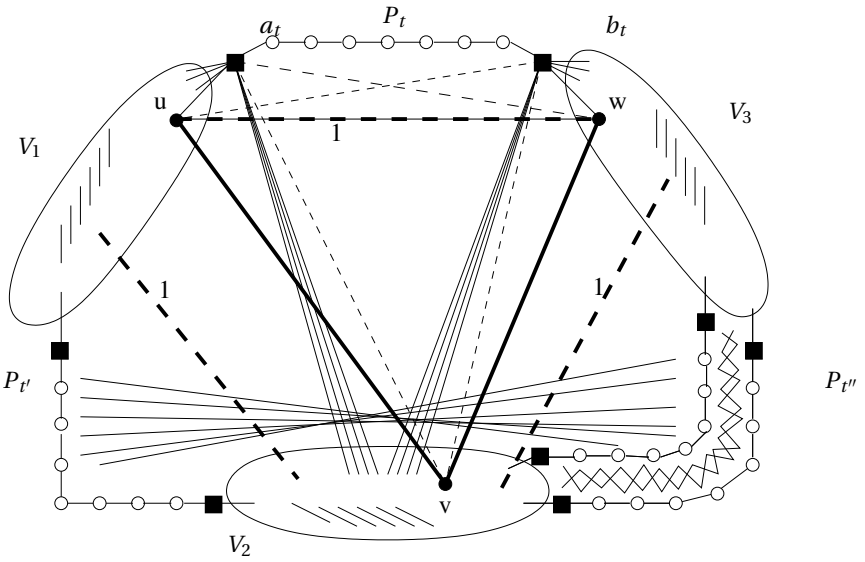


FIGURE 2.3 : Illustration de la construction du graphe $H := (V_H, E_H)$ pour le problème C_l-FREE EDGE DELETION.

Affirmation 2.42. *Un ensemble de sommets $C \in V_H$ induit un cycle de longueur l si et seulement si G contient un triangle $\{u, v, w\}$, avec $e := uw$ une 1-arête et uv, vw deux arêtes non colorées, telles que $C := P_t \cup \{v\}$, $t := (e, v)$.*

Preuve. Par construction, si G contient un triangle $\{u, v, w\}$ avec une unique 1-arête $e := uw$, alors $C := P_t \cup \{v\}$ (avec $t := (e, v)$) induit un cycle de longueur l dans H (rappelons que les 1-arêtes de G ont été supprimées dans H). Inversement, soit C un cycle induit de longueur l dans H. Puisque V_1, V_2 et V_3 sont des cliques de H, il suit que $|C \cap V| \leqslant 6$, et donc C doit intersecter l'ensemble de sommets U. Observons de plus que C doit également intersecter V : en effet, dans le cas contraire, nous aurions $|C| \leqslant 4$ (rappelons que les sommets de chemins distincts sont adjacents). Nous prouvons par contradiction que $C \cap U$ est contenu dans un *unique* chemin P_t. Dans un premier temps, remarquons que $C \cap U$ ne peut intersecter trois chemins $P_t, P_{t'}, P_{t''}$, puisque cela impliquerait l'existence de trois sommets induisant un C_3. Supposons maintenant que $C \cap U$ intersecte deux chemins distincts P_t, $P_{t'}$ $(t \neq t')$. Si $C \cap U$ contient deux sommets de P_t et deux sommets de $P_{t'}$, alors ces quatre sommets induisent un C_4, ce qui est impossible. De manière similaire, si $C \cap U$ contient trois sommets de P_t et un sommet de $P_{t'}$, alors ce sommet aurait degré au moins 3 dans C, ce qui est impossible (tout sommet d'un cycle a degré exactement 2). Il suit que $|C \cap P_t| \leqslant 2$ et

$|C \cap P_{t'}| = 1$. Puisque $C \cap V \neq \emptyset$, les sommets de $C \cap P_t$ correspondent aux extrémités de P_t. Maintenant, par définition de C, un sommet de $C \cap P_t$ est non-incident à au moins $l-3 \geqslant 3$ sommets de $C \cap V$, et ne peut donc correspondre ni à p_t^1 ni à p_t^{l-3} par définition des adjacences de ces sommets : contradiction. Ainsi, il existe un chemin P_t, avec $t := (e, v)$ et $e := uv$, contenant les sommets de $C \cap U$. Deux cas sont alors possibles : (i) $C \cap P_t \subseteq \{p_t^1, p_t^{l-3}\}$ ou (ii) $P_t \subseteq C$. Dans le premier cas, p_t^1 ou p_t^{l-3} est non-adjacent à au moins $l-3 \geqslant 3$ sommets de $C \cap V$, ce qui contredit la définition des adjacences de ces sommets. Il suit que (ii) est vérifié, et C est constitué des sommets de P_t plus trois sommets $x, y, z \in V$, où x est (uniquement) adjacent à p_t^1 dans C, z adjacent uniquement à p_t^{l-3} et y non-adjacent à p_t^1 et p_t^{l-3}. Puisque v est le seul sommet de V non incident à la fois à p_t^1 et p_t^{l-3}, il suit que $y = v$. Cela implique que $x = u$ et $z = w$ (puisque ce sont les seuls sommets non-incidents à p_t^{l-3} et p_t^1, respectivement). Finalement, l'existence du chemin P_t implique l'existence du triangle $\{u, v, w\}$ dans G. Comme $uw, wv \in E_H$ et $uw \notin E_H$, uw est l'unique 1-arête du triangle $\{u, v, w\}$ dans G. Ceci conclut la preuve de l'Affirmation 2.42. ⋄

Affirmation 2.43. *L'instance* (G, B, k) *est une instance positive de* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE *si et seulement si l'instance* (H, S, k') *est une instance positive de* C_l-FREE EDGE DELETION.

Preuve. Supposons qu'il existe un ensemble F d'au plus k' arêtes autorisées tel que $H' := (V_H, E_H \setminus F)$ est C_l-free. Nous définissons l'arête-bicoloration de E comme suit : pour toute arête $e \in E_H$, $B'(e) = 1$ si $e \in F$, et $B'(e) = 0$ sinon. De plus, puisque toute arête $e \in E(G) \setminus E_H$ est une 1-arête, nous posons $B'(e) = 1$ pour de telles arêtes. Comme B n'assigne aucune 0-arête par hypothèse, il suit que B' étend B et a un poids d'au plus $|F| + k_1 \leqslant k' + k_1 = k$. Nous prouvons maintenant que B' est une arête-bicoloration valide de G. Soit $t := (e, v)$, avec $e := uw \in E(G)$, une paire telle que $\{u, v, w\}$ induit un triangle de G. Supposons par contradiction et sans perte de généralité que $B(uw) = 1$, $B'(uv) = B'(vw) = 0$. Dans ce cas, $P_t \cup \{v\}$ induit un cycle de longueur dans H', ce qui est impossible. Inversement, supposons que B' soit une arête-bicoloration valide de G, de poids au plus k, qui étend B. Soit $F \subseteq E(G)$ l'ensemble d'arêtes non-colorées par B telles que $B'(e) = 1$. Par construction, F est un ensemble d'arêtes autorisées de H et a taille au plus $k - k_1$ (rappelons que les 1-arêtes de G ne sont pas ajoutées dans F). Comme B' est une arête-bicoloration valide de G, l'Affirmation 2.42 implique que $H' := (V_H, E_H \setminus F)$ ne contient pas de cycle induit de longueur l. ⋄

Cela conclut la preuve du Théorème 2.40. □

En modifiant légèrement la construction décrite précédemment, nous obtenons le résultat suivant (présenté dans l'Annexe A).

Théorème 2.44. *Le problème* P_l-FREE EDGE DELETION *n'admet pas de noyau polynomial pour tout* $l \geqslant 7$, *sauf si* $NP \subseteq coNP/poly$.

Remarque. En utilisant la technique dite de *cross-composition* [18] (Section 2.3), nous avons récemment amélioré le Théorème 2.40, en démontrant le résultat suivant.

Théorème 2.45. *Le problème* C_l-FREE EDGE DELETION *n'admet pas de noyau polynomial pour tout* $l \geqslant 4$, *sauf si* $NP \subseteq coNP/poly$.

Partie I. Edition de graphes - réduire les branches

Contexte général. Dans la première partie de cette thèse, nous présentons des algorithmes de noyau pour des problèmes **d'édition de graphes**. Nous rappelons la définition formelle de ce problème [3].

$\mathscr{G}$-EDITION :   **Entrée** : Un graphe $G := (V, E)$, un entier k.   **Paramètre** : k.   **Question** : Existe-t-il un ensemble $F \subseteq (V \times V)$ de taille au plus k tel que le graphe $H := (V, E \triangle F)$ appartient à la classe $\mathscr{G}$?

Ces problèmes sont largement étudiés dans la littérature [47, 68, 79, 94, 100, 123, 138, 153], et sont généralement NP-Complets [35, 117, 147]. Certains de ces problèmes font partie des problèmes NP-Complets **historiques**, comme par exemple VERTEX COVER [102], qui peut s'énoncer comme un problème d'édition de graphe de la manière suivante : existe-il un ensemble $C \subseteq V$ de taille au plus k tel que le graphe $G \setminus C$ est un ensemble indépendant ? D'autres problèmes d'édition de graphes sont des problèmes majeurs de la théorie des graphes, comme notamment FEEDBACK VERTEX SET [139, 151], consistant à supprimer un ensemble d'au plus k sommets d'un graphe pour obtenir une forêt, ou bien encore MULTICUT [22, 44, 78] qui étant donnés un graphe et un ensemble de requêtes (*i.e.* paires de sommets de G) consiste lui à trouver un ensemble d'au plus k arêtes dont la suppression **déconnecte** toutes les requêtes du graphe.

Applications. Les problèmes d'édition de graphes trouvent des applications dans de nombreux domaines, comme par exemple le traitement d'image [147], les bases de données [150] et surtout la bioinformatique [37, 82, 100, 148]. Un exemple de base liant ces deux domaines de recherche est le suivant : supposons que l'on récupère expérimentalement un ensemble de données : par exemple,

3. Les problèmes que nous considérons dans la suite modifient principalement les **arêtes** du graphe. Nous préciserons lorsque les problèmes étudiés supprimeront des sommets du graphe.

des **ressemblances** entre espèces. Si les données étaient d'une certaine manière **parfaites**, elles devraient vérifier une propriété particulière : par exemple, des classes d'équivalence entre espèces devraient se former. Cependant, pour diverses raisons (problème lors de l'acquisition des données, bruit, ...) ces propriétés ne sont pas vérifiées mais sont **proches** de l'être. En ce sens, réaliser un petit nombre de modifications sur les données considérées permettraient d'obtenir des données vérifiant la propriété désirée. Si l'on formule cet exemple en termes de graphes, les espèces seront représentées par les sommets, et deux sommets seront reliés par une arête si leur ressemblance dépasse un seuil prédéfini. Le cas idéal serait donc que ce graphe soit une union disjointe de cliques, et le problème d'édition correspondant est CLUSTER EDITING, étudié dans le **Chapitre 2**.

Résultats connus. L'étude de la complexité paramétrée des problèmes d'édition de graphes a connu une attention particulière ces dernières années [22, 37, 123, 141]. Comme dans le cadre des problèmes classiques, VERTEX COVER constitue un problème de base. En effet, il s'agit d'un des premiers problèmes à avoir été démontré FPT [57, 129]. De plus, un résultat de Buss [29] impliquait l'existence d'un noyau quadratique pour le problème, et cela avant même que la notion de noyau soit formellement définie. Par la suite, de nombreux résultats concernant la complexité paramétrée des problèmes d'édition de graphes ont été démontrés. En 1994, Kaplan et al. [100, 101] ont ainsi prouvé que les problèmes CHORDAL COMPLETION et PROPER INTERVAL COMPLETION admettaient des algorithmes FPT, établissant l'un des premiers algorithmes FPT non trivial pour des problèmes d'édition de graphes. En 1996, Cai [31] a démontré que le problème $\mathcal{G}$-ÉDITION (version édition d'arête ou suppression de sommets) était FPT dès lors que $\mathcal{G}$ était une classe de graphes **héréditaire** caractérisée par une famille **finie de sous-graphes induits interdits**. Plus récémment, grâce à l'introduction de nouvelles techniques permettant d'élaborer des algorithmes FPT, de nouveaux résultats ont été établis pour des problèmes ouverts depuis de nombreuses années [141]. Ainsi, il a été démontré que les problèmes CHORDAL DELETION [123] (utilisant la technique dite de **compression itérative** [141]) et INTERVAL COMPLETION [94] admettent des algorithmes FPT. Il est important de noter également qu'il existe quelques problèmes d'édition de graphes n'admettant pas d'algorithme FPT : par exemple, WHEEL-FREE DELETION est $W[2]$-Difficile [118].

En ce qui concerne l'existence de noyaux pour de tels problèmes, de nombreux résultats sont apparus dans la littérature ces dernières années. En particulier, le problème CLUSTER EDITING a été longuement étudié, les premiers noyaux obtenus contenant $O(k^2)$ sommets [136], avant d'être améliorés à $4k$ [84] puis récemment à $2k$ sommets [37]. Le problème VERTEX COVER a connu des améliorations similaires, le meilleur noyau connu contenant $2k-c$ sommets pour toute constante $c>0$ [149]. Notons également que l'algorithme FPT décrit par Kaplan et al. [100] pour le problème CHORDAL COMPLETION utilisait l'existence d'un algorithme de noyau quadratique pour ce problème. Enfin, mentionnons également que les problèmes FEEDBACK VERTEX SET [151] et CLUSTER VERTEX DELETION [152] admettent des noyaux quadratiques, alors que la question est à l'heure actuelle ouverte pour les problèmes CHORDAL DELETION [123] et INTERVAL COMPLETION [94] (voir **Chapitre 9**, **Section 9.3**). Remarquons également que le problème $\mathcal{G}$-VERTEX DELETION admet un algorithme de noyau polynomial lorsque $\mathcal{G}$ est caractérisée par un ensemble fini de sous-graphes induits interdits [1]. En ce qui concerne l'édition d'arêtes, suivant son résultat stipulant que le problème $\mathcal{G}$-EDITION est FPT dès lors que $\mathcal{G}$ est héréditaire et caractérisée par une famille finie de

sous-graphes induis interdits, Cai [15] a conjecturé que de tels problèmes admettaient également des algorithmes de noyau polynomiaux. Cependant, les résultats récemment introduits par Bodlaender et al. [17] ont permis d'établir que certains problèmes d'édition de graphes n'admettaient pas de noyaux polynomiaux [111]. Nous avons ainsi vu dans le **Chapitre 2** que cela pouvait être le cas même lorsque la classe cible est une classe de graphes très restrictive. De ce fait, déterminer l'existence d'un noyau polynomial devient un enjeu de recherche important, aussi bien d'un point de vue théorique que pratique.

Résultats présentés. Dans cette partie, nous suivons cette ligne de recherche, et établissons un certain nombre de résultats pour des problèmes d'édition de graphes. En particulier, nous développons les **premiers** algorithmes de noyau connus pour plusieurs problèmes d'édition de graphe. Pour ce faire, nous introduisons une technique basée sur une décomposition en **branches** du graphe, que nous présentons dans la suite de cette introduction. Cette technique nous permet notamment d'obtenir des noyaux polynomiaux pour les problèmes CLOSEST 3-LEAF POWER [11, 12] et PROPER INTERVAL COMPLETION [13], admettant tous deux des algorithmes FPT [53, 100]. Ces résultats seront détaillés dans les **Chapitres 3 et 4**. Pour conclure cette Partie, nous décrirons un algorithme de noyau pour le problème COGRAPH EDITION, en nous basant sur l'**arbre de décomposition modulaire** d'une instance du problème (**Chapitre 5**).

Règles de réduction génériques pour le problème $\mathcal{G}$-EDITION

Dans cette section, nous présentons des règles de réduction *génériques* pouvant s'appliquer sur de nombreux problèmes d'édition. Nous démontrons en particulier comment utiliser ces règles pour obtenir un noyau quadratique pour le problème CLUSTER EDITING.

Supprimer les composantes connexes. L'une des règles de réduction les plus classiques [12, 13, 83, 123] (notamment utilisée dans le noyau pour VERTEX COVER présenté dans la **Section 2.2.1**) consiste à supprimer d'une instance (G,k) du problème $\mathcal{G}$-EDITION toute composante connexe appartenant à la classe $\mathcal{G}$.

Règle 2.5 (Composante Connexe)**.** *Soit (G,k) une instance du problème $\mathcal{G}$-EDITION où $\mathcal{G}$ est héréditaire, close par union disjointe et dont l'appartenance peut être décidée en temps polynomial. Supprimer de G toute composante connexe C telle que $G[C]$ appartient à $\mathcal{G}$.*

Lemme 2.46. *La Règle 2.5 est valide et peut être appliquée en temps polynomial.*

Preuve. Soient $G' := G \setminus C$ et F une k-édition de G. Puisque $\mathcal{G}$ est héréditaire, il suit que F est une k-édition de G'. Inversement, soit F' une k-édition de G'. Puisque $G[C]$ appartient à $\mathcal{G}$ et vu que $\mathcal{G}$ est stable par union disjointe, il suit que le graphe $(G' \triangle F') \cup G[C]$, qui correspond à $G \triangle F'$, appartient à $\mathcal{G}$. Ainsi, F' est une k-édition de G. Comme l'appartenance à $\mathcal{G}$ peut être décidée en temps polynomial, il suit que la règle peut s'appliquer en temps polynomial. □

Réduire les sunflowers. Nous détaillons maintenant une règle de réduction considérant les *sunflowers* d'une instance $G := (V,E)$ du problème $\mathcal{G}$-EDITION. Encore une fois, cette règle de réduction

est largement utilisée dans la littérature [13, 21, 82, 83], et nous servira à de nombreuses reprises dans la suite de cette thèse (**Chapitres 4, 5 et 7**). Nous adapterons la définition suivante lorsque nous étudierons des problèmes d'édition de relations.

Définition 2.47 (Sunflower)**.** *Soit* (G,k) *une instance de* $\mathcal{G}$-EDITION *où* $\mathcal{G}$ *est caractérisée par une famille de sous-graphes induits interdits. Un ensemble* $\mathcal{S} := \{C_1,\dots,C_m\}$ *de sous-graphes induits interdits est une* sunflower *de* G *s'il existe* $u,v \in V$ *tels que pour tous* $1 \leqslant i < j \leqslant m$, $C_i \cap C_j = \{u,v\}$. *La paire* $\{u,v\}$ *est appelée le* centre *de* $\mathcal{S}$.

Nous verrons que cette définition peut être relaxée suivant le type d'édition autorisé (Chapitre 4).

Règle 2.6. *Soient* (G,k) *une instance de* $\mathcal{G}$-EDITION, *et* $\mathcal{S} := \{C_1,\dots,C_m\}$ *une* sunflower *de* G *de centre* $\{u,v\}$, *avec* $m > k$. *Transformer* G *en* $G \triangle \{\{u,v\}\}$ *et décrémenter* k *de* 1.

Lemme 2.48. *La Règle 2.6 est valide et peut être appliquée en temps polynomial.*

Preuve. Nous démontrons que toute k-édition F de G doit contenir la paire $\{u,v\}$. Supposons par contradiction que ce ne soit pas le cas, et considérons une k-édition F de G ne contenant pas $\{u,v\}$. Par définition de $\mathcal{S}$, F doit contenir une paire pour tout sous-graphe induit interdit C_i, $i \in [m]$. Comme $m > k$, il suit que $|F| > k$, une contradiction. Ainsi, toute k-édition doit contenir la paire $\{u,v\}$, et la règle est donc valide. Une telle structure peut être détectée en temps polynomial dans la mesure où les sous-graphes induits interdits sont finis. □

Observons que la Règle 2.6 peut être adaptée de manière similaire aux problèmes $\mathcal{G}$-COMPLETION et $\mathcal{G}$-EDGE DELETION.

Réduire les cliques critiques. Nous démontrons maintenant que la notion de clique critique peut également être exploitée dans le cadre du problème $\mathcal{G}$-EDITION, lorsque la classe de graphes est héréditaire et close par ajout de vrai jumeau. En particulier, nous prouvons l'existence d'une édition optimale qui *préserve* les cliques critiques d'une instance du problème. La version d'optimisation du problème $\mathcal{G}$-EDITION consiste à trouver un ensemble d'arêtes de taille minimum dont l'édition dans G permet d'obtenir un graphe appartenant à $\mathcal{G}$.

Lemme 2.49 ([11, 12])**.** *Soit* $\mathcal{G}$ *une classe de graphes héréditaire et close par ajout de vrai jumeau. Pour toute instance* $G := (V,E)$ *du problème d'optimisation* $\mathcal{G}$-EDITION, *il existe une édition optimale* F *telle que toute clique critique de* G *est contenue dans une clique critique de* $H := G \triangle F$.

Preuve. Soit F une édition optimale maximisant le nombre m de cliques critiques de G qui sont cliques modules dans H. Formellement, nous supposons $\mathcal{C}_G := \{C_1,\dots,C_c\}$ et $m < c$ (*i.e.* $C_1,\dots,C_m$ sont cliques modules dans H et $C_{m+1},\dots,C_c$ ne sont plus cliques modules dans H). Soit u un sommet de C_{m+1} tel que le nombre de paires de F contenant u est minimum, et soit $C_u := C_{m+1} \setminus \{u\}$. Nous considérons maintenant le graphe $H_u := H \setminus C_u$. Puisque $\mathcal{G}$ est héréditaire, le graphe H_u appartient à la classe $\mathcal{G}$. De plus, comme $\mathcal{G}$ est stable par ajout de vrai jumeau, insérer les sommets de C_u dans H_u comme vrais jumeaux de u permet d'obtenir un graphe H' qui appartient à $\mathcal{G}$. Par construction, le graphe H' est obtenu à partir de G en éditant les paires F' contenues dans

$E(G) \triangle E(H')$. De plus, par choix de u, nous savons que $|F'| \leqslant |F|$, ce qui implique que F' est une édition optimale. Pour conclure, nous savons par construction que $C_1, \ldots, C_m$ et C_{m+1} sont des cliques modules dans H', impliquant le résultat. □

Une implication directe du Lemme 2.49 est que, étant donnés une classe de graphes $\mathscr{G}$ héréditaire et close par ajout de vrai jumeau et une instance $G := (V, E)$ de $\mathscr{G}$-EDITION, il existe *toujours* une édition optimale telle que :

(i) toute arête entre deux sommets d'une clique critique de G est une arête de $G + F$ et,

(ii) étant données deux cliques critiques C et C' de G, $(V(C) \times V(C')) \subseteq E(G \triangle F)$ ou $(V(C) \times V(C')) \cap E(G \triangle F) = \emptyset$.

En d'autres termes, (ii) implique qu'il existe toujours une édition optimale qui, étant données deux cliques critiques C et C' de G, édite toute ou aucune arête entre C et C'. Dans la suite de cette partie, nous assumons que toute édition considérée vérifie cette propriété.

Remarques :

- Les résultats précédents sont également vérifiés pour les problèmes $\mathscr{G}$-COMPLETION et $\mathscr{G}$-EDGE DELETION.
- Les arguments précédents sont également vérifiés si l'on considère les *ensembles indépendants critiques* (définis de manière analogue aux cliques critiques) et une classe de graphes close par ajout de faux jumeau. Un tel résultat a été démontré de manière indépendante par Komusiewicz et Uhlmann. [107], obtenant un noyau cubique pour le problème MINIMUM FLIP CONSENSUS TREE.

Règle 2.7. *Soit (G, k) une instance de $\mathscr{G}$-*EDITION *où $\mathscr{G}$ est héréditaire et close par ajout de vrai jumeau. Soit C une clique critique de G telle que $|C| > k+1$. Supprimer $|C| - (k+1)$ sommets arbitraires de C dans G.*

Lemme 2.50. *La Règle 2.7 est valide et peut être appliquée en temps polynomial.*

Preuve. Soit C une clique critique de G telle que $|C| > k+1$, et soit G' le graphe où $|C| - (k+1)$ sommets de C ont été supprimés. Nous utilisons C' pour désigner la clique critique de G' composée des sommets de C présents dans G'. Comme la classe $\mathscr{G}$ est héréditaire, nous savons que toute k-édition pour G est une k-édition pour G'. Inversement, soit F' une k-édition pour G'. Par le Lemme 2.49, nous pouvons supposer que F' contient toute ou aucune arête entre deux cliques critiques. De ce fait, comme $|C'| = k+1$, F' ne contient aucune arête incidente à C', et les sommets de C' ne sont pas affectés par F'. Cela implique que les sommets de C' ont le même voisinage dans G' et $G' + F'$; comme les sommets de $C \setminus C'$ sont des vrais jumeaux de C', et comme la classe $\mathscr{G}$ est close par ajout de vrai jumeau, il suit que le graphe $G + F'$ appartient à $\mathscr{G}$. Le fait que la règle puisse être appliquée en temps polynomial découle de l'Observation 1.7, qui permet de calculer $\mathscr{C}(G)$ en temps $O(n+m)$. De ce fait, comme $\mathscr{C}_G$ est linéaire en la taille de G, les cliques critiques de taille $k+1$ peuvent simplement être détectées. □

Noyau quadratique pour CLUSTER EDITING

Nous présentons un algorithme de noyau quadratique pour le problème CLUSTER EDITING, en utilisant les règles de réduction présentées précédemment. Ce problème admet un noyau linéaire [37, 85], notamment basé sur la notion de décomposition en couronne d'un graphe. Afin de démontrer la validité des Règles 2.5 et 2.7 dans le cas de ce problème, nous devons démontrer que les unions disjointes de cliques sont héréditaires et closes par ajout de vrai jumeau, ce qui est vérifié par définition même de ces graphes. Nous obtenons alors le résultat suivant.

Théorème 2.51. *Le problème* CLUSTER EDITING *admet un noyau avec* $O(k^2)$ *sommets.*

Preuve. Soit (G,k) une instance positive de CLUSTER EDITING réduite par les Règles 2.5 et 2.7. Soit F une k-édition de G, et $H := G \triangle F$ l'union disjointe de cliques correspondante. Nous supposons que H se décompose en p cliques $\{C_1,\dots,C_p\}$. Comme G est réduit par la Règle 2.5, G contient au plus k composantes connexes et donc $p \leqslant 2k$. Par définition, nous savons que H contient au plus $2k$ sommets affectés. Nous considérons l'ensemble $A \subseteq V(H)$ de ces sommets. Pour tout $i \in [p]$, nous définissons $A_i := A \cap C_i$ et $T_i := C_i \setminus A_i$. Par définition, T_i est un ensemble de sommets non affectés strictement contenu dans une clique de H, et correspond donc à une clique critique de G. Comme G est réduit par la Règle 2.7, il suit que $|T_i| \leqslant k+1$ pour tout $i \in [p]$, ce qui implique : $\sum_{i=1}^{p} |T_i| \leqslant 2k \cdot (k+1)$. Comme $V(H) = \cup_{i=1}^{p} T_i \cup A$, il suit que :

$$V(G) \leqslant \sum_{i=1}^{p} |T_i| + |A| \leqslant 2k \cdot (k+1) + 2k = 2k^2 + 4k$$

ce qui implique le résultat. □

Réduction via la notion de branches

Intuition. L'idée générale de la notion de **branches** dans des algorithmes de noyau est de **réduire** un ensemble de sommets qui induit un graphe qui appartient à la classe de graphes cible $\mathscr{G}$. Observons que cette idée est **naturelle** lorsque l'on cherche à réduire une instance du problème $\mathscr{G}$-EDITION. En effet, cette dernière a déjà été utilisée dans plusieurs algorithmes de noyau [12, 136], en particulier via la règle de réduction suivante (qui correspond à la Règle 2.5 présentée dans la **Section 2.3.5**).

> *Soit* (G,k) *une instance du problème* $\mathscr{G}$-EDITION *où* $\mathscr{G}$ *est héréditaire et close par union disjointe. Supprimer de* G *toute composante connexe* C *telle que* $G[C]$ *appartient à* $\mathscr{G}$.

Comme nous le verrons par la suite, la Règle 2.5 est un cas particulier des règles de réduction via la notion de branches. Une autre application de cette idée a permis d'obtenir un noyau linéaire pour le problème CLUSTER EDITING, et correspond elle à la Règle 2.7 de la **Section 2.3.5** : étant donnée une clique critique de G de taille arbitraire, il est possible de ne préserver que $k+1$ sommets de cette clique critique. Cette observation provient du fait que toute l'information **utile** contenue dans une telle clique réside dans sa **frontière**, et qu'il est ainsi sûr de ne préserver qu'un petit nombre de ses sommets. Dans un travail réalisé avec Stéphane BESSY et Christophe PAUL [11, 12], nous

nous appuyons sur cette idée et introduisons la notion de **branche** pour le problème CLOSEST 3-LEAF POWER, qui est une généralisation non-triviale des règles précédemment mentionnées. Nous donnons un aperçu général de cette méthode, qui est particulièrement adaptée pour le problème $\mathcal{G}$-EDITION où les membres de la classe $\mathcal{G}$ admettent une **décomposition d'adjacence**.

Décomposition d'adjacence . *Soit $\mathcal{G}$ une classe de graphes. Nous disons que $\mathcal{G}$ admet une* **décomposition d'adjacence** *si pour tout graphe $G := (V, E)$ appartenant à $\mathcal{G}$, il existe une collection $\mathcal{V} := \{V_1, \ldots, V_l\}$ couvrant V telle que $V_i \subseteq V$, $i \in [l]$, et telle que V_i, V_j, $i \neq j$ sont connectés en respectant une certaine propriété d'adjacence $P_{\mathcal{G}}$.*

Comme exemple, observons que les **graphes triangulés** admettent une décomposition d'adjacence, connue sous le nom de **graphe des cliques** [71]. Étant donné un graphe triangulé $G := (V, E)$, nous définissons la collection $\mathcal{V}$ comme la collection des cliques maximales de G. Maintenant, le graphe des cliques $C(G) := (\mathcal{V}, E_c)$ est défini comme le graphe dont les sommets sont les éléments de $\mathcal{V}$ et la relation d'adjacence $P_{triangulé}$ est : il existe une arête entre deux sommets V_i, V_j de $C(G)$ si leur intersection est un séparateur minimal de G [71]. Comme nous le démontrerons par la suite, les 3-leaf powers et les graphes d'intervalle propre admettent une telle décomposition. Nous donnons maintenant une définition générale de la notion de **branches**, qui devra être ajustée pour chaque problème considéré.

Branches . *Soient $\mathcal{G}$ une classe de graphes admettant une décomposition d'adjacence $(\mathcal{V}, P_{\mathcal{G}})$, et $G := (V, E)$ un graphe de $\mathcal{G}$. Nous disons qu'un ensemble $B \subseteq V$ est une* **branche** *de G si :*
- *le graphe $G[B]$ appartient à la classe $\mathcal{G}$,*
- *les arêtes contenues dans $E(\delta(B), N_{G\setminus B}(\delta(B)))$ respectent la propriété d'adjacence $P_{\mathcal{G}}$.*

Détecter les branches. Afin d'obtenir un algorithme de noyau, une condition nécessaire est que les branches puissent être détectées en temps polynomial. Encore une fois, cette phase dépend du problème considéré. Nous verrons dans les **Chapitres 3 et 4** que cela peut être fait pour les problèmes CLOSEST 3-LEAF POWER et PROPER INTERVAL COMPLETION.

Réduire les branches. Comme évoqué précédemment, l'idée principale de la notion de branche est la suivante. Soit (G, k) une instance de $\mathcal{G}$-EDITION où $\mathcal{G}$ admet une décomposition d'adjacence $(\mathcal{V}, P_{\mathcal{G}})$. Soit $B \subseteq V$ une branche de G. L'information principale contenue dans B se situe alors dans sa **frontière**. En d'autres termes, l'ensemble $B^R := B \setminus \delta(B)$ contient **trop d'information**, et il est possible de démontrer *pour certains problèmes* que cet ensemble peut être remplacé de manière valide par un graphe **équivalent** de taille bornée en k. Notons toutefois que cette partie est spécifique au problème considéré, et ne semble donc pas pouvoir s'appliquer de manière générique sur tous les problèmes admettant une décomposition d'adjacence. Informellement, une règle de réduction réduisant les branches d'un graphe répond donc aux lignes suivantes :

> *Soit (G, k) une instance de $\mathcal{G}$-EDITION où $\mathcal{G}$ admet une décomposition d'adjacence. Soit $B \subseteq V$ une branche de G. Remplacer $B^R := B \setminus \delta(B)$ par un* ***graphe équivalent*** *de taille bornée par une fonction de k.*

Borner la taille du noyau. Nous présentons maintenant la raison pour laquelle une telle règle de réduction permet d'obtenir des algorithmes de noyau polynomiaux. Soit (G,k) une instance de $\mathscr{G}$-EDITION où $\mathscr{G}$ admet une décomposition d'adjacence $(\mathcal{V},P_{\mathscr{G}})$. Par souci de simplicité, nous supposons que $\mathcal{V}$ est une partition de V. Soient F une k-édition de G et $H := G \triangle F$. Nous définissons un sommet de G comme **affecté** s'il est contenu dans une paire de F, et un ensemble de $\mathcal{V}$ comme **affecté** s'il contient un sommet affecté. Maintenant, nous considérons les composantes connexes du graphe $H \setminus A(\mathcal{V})$, où $A(\mathcal{V})$ dénote les ensembles affectés de $\mathcal{V}$. Par définition, ces composantes définissent des **branches** de G, dont la frontière doit être attachée au reste du graphe en respectant la propriété $P_{\mathscr{G}}$ (puisque ce sont des sommets non-affectés). Comme $|F| \leqslant k$ par définition, il existe au plus $2k$ sommets affectés dans H, et donc $O(k)$ branches. De ce fait, trouver une manière de réduire **efficacement** ces branches permettrait d'obtenir des noyaux polynomiaux. Cette partie de l'algorithme de noyau est spécifique au problème.

Lien avec les protrusions

Nous présentons maintenant la notion de **protrusions** introduite par Bodlaender et al. [17], qui considère une décomposition arborescente d'un graphe. En utilisant les protrusions, Bodlaender et al. [17] obtiennent plusieurs noyaux polynomiaux pour des problèmes d'édition définis sur des graphes de **genre** borné, établissant un cadre général pour concevoir des algorithmes de noyau polynomiaux pour de tels problèmes. Bien que le concept de protrusion soit similaire à la notion de branches, certaines propriétés additionnelles sont requises sur le problème considéré afin de pouvoir utiliser cette technique. Comme mentionné précédemment, le graphe doit pouvoir être plongé dans une surface de genre borné. De plus, les problèmes considérés doivent pouvoir s'exprimer en **Counting Monadic Second Order Logic (CMSO)**, et être **compacts** [17].

Protrusions. Informellement, une protrusion est un sous-graphe de treewidth constante séparé du reste du graphe par un nombre constant de sommets. L'idée permettant de réduire le graphe est la suivante [17] :

> *S'il existe un séparateur de taille constante dont la suppression permet d'obtenir une composante connexe de grande taille et de treewidth constante, alors il est possible de remplacer cette composante connexe par un graphe de petite taille.*

Si l'idée de cette règle de réduction est similaire à celle présentée dans la section précédente, remarquons cependant que la notion de branche utilise la décomposition d'adjacence propre à la classe $\mathscr{G}$ considérée, ce qui n'est pas le cas ici. Les résultats présentés dans [17] s'appliquent au problème p-MIN-CMSO (resp. p-EQ-CMSO, p-MAX-CMSO), où l'instance consiste en un graphe $G := (V,E)$ et un entier k et où l'objectif est de déterminer l'existence d'un ensemble de sommets/arêtes S de taille au plus (resp. exactement, au moins) k dont l'édition satisfait la propriété CMSO $\mathscr{G}$. Par souci de simplicité, nous mentionnons uniquement les résultats concernant p-MIN-CMSO. La version **annotée** de ce problème possède un ensemble $Y \subseteq V$ de sommets **annotés** et il est requis que la solution S soit un sous-ensemble de Y. Dans ce qui suit, nous considérons la version annotée du problème p-MIN-CMSO, vu que les résultats obtenus pour les problèmes annotés peuvent être propagés aux versions classiques des problèmes (Corollaire 1 dans [17]).

Définition 2.52 (r-protrusion)**.** *Étant donné un graphe $G := (V,E)$, un ensemble $X \subseteq V$ est une r-**protrusion** de G si $treewidth(G[X]) \leqslant r$ et $|\delta(X)| \leqslant r$.*

Réduire les protrusions. Soit $X \subseteq V$ une r-protrusion. L'objectif d'une règle de réduction considérant les protrusions est de **remplacer** l'ensemble de sommets contenus dans $X \setminus \delta(X)$ par un graphe équivalent de taille bornée. Considérer la version annotée des problèmes est utile pour ce faire, puisque nous pouvons alors supposer que les sommets de $X \cap Y$ sont contenus dans $\delta(X)$, et ainsi $S \cap X \subseteq \delta(X)$ pour toute solution S. De ce fait, l'information **importante** contenue dans une r-protrusion se situe dans sa **frontière**. Comme cette dernière a une taille constante par définition, il est alors possible de la préserver tout en remplaçant $X \setminus \delta(X)$ par un graphe équivalent et de taille bornée. Formellement, les règles permettant de réduire les protrusions fonctionnent comme suit.

Lemme 2.53. *Soit Π_A un problème annoté de type p-MIN-CMSO. Pour tous entiers r et g, il existe une constante $c > 0$ et un algorithme qui prend en entrée :*

- *un graphe $G := (V,E)$ pouvant être plongé dans une surface de genre g,*
- *un ensemble $Y \subseteq V$ de sommets annotés,*
- *une r-protrusion X avec $|X| > c$ telle que $Y \cap X \subseteq \delta(X)$,*

et retourne en temps $O(|X|)$:

- *un graphe $G' := (V',E')$ pouvant être plongé dans une surface de genre g,*
- *un ensemble de sommets annotés $Y' \subseteq V'$*

tels que $|V'| < |V|$ et (G',Y',k) est une instance positive si et seulement si (G,Y,k) est une instance positive.

Il reste maintenant à calculer une r-protrusion respectant les conditions du Lemme 2.53. Cela peut être réalisé à partir d'une protrusion contenant un nombre de sommets suffisamment grand (par rapport au paramètre k). Finalement, une telle r-protrusion peut être détectée en temps polynomial en utilisant une décomposition arborescente du graphe G (Section 5.1 dans [17]), permettant d'arriver aux résultats suivants. Un problème paramétré est *compact* lorsqu'il possède certaines propriétés en relation avec la notion de genre borné [17].

Théorème 2.54 ([17])**.** *Soit Π un problème de type p-MIN-CMSO défini sur des graphes de genre borné tels que Π ou $\overline{\Pi}$ est compact. La version annotée Π_A admet un noyau quadratique*[4]*.*

Basé sur la notion d'**index entier fini** [17], Bodlaender et al. améliorent les résultats précédents en noyaux **linéaires** pour plusieurs problèmes. Cela provient du fait que, lorsque le problème considéré a un index entier fini, une r-protrusion respectant les conditions du Lemme 2.53 peut être obtenue à partir d'une r-protrusion de taille **constante** (ne dépendant donc plus du paramètre k). Cela permet d'obtenir les résultats (partiels) suivants.

Théorème 2.55 ([17])**.** *Les problèmes* FEEDBACK VERTEX SET, DOMINATING SET, CONNECTED VERTEX COVER, EDGE DOMINATING SET *définis sur des graphes de genre borné admettent des noyaux avec un nombre linéaire de sommets.*

4. Les noyaux obtenus sont quadratique pour p-MAX-CMSO et cubiques pour p-EQ-CMSO.

CHAPITRE

3 Noyau cubique pour CLOSEST 3-LEAF POWER

Dans ce chapitre, nous illustrons la notion de **branches** définie dans l'introduction de la Partie I sur le problème CLOSEST 3-LEAF POWER. Introduite par Lin et al. [116] dans le contexte de la phylogénie, la classe des p-leaf powers a depuis reçu une grande attention dans la littérature [34, 53, 54, 114]. Un graphe $G := (V, E)$ est un **p-leaf power** s'il peut être obtenu à partir d'un arbre T dont les feuilles sont en bijection avec V et telles que $uv \in E(G) \Leftrightarrow d_T(u, v) \leqslant p$. Il a été établi que la reconnaissance de ces graphes est polynomiale pour $p \leqslant 5$ [24, 26, 34], et ouverte pour $p > 6$. Nous considérons le problème NP-Complet [54] CLOSEST p-LEAF POWER, où l'objectif est d'éditer au plus k arêtes d'un graphe afin de le transformer en un p-leaf power. Pour $p \leqslant 4$, des algorithmes FPT sont connus [53, 54], le problème CLOSEST 3-LEAF POWER pouvant être résolu en temps $O^*(2^k)$ [53]. Cependant, la complexité paramétrée reste ouverte pour $p > 5$. De plus, l'existence d'un noyau polynomial est une question ouverte pour ces valeurs de p. Dans un travail réalisé avec Stéphane BESSY et Christophe PAUL [11, 12], nous démontrons l'existence d'un noyau polynomial pour $p = 3$. Nous donnons maintenant une définition formelle du problème.

CLOSEST 3-LEAF POWER :
Entrée : Un graphe $G := (V, E)$, et un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq (V \times V)$ de taille au plus k tel que le graphe $H := (V, E \triangle F)$ est un 3-leaf power ?

Théorème 3.1. *Les problèmes* CLOSEST 3-LEAF POWER, 3-LEAF POWER COMPLETION *et* 3-LEAF POWER EDGE-DELETION *admettent un noyau avec $O(k^3)$ sommets.*

3.1 Définition et caractérisation des 3-leaf powers

Nous définissons dans un premier temps la notion de *p-leaf power*. Nous décrirons ensuite plus précisément les classes de graphes correspondant aux p-leaf powers pour $p \leqslant 3$.

Définition 3.2 (p-leaf power)**.** *Soit T un arbre non enraciné dont les feuilles sont en bijection avec les sommets d'un ensemble V. Le* p-leaf power *de T est le graphe $T_p := (V,E)$, où $E := \{\{u,v\} : u,v \in V$ et $d_T(u,v) \leqslant p\}$. L'arbre T est appelé le* p-leaf root *de T_p.*

p-leaf power

Nous disons qu'un graphe $G := (V,E)$ est un *p-leaf power* s'il existe un arbre T dont les feuilles sont en bijection avec les sommets V de G et vérifiant $uv \in E(G) \Leftrightarrow d_T(u,v) \leqslant p$.

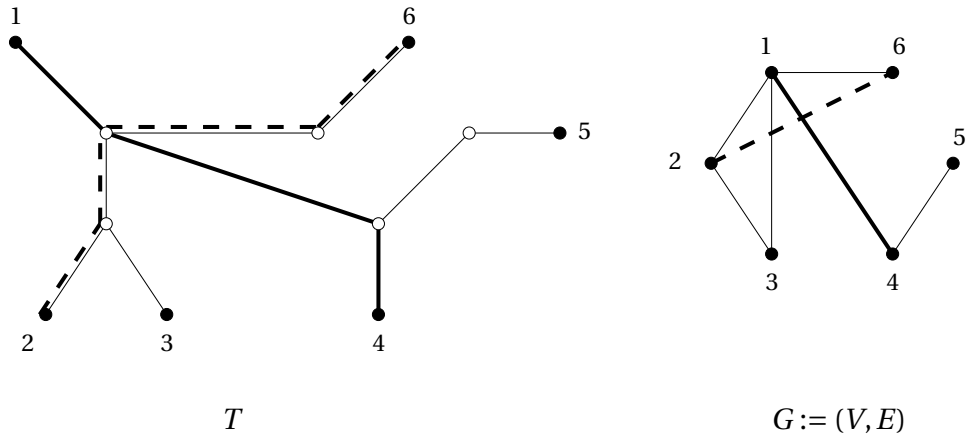


FIGURE 3.1 : Un 3-leaf power $G := (V,E)$ et son 3-leaf root T.

Nous donnons maintenant de simples caractérisations des p-leaf powers pour $p \leqslant 2$. Nous verrons par la suite qu'il existe des caractérisations plus complexes pour les p-leaf powers, notamment pour $p = 3$ [54].

0-leaf power. Par définition, un graphe $G := (V,E)$ est un 0-leaf power s'il peut être obtenu à partir d'un arbre T dont les feuilles sont en bijection avec les sommets V de G et tel que $uv \in E(G)$ si et seulement si les feuilles correspondant aux sommets u et v dans T vérifient $d_T(u,v) = 0$. Comme cela est impossible (sauf si $u = v$), il suit que le graphe G ne contient pas d'arête.

1-leaf power. De manière similaire, la seule configuration possible pour que deux feuilles d'un arbre T soit à distance 1 est que l'arbre soit réduit à deux sommets reliés par une arête. Il suit que les 1-leaf powers correspondent soit au graphe complet K_2 (constitué de deux sommets et d'une arête), soit à des graphes sans arêtes.

2-leaf power. Étant donné un arbre T, deux feuilles u et v de T vérifient $d_T(u,v) = 2$ si et seulement si u et v sont attachées au même noeud interne x de T. Ainsi, dans tout 2-leaf power G de T, l'ensemble des feuilles F_x attachées à un noeud x de T forme une clique de G, qui n'est adjacente

à aucun sommet correspondant à une feuille attachée à un noeud y de T, avec $x \neq y$. Il suit qu'un graphe G est un 2-leaf power si et seulement si G est une union disjointe de cliques.

Remarque. Par ce qui précède, le problème 2-LEAF POWER EDITION est équivalent au problème CLUSTER EDITING étudié dans le Chapitre 2.

Nous donnons maintenant une caractérisation des 3-leaf powers en termes de sous-graphes induits interdits.

Théorème 3.3 (Sous-graphes induits [24, 26])**.** *Un graphe $G := (V, E)$ est un* 3-leaf power *si et seulement s'l ne contient pas de* gem, dart, bull *ou* hole *comme sous-graphe induit. (voir Figure 3.2)*

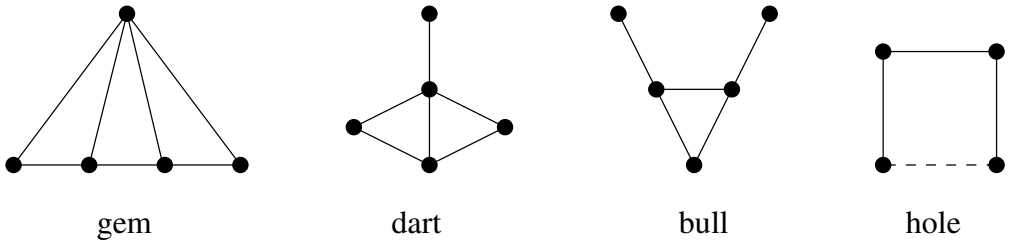


FIGURE 3.2 : Les sous-graphes induits interdits caractérisant la classe des 3-leaf powers.

3.1.1 Un outil combinatoire approprié : les cliques critiques

Nous établissons maintenant une corrélation entre la notion de cliques critiques (définie dans le Chapitre 1) et les 3-leaf powers. L'intérêt de l'utilisation de cet outil combinatoire dans l'étude des *leaf powers* provient de la remarque suivante :

Remarque. Soient $G := (V, E)$ un *p-leaf power* et T son *p-leaf root* associé. Soient x un noeud interne de T et F_x l'ensemble des feuilles attachées à x. Alors F_x définit une clique module de G (à condition que $p \geqslant 2$) : en effet, toute paire u, v appartenant à F_x est à distance 2 dans T et correspond donc à une arête de G. De plus, u et v sont à égale distance de toute feuille $w \notin F_x$, impliquant que l'ensemble correspondant à F_x dans G définit une clique ayant le même voisinage. En supposant sans perte de généralité que le modèle T est *optimal* (*i.e.* les vrais jumeaux de G sont attachés au même noeud interne de T), F_x définit alors une clique module *maximale, i.e.* une clique critique.

La notion de cliques critiques permet en particulier d'obtenir une caractérisation simple pour les 3-leaf powers, formulée comme suit.

Théorème 3.4 ([24, 26])**.** *Un graphe $G := (V, E)$ est un* 3*-leaf power si et seulement si son graphe des cliques critiques $\mathscr{C}(G)$ est une forêt.*

Rappelons que le graphe des cliques critiques $\mathscr{C}(G)$ d'un graphe $G := (V, E)$ peut-être vu comme un sous-graphe de G induit en conservant un *unique sommet* par clique critique de G. Ainsi, le graphe des cliques critiques de G contient toute l'*information structurelle* de G. En particulier, nous utiliserons le Théorème 3.4 de manière extensive afin d'élaborer nos règles de réduction.

Décomposition d'adjacence. La notion de cliques critiques et le Théorème 3.4 permettent de définir une décomposition d'adjacence pour les 3-leaf powers. En effet, soit $G := (V, E)$ un 3-leaf power. Nous posons $\mathcal{V} := \mathscr{C}_G$ (*i.e.* $\mathcal{V}$ correspond aux cliques critiques de G), et la relation d'adjacence $P_{\mathscr{G}}$ est définie par le graphe des cliques critiques $\mathscr{C}(G)$.

Join-composition. Nous donnons maintenant une caractérisation des 3-leaf powers qui sera utile pour prouver la validité de nos règles de réduction. Cette dernière est basée sur la notion de *join-composition*, qui est une généralisation de l'application Série définie dans le Chapitre 1. Étant donnés deux graphes disjoints $G_1 := (V_1, E_1)$ et $G_2 := (V_2, E_2)$, et $S_1 \subseteq V_1$, $S_2 \subseteq V_2$, la *join-composition* de G_1 et G_2 sur S_1 et S_2, dénotée $(G_1, S_1) \otimes (G_2, S_2)$ est le graphe $H := (V_1 \cup V_2, E_1 \cup E_2 \cup (S_1 \times S_2))$ (voir Figure 3.3).

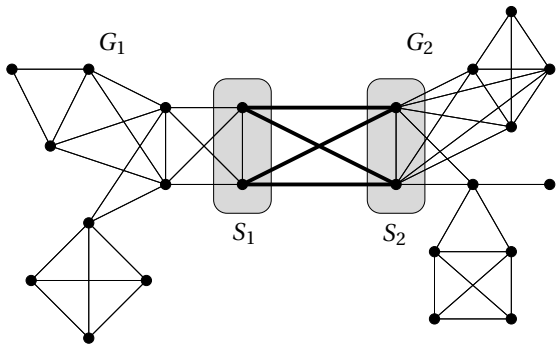


FIGURE 3.3 : Illustration de la notion de *join-composition* entre deux graphes G_1 et G_2.

Lemme 3.5 ([12])**.** *Soient $G_1 := (V_1, E_1)$ et $G_2 := (V_2, E_2)$ deux 3-leaf powers connexes disjoints. Le graphe $H := (G_1, S_1) \otimes (G_2, S_2)$, avec $S_1 \subseteq V_1$, $S_2 \subseteq V_2$ est un 3-leaf power si et seulement si une des conditions suivantes est vérifiée :*

(i) S_1 et S_2 sont des cliques de G_1 et G_2, respectivement, et si S_1 (resp. S_2) n'est pas une clique critique alors G_1 (resp. G_2) est une clique ;

(ii) il existe un sommet $v \in V_1$ tel que $S_1 := N_{G_1}[v]$ et G_2 est une clique.

Plus particulièrement, nos preuves utiliseront le fait que (i) implique que le graphe $H := (G_1, S_1) \otimes (G_2, S_2)$ est un 3-leaf power. Pour conclure cette partie, nous prouvons une observation qui sera utile dans l'élaboration de nos règles de réduction.

Observation 3.6. *Soient $G := (V, E)$ un 3-leaf power et $C \subseteq V$ une clique critique de G. Pour tout ensemble $S \subseteq V$, $S \neq C$, si la clique induite par $C \setminus S$ n'est pas critique dans $G[V \setminus S]$, alors la composante connexe de $G[V \setminus S]$ contenant $C \setminus S$ est une clique.*

Preuve. Supposons que la clique $C \setminus S$ ne soit pas une clique critique de $G[V \setminus S]$. En d'autres termes, puisque $C \setminus S$ est une clique module de $G[V \setminus S]$, $C \setminus S$ n'est pas maximale. Soit $u \in V \setminus S$ tel que $(C \setminus S) \cup \{u\}$ est une clique module de $G[V \setminus S]$. Par définition, u appartient à une clique critique C' adjacente à C dans G. Puisque G est un 3-leaf power, le Théorème 3.4 implique que $\mathscr{C}(G)$ est

une forêt. Ainsi, S doit contenir l'ensemble des cliques critiques de G adjacente à C' (sauf C), et également l'ensemble des cliques critiques de G adjacentes à C (sauf C'). Cela implique donc que la composante connexe de $G[V \setminus S]$ contenant $C \setminus S$ est un sous-ensemble de $C \cup C'$, *i.e.* une clique. □

3.1.2 Branches

Structures à réduire. Nous allons maintenant définir la notion de *branches* pour le problème CLOSEST 3-LEAF POWER. Pour cela, nous décrivons quels types de branches devront être réduits pour obtenir un algorithme de noyau polynomial pour ce problème. Pour ce faire, considérons une instance positive (G,k) de CLOSEST 3-LEAF POWER. Soient F une k-édition de G et $H := G \triangle F$ le 3-leaf power correspondant. Par le Théorème 3.4, nous savons que $\mathscr{C}(H)$ est une forêt. Par souci de simplicité, nous supposons ici qu'il s'agit d'un arbre. Considérons la collection $\mathscr{A}$ des cliques critiques de $\mathscr{C}(H)$ contenant un sommet affecté. Par définition de F, nous avons $|\mathscr{A}| \leqslant 2k$. Nous considérons également un arbre minimal T couvrant les cliques critiques contenues dans $\mathscr{A}$ dans $\mathscr{C}(H)$. Observons qu'une clique critique non affectée C de $T \setminus \mathscr{A}$ associée à l'ensemble de sommets *pendants* à C dans $\mathscr{C}(H) \setminus T$ définit une branche de G, dans la mesure où cet ensemble induit un 3-leaf power dont une seule clique critique (à savoir C) est connectée au reste du graphe. Remarquons également qu'un ensemble de sommets *connexe* de $\mathscr{C}(H) \setminus T$ *pendants* à une clique critique *affectée* de $\mathscr{A}$ définit une telle branche. De plus, en supprimant les sommets de $\mathscr{A}$ ainsi que les sommets de degré au moins 3 (dont le nombre est borné par $2k$ également) de cet arbre, les composantes connexes obtenues sont composées de cliques critiques non affectées et induisent donc des 3-leaf powers. Par construction, ces composantes connexes correspondent donc à une branche de G dont la frontière peut être partitionnée en deux cliques critiques. Ainsi, réduire les branches où la frontière consiste en une ou deux cliques critiques est suffisant pour obtenir un noyau polynomial. Nous verrons par la suite que borner le nombre de sommets contenus dans le graphe $\mathscr{C}(H)$ permet également de borner le nombre de sommets contenus dans $\mathscr{C}(G)$, la Règle 2.7 permettant de réduire les cliques critiques pouvant être appliquée pour ce problème (Observation 3.9).

Branches. Nous définissons maintenant formellement la notion de *branches* pour le problème CLOSEST 3-LEAF POWER.

Définition 3.7 (branches)**.** *Soient (G,k) une instance de* CLOSEST 3-LEAF POWER *et $B \subseteq V$. L'ensemble B est une* branche *de G si B est l'union de cliques critiques $\{C_1,\dots,C_r\}$ telles que le sous-graphe de $\mathscr{C}(G)$ induit par $\{C_1,\dots,C_r\}$ est un arbre.*

Définition 3.8 (l-branches)**.** *Soient (G,k) une instance de* CLOSEST 3-LEAF POWER *et $B \subseteq V$ une branche de G composée des cliques critiques $\{C_1,\dots,C_r\}$. Une clique critique C_i, $i \in [r]$ est un* point d'attachement *de B si $N_{G\setminus B}(C_i) \neq \emptyset$. Une branche contenant l points d'attachement est appelée une l-branche.*

1-branche
2-branche

Comme expliqué précédemment, nous nous intéressons uniquement aux 1-branches et aux 2-branches pour concevoir nos règles de réduction (voir Figure 3.4). Dans le reste de ce chapitre, le point d'attachement d'une 1-branche sera dénoté A_1, ceux d'une 2-branche A_1 et A_2, respectivement. De plus, nous posons $B^R := B \setminus A_1$ dans le cas d'une 1-branche, et $B^R := B \setminus (A_1 \cup A_2)$ dans le cas d'une 2-branche.

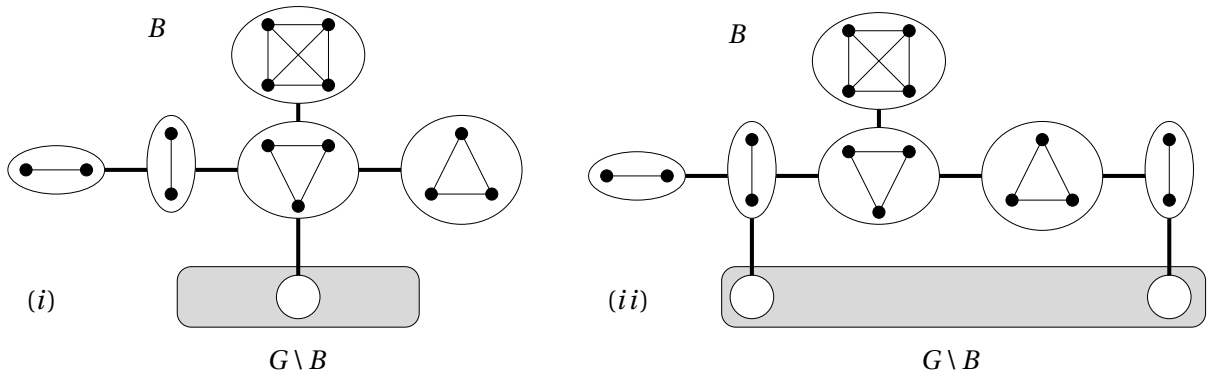


FIGURE 3.4 : (i) Une 1-branche dans une instance (G,k) de CLOSEST 3-LEAF POWER ; (ii) Une 2-branche dans une instance (G,k) de CLOSEST 3-LEAF POWER.

3.2 Règles de réduction via la notion de *branches*

Dans un premier temps, nous démontrons que les Règles 2.5 et 2.7 présentées Section 2.3.5 s'appliquent sur le problème CLOSEST 3-LEAF POWER. Nous rappelons l'énoncé de ces règles de réduction. Conformément aux résultats présentés dans la Section 2.3.5 (Lemmes 2.46 et 2.50), la validité de ces règles de réduction est due à l'Observation suivante.

Observation 3.9. *La classe des p-leaf powers est héréditaire, close par union disjointe et par ajout de vrai jumeau.*

Preuve. Soient $G := (V,E)$ un p-leaf power et T son p-leaf root. Premièrement, observons que tout sous-graphe induit de $G[S]$, $S \subseteq V$, est également un p-leaf power, son p-leaf root correspondant à l'arbre T où les feuilles correspondantes à $V \setminus S$ ont été supprimées. Soient $G' := (V',E')$ un p-leaf power et T' son p-leaf root. Le graphe $G \oplus G'$ est également un p-leaf power : en effet, l'arbre obtenu en reliant T et T' sur deux noeuds internes quelconques par un chemin de longueur p est un p-leaf root de $G \oplus G'$. Finalement, si un vrai jumeau u' est ajouté à un sommet u de G, alors l'arbre obtenu à partir de T en ajoutant une feuille correspondant à u' attachée au père de u est un p-leaf root pour $G \cup \{u'\}$. □

Règle 3.1 (Composante connexe)**.** *Soit (G,k) une instance de* CLOSEST 3-LEAF POWER. *Supprimer toute composante connexe C de G telle que $G[C]$ est un 3-leaf power.* Règle 2.5

Règle 3.2 (Vrai jumeau)**.** *Soient $G := (V,E)$ un graphe et C une clique critique de G telle que $|C| > k+1$. Supprimer $|C| - (k+1)$ sommets arbitraires de C.* Règle 2.7

Lemme 3.10. *Les Règles 3.1 et 3.2 sont valides et peuvent être appliquées en temps polynomial.* Lemmes 2.46 et 2.50

Par l'Observation 3.9, nous savons que la classe des 3-leaf powers vérifie les conditions du Lemme 2.49. Ainsi, il existe toujours une édition (resp. complétion, suppression) optimale qui, étant données deux cliques critiques C et C' de G, édite toute ou aucune arête entre C et C'. Dans la suite

de ce chapitre, nous assumons que toute édition (resp. complétion, suppression) considérée vérifie cette propriété.

Remarque. La Règle 3.2 est un cas particulier de règle de réduction via la notion de branche : en effet, une clique critique C d'une instance (G,k) de CLOSEST 3-LEAF POWER est une branche particulière, composée d'un unique point d'attachement.

Nous décrivons maintenant plusieurs règles de réduction nous permettant de réduire les 1-branches (Section 3.2.1) et les 2-branches (Section 3.2.2) d'une instance de CLOSEST 3-LEAF POWER.

3.2.1 Couper les 1-branches

Dans un premier temps, nous établissons un résultat permettant de déterminer quels sommets d'une 1-branche sont affectés par une édition optimale. Ce résultat nous permettra par la suite d'énoncer une règle réduisant la taille des 1-branches d'une instance de CLOSEST 3-LEAF POWER.

Lemme 3.11. *Soient (G,k) une instance de* CLOSEST 3-LEAF POWER *et $B \subseteq V$ une 1-branche de G. Il existe une édition optimale F telle que :*

(i) l'ensemble des sommets affectés de B est inclus dans $A_1 \cup N_B(A_1)$,

(ii) F ne contient aucune paire de la forme $(B^R \times B^R)$ et,

(iii) dans le graphe $H := G \triangle F$, les sommets de $N_B(A_1)$ sont adjacents aux mêmes sommets *de $V \setminus B^R$.*

Preuve. Soit F une édition optimale de G. Nous construisons à partir de F une nouvelle édition optimale F' respectant les propriétés du Lemme 3.11.

Soit A'_1 la clique critique de $H := G \triangle F$ contenant A_1. Rappelons que A'_1 est bien définie par le Lemme 2.49. Puisque F contient toute ou aucune paire entre deux cliques critiques, l'ensemble des cliques critiques $\{C_1,\dots,C_l\}$ dont les sommets appartiennent à $N_B(A_1)$ peut être partitionné en deux : les cliques critiques $\{C_1,\dots,C_c\}$ contenues dans A'_1 ou adjacentes à A'_1 dans H, et les cliques critiques $\{C_{c+1},\dots,C_l\}$ non adjacentes à A'_1 dans H. Pour tout $i \in [l]$, nous notons CC_i la composante connexe de $G[B^R]$ contenant C_i. Nous considérons les trois graphes suivants : G_1, le sous-graphe de G induit par $\{CC_1,\dots,CC_c\}$; G_2, le sous-graphe de G induit par $\{CC_{c+1},\dots,CC_l\}$; et enfin G_3, le sous-graphe de H induit par les sommets de $V(G) \setminus B^R$ (voir Figure 3.5). Puisque la classe des 3-leaf powers est héréditaire, ces trois graphes sont des 3-leaf powers.

$A''_1 := A'_1 \setminus B^R$

Maintenant, nous savons par l'Observation 3.6 que si $A''_1 := A'_1 \setminus B^R$ n'est pas une clique critique de G_3, alors la composante connexe contenant A''_1 dans G_3 est une clique. De même, pour tout $i \in [c]$, si C_i n'est pas une clique critique de G_1 alors CC_i induit une clique. Par le Lemme 3.5, il suit que le graphe $H' := G_2 \oplus G'$, où $G' := (G_3, A''_1) \otimes (G_1, \{C_1,\dots,C_c\})$ est un 3-leaf power (voir Figure 3.5). Par construction, l'ensemble F' tel que $H' = G \triangle F'$ est un sous-ensemble de F, impliquant $|F'| \leqslant |F|$. De plus, les sommets de B affectés par F' sont contenus dans $A_1 \cup N_B(A_1)$ et F ne contient aucune paire de la forme $(B^R \times B^R)$, ce qui prouve (i) et (ii).

Nous considérons maintenant une édition optimale F vérifiant (i) et (ii). Pour démontrer (iii), nous étudions la relation entre les cliques C_i et A''_1 dans le graphe $H := G \triangle F$. Supposons qu'il existe $i \in [c]$: cela signifie que le nombre d'arêtes manquantes entre C_i et A''_1 dans G est inférieur ou égal

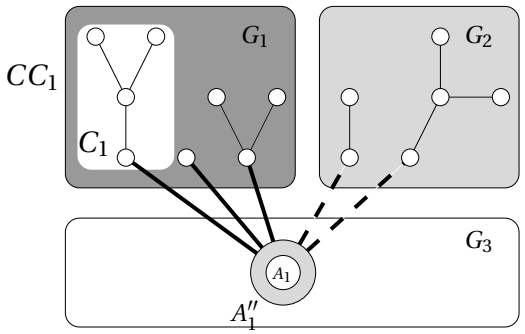


FIGURE 3.5 : Le graphe $H' := G_2 \oplus ((G_3, A_1'') \otimes (G_1, \{A_1, \dots, C_c\}))$ est un 3-leaf power par le Lemme 3.5.

au nombre d'arêtes présentes entre C_i et A_1 dans G (dont la suppression permettrait d'obtenir un 3-leaf power par le Lemme 3.5). Il suit que $|C_i| \cdot |A_1'' \setminus A_1| \leqslant |C_i| \cdot |A_1|$. De même, s'il existe $c < i \leqslant l$, alors $|A_1| \leqslant |A_1'' \setminus A_1|$. Ainsi, si les deux cas sont vérifiés, cela implique que $|A_1| = |A_1'' \setminus A_1|$, et, pour tout $i \in [l]$, ajouter ou supprimer toutes les arêtes entre C_i et A_1'' requiert le même nombre d'éditions. De ce fait, il est possible de modifier F' de sorte que les sommets de $N_B(A_1)$ aient le même voisinage dans H'. □

Règle 3.3 (1-branche)**.** *Soient* (G, k) *une instance de* CLOSEST 3-LEAF POWER *et* $B \subseteq V$ *une* 1*-branche de* G*. Supprimer de* G *les sommets de* B^R *et ajouter une nouvelle clique critique* C_1 *voisine de* A_1*, de taille* $\min\{|N_B(A_1)|, k+1\}$.

Nous démontrons dans un premier temps que cette règle de réduction est valide. Nous verrons dans la section suivante comment l'appliquer en temps polynomial.

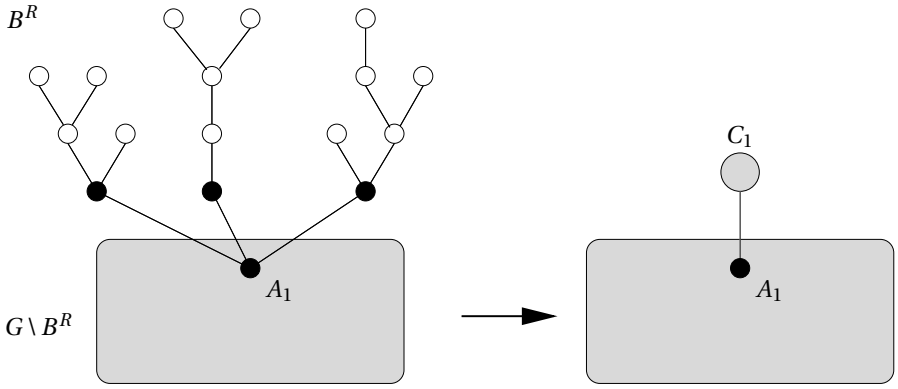


FIGURE 3.6 : Illustration de la Règle 3.3.

Lemme 3.12. *La Règle 3.3 est valide.*

Preuve. Nous dénotons par G' le graphe obtenu à partir de G en supprimant les sommets de B^R et en ajoutant une nouvelle clique critique C_1 voisine de A_1, de taille $\min\{|N_B(A_1)|, k+1\}$. Soit F une

k-édition de G. Par le Lemme 3.11, nous pouvons supposer que les seuls sommets de B affectés par F sont contenus dans $A_1 \cup N_B(A_1)$. De plus, tous les sommets de $N_B(A_1)$ ont le même voisinage dans $H \setminus B^R$, $H := G \triangle F$, et sont donc édités de la même manière par F. En particulier, si $|N_B(A_1)| > k$, aucun sommet de $N_B(A_1)$ ne sera affecté par F, et le graphe $((G \setminus B^R \triangle F), A_1) \otimes (C_1, C_1)$ est un 3-leaf power (Lemme 3.5), impliquant que F est une k-édition de G'. Dans le cas inverse, remplacer les éditions de F de la forme $(N_B(A_1) \times V \setminus N_B(A_1))$ par $(C_1 \times V \setminus C_1)$ permet d'obtenir une k-édition pour G'. Inversement, soit F' une k-édition pour G'. Soit A'_1 la clique critique contenant A_1 dans $H' := G' \triangle F'$. Si F' n'affecte pas les sommets de C_1, alors le graphe H obtenu à partir de H' en supprimant C_1 et en faisant une join-composition entre $(G[B^R], N_B(A_1))$ et $(H' \setminus C_1, A'_1)$ est un 3-leaf power par le Lemme 3.5. Observons que les arêtes ainsi ajoutées sont présentes dans G puisque, dans ce cas, $A_1 = A'_1$. Dans le cas contraire, F' édite tous les sommets de C_1 de la même manière, et une join-composition entre $(G[B^R], N_B(A_1))$ et $(H' \setminus C_1, A'_1)$ permet d'obtenir un 3-leaf power où les éditions de la forme $(C_1 \times V \setminus C_1)$ ont été remplacées par $(N_B(A_1) \times V \setminus N_B(A_1))$. □

Nous considérons maintenant le cas où plusieurs 1-branches sont attachées au graphe sur le même voisinage.

Lemme 3.13. *Soit (G, k) une instance positive de* CLOSEST 3-LEAF POWER. *Soit $\{B_1, \ldots, B_r\}$ un ensemble de 1-branches dont les points d'attachement $\{A_1^1, \ldots, A_1^r\}$ ont le même voisinage N dans $V \setminus \cup_{i=1}^r B_i$. Si $r > 2k+1$, alors il existe une k-édition F telle que :*

(i) N est une clique critique de $H := G \triangle F$ et,

(ii) aucun sommet contenu dans $\cup_{i=1}^r B_i$ n'est affecté.

Preuve. Nous prouvons tout d'abord (i) par contradiction. Soit F une k-édition de G. Dans un premier temps, supposons que N ne soit pas une clique dans $H := G \triangle F$, *i.e.* qu'il existe deux sommets $u, v \in N$ tels que $uv \notin E(H)$. Pour toute paire de sommets $v_i \in A_1^i$, $v_j \in A_1^j$, avec $i \neq j$, l'ensemble $\{u, v, v_i, v_j\}$ ne peut pas induire un cycle sans corde dans H. Comme $uv \notin E(H)$, cela signifie que F affecte les sommets de A_1^i ou A_1^j. Il suit qu'au plus un point d'attachement n'est pas affecté par F. Comme $r > 2k+1$, F contient au moins $k+1$ paires : contradiction.

Supposons maintenant que N ne soit pas un module dans H. Cela implique qu'il existe $w \notin N$ et $u, v \in N$ tels que, sans perte de généralité, $uw \in E(H)$ et $vw \notin E(H)$. Comme $|F| \leqslant k$ et $r > 2k+1$, il existe deux sommets $v_i \in A_1^i$ et $v_j \in A_1^j$, $i \neq j$, non affectés par F (rappelons que F affecte au plus $2k$ sommets, donc au plus $2k$ points d'attachement). Par construction, l'ensemble $\{u, v, w, v_i, v_j\}$ induit alors un *dart* dans H, contredisant le Théorème 4.3.

Finalement, nous montrons que N est une clique critique de H. Supposons que ce ne soit pas le cas. Puisque N est une clique module de H par les arguments précédents, il suit qu'il existe un sommet $u \in V(G) \setminus (\cup_{i=1}^r B_i \cup N)$ tel que $N \cup \{u\}$ est une clique module de H. Comme $|F| \leqslant k$ et $r > 2k+1$, N est adjacent à au moins $k+1$ sommets de $\cup_{i=1}^r B_i$ dans H, ce qui implique que u est adjacent à au moins $k+1$ sommets de $\cup_{i=1}^r B_i$ dans H. Il suit que $|F| > k$, ce qui est impossible.

Pour conclure la preuve du Lemme 3.13, nous prouvons (ii). Pour cela, nous procédons comme dans la preuve du Lemme 3.11 : soient G_1 le graphe induit par $\cup_{i=1}^r B_i$ dans G et G_2 le graphe induit par $V(G) \setminus \cup_{i=1}^r B_i$ dans H. Comme la classe des 3-leaf powers est héréditaire, G_1 et G_2 sont des 3-leaf powers. De plus, comme N est une clique critique de H, si N n'est pas une clique critique dans G_2 alors la composante connexe contenant N dans G_2 est une clique. Par le Lemme 3.5, il suit

que le graphe $H' := (G_1, \{A_1^1, \ldots, A_1^r\}) \otimes (G_2, N)$ est un 3-leaf power. Par construction, l'ensemble F' tel que $H' := G \triangle F'$ est un sous-ensemble de F, impliquant $|F'| \leqslant |F|$. De plus, aucun sommet de $\cup_{i=1}^r B_i$ n'est affecté : en effet, les sommets de $\cup_{i=1}^r A_1^i$ sont adjacents à N dans G et les éditions de F incidentes aux sommets de $\cup_{i=1}^r B_i$ ne sont pas conservées dans F'. □

Par le Lemme 3.13, nous savons que s'il existe une k-édition pour une instance (G,k) de CLOSEST 3-LEAF POWER, alors préserver seulement $2k+2$ points d'attachement permet d'obtenir une règle de réduction valide.

Règle 3.4. *Soient (G,k) une instance de* CLOSEST 3-LEAF POWER, *et $\{B_1, \ldots, B_r\}$ un ensemble de 1-branches dont les points d'attachement $\{A_1^1, \ldots, A_1^r\}$ ont le même voisinage N dans $V \setminus \cup_{i=1}^r B_i$. Si $r > 2k+2$, alors supprimer de G les sommets de $\cup_{i=2k+3}^r B_i$.*

Lemme 3.14. *La Règle 3.4 est valide.*

Preuve. Nous dénotons par G' le graphe obtenu à partir de G en supprimant les sommets de $\cup_{i=2k+3}^r B_i$. Soient F une k-édition de G et $H := G \triangle F$. Par le Lemme 3.13, nous pouvons supposer que F n'affecte aucun sommet de $\cup_{i=1}^r B_i$. Ainsi, le graphe $G' \triangle F$ est un sous-graphe induit de H. Comme la classe des 3-leaf powers est héréditaire, il suit que F est une k-édition de G'. Inversement, soit F' une k-édition de G'. Par le Lemme 3.13, nous savons qu'aucun sommet de $\cup_{i=1}^{2k+2} B_i$ n'est affecté par F'. Ainsi, le voisinage des points d'attachement $\{A_1^1, \ldots, A_1^{2k+2}\}$ est exactement N dans H'. Comme de plus N est une clique critique de H', il suit que réaliser une join-composition entre (H', N) et $(G[\cup_{i=2k+3}^r B_i], \{A_1^{2k+3}, \ldots, A_1^r\})$ permet d'obtenir un 3-leaf power H (Lemme 3.5), construit à partir de G en éditant toutes les paires contenues dans F'. □

Remarque. Lors de l'application de la Règle 3.4, un ensemble d'exactement $2k+1$ branches est conservé. Dans la suite, nous utiliserons la Règle 3.3 afin de borner la taille des branches restantes, et ainsi borner le nombre de sommets contenus après application de la Règle 3.4.

3.2.2 Couper les 2-branches

Nous démontrons maintenant comment réduire les 2-*branches* d'une instance (G,k) de CLOSEST 3-LEAF POWER. Soit $B \subseteq V$ une 2-branche de G. Nous disons que B est *propre* si A_1 et A_2 sont des feuilles du graphe $\mathscr{C}(G[B])$, et dénotons par P_1 et P_2 les cliques critiques voisines de A_1 et A_2 dans $\mathscr{C}(G[B])$. De plus, nous dénotons par $path(B) := \{P_1, \ldots, P_2\}$ l'unique chemin de $\mathscr{C}(G[B])$ reliant les points d'attachement A_1 et A_2. Une *coupe de taille minimum* de $path(B)$ est un ensemble $F \subseteq E$ de taille minimum tel que $G[B] \setminus F$ ne contient aucun chemin reliant A_1 à A_2. Observons en particulier que $F = (P_i \times P_{i+1})$ pour une certaine clique critique P_i contenue dans $path(B)$.

Lemme 3.15. *Soient (G,k) une instance de* CLOSEST 3-LEAF POWER *et $B \subseteq V$ une 2-branche propre de G telle que $path(B)$ contient au moins 5 cliques critiques. Il existe une édition optimale F telle que :*

(i) si $path(B)$ est un sous-graphe non connexe de $G \triangle F$, alors F peut contenir les arêtes correspondant à une coupe minimum de $path(B)$;

(ii) dans tous les cas, les autres sommets affectés de B appartiennent à $A_1 \cup P_1 \cup P_2 \cup A_2$.

Preuve. Soient F une édition optimale et $H := G \triangle F$. Nous construisons une édition F' telle que $|F'| \leqslant |F|$ et vérifiant les conditions du Lemme 3.15. Pour cela, nous étudions le comportement du chemin $path(B)$ dans H. Nous notons par A'_1 et A'_2 les cliques critiques de H qui contiennent A_1 et A_2, respectivement. Remarquons que le cas $A'_1 = A'_2$ est possible, mais peut être considéré de manière similaire. De plus, nous posons $A''_1 := A'_1 \setminus B^R$ et $A''_2 := A'_2 \setminus B^R$ (voir Figure 3.9).

$A''_i := A'_i \setminus B^R$

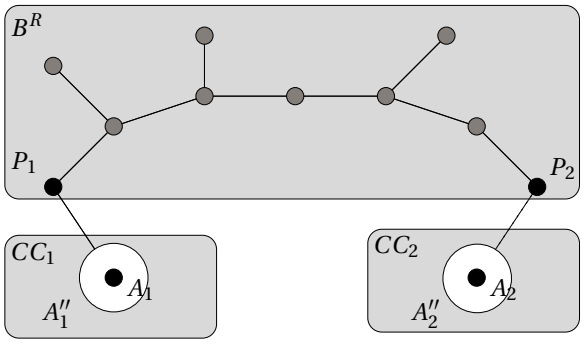


FIGURE 3.7 : Illustration du **Cas** 2 lorsque A_1 et A_2 appartiennent à différentes composantes connexes CC_1 et CC_2.

L'observation suivante nous permettra de construire la solution optimale désirée dans plusieurs cas.

Observation 3.16 (Classique)**.** *Soient F une édition optimale de G et $F_1 \subseteq F$. Si F_2 est une édition optimale du graphe $G \triangle F_1$, alors $F_1 \cup F_2$ est une édition optimale de G.*

Cas 1. F **déconnecte** $path(B)$**.** Supposons dans un premier temps que F contienne l'ensemble d'arêtes $F_1 := (A_1 \times P_1)$, et considérons le graphe $H_1 := G \triangle F_1$. Remarquons que $B_1 := B \setminus A_1$ est une 1-branche de P_1, ayant pour point d'attachement A_2. En utilisant le Lemme 3.11, nous savons qu'il existe une édition optimale F_2 pour H_1 telle que les seuls sommets affectés de B_1 sont contenus dans $A_2 \cup P_2$. Par l'Observation 3.16, il suit que $F' := F_1 \cup F_2$ est une édition optimale de G respectant les conditions du Lemme 3.15. Nous procédons de manière similaire lorsque F contient $F_2 := (A_2 \times P_2)$.

$(A_1 \times P_1) \subseteq F$

Supposons maintenant que F ne contienne ni $(A_1 \times P_1)$ ni $(A_2 \times P_2)$. Dans ce cas, par hypothèse, il existe un ensemble $F_1 \subseteq F$ correspondant à une coupe *minimale* de $path(B)$, disjoint de $(A_1 \times P_1)$ et $(A_2 \times P_2)$. Comme précédemment, nous considérons le graphe $H_1 := G \triangle F_1$. Par construction, B se décompose en deux 1-branches dans H_1 : B_1 contenant A_1 comme point d'attachement et B_2 contenant A_2 comme point d'attachement. Comme précédemment, le Lemme 3.11 s'applique sur B_1 et B_2, permettant d'obtenir une édition optimale F_2 de H_1 dont les seuls sommets affectés de B_1 et B_2 sont contenus dans $A_1 \cup P_1 \cup P_2 \cup A_2$. Ainsi, $F' := F_1 \cup F_2$ est une k-édition respectant les propriétés désirées. En effet, si F_1 ne correspond pas à une coupe de taille *minimum* dans $path(B)$, alors l'édition $F' := (F \setminus F_1) \cup F_c$ où F_c correspond à une coupe minimum de $path(B)$ est une édition optimale vérifiant $|F'| < |F|$, contredisant l'optimalité de F.

$(A_1 \times P_1) \cap F \neq \emptyset$

Cas 2. F ne déconnecte pas $path(B)$. Soient CC_1 et CC_2 les composantes connexes de $H \setminus B^R$ contenant A_1 et A_2, respectivement. Nous considérons tout d'abord le cas où CC_1 et CC_2 sont deux composantes connexes distinctes de $H \setminus B^R$. Par définition de la 2-branche B, nous savons que $G[B^R]$ est un 3-leaf power. De plus, comme $path(B)$ contient au moins 5 cliques critiques par hypothèse, nous savons que P_1 et P_2 sont deux cliques critiques de $G[B^R]$. Par définition de H, les sous-graphes $H[CC_1]$ et $H[CC_2]$ définissent également des 3-leaf powers, qui sont des cliques si A_1'' (resp. A_2'') ne sont pas des cliques critiques (Observation 3.6). Par le Lemme 3.5, il suit que réaliser la join-composition suivante : $H' := (H[CC_1], A_1'') \otimes (G[B^R], P_1)$ et $(H', P_2) \otimes (H[CC_2], A_2'')$ permet d'obtenir un 3-leaf power. Observons que durant cette-join composition, aucune paire non contenue dans F n'a été éditée : en effet, puisque $(A_1 \times P_1) \subseteq E(H)$ par hypothèse, nous avons $(A_1'' \times P_1) \subseteq E(H)$. Le même argument est valable pour A_2'' et P_2, impliquant $(A_2'' \times P_2) \subseteq E(H)$. De ce fait, si F affecte des sommets contenus dans $B^R \setminus (P_1 \cup P_2)$, alors supprimer ces arêtes de F permet d'obtenir une édition de taille strictement plus petite, contredisant l'optimalité de F. Finalement, supposons que A_1 et A_2 appartiennent à la même composante connexe CC de $H \setminus B^R$. Soient a_1 et a_2 deux sommets de A_1' et A_2', respectivement. Dans le cas où $A_1' = A_2'$, nous supposons $a_1 = a_2$. Considérons maintenant deux chemins π_B et π_{CC} reliant a_1 et a_2 dans H comme suit : π_B est obtenu en prenant un sommet p_i pour toute clique critique P_i de $path(B)$ (ce qui définit bien un chemin vu que $path(B)$ est connecté dans H), et π_{CC} est un plus court chemin entre a_1 et a_2 dans $H[CC]$. Par construction, les sommets $\{c_1, \ldots, c_{|\pi_{CC}|}\}$ appartenant à π_{CC} sont contenus dans différentes cliques critiques de H, disons $\{C_1, \ldots, C_{|\pi_{CC}|}\}$, avec $C_1 = A_1'$ et $C_{|\pi_{CC}|} = A_2'$. Ainsi, l'union de ces deux chemins permet d'obtenir un cycle C de H (non-induit) ayant une longueur supérieure ou égale à 5. Nous démontrons maintenant le résultat suivant, qui nous permettra de conclure

Affirmation 3.17. *Soit $G := (V, E)$ un 3-leaf power. Tout cycle C de longueur au moins 5 dans G contient quatre sommets distincts $\{a, b, c, d\}$ (apparaissant dans cet ordre dans C), avec $ab, cd \in E(C)$, tels que $ad \in E$, $ac \in E$ et $bd \in E$.*

Preuve. Comme la classe des 3-leaf powers est héréditaire (Observation 3.9), le sous-graphe G_C induit par C dans G est un 3-leaf power contenant au moins 5 sommets. Comme G_C n'est pas un arbre, il contient une clique critique P de taille supérieure ou égale à 2. Soient a et d deux sommets distincts de P. Comme $|C| \geqslant 5$, observons qu'il existe deux sommets distincts b et c, également distincts de a et d, tels que a, b, c et d apparaissent dans cet ordre dans C. De plus, ces sommets sont tels que ab et cd sont des arêtes de G_C. Comme P est une clique critique, il suit que $ad \in E$, $ac \in E$ et $bd \in E$. ◇

Comme H est un 3-leaf power, l'Affirmation 3.17 implique qu'il existe deux arêtes $e := ab$ et $f := cd$ de C telles que les arêtes ac et bd appartiennent à $E(H)$. Par construction, au plus une des arêtes e et f appartient à π_{CC} (sinon π_{CC} ne serait pas un plus court chemin entre a_1 et a_2). Nous étudions les différentes configurations possibles afin de conclure.

(*a*) *Les arêtes e et f appartiennent à π_B.* Sans perte de généralité, nous supposons $a = p_i$, $b = p_{i+1}$, $c = p_j$ et $d = p_{j+1}$, avec $i+1 < j$. Par l'Affirmation 3.17, F contient les paires $(P_i \times P_j) \cup (P_{i+1} \times P_{j+1})$. Observons maintenant que $\min\{|P_i| \cdot |P_{i+1}|, |P_j| \cdot |P_{j+1}|\} < |P_i| \cdot |P_j| + |P_{i+1}| \cdot |P_{j+1}|$, et supposons sans perte de généralité que $\min\{|P_i| \cdot |P_{i+1}|, |P_j| \cdot |P_{j+1}|\} = |P_i| \cdot |P_{i+1}|$. Nous modifions F en *coupant*

les arêtes entre P_i et P_{i+1}. Soit :

$$F' := (F \setminus (V \times B^R)) \cup (P_i \times P_{i+1})$$

De plus, si $P_i \neq A_1$ (resp. $P_{i+1} \neq A_2$) nous ajoutons à F' les paires $F_1 := ((A_1'' \setminus A_1) \times P_1) \subseteq F$ (resp. $F_2 := ((A_2'' \setminus A_2) \times P_2) \subseteq F$). Dans tous les cas, nous avons $|F'| < |F|$ par construction (voir Figure 3.8). En utilisant à nouveau le Lemme 3.5, nous déduisons que le graphe $G \triangle F'$ est un 3-leaf power, contredisant l'optimalité de F.

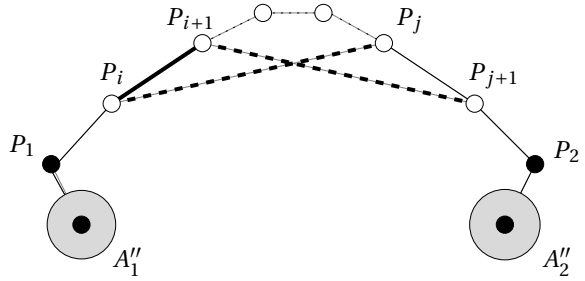


FIGURE 3.8 : Illustration du cas (*a*). L'ensemble d'arêtes en gras correspond aux arêtes que nous allons supprimer.

(*b*) *L'arête e appartient à* π_B *et f à* π_{CC}. Sans perte de généralité, nous supposons $a = p_i$, $b = p_{i+1}$, $c = c_{j+1}$, $d = c_j$. Comme précédemment, par l'Affirmation 3.17, F contient les paires $(P_i \times C_{j+1}) \cup (P_{i+1} \times C_j)$. De nouveau, observons que $\min\{|P_i| \cdot |P_{i+1}|, |C_j| \cdot |C_{j+1}|\} < |P_i| \cdot |C_{j+1}| + |P_{i+1}| \cdot |C_j|$, et supposons dans un premier temps que $\min\{|P_i| \cdot |P_{i+1}|, |C_j| \cdot |C_{j+1}|\} = |P_i| \cdot |P_{i+1}|$. Nous considérons alors l'ensemble :

$$F' := (F \setminus (V \times B^R)) \cup (P_i \times P_{i+1})$$

Comme précédemment, si $P_i \neq A_1$ (resp. $P_{i+1} \neq A_2$) nous ajoutons à F' les paires $F_1 \subseteq F$ (resp. $F_2 \subseteq F$). Il suit que $|F'| < |F|$ et, par le Lemme 3.5, $G \triangle F'$ est un 3-leaf power, contredisant l'optimalité de F. Finalement, si $\min\{|P_i| \cdot |P_{i+1}|, |C_j| \cdot |C_{j+1}|\} = |C_j| \cdot |C_{j+1}|$, nous considérons l'ensemble :

$$F' := (F \setminus (V \times B^R)) \cup (C_j \times C_{j+1}) \cup F_1 \cup F_2$$

De nouveau, $|F'| < |F|$ et le Lemme 3.5 permet d'établir que le graphe $G \triangle F'$ est un 3-leaf power, contredisant l'optimalité de F. Cela conclut la preuve du Lemme 3.15. □

Règle 3.5. *Soient* (G, k) *une instance de* CLOSEST 3-LEAF POWER *et* $B \subseteq V$ *une* 2*-branche propre de* G *telle que* $path(B)$ *contient au moins* 5 *cliques critiques. Supprimer de* G *les sommets contenus dans* $B^R \setminus (P_1 \cup P_2)$ *et ajouter quatre nouvelles cliques critiques comme suit :*

(i) C_1 *(resp.* C_2*) de taille* $k+1$ *et adjacente à* P_1 *(resp.* P_2*) ;*

(ii) C_1' *(resp.* C_2'*) adjacente à* C_1 *et* C_2' *(resp.* C_2 *et* C_1'*), telles que la valeur de* $|C_1'| \cdot |C_2'|$ *est égale à la valeur d'une coupe minimum de* $path(B)$.

Nous montrons que cette règle peut être appliquée de manière valide. La complexité nécessaire pour l'appliquer sera décrite dans la prochaine section.

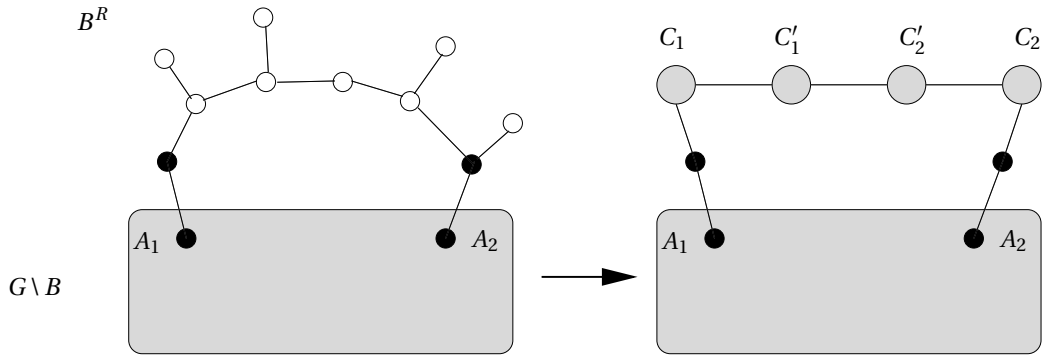


FIGURE 3.9 : Illustration de la Règle 3.5.

Lemme 3.18. *La Règle 3.5 est valide.*

Preuve. Nous dénotons par G' le graphe réduit par la Règle 3.5, et démontrons que G admet une k-édition si et seulement si G' admet une k-édition. Soit F une k-édition de G. Par le Lemme 3.15, nous pouvons supposer que les seuls sommets de B affectés par F sont contenus dans $A_1 \cup P_1 \cup P_2 \cup A_2$ ainsi que possiblement M, où M correspond aux extrémités des arêtes d'une coupe de taille minimum de $path(B)$. Par construction, remplacer les paires de F de la forme $(M \times M)$ par $(C'_1 \times C'_2)$ permet d'obtenir une k-édition pour G'.

Inversement, soit F' une k-édition de G'. Par construction de G', F' peut affecter uniquement les sommets de $A_1 \cup P_1 \cup P_2 \cup A_2$, ainsi que possiblement $(C'_1 \times C'_2)$: en effet, les cliques critiques C_1 et C_2 contiennent toutes deux $k+1$ sommets et ne sont donc pas affectées par F. Comme les sommets $A_1 \cup P_1 \cup P_2 \cup A_2$ sont inchangés entre G et G', et comme $|C'_1| \cdot |C'_2|$ correspond à la taille minimum d'une coupe de $path(B)$, il suit qu'une k-édition pour G peut être obtenue à partir de F' en préservant les éditions de F' et en remplaçant $(C'_1 \times C'_2)$ par $(P_i \times P_{i+1})$, où $|P_i| \cdot |P_{i+1}|$ est égal à la taille minimum d'une coupe de $path(B)$ dans G. □

3.3 Borner la taille d'une instance réduite

Détecter les branches en temps polynomial. Nous démontrons maintenant que les règles de réduction énoncées dans les sections précédentes permettent d'obtenir un algorithme de noyau pour le problème CLOSEST 3-LEAF POWER. Pour ce faire, nous devons démontrer (i) que l'application de ces règles peut s'effectuer en temps polynomial (Lemmes 3.10 et 3.10) et (ii) que la taille d'une instance positive réduite par ces règles peut-être bornée par un polynôme en k (Théorème 3.20).

Lemme 3.19. *Soit* (G,k) *une instance de* CLOSEST 3-LEAF POWER. *Les règles de réduction 3.3 à 3.5 peuvent être appliquées en temps linéaire.*

Preuve. En utilisant l'Observation 1.7, nous construisons le graphe des cliques critiques $\mathscr{C}(G)$ en temps $O(n+m)$. Pour identifier les 1-branches de G, un parcours peut-être utilisé sur $\mathscr{C}(G)$: en effet, les 1-branches de G correspondent à des sous-arbres induits dans $\mathscr{C}(G)$, rattachés au graphe par un unique point d'attachement. Nous supposons désormais que le graphe G est réduit par les

Règles 3.3 et 3.4. Dans ce cas, les 2-branches de G correspondent à des chemins induits de $\mathscr{C}(G)$, sur lesquels des sommets de degré 1 sont éventuellement attachés. Pour détecter les 2-branches, nous enlevons donc dans un premier temps les sommets de degré 1 dans $\mathscr{C}(G)$, puis identifions dans $\mathscr{C}(G)$ les chemins induits (*i.e.* contenant uniquement des sommets de degré 2). Pour ce faire, nous calculons les composantes connexes du graphe induit par les sommets de degré 2 dans le graphe ainsi obtenu. Une 2-branche de G correspond alors à une telle composante connexe, ainsi qu'aux deux voisins des extrémités de ce chemin. □

Algorithme de noyau. Nous démontrons maintenant le résultat principal de cette section.

Théorème 3.20. *Le problème* CLOSEST 3-LEAF POWER *admet un noyau avec* $O(k^3)$ *sommets.*

Preuve. Soit (G,k) une instance positive de CLOSEST 3-LEAF POWER réduite par les Règles 3.1 à 3.5. Soit F une k-édition de G, et soit $H := G \triangle F$ le 3-leaf power correspondant. Nous prouvons dans un premier temps que $\mathscr{C}_H \in O(k^2)$.

Nous supposons dans la suite de cette preuve que H possède $p \geqslant 1$ composantes connexes, et dénotons $\{H_1,\dots,H_p\}$ ces composantes connexes. Une clique critique de H est *affectée* si elle contient un sommet incident à une paire de F. Soit A l'ensemble des cliques critiques affectées de H. Comme $|F| \leqslant k$, nous savons que $|A| \leqslant 2k$. Pour tout $i \in [p]$, A_i dénote l'ensemble des cliques critiques affectées de H_i. Par ce qui précède, nous avons $\sum_{i=1}^{p} |A_i| = |A| \leqslant 2k$. Puisque H est un 3-leaf power, le Théorème 3.4 implique que $\mathscr{C}(H)$ est une forêt. En d'autres termes, cela signifie que $\mathscr{C}(H_i)$ est un arbre pour tout $i \in [p]$. Soit T_i le sous-arbre minimal de $\mathscr{C}(H_i)$ couvrant les sommets de A_i, $i \in [p]$. Par définition, si T_i' est un sous-arbre maximal de $\mathscr{C}(H_i) \setminus T_i$, T_i' ne contient aucune clique critique affectée et correspond donc à une 1-branche de G, qui aura été réduite par la Règle 3.3 ou 3.4. Pour tout $i \in [p]$, nous dénotons par $A_i^{\geqslant 3}$ l'ensemble des sommets de degré au moins 3 dans T_i (rappelons que les feuilles de T_i sont contenues dans A_i). De ce fait, $\sum_{i=1}^{p} |A_i^{\geqslant 3}| \leqslant \sum_{i=1}^{p} |A_i| \leqslant 2k$. Par construction, les composantes connexes de $T_i \setminus (A_i \cup A_i^{\geqslant 3})$ sont des chemins, et il existe au plus $(|A_i| + |A_i^{\geqslant 3}|) - 1$ tels chemins. Observons en particulier que $\sum_{i=1}^{p} (|A_i| + |A_i^{\geqslant 3}| - 1) \leqslant 4k - p \leqslant 4k$. De plus, par construction, chaque chemin correspond au $path(B)$ d'une 2-branche B de G, et a donc été réduit par la Règle 3.5. Ces différentes remarques nous amènent à l'observation suivante :

Observation 3.21. *Soit H_i une composante connexe de H, $i \in [p]$. Les propriétés suivantes sont vérifiées dans $\mathscr{C}(H)$ (voir Figure 3.10) :*

(i) tout sommet de $T_i \setminus A_i$ a au plus un sommet pendant (Règle 3.3),

(ii) toute clique critique de A_i a au plus $4k+2$ sommets pendants (Règle 3.3 et 3.4) et,

(iii) tout chemin de $\mathscr{C}(H_i) \setminus (A_i \cup A_i^{\geqslant 3})$ comporte au plus 8 sommets (Règles 3.3 et 3.5).

Nous pouvons désormais borner le nombre de sommets du graphe des cliques critiques $\mathscr{C}(H)$. Soit $i \in [p]$. Par la propriété (i) de l'Observation 3.21, toute clique critique de $A_i^{\geqslant 3}$ a au plus un sommet pendant. Par (ii), toute clique critique de A_i a au plus $4k+2$ sommets pendants et enfin, par (iii), toute 2-branche de $\mathscr{C}(H_i)$ comporte au plus 8 sommets. Finalement, T_i comporte $(|A_i| + |A_i^{\geqslant 3}|)$ sommets reliés par $(|A_i| + |A_i^{\geqslant 3}|) - 1$ chemins correspondant à des 2 branches. En sommant ces observations sur les p composantes connexes de H, nous obtenons le résultat suivant :

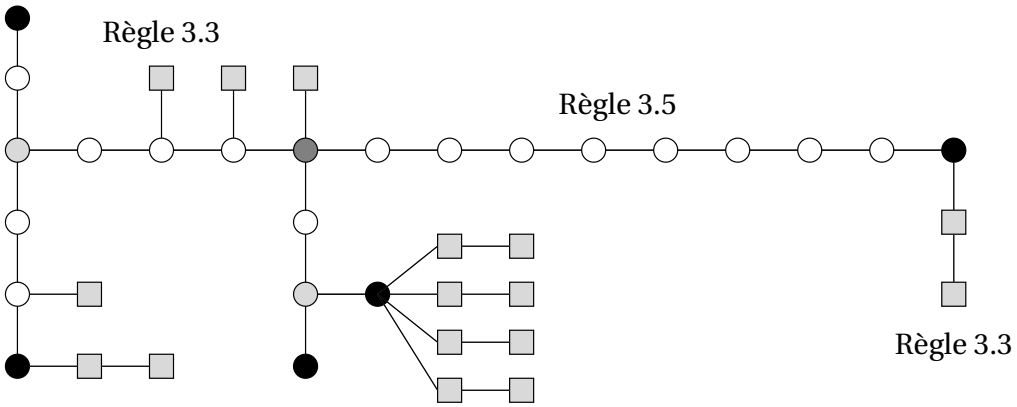


FIGURE 3.10 : Illustration des propriétés de l'Observation 3.21. Les sommets noirs représentent les cliques critiques de $\mathscr{A}$, les sommets ronds et gris les cliques critiques de $\mathscr{A}^{\geqslant 3}$ et les sommets carrés les 1-branches réduites.

$$
\begin{aligned}
|\mathscr{C}(H)| &= \sum_{i=1}^{p}((|A_i| + |A_i^{\geqslant 3}| - 1)\cdot 8 + |A_i^{\geqslant 3}|\cdot 2 + |A_i|\cdot(4k+3)) && (3.1)\\
&\leqslant 4k\cdot 8 + 2k\cdot 2 + 2k\cdot(4k+3) && (3.2)\\
&= 8k^2 + 42k && (3.3)
\end{aligned}
$$

Pour conclure, nous utilisons le résultat suivant.

Affirmation 3.22 ([136])**.** *Soient $G := (V,E)$ un graphe et $H := (V, E \triangle \{\{u,v\}\})$ avec $\{u,v\} \in V \times V$. Alors $|\mathscr{C}_H| \leqslant |\mathscr{C}_G| + 4$.*

Par l'Affirmation 3.22, nous savons que toute modification d'une paire $\{u,v\}$ dans le graphe H implique $|\mathscr{C}_{H'}| \leqslant |\mathscr{C}_H| + 4$, où H' dénote le graphe $H \triangle \{\{u,v\}\}$. Comme $|F| \leqslant k$, il suit que $|\mathscr{C}_G| \leqslant |\mathscr{C}_H| + 4k$, et donc $|\mathscr{C}_G| \leqslant 8k^2 + 46k$. Comme le graphe G est réduit par la Règle 3.2, G contient au plus $|\mathscr{C}_G|\cdot(k+1)$ sommets, soit au plus $8k^3 + 54k + 46k$ sommets. □

3.4 Noyaux cubiques pour 3-LEAF POWER COMPLETION et EDGE-DELETION

Nous prouvons maintenant que des résultats similaires peuvent être obtenus pour les problèmes de *complétion* et de *suppression*, où la seule édition autorisée est respectivement l'ajout et la suppression d'arêtes. Nous verrons que la majorité des règles de réduction utilisées pour le problème CLOSEST 3-LEAF POWER sont également valides pour les problèmes 3-LEAF POWER COMPLETION et 3-LEAF POWER EDGE-DELETION. Cela dit, une adaptation de la règle de réduction des 2-branches est nécessaire pour le problème 3-LEAF POWER COMPLETION. Dans la suite de cette section, nous démontrons uniquement les résultats différant de la Section 3.2. Comme nous allons le

voir, les règles de réduction pour CLOSEST 3-LEAF POWER sont valides pour 3-LEAF POWER EDGE-DELETION, et seule la Règle 3.5 requiert une approche différente pour le problème 3-LEAF POWER COMPLETION.

Lemme 3.23 ([12])**.** *Les Règles 3.1 à 3.4 sont valides et peuvent être appliquées en temps polynomial pour les problèmes* 3-LEAF POWER COMPLETION *et* 3-LEAF POWER EDGE-DELETION.

Lemme 3.24 ([12])**.** *La Règle 3.5 est valide pour le problème* 3-LEAF POWER EDGE-DELETION.

Remarque. La Règle 3.5 n'est pas valide pour le problème 3-LEAF POWER COMPLETION : en effet, dans la preuve du Lemme 3.15, lorsque nous considérons le cycle C contenant des sommets de B, il se peut que l'édition optimale doive supprimer les arêtes reliant deux cliques critiques consécutives de C, ce qui n'est pas possible dans le cas où seuls des ajouts d'arêtes sont autorisés.

Afin de corriger cela, nous démontrons un résultat technique permettant de borner la longueur d'un cycle induit de toute instance positive du problème 3-LEAF POWER COMPLETION. Ce résultat est valable pour tous les problèmes d'édition devant *supprimer* les cycles induits de longueur supérieure ou égale à 4.

Lemme 3.25. *Soient* (G,k) *une instance de* 3-LEAF POWER COMPLETION *et* $B \subseteq V$ *une* 2*-branche propre de* G *telle que* $path(B)$ *contient au moins* $k+4$ *cliques critiques. Si* A_1 *et* A_2 *appartiennent à la même composante connexe de* $G \setminus B^R$*, alors il n'existe pas de* k*-complétion pour* G*.*

Preuve. Comme A_1 et A_2 appartiennent à la même composante connexe dans $G \setminus B^R$, G contient un cycle induit C de taille supérieure ou égale à $k+4$. Pour toute k-complétion F de G, nous savons par le Théorème 3.4 qu'il existe $F' \subseteq F$ tel que $\mathscr{C}(C+F')$ est un arbre. En particulier, F' est une *triangulation* de C [53]. Comme toute triangulation d'un cycle de longueur q contient au moins $q-3$ cordes [53], cela implique que $|F'| > k$, contredisant la définition de F. Il suit que G n'admet pas de k-complétion. □

Nous définissons maintenant une règle réduisant les 2-branches pour le problème 3-LEAF POWER COMPLETION.

Règle 3.6. *Soient* (G,k) *une instance de* 3-LEAF POWER COMPLETION *et* $B \subseteq V$ *une* 2*-branche propre de* G *dont le chemin* $path(B)$ *contient au moins* $k+4$ *cliques critiques.*

- *Si les points d'attachement* A_1 *et* A_2 *appartiennent à la même composante connexe de* $G \setminus B^R$*, alors il n'existe pas de* k*-complétion de* G*.*
- *Sinon, supprimer de* G *les sommets de* $B^R \setminus (P_1 \cup P_2)$*, et ajouter deux cliques critiques adjacentes* C_1 *et* C_2 *de taille* $k+1$*, respectivement voisines de* P_1 *et* P_2*.*

Lemme 3.26. *La Règle 3.6 est valide.*

Preuve. Observons dans un premier temps que le premier point provient du Lemme 3.25. Supposons maintenant que les cliques d'attachement A_1 et A_2 appartiennent à différentes composantes connexes de $G \setminus B^R$. Nous démontrons qu'il existe toujours une complétion optimale n'affectant que des sommets de $A_1 \cup P_1 \cup A_2 \cup P_2$. Soit F une complétion optimale de G. Nous dénotons par A'_1 (resp. A'_2) la clique critique de $H := G+F$ contenant A_1 (resp. A_2). Remarquons que A_1 et A_2 appartiennent à différentes composantes connexes CC_1 et CC_2 de $H \setminus B^R$: en effet, dans

le cas contraire, F doit trianguler un cycle de longueur supérieure ou égale à $k+4$, ce qui est impossible par le Lemme 3.25. Comme précédemment, en utilisant le Lemme 3.5 et l'Observation 3.6, le graphe $H' := (H[CC_1], A'_1 \setminus B^R) \otimes (G[B^R], P_1)$ est un 3-leaf power. De même, le graphe $H'' := (H[CC_2], A'_2 \setminus B^R) \otimes (H', P_2)$ est un 3-leaf power. Finalement, la complétion F'' telle que $H'' := G + F''$ est un sous-ensemble de F respectant les propriétés désirées.

Supposons maintenant que G possède une k-complétion, et dénotons par G' le graphe réduit par la Règle 3.6. Par les arguments précédents, nous savons que les sommets de B affectés par F sont contenus dans $A_1 \cup P_1 \cup P_2 \cup A_2$. Comme ces sommets sont toujours présents dans G', F est une k-complétion de G. Inversement, soit F' une k-complétion de G'. Par construction de G', F n'affecte que les sommets de $A_1 \cup P_1 \cup P_2 \cup A_2$, les cliques critiques C_1 et C_2 contenant $k+1$ sommets. Ainsi le graphe obtenu à partir de $G' + F$ en *remplaçant* les cliques critiques C_1 et C_2 par $path(B) \setminus (P_1 \cup P_2)$ est un 3-leaf power obtenu à partir de G en ajoutant les paires contenues dans F' (Lemme 3.5). Il suit que F' est une k-complétion de G. □

Théorème 3.27. *Les problèmes* 3-LEAF POWER COMPLETION *et* 3-LEAF POWER EDGE-DELETION *admettent un noyau avec* $O(k^3)$ *sommets.*

CHAPITRE

4

Noyau polynomial pour PROPER INTERVAL COMPLETION

Dans ce chapitre, nous utilisons à nouveau la notion de **branches** pour obtenir un **noyau polynomial** pour le problème PROPER INTERVAL COMPLETION. Ce problème NP-Complet [76] trouve de nombreuses applications en bioinformatique [103]. Le problème PROPER INTERVAL COMPLETION admet un algorithme FPT de complexité $O^*(2^{4^k} \cdot m)$ [100, 101]. Ce dernier, établi par **Kaplan et al.** en **1994**, constitue l'un des premiers algorithmes FPT non-trivial pour un problème d'édition de graphes. Cependant, comme pour le problème CLOSEST 3-LEAF POWER, l'existence d'un noyau polynomial était à ce jour inconnue. Dans un travail réalisé avec Stéphane BESSY [13], nous utilisons une caractérisation par **sous-graphes induits interdits** [156] et par **ordre parapluie** [121] pour définir des **branches** et ainsi obtenir le premier noyau polynomial pour le problème PROPER INTERVAL COMPLETION, défini comme suit.

PROPER INTERVAL COMPLETION :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq (V \times V) \setminus E$ de taille au plus k tel que le graphe $H := (V, E \cup F)$ est un graphe d'intervalle propre ?

Théorème 4.1. *Le problème* PROPER INTERVAL COMPLETION *admet un noyau avec* $O(k^5)$ *sommets.*

4.1 Définition et caractérisation des graphes d'intervalles propres

Définition 4.2 (Graphe d'intervalle propre)**.** *Un graphe $G := (V, E)$ est un* graphe d'intervalle propre *si et seulement s'il admet une représentation sur la droite réelle telle que :* intervalle propre

(i) les sommets de G sont en bijection avec des intervalles de la droite réelle,

(ii) $uv \in E$ si et seulement si $I_u \cap I_v \neq \emptyset$, où I_u et I_v représentent les intervalles associés aux sommets u et v, respectivement et,

(iii) pour tous sommets $u, v \in V$, $I_u \not\subset I_v$.

Nous énonçons maintenant plusieurs résultats connus permettant de caractériser la classe des graphes d'intervalle propre. Ces caractérisations nous permettront par la suite de définir la notion de *branche* pour le problème PROPER INTERVAL COMPLETION.

Théorème 4.3 (Sous-graphes induits [156])**.** *Un graphe $G := (V, E)$ est un* graphe d'intervalle propre *si et seulement s'l ne contient pas de* griffe, 3-sun, net *ou* hole *comme sous-graphe induit. (voir Figure 4.1)*

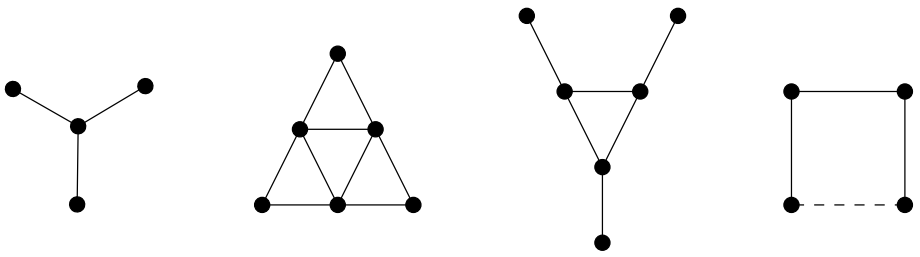


FIGURE 4.1 : Les sous-graphes induits interdits caractérisant la classe des graphes d'intervalle propre.

La *griffe* est le graphe biparti $K_{1,3}$. En dénotant la bipartition par $(\{c\}, \{l_1, l_2, l_3\})$, nous définissons c comme le *centre* de la griffe, et $\{l_1, l_2, l_3\}$ comme ses *feuilles*.

Théorème 4.4 (Propriété parapluie [121])**.** *Un graphe $G := (V, E)$ est un graphe d'intervalle propre si et seulement si ses sommets V admettent un ordre $\sigma := v_1 \dots v_n$ (appelé* ordre parapluie*) vérifiant la propriété suivante : étant donnés $v_i v_j \in E$ avec $i < j$, $v_i v_l, v_l v_j \in E$ pour tout $i < l < j$.* ordre parapluie

Dans la suite de ce chapitre, nous associons un ordre parapluie σ_G à tout graphe d'intervalle propre $G := (V, E)$. Nous utiliserons cet ordre de manière extensive afin de prouver la validité de nos règles de réduction. En particulier, nous utiliserons les notations et résultats suivants.

Définition 4.5 (Arête extrémale)**.** *Soit $G := (V, E)$ un graphe d'intervalle propre et σ_G son ordre parapluie. Une arête uv est dite* extrémale *(par rapport à σ_G) s'il n'existe pas d'arête $u'v'$ différente de uv telle que $u' \leq_{\sigma_G} u$ et $v \leq_{\sigma_G} v'$.* arête extrémale

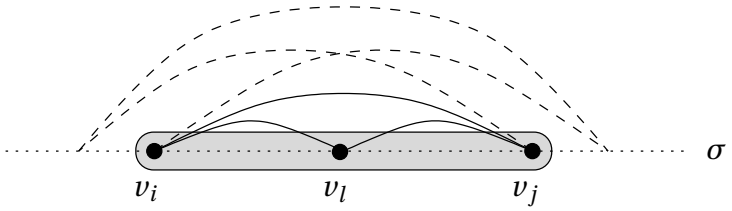


FIGURE 4.2 : Une illustration de la propriété parapluie (Théorème 4.4). L'arête $v_i v_j$ représentée est extrémale (Définition 4.5). [1]

Observation 4.6. *Soient* (G,k) *une instance de* PROPER INTERVAL COMPLETION *et* F *une complétion optimale de* G. *Soient* $H := G + F$ *le graphe d'intervalle propre correspondant et* σ_H *son ordre parapluie. Toute arête extrémale de* σ_H *est une arête de* G.

Preuve. Supposons par contradiction qu'il existe une arête extrémale e de σ_H qui corresponde à une paire de F. Par définition, σ_H est toujours un ordre parapluie si on supprime l'arête e de F, contredisant l'optimalité de F. □

Remarque. Étant donnés un graphe d'intervalle propre $G := (V,E)$ et un ordre parapluie σ_G, tout ensemble de vrais jumeaux de G est consécutif dans σ_G [50]. De plus, l'ordre σ_G est *unique* à *permutation des vrais jumeaux* et *renversement de l'ordre induit par une composante connexe* près [50]. Remarquons également que pour tout arête uv vérifiant $u <_{\sigma_G} v$, l'ensemble $\{w \in V \,:\, u \leqslant_{\sigma_G} w \leqslant_{\sigma_G} v\}$ est une clique de G. Notons finalement que pour tout $1 \leqslant i < n$, la partition $(\{v_1,\ldots,v_i\},\{v_{i+1},\ldots,v_n\})$ est un join de G.

Décomposition d'adjacence. Le Théorème 4.4 nous permet d'établir la notion de *décomposition d'adjacence* pour le problème PROPER INTERVAL COMPLETION. Soient $G := (V,E)$ un graphe d'intervalle propre et $\sigma_G := v_1 \ldots v_n$ son ordre parapluie associé. Considérons la collection $\mathcal{V} := \{V_1,\ldots,V_r\}$, où $V_1 := \{v_1,\ldots,v_{l_1}\}$ et v_{l_1} dénote le voisin de v_1 d'indice maximal dans σ_G. Par le Théorème 4.4, V_1 est une clique de G. Maintenant, pour tout $2 \leqslant i \leqslant r$, nous posons $V_i := \{v_{l_{i-1}+1},\ldots,v_{l_i}\}$, v_{l_i} dénotant le voisin de $v_{l_{i-1}+1}$ d'indice maximal dans σ_G. Observons que V_i est également une clique de G. La relation d'adjacence $P_{\mathcal{G}}$ est également définie en utilisant le Théorème 4.4. En effet, pour tout $1 \leqslant i < l$, les ensembles V_i et V_{i+1} sont connectés par un *join* dans G. En d'autres termes, le graphe $G[V_i \cup V_{i+1}]$ est un graphe *threshold* : pour tous $v_p, v_q \in V_i$, $p < q$, $N_{V_{i+1}}(v_p) \subseteq N_{V_{i+1}}(v_q)$ est vérifié. Ainsi, un graphe d'intervalle propre peut être décomposé en un *chemin de cliques*, dont les cliques consécutives sont connectées par un join (Figure 4.3). [2]

1. *Conventions.* Les ensembles grisés définissent des cliques du graphe considéré. Lorsque deux ensembles A et B sont reliés par une arête, cela signifie que les sommets de A dominent les sommets de B.

2. Pour des raisons de simplicité, nous considérons les branches sous la forme de leur *ordre parapluie* plutôt que du chemin de cliques.

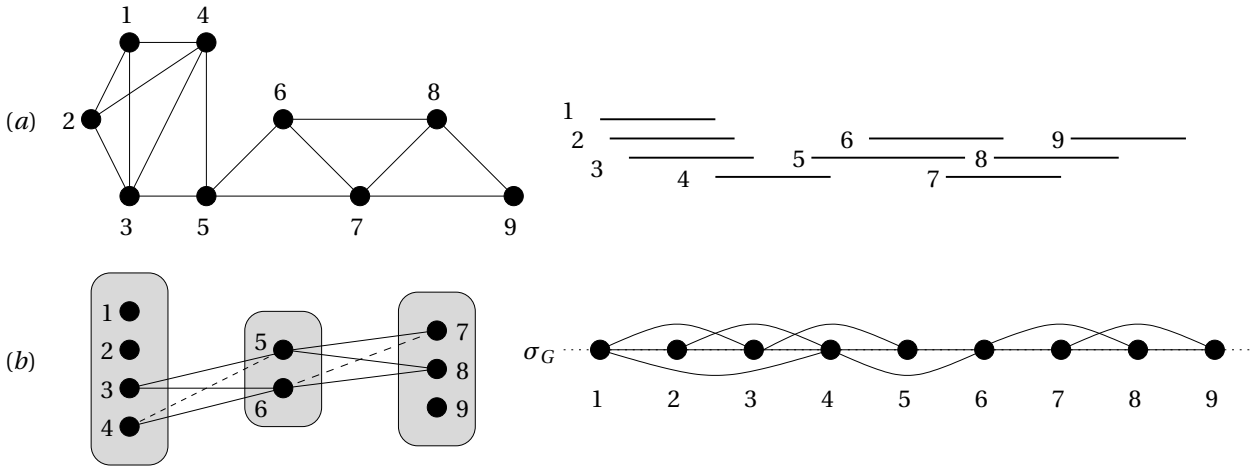


FIGURE 4.3 : (*a*) Un graphe d'intervalle propre $G := (V, E)$ ainsi que sa représentation associée. (*b*) La décomposition en cliques de G et son ordre parapluie associé σ_G.

4.1.1 Branches et K-joins

Structures à réduire. Nous avons mentionné dans la Section 4.1 que la classe des graphes d'intervalle propre admettait une décomposition d'adjacence, et qu'ainsi la notion de branches pouvait être considérée pour ce problème. Afin de déterminer quels types de branches il sera nécessaire de réduire, nous considérons une instance positive (G, k) du problème PROPER INTERVAL COMPLETION, et une k-édition F de G. Nous posons $H := G + F$ pour dénoter le graphe d'intervalle propre résultant, et σ_H pour son ordre parapluie associé. Par définition de F, nous savons que H possède p sommets affectés $\{a_1, \ldots, a_p\}$ avec $p \leqslant 2k$. Supposons sans perte de généralité que ces sommets vérifient $a_1 <_{\sigma_H} \ldots <_{\sigma_H} a_p$. Maintenant, observons que l'ensemble de sommets L (resp. R) situés avant (resp. après) a_1 (resp. a_p) induit un graphe d'intervalle propre. De plus, la partition $(L, G \setminus L)$ (resp. $(R, G \setminus R)$) est un join de G, impliquant que L et R sont tous deux attachés au reste du graphe en respectant la propriété d'adjacence $P_{\mathcal{G}}$. Remarquons également que l'ensemble des sommets contenus dans $\delta(L)$ et $\delta(R)$ forment une clique par la définition de σ_H. De manière similaire, l'ensemble de sommets S_i contenus entre a_i et a_{i+1}, $i \in [p]$, induit un graphe d'intervalle propre dont la *frontière* peut être partitionnée en deux cliques, étant attachées au reste du graphe en respectant $P_{\mathcal{G}}$. Ainsi, ce sont ces structures particulières que nous allons devoir réduire afin d'obtenir un algorithme de noyau polynomial pour PROPER INTERVAL COMPLETION.

Branches. Nous définissons maintenant formellement la notion de *branche*, qui correspond à une partie du graphe *induisant* un graphe d'intervalle propre et dont la frontière est connectée au reste du graphe en respectant $P_{\mathcal{G}}$.

Définition 4.7 (1-branche)**.** *Soient* (G, k) *une instance de* PROPER INTERVAL COMPLETION*, et* $B \subseteq V$*.* 1-branche
L'ensemble B *est appéle une* 1-branche *de* G *si les propriétés suivantes sont vérifiées :*

(i) le graphe $G[B]$ *est un graphe d'intervalle propre connexe admettant un ordre parapluie* $\sigma_B := b_1 \ldots b_{|B|}$ *et,*

(ii) L'ensemble $V \setminus B$ peut être partitionné en deux ensembles R et C tels que :
- *il n'y a aucune arête entre B et C,*
- *tout sommet de R a un voisin dans B,*
- *il n'y a aucune arête entre $B \setminus N_B[b_{|B|}]$ et R,*
- *pour tous $b_i, b_j \in N_B[b_{|B|}]$, $i < j$, $N_R(b_i) \subseteq N_R(b_j)$.*

Nous dénotons par A_d l'ensemble $N_B[b_{|B|}]$. Observons que A_d induit une clique dans G. Nous appelons A_d la *clique d'attachement* de B, et posons $B^R := B \setminus A_d$. Observons en particulier que A_d contient *tous* les sommets de B ayant des voisins dans $G \setminus B$ (des sommets de A_d peuvent cependant n'avoir aucun voisin dans $G \setminus B$).

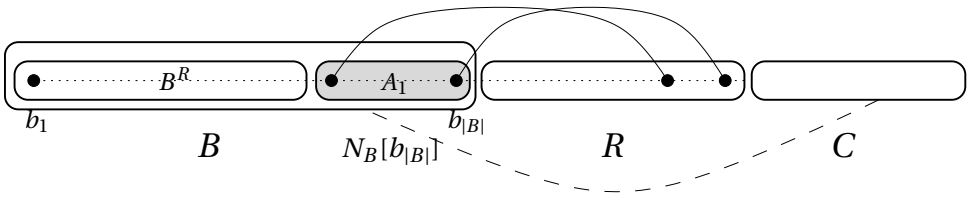


FIGURE 4.4 : Illustration de la notion de 1-branche pour le problème PROPER INTERVAL COMPLETION.

2-branche

Définition 4.8 (2-branche)**.** *Soient (G,k) une instance de* PROPER INTERVAL COMPLETION *et $B \subseteq V$. L'ensemble B est appéle une* 2-branche *de G si les propriétés suivantes sont vérifiées :*

(i) le graphe $G[B]$ est un graphe d'intervalle propre connexe admettant un ordre parapluie $\sigma_B := b_1 \dots b_{|B|}$ et,

(ii) L'ensemble $V \setminus B$ peut être partitionné en trois ensembles L, R et C tels que :
- *il n'y a aucune arête entre B et C,*
- *tout sommet de L (resp. R) a un voisin dans B,*
- *il n'y a aucune arête entre $B \setminus N_B[b_{|B|}]$ et R,*
- *il n'y a aucune arête entre $B \setminus N_B[b_1]$ et L,*
- *pour tous $b_i, b_j \in N_B[b_{|B|}]$, $i < j$, $N_R(b_i) \subseteq N_R(b_j)$. De même, pour tous $b_i, b_j \in N_B[b_1]$, $i < j$, $N_L(b_j) \subseteq N_L(b_i)$.*

cliques d'attachement

Comme précédemment, nous dénotons par A_d l'ensemble $N_B[b_{|B|}]$. De plus, nous appelons A_g l'ensemble $N_B[b_1]$. Les ensembles A_g et A_d sont appelés *cliques d'attachement* de B, et nous posons $B^R := B \setminus (A_g \cup A_d)$.

Remarques :
- En ce qui concerne les 2-branches, les ensembles L et R peuvent être inversés selon l'ordre σ_B considéré. Ainsi, dans la suite de ce chapitre, nous considérons des 2-branches B pour lesquelles l'ordre parapluie σ_B est *fixé*, permettant ainsi de fixer également les ensembles L et R.
- De plus, les cas où $L = \emptyset$ ou $R = \emptyset$ sont possibles, et correspondent exactement à la Définition 4.7.

– Finalement, lorsque $B^R = \emptyset$, il est possible qu'un sommet de R ou de L soit adjacent à tous les sommets de B. Dans un tel cas, nous dénotons par N l'ensemble des sommets de $G \setminus B$ dominant B, et supprimerons N de R et L. Dans un abus de notation, nous appellerons R (resp. L) l'ensemble $R \setminus N$ (resp. $L \setminus N$). Nous préciserons dès que nécessaire l'utilisation d'un tel ensemble N.

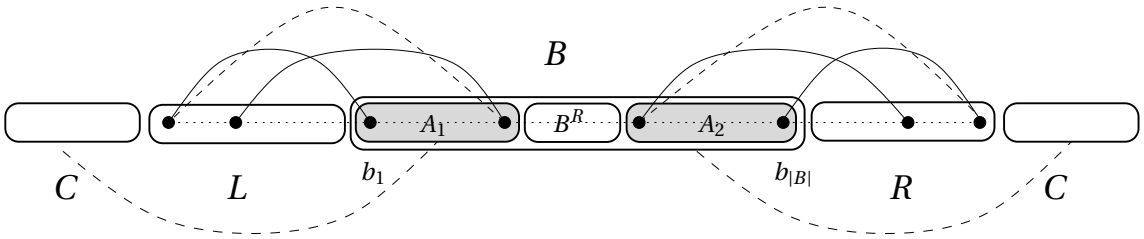


FIGURE 4.5 : Illustration de la notion de 2-branche pour le problème PROPER INTERVAL COMPLETION.

Dans tous les cas, dans un abus de notation et par souci de simplicité, nous utiliserons B pour dénoter à la fois une branche et son ordre parapluie associé σ_B. Ainsi, les *premiers sommets de B* correspondent aux premiers sommets de σ_B. De la même manière, les *premiers sommets de* B^R correspondent aux premiers sommets de l'ordre σ_{B^R}.

K-join. Dans les deux définitions, lorsque le graphe d'intervalle propre $G[B]$ est une *clique*, nous appelons B un *K-join*. Remarquons que, dans une 1- ou 2-branche, pour toute arête extrémale uv de σ_B l'ensemble $\{w \in B : u \leqslant_{\sigma_B} w \leqslant_{\sigma_B} v\}$ définit un K-join de G. Cependant, cette décomposition d'adjacence en K-joins n'est pas unique : par exemple, les K-joins correspondant aux arêtes extrémales de σ_B ne sont pas disjoints. Comme nous l'avons vu précédemment, il est possible de raffiner cette décomposition en considérant des K-joins *disjoints*. K-join

Décomposition en K-joins. Nous exprimons maintenant la décomposition des graphes d'intervalle propre en termes de K-joins. Plus précisément, nous donnons une décomposition pour les *branches*, que nous utiliserons dans la suite de ce chapitre. Afin d'être aussi général que possible, nous exprimons cette décomposition sur les 2-branches (Définition 4.8). Cependant, cette dernière est également valide pour les 1-branches (Définition 4.7). Soient (G,k) une instance de PROPER INTERVAL COMPLETION, $B \subseteq V$ une 2-branche de G et $\sigma_B := b_1 \dots b_{|B|}$ l'ordre parapluie associé. Nous redéfinissons A_g comme l'ensemble $\{b_1, \dots, b_{l'}\}$, où $b_{l'}$ dénote le voisin de b_1 d'indice maximal dans σ_B. De manière similaire, nous posons $A_d := \{b_l, \dots, b_{|B|}\}$, où b_l est le voisin de $b_{|B|}$ d'indice minimal dans σ_B. Nous définissons la *décomposition en K-joins* $\mathscr{B} := \{B_1, \dots, B_r\}$ de B comme suit : pour tout $i \in [r]$, nous posons $B_i := \{b_{l_{i-1}+1}, \dots, b_{l_i}\}$, où b_{l_i} est le voisin de $b_{l_{i-1}+1}$ d'indice maximal dans σ_B. Le premier K-join de $\mathscr{B}$ est A_g (et donc, $l_0 = 0$ et $l_1 = l'$) et une fois B_{i-1} défini, nous posons $B_i := \{b_{l_{i-1}+1}, \dots, b_{l_i}\}$ si $b_{l_{i-1}+1} \notin A_d$, et $B_i := \{b_{l_{i-1}+1}, \dots, b_{|B|}\}$ dans le cas contraire (voir Figure 4.6).

Définition 4.9 (K-join propre)**.** *Soit* (G,k) *une instance de* PROPER INTERVAL COMPLETION. *Un K-join de G est* propre *lorsque ses sommets ne sont contenus ni dans une griffe ni dans un* C_4.

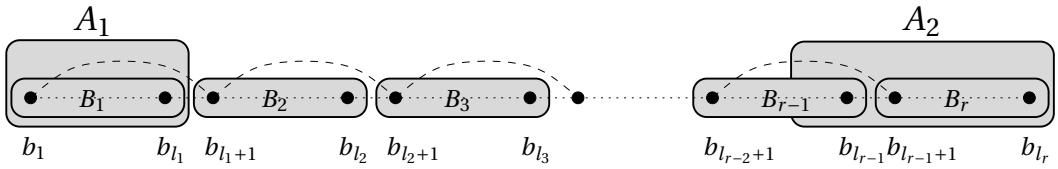


FIGURE 4.6 : Une décomposition en K-joins d'une 2-branche.

Observation 4.10. *Soit (G,k) une instance de* PROPER INTERVAL COMPLETION. *Tout sous-ensemble d'un K-join (resp. K-join propre) de G définit un K-join (resp. K-join propre) de G.*

Preuve. Soient $B \subseteq V$ un K-join (propre ou non) de G d'ordre parapluie σ_B, et $B' \subseteq B$. Par définition d'un K-join, les sommets de B' ordonnés selon $\sigma_{B'}$ respectent les propriétés d'adjacence avec $G \setminus B$ décrites dans les Définitions 4.7 et 4.8. Comme $G[B]$ est une clique, il suit que les sommets de $B \setminus B'$ dominent les sommets de B', qui définit ainsi un K-join de G. □

4.2 Règles de base

Nous décrivons plusieurs règles de réduction *classiques* pour le problème PROPER INTERVAL COMPLETION, qui nous seront utiles afin d'établir notre algorithme de noyau. La validité de ces règles de réduction découle des Lemmes 2.46, 2.48 et 2.50 présentés dans la Section 2.3.5. Nous démontrons donc uniquement que les conditions pour que ces règles de réduction puissent s'appliquer sont vérifiées par la classe des graphes d'intervalle propre.

Observation 4.11. *La classe des graphes d'intervalle propre est héréditaire, close par union disjointe et par ajout de vrai jumeau.*

Preuve. Soit $G := (V,E)$ un graphe d'intervalle propre. Dans un premier temps, observons que tout sous-graphe induit $G[S]$, $S \subseteq V$, est un graphe d'intervalle propre, dont la représentation sur la droite réelle correspond à celle de G où les intervalles correspondant aux sommets de $V \setminus S$ ont été supprimés. De manière similaire, étant donné $G' := (V',E')$ un graphe d'intervalle propre, $G \oplus G'$ est un graphe d'intervalle propre (il suffit de faire une union disjointe des représentations sur la droite réelle). Pour conclure, si le graphe G_u est obtenu à partir de G en ajoutant un vrai jumeau u' à un sommet u, alors copier l'intervalle correspondant à u permet d'obtenir une représentation sur la droite réelle pour G_u. □

Nous rappelons l'énoncé formel de ces règles de réduction.

Règle 2.5 **Règle 4.1** (Composante connexe)**.** *Soit (G,k) une instance de* PROPER INTERVAL COMPLETION. *Supprimer toute composante connexe C de G telle que $G[C]$ induit un graphe d'intervalle propre.*

Lemme 2.46 **Lemme 4.12.** *La Règle 4.1 est valide et peut être appliquée en temps polynomial.*

Règle 2.7 **Règle 4.2** (Vrai jumeau)**.** *Soient (G,k) une instance de* PROPER INTERVAL COMPLETION *et $C \subseteq V$ un ensemble de vrais jumeaux de G tel que $|C| > k$. Supprimer $|C| - (k+1)$ sommets arbitraires de C.*

Lemme 2.50 **Lemme 4.13.** *La Règle 4.2 est valide et peut être appliquée en temps polynomial.*

Règle 4.3 (Sunflower)**.** *Soit* (G,k) *une instance de* PROPER INTERVAL COMPLETION. *Soit* $\mathscr{S} := \{C_1,\dots,C_m\}$, $m > k$ *un ensemble de griffes ayant deux feuilles* u, v *en commun mais différentes troisièmes feuilles. Ajouter* uv *à* F *et décrémenter* k *de* 1. Règle 2.6
Soit $\mathscr{S} := \{C_1,\dots,C_m\}$, $m > k$ *un ensemble de* C_4 *distincts ayant une non-arête* $uv \in \bar{E}$ *en commun. Ajouter* uv *à* F *et décrémenter* k *de* 1.

Lemme 4.14. *La Règle 4.3 est valide et peut être appliquée en temps polynomial.* Lemme 2.48

4.3 Règles de réduction via la notion de *branches*

Dans cette section nous décrivons les règles de réduction permettant de réduire les 1-branches et 2-branches d'une instance (G,k) de PROPER INTERVAL COMPLETION. Nous démontrons dans un premier temps que les K-joins *propres* d'une instance (G,k) de PROPER INTERVAL COMPLETION (qui sont des branches particulières) peuvent être réduits. Nous verrons par la suite que tout K-join de taille suffisamment grande (par rapport au paramètre k) contient un K-join propre (Section 4.3.2), ce qui nous permettra de réduire la taille des K-joins de G (Section 4.3.3).

4.3.1 Borner la taille d'un K-join propre

Nous décrivons maintenant une règle de réduction qui permet de borner le nombre de sommets contenus dans un K-join propre. Bien que particulièrement technique à prouver valide, cette règle de réduction constitue l'outil principal de notre algorithme de noyau. Nous verrons dans la Section 4.4 comment cette règle peut être appliquée en temps polynomial.

Règle 4.4 (K-join)**.** *Soient* (G,k) *une instance de* PROPER INTERVAL COMPLETION *et* $B \subseteq V$ *un K-join propre contenant au moins* $2k+2$ *sommets. Soient* B_L *les* $k+1$ *premiers sommets de* B, B_R *les* $k+1$ *derniers sommets de* B *et* $M := B \setminus (B_L \cup B_R)$. *Supprimer l'ensemble de sommets* M *de* G.

Lemme 4.15. *La Règle 4.4 est valide.*

Preuve. Nous dénotons le graphe réduit par $G' := G \setminus M$. Conformément aux notations précédentes, nous supposons $\sigma_B := b_1 \dots b_{|B|}$. Remarquons tout d'abord que toute k-complétion de G est une k-complétion de G' puisque la classe des graphes d'intervalle propres est héréditaire (Observation 4.11). Soit F' une k-complétion de G'. Nous dénotons par $H := G' + F'$ le graphe d'intervalle propre correspondant et par $\sigma_H := h_1 \dots h_{|H|}$ son ordre parapluie associé. Nous prouvons que les sommets de M peuvent être insérés dans σ_H (quitte à modifier cet ordre) pour obtenir un ordre parapluie pour G, et ce sans ajouter de nouvelles arêtes. En fait, durant le processus, certaines arêtes de F' pourront même être supprimées. Il suivra directement que G admet une k-complétion. Pour ce faire, nous avons tout d'abord besoin d'une propriété structurelle sur G. Comme mentionné dans la Section 4.1.1, nous considérons de manière particulière les sommets de $L \cup R$ dominant B. Nous posons ainsi $N := \bigcap_{b \in B} N_G(b) \setminus B$ et, de manière abusive, nous dénotons L (resp. R) l'ensemble $L \setminus N$ (resp. $R \setminus N$).

Affirmation 4.16. *Les ensembles* L *et* R *sont des cliques de* G.

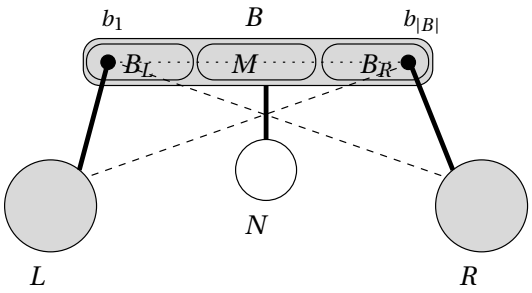


FIGURE 4.7 : Les propriétés structurelles du K-join B dans G.

Preuve. Nous démontrons que R est une clique de G, la preuve pour L utilisant les mêmes arguments. Supposons que R contienne deux sommets u et v tels que $uv \notin E(G)$. Par construction, aucun sommet de R n'est adjacent à b_1 : en effet, si un tel sommet existait, il serait adjacent à tous les sommets de B par définition d'un K-join et devrait donc se trouver dans N. De même, tout sommet de R est adjacent au sommet $b_{|B|}$. Il suit que $\{b_{|B|}, b_1, u, v\}$ définit une griffe dans G contenant des sommets de B, contredisant le fait que B soit propre. ◇

L'observation suivante est une implication directe de la définition de K-join et sera utilisée dans le reste de la preuve.

Observation 4.17. *Soit $r \in R$ tel que $N_B(r) \cap B_L \neq \emptyset$. Alors $M \subseteq N_B(r)$. De manière similaire, pour tout sommet $l \in L$ tel que $N_B(l) \cap B_R \neq \emptyset$, $M \subseteq N_B(l)$.*

Nous utilisons ces deux résultats pour construire un ordre parapluie pour G en insérant les sommets de M dans σ_H. Nous dénotons respectivement par b_p et b_d les premier et dernier sommets de $B \setminus M$ apparaissant dans σ_H. Nous posons également $B_H := \{u \in V(H) : b_p \leq_{\sigma_H} u \leq_{\sigma_H} b_d\}$. Remarquons en particulier que B_H est une clique dans H puisque l'arête $b_p b_d$ appartient à $E(G)$ par définition. De plus, nous avons également $B \setminus M \subseteq B_H$. Nous modifions maintenant l'ordre σ_H comme suit : si u et v sont vrais jumeaux dans H, sont consécutifs dans σ_H et vérifient $u <_{\sigma_H} v <_{\sigma_H} b_p$ et $N_M(v) \subseteq N_M(u)$, alors nous inversons u et v dans σ_H. Remarquons que ces échanges s'arrêtent lorsque les vrais jumeaux considérés sont ordonnés en fonction du join entre $\{u \in V(H) : u <_{\sigma_H} b_p\}$ et M. Nous procédons de manière similaire pour les vrais jumeaux situés après b_d dans σ_H. L'ordre ainsi obtenu est également un ordre parapluie pour H : en fait, nous avons simplement renommé certains sommets de σ_H. Dans un abus de notation, nous appelons σ_H l'ordre ainsi obtenu.

Affirmation 4.18. *Pour tout sommet $m \in M$, l'ensemble $B_H \cup \{m\}$ est une clique de G. De ce fait, comme M est une clique de G, l'ensemble $B_H \cup M$ est une clique de G.*

Preuve. Soient $m \in M$ et $u \in B_H$. Nous prouvons que $um \in E(G)$. Observons tout d'abord que le résultat est trivialement vérifié si $u \in B$. Supposons donc que $u \notin B$. Comme B_H est une clique dans H, il suit que u est adjacent à tous les sommets de $B \setminus M = B_L \cup B_R$ dans H. Puisque B_L et B_R contiennent tous deux $k+1$ sommets, il suit que $N_G(u) \cap B_L \neq \emptyset$ et $N_G(u) \cap B_R \neq \emptyset$ (sinon nous aurions $|F| > k$). De ce fait, u appartient à $L \cup N \cup R$ et $um \in E(G)$ par l'Observation 4.17. ◇

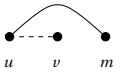
Affirmation 4.19. *Soient m un sommet de M et σ_m l'ordre obtenu à partir de σ_H en supprimant B_H et en insérant m à la place de B_H. L'ordre σ_m vérifie la propriété parapluie.*

Preuve. Nous démontrons le résultat par contradiction. Supposons que σ_m ne respecte pas la propriété parapluie, *i.e.* qu'il existe (sans perte de généralité) deux sommets u et v de $V(H) \setminus B_H$ tels que l'une des configurations suivantes soit vérifiée :

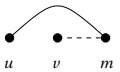
$$u <_{\sigma_m} v <_{\sigma_m} m \quad : \quad um \in E(H) \text{ et } uv \notin E(H) \tag{4.1}$$

$$u <_{\sigma_m} m <_{\sigma_m} v \quad : \quad um \notin E(H) \text{ ou } vm \notin E(H) \text{ et } uv \in E(H) \tag{4.2}$$

$$u <_{\sigma_m} v <_{\sigma_m} m \quad : \quad um \in E(H) \text{ et } vm \notin E(H) \tag{4.3}$$



Nous supposons dans un premier temps que (4.1) est vérifiée. Puisque $uv \notin E(H)$ et σ_H est un ordre parapluie de H, il suit que $uw \notin E(H)$ pour tout sommet w appartenant à B_H (rappelons que m est inséré à la place de B_H dans σ_H). Cela implique que $uw \notin E(G)$ pour tout $w \in B_H$, et donc $N_G(u) \cap B_L = \emptyset$ et $N_G(u) \cap B_R = \emptyset$, ce qui est impossible vu que $um \in E(G)$. Ensuite, nous supposons que (4.2) est vérifiée , avec $um \notin E(H)$. Puisque $uv \in E(H)$ et σ_H est un ordre parapluie, $B_H \subseteq N_H(u)$, et en particulier $B_L \cup B_R \subseteq N_H(u)$. Comme $|B_L| = |B_R| = k+1$, nous savons que $N_G(u) \cap B_L \neq \emptyset$ et $N_G(u) \cap B_R \neq \emptyset$. Dans ce cas, l'Observation 4.17 implique que $um \in E(G)$, et nous obtenons une contradiction. Le cas $vm \notin E(H)$ se prouve de manière similaire. Pour conclure, nous supposons que (4.3) est vérifiée. Nous choisissons le premier sommet u satisfaisant cette propriété par rapport à l'ordre σ_m. En d'autres termes, nous supposons que $wm \notin E(G)$ pour tout $w <_{\sigma_m} u$. De manière similaire, nous choisissons v comme le premier sommet vérifiant $u <_{\sigma_m} v$ et $vm \notin E(G)$. Comme $um \in E(G)$, nous savons par définition que u appartient $L \cup N \cup R$. De plus, comme $vm \notin E(G)$, nous savons que v appartient à $L \cup R \cup C$. Nous devons donc étudier différentes configurations :



(i) $u \in N$: dans ce cas, nous avons $B \subseteq N_G(u)$ par définition de l'ensemble N. En particulier, cela implique que $ub_d \in E(G)$. Puisque σ_H est un ordre parapluie de H, il suit que $vb_d \in E(H)$ et $B_H \subseteq N_H(v)$ (voir Figure 4.8). Puisque $|B_L| = |B_R| = k+1$, et vu que $|F| \leqslant k$, nous savons que $N_G(v) \cap B_L \neq \emptyset$ et $N_G(v) \cap B_R \neq \emptyset$. Comme précédemment, l'Observation 4.17 implique alors que $vm \in E(G)$, contredisant l'hypothèse.

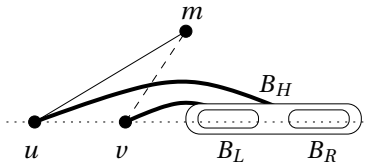


FIGURE 4.8 : Illustration du cas (i).

(ii) $u \in R$, $v \notin R$: puisque $um \in E(G)$, $B_R \subseteq N_G(u)$. Vu que σ_H est un ordre parapluie, et comme $u <_{\sigma_H} v$, nous avons $B_R \subseteq N_H(v)$ (rappelons que $B_R \subseteq B_H$). Maintenant, comme $|B_R| = k+1$, il suit que $N_G(v) \cap B_R \neq \emptyset$. Encore une fois, l'Observation 4.17 nous permet de conclure que $vm \in E(G)$, contredisant l'hypothèse.

(iii) $u, v \in R$: le dernier cas à considérer est quand u et v appartiennent à R. Observons que, dans ce cas, l'Affirmation 4.16 implique que $uv \in E(G)$. Cependant, u et v ne peuvent pas être vrais

jumeaux dans H : dans le cas contraire, puisque $N_M(v) \subseteq N_M(u)$, u et v auraient été inversés lors de notre phase préliminaire de modification de l'ordre σ_H. Cela implique qu'il existe un sommet $w \in V(H)$ qui distingue u de v dans H. Nous considérons les différents cas possibles.

(a) Supposons tout d'abord que $w <_{\sigma_H} u$ et $uw \in E(H)$, $vw \notin E(H)$. Nous choisissons le premier sommet w satisfaisant ces propriétés par rapport à l'ordre σ_H. Deux cas peuvent arriver : dans un premier temps, supposons que l'arête uw soit une arête de G. Alors, comme $wm \notin E(G)$ par choix de u, $\{u, v, w, m\}$ induit une griffe de G contenant un sommet de B, ce qui est impossible (voir Figure 4.9 (*a*) en ignorant le sommet u'). Il suit que $uw \in F$. Par l'Observation 4.6, nous savons que uw n'est pas une arête extrémale de σ_H. Par choix de w et puisque $vw \notin E(H)$, il existe un sommet $u' \in V(H)$ tel que $u <_{\sigma_H} u' <_{\sigma_H} v$ et $u'w$ est une arête extrémale de σ_H. En particulier, observons que $u'w \in E(G)$ par l'Observation 4.6. Par choix de v, nous avons $u'm \in E(G)$ et donc $u' \in L \cup R \cup N$. Remarquons maintenant que $u'v \notin E(G)$: en effet, dans ce cas $\{u', v, w, m\}$ induirait une griffe dans G contenant m, ce qui est impossible. Puisque R est une clique de G par l'Affirmation 4.16, il suit que $u' \in L \cup N$. De plus, puisque $u'm \in E(G)$, nous savons que $B_L \subseteq N_G(u')$, et donc $B_L \subseteq N_H(u')$. Comme $u' <_{\sigma_H} v <_{\sigma_H} B_L$, nous avons alors $B_L \subseteq N_H(v)$. Nous concluons ce cas comme dans la configuration (ii) : puisque $|B_L| = k+1$, $N_G(v) \cap B_L \neq \emptyset$ et donc $vm \in E(G)$ (Observation 4.17).

(b) Dans la suite, nous pouvons donc supposer que tous les sommets distinguant u et v sont situés *après* u dans σ_H. Autrement dit, $uw' \in E(H)$ implique $vw' \in E(H)$ pour tout $w' <_{\sigma_H} u$. Supposons maintenant qu'il existe $w \in V(H)$ tel que $b_d <_{\sigma_H} w$ et $uw \notin E(H)$, $vw \in E(H)$. En particulier, cela signifie que $B_L \subseteq N_H(v)$. Comme $|B_L| = k+1$, nous avons $B_L \cap N_G(v) \neq \emptyset$ et donc $vm \in E(G)$ par l'Observation 4.17. Supposons maintenant qu'il existe un sommet $w \in V(H)$ distinguant u et v et vérifiant $v <_{\sigma_H} w <_{\sigma_H} b_p$, $uw \notin E(H)$, $vw \in E(H)$. Dans ce cas, puisque $uw \notin E(H)$, il suit que $B_H \cap N_H(u) = \emptyset$, et donc $B \cap N_G(u) = \emptyset$, ce qui est impossible puisque $u \in R$. En dernier lieu, nous supposons donc qu'il existe $w \in B_H$ distinguant u et v, *i.e.* $b_p <_{\sigma_H} w <_{\sigma_H} b_d$ est tel que $uw \notin E(H)$ et $vw \in E(H)$. Comme précédemment, nous choisissons le dernier sommet (par rapport à σ_H) distinguant u de v. En d'autres termes, nous supposons que $vw'' \notin E(H)$ pour tout $w <_{\sigma_H} w''$ (voir Figure 4.9 (*b*) en ignorant le sommet u').

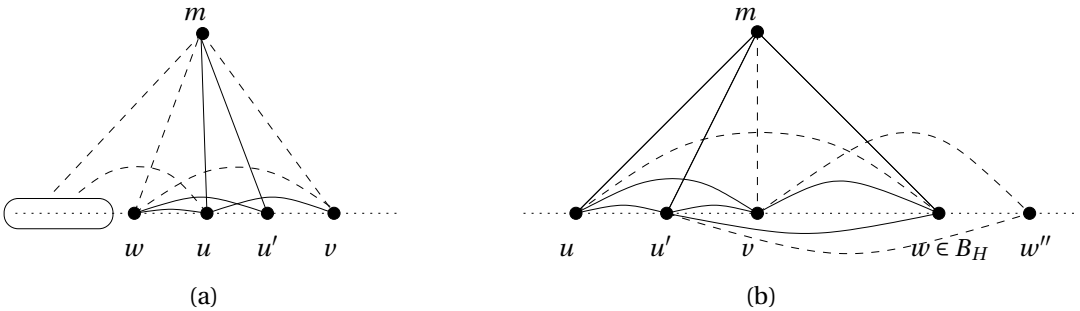


FIGURE 4.9 : Cas (*a*) : u et v sont distingués par un sommet $w <_{\sigma_H} u$. Cas (*b*) : u et v sont distingués par un sommet $w \in B_H$.

Si l'arête vw appartient à $E(G)$, alors l'ensemble $\{u, m, w, v\}$ induit un C_4 dans C contenant un sommet de B, ce qui est impossible par hypothèse. Donc, l'arête vw appartient à F et n'est pas extrémale par l'Observation 4.6. En particulier, par choix de w, cela implique qu'il existe un sommet $u' \in V(H)$ tel que $u <_{\sigma_H} u' <_{\sigma_H} v$ et $u'w$ est une arête extrémale de σ_H (et appartient donc à $E(G)$). Maintenant, par choix de v, nous savons que $u'm \in E(G)$. De plus, les sommets u' et v sont des vrais jumeaux dans H : en effet, supposons par contradiction que ce ne soit pas le cas. Il existe alors un sommet d qui distingue u de v. Cependant, par construction, d ne peut pas être tel que $d <_{\sigma_H} u$ (autrement d distinguerait u et v). De plus, d ne peut pas vérifier $u <_{\sigma_H} d <_{\sigma_H} w$ (dans ce cas, d serait adjacent à u' et v) ni $w <_{\sigma_H} d$ (par choix de w). Il n'existe donc pas de tel sommet d, et u' et v sont vrais jumeaux dans H. Cette observation induit une contradiction puisque nous avons supposé par construction que $N_M(x) \subseteq N_M(y)$ pour tous sommets x et y tels que $x <_{\sigma_H} y <_{\sigma_H} b_p$.

Les configurations où u appartient à L sont similaires. Ceci conclut donc la preuve de l'Affirmation 4.19. ◇

Nous prouvons maintenant l'affirmation suivante, qui nous permettra de conclure. Comme nous allons le voir, la preuve de l'Affirmation 4.20 n'utilise à aucun moment le fait que les sommets de H n'appartiennent pas à M. Il suivra alors que les sommets de M peuvent être insérés dans σ_H de manière itérative en préservant un ordre parapluie à chaque étape.

Affirmation 4.20. *Tout sommet $m \in M$ peut être ajouté au graphe H tout en préservant la propriété parapluie.*

Preuve. Soit $m \in M$. Nous dénotons par H_m le graphe $H \cup \{m\}$, et considérons ce graphe en ordonnant $V(H)$ en fonction de l'ordre σ_H (voir Figure 4.10). Soit h_i (resp. h_j) le sommet de σ_H d'indice minimal (resp. maximal) tel que $h_i m \in E(G)$ (resp. $mh_j \in E(G)$). Par construction, nous avons $h_{i-1}m, h_{j+1}m \notin E(G)$ et l'Affirmation 4.18 implique que $h_{i-1} <_{\sigma_H} b_p$ et $b_d <_{\sigma_H} h_{j+1}$. De plus, nous avons $B_H \subseteq N_{H_m}(m)$, où $N_{H_m}(m) = \{w \in V(H) \ : \ h_i \leqslant_{\sigma_H} w \leqslant_{\sigma_H} h_j\}$ (Affirmation 4.19). Finalement, remarquons que $h_{i-1}h_{j+1} \notin E(G)$: en effet, dans le cas contraire, l'ordre σ_m défini dans l'Affirmation 4.19 ne serait pas un ordre parapluie. Cette situation est décrite dans la Figure 4.10. Pour tout sommet $v \in N_{H_m}(m)$, nous définissons $N^-(v)$ (resp. $N^+(v)$) comme l'ensemble de sommets $\{w \in V(H) \ : \ w <_{\sigma_H} h_i \text{ et } wv \in E(H)\}$ (resp. $\{w \in V(H) \ : \ h_j <_{\sigma_H} w \text{ et } wv \in E(H)\}$). Observons que pour tout sommet $v \in N_{H_m}(m)$, s'il existe deux sommets $x \in N^-(v)$ et $y \in N^+(v)$ tels que $xv, yv \in E(G)$, alors $\{v, x, y, m\}$ induit une griffe de G contenant m, ce qui est impossible. Nous considérons maintenant le sommet $c_{h_{i-1}}$ défini comme le voisin de h_{i-1} d'indice maximal dans σ_H. De manière similaire, nous définissons $c_{h_{j+1}}$ comme le voisin de h_{j+1} d'indice minimal dans σ_H. Remarquons que comme $h_{i-1}h_{j+1} \notin E(G)$, $c_{h_{i-1}}$ et $c_{h_{j+1}}$ appartiennent tous deux à $N_{H_m}(m)$. Nous étudions le comportement de ces deux sommets pour conclure la preuve. Supposons dans un premier temps que $c_{h_{j+1}} \leqslant_{\sigma_H} c_{h_{i-1}}$. Nous posons $U := \{w \in V(H) \ : \ c_{h_{j+1}} \leqslant_{\sigma_H} w \leqslant_{\sigma_H} c_{h_{i-1}}\}$. Remarquons que nous avons $c_{h_{i-1}} <_{\sigma_H} b_d$ et $b_p <_{\sigma_H} c_{h_{j+1}}$: en effet, supposons par exemple et par contradiction que $b_d \leqslant_{\sigma_H} c_{h_{i-1}}$. Dans ce cas, $B_H \subseteq N_H(h_{i-1})$ et donc, en utilisant les mêmes arguments que précédemment nous concluons que $h_{i-1}m \in E(G)$: contradiction. Il suit en particulier que U est inclus dans B_H. Nous définissons maintenant $U_1 \subseteq U$ comme l'ensemble de sommets $u \in U$ tels qu'il existe $w \in N^+(u)$ vérifiant $uw \in E(G)$, et posons $U_2 := U \setminus U_1$. Soit $u_1 \in U_1$: par construction,

observons que $u_1 w' \in F$ pour tout sommet $w' \in N^-(u_1)$. En effet, comme expliqué précédemment, si $u_1 w' \in E(G)$ alors l'ensemble $\{u_1, w, w', m\}$ induit une griffe de G, ce qui est impossible. De manière similaire, étant donné $u_2 \in U_2$, $u_2 w'' \in F$ pour tout sommet $w'' \in N^+(u_2)$. Nous réordonnons maintenant les sommets de U comme suit : nous plaçons tout d'abord les sommets de U_2 puis les sommets de U_1, respectant l'ordre induit par σ_H sur chacun de ces ensembles. De plus, nous supprimons de $E(H)$ toutes les arêtes entre U_1 et $N^-(U_1)$ et toutes les arêtes entre U_2 et $N^+(U_2)$. Rappelons que ces arêtes appartiennent à F par construction. Nous affirmons que l'insertion du sommet m *entre* U_1 et U_2 dans l'ordre ainsi obtenu permet de créer un ordre parapluie pour le graphe $H \cup \{m\}$ (voir Figure 4.10). En effet, par l'Affirmation 4.19 et puisque $U \subseteq B_H$, nous savons que l'ordre parapluie est respecté entre m et les sommets de $V(H) \setminus B_H$.

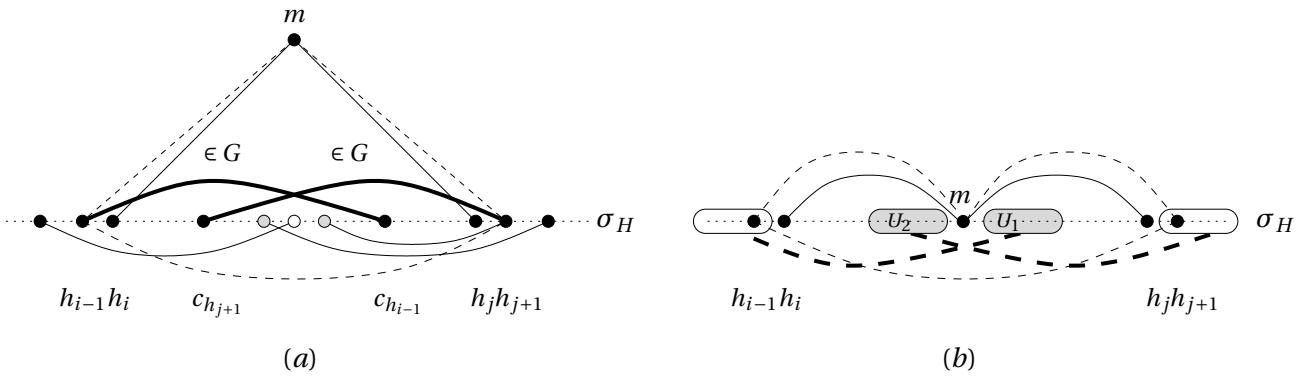


FIGURE 4.10 : Illustration du nouvel ordre appliqué sur σ_H. Sur la gauche, les sommets gris représentent les sommets de U_2, et le sommet blanc est un sommet de U_1.

Maintenant, remarquons que : (i) il n'existe aucune arête entre U_1 et $\{w \in V \,:\, w \leqslant_{\sigma_H} h_{i-1}\}$; (ii) il n'existe aucune arête entre U_2 et $\{w \in V \,:\, h_{j+1} \leqslant_{\sigma_H} w\}$; (iii) toutes les arêtes entre $N_{H_m}(m)$ et $U_1 \cup U_2$ sont préservées ; (iv) les arêtes entre U_1 et $\{w \in V \,:\, h_{j+1} \leqslant_{\sigma_H} w\}$ et entre U_2 et $\{w \in V \,:\, w \leqslant_{\sigma_H} h_{i-1}\}$ sont inchangées. Toutes ces propriétés impliquent que le nouvel ordre respecte la propriété parapluie, ce qui conclut cette partie de la preuve.

Il reste maintenant à considérer le cas où $c_{h_{i-1}} <_{\sigma_H} c_{h_{j+1}}$. Pour ce faire, nous dénotons par c_{h_i} (resp. c_{h_j}) le voisin de h_i (resp. h_j) dans $N_{H_m}(m)$ d'indice maximal (resp. minimal) dans σ_H. Observons en particulier que $c_{h_{i-1}} \leqslant_{\sigma_H} c_{h_i}$ et $c_{h_j} \leqslant_{\sigma_H} c_{h_{j+1}}$ (voir Figure 4.11). Deux cas peuvent se produire :

(i) Dans un premier temps, supposons que $c_{h_i} <_{\sigma_H} c_{h_j}$ (voir Figure 4.11). En particulier, cela implique que $h_i h_j \notin E(G)$. Si c_{h_i} et c_{h_j} sont consécutifs dans σ_H, alors insérer m entre c_{h_i} et c_{h_j} résulte en un ordre parapluie : en effet, c_{h_j} (resp. c_{h_i}) n'a aucun voisin avant (resp. après) h_i (resp. h_j). Maintenant, s'il existe un sommet $w \in V$ tel que $c_{h_{i-1}} <_{\sigma_H} w <_{\sigma_H} c_{h_{j+1}}$, alors l'ensemble $\{m, h_i, w, h_j\}$ induit une griffe de G contenant un sommet de B, ce qui n'est pas possible.

(ii) Le second cas à considérer est $c_{h_j} \leqslant_{\sigma_H} c_{h_i}$. Remarquons que, dans ce cas, $h_i c_{h_i} \notin E(G)$ et $h_j c_{h_j} \notin E(G)$. Ainsi, m et les sommets de $\{w \in V \,:\, c_{h_j} <_{\sigma_H} w <_{\sigma_H} c_{h_i}\}$ sont des vrais jumeaux du graphe H_m : en effet, leur voisinage commun est $\{w \in V \,:\, h_i \leqslant_{\sigma_H} w \leqslant_{\sigma_H} h_j\}$. De ce fait, in-

sérer m juste avant c_{h_i} (ou n'importe où entre c_{h_i} et c_{h_j} ou juste après c_{h_j}) résulte en un ordre parapluie.

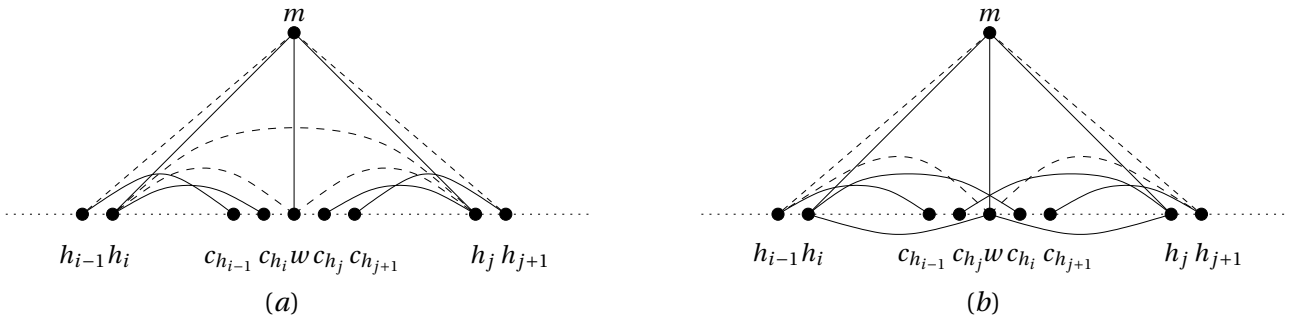


FIGURE 4.11 : Les différentes configurations possibles pour $c_{h_{i-1}} <_{\sigma_H} c_{h_{j+1}}$. Cas (a) : $c_{h_i} <_{\sigma_H} c_{h_j}$. Cas (b) : $c_{h_j} \leqslant_{\sigma_H} c_{h_i}$.

◇

Cela conclut la preuve du Lemme 4.15. □

4.3.2 Extraire un K-join propre depuis un K-join

Soit (G,k) une instance de PROPER INTERVAL COMPLETION. Nous démontrons maintenant qu'il est possible d'extraire un K-join propre de tout K-join de G contenant un nombre suffisant de sommets (par rapport au paramètre k).

Lemme 4.21. *Soient (G,k) une instance positive de* PROPER INTERVAL COMPLETION *réduite par la Règle 4.3. Il existe au plus k^2 griffes avec des ensembles de feuilles distincts dans G, et au plus k^2+2k sommets de G sont feuilles de griffes. De manière similaire, il existe au plus $2k^2+2k$ sommets de G contenus dans des C_4.*

Preuve. Soit F une k-complétion de G. Puisque G est une instance positive de PROPER INTERVAL COMPLETION, toute griffe et tout C_4 de G contient une non-arête qui appartient à l'ensemble F. Soit uv une paire de F. Comme G est réduit par la Règle 4.3, il existe au plus k sommets de G qui forment les feuilles d'une griffe avec u et v. De ce fait, il existe au plus $(k+2)k = k^2+2k$ sommets de G qui sont des feuilles de griffes (rappelons que G contient au plus k griffes avec des feuilles *disjointes*). Cela implique également que G contient au plus k^2 griffes aces des ensembles de feuilles distincts. De manière similaire, il existe au plus k non-arêtes de G, et donc au plus $2k$ sommets, qui forment un C_4 avec la non-arête uv. Ainsi, au plus $k\cdot(2k+2) = 2k^2+2k$ sommets de G sont contenus dans un C_4. □

Lemme 4.22. *Soient (G,k) une instance positive de* PROPER INTERVAL COMPLETION *réduite par les Règles 4.2 et 4.3, et $B \subseteq V$ un K-join de G. Il existe au plus k^3+4k^2+5k+1 sommets de B contenus dans une griffe ou un C_4.*

Preuve. Par le Lemme 4.21, il existe au plus $3k^2+4k$ sommets de G qui sont feuilles d'une griffe ou contenus dans un C_4. Dans un premier temps, nous supprimons ces sommets de B, et dénotons par B' l'ensemble ainsi obtenu. Remarquons que B' définit également un K-join de G (Observation 4.10). Maintenant, nous supprimons de B' tous les sommets n'appartenant pas à une griffe et dénotons par B'' l'ensemble obtenu. Comme précédemment, B'' définit un K-join de G. Enfin, nous contractons chaque ensemble de vrais jumeaux de B'', et appelons abusivement B'' l'ensemble ainsi obtenu. Comme G est réduit par la Règle 4.2, nous savons que tout ensemble contracté contient au plus $k+1$ sommets.

Nous considérons maintenant un ordre parapluie $\sigma_{B''} := b_1 \dots b_r$ de l'ensemble B'', où $r = |B''|$. Observons que, par construction, tout sommet de B'' est centre d'une griffe de G. Nous allons construire un ensemble de $r-1$ griffes avec des ensembles de feuilles distincts, ce qui impliquera $r \leq k^2+1$ (Lemme 4.21). Pour ce faire, remarquons que puisque b_i et b_{i+1} ne sont pas des vrais jumeaux pour $i \in [r-1]$, il existe un sommet $c_i \notin B''$ qui distingue b_i et b_{i+1}. En d'autres termes, le sommet c_i est tel que $b_i c_i \notin E$ et $b_{i+1} c_i \in E$ ou $b_i c_i \in E$ et $b_{i+1} c_i \notin E$. Nous supposons sans perte de généralité que la première configuration est vérifiée. Comme B'' est un K-join, il suit que tous les sommets c_i, $i \in [r-1]$ sont distincts. Pour tout $i \in [r-1]$, nous allons construire une griffe contenant c_i comme feuille. Comme b_{i+1} est centre d'une griffe par construction, il existe un ensemble indépendant $\{u,v,w\} \subseteq V \setminus B$ qui est dominé par b_{i+1}. Si $c_i \in \{u,v,w\}$, nous avons trouvé une griffe contenant c_i comme feuille. De ce fait, nous supposons que $c_i \notin \{u,v,w\}$. En particulier, cela implique que b_i domine $\{u,v,w\}$: en effet, dans le cas contraire, un des sommets de $\{u,v,w\}$ serait adjacent à b_{i+1} et non adjacent à b_i, et nous choisirions ce sommet comme étant c_i.

Nous étudions la relation entre $\{u,v,w\}$ et c_i pour conclure. Si deux éléments de $\{u,v,w\}$, disons u et v, sont adjacents à c_i, alors $\{u,c_i,v,b_i\}$ induit un C_4 de G contenant b_i, ce qui est impossible. Il suit qu'au moins deux éléments de $\{u,v,w\}$, disons u et v, ne sont pas adjacents à c_i. Cela implique alors que $\{b_{i+1},u,v,c_i\}$ est une griffe de centre b_{i+1} contenant c_i comme feuille. Observons que toutes les griffes ainsi construites ont des ensembles de feuilles distincts, vu que que les sommets c_i, $i \in [r-1]$, sont distincts. Par le Lemme 4.21, il existe au plus k^2 telles griffes (*i.e.* $r-1 \leqslant k^2$), impliquant que B'' contient au plus k^2+1 éléments et, par construction, B'' contient au plus $(k+1)\cdot(k^2+1)$ sommets. Comme B' est obtenu à partir de B'' en ajoutant des sommets non contenus dans des griffes ou des C_4, B' contient au plus $(k+1)\cdot(k^2+1)$ sommets contenus dans des griffes ou des C_4. En regroupant les différents arguments, nous obtenons donc que B contient au plus $(k+1)\cdot(k^2+1)+3k^2+4k = k^3+4k^2+5k+1$ sommets appartenant à des griffes ou des C_4 dans G. □

Remarque. Puisque tout sous-ensemble d'un K-join est un K-join par l'Observation 4.10, le Lemme 4.22 implique que tout K-join B de taille supérieure à k^3+4k^2+5k+1 contient un K-join propre B_p tel que $|B_p| \geqslant |B| - (k^3+4k^2+5k+1)$.

4.3.3 Borner la taille d'un K-join

Pour conclure, nous démontrons que la Règle 4.4 permet de borner la taille des K-joins d'une instance (G,k) de PROPER INTERVAL COMPLETION.

Observation 4.23. *Soit (G,k) une instance positive de* PROPER INTERVAL COMPLETION *réduite par les Règles 4.2 à 4.4. Tout K-join de G contient au plus k^3+4k^2+7k+3 sommets.*

Preuve. Soit $B \subseteq V$ un K-join de G. Supposons par contradiction que $|B| > k^3 + 4k^2 + 7k + 3$. Par le Lemme 4.22, nous savons que B contient un K-join propre de taille supérieure à $|B| - (k^3 + 4k^2 + 5k + 1) \geqslant 2k + 1$. Il suit que le graphe G n'est pas réduit par la Règle 4.4, ce qui est une contradiction. □

4.3.4 Couper les 1-branches

Nous démontrons maintenant comment réduire les 1-branches d'une instance (G, k) de PROPER INTERVAL COMPLETION. Dans un premier temps, nous démontrons qu'il existe toujours une k-complétion n'affectant que les derniers sommets d'une 1-branche (voir Figure 4.12). Cela nous permettra en particulier de concevoir une règle de réduction *supprimant* les autres sommets. Les complétions étant indépendantes entre les composantes connexes de G, nous pouvons supposer sans perte de généralité que G est connexe.

Lemme 4.24 (1-branche)**.** *Soient (G, k) une instance connexe de* PROPER INTERVAL COMPLETION *et $B \subseteq V$ une 1-branche de G. Supposons que $|B^R| \geqslant 2k + 1$ et dénotons B_d les 2k +1 derniers sommets de B^R. Il existe une k-complétion F telle que l'ordre parapluie du graphe $G + F$ respecte l'ordre induit par σ_B sur l'ensemble $B_{b_F} := \{w \in B : b_1 \leqslant_{\sigma_B} w \leqslant_{\sigma_B} N_B[b_F]\}$, où $b_F \in B_d$. De plus, les sommets de B_{b_F} sont les premiers dans l'ordre parapluie de $G + F$.*

Preuve. Soient F une k-complétion de G, $H := G + F$ le graphe d'intervalle propre correspondant et σ_H l'ordre parapluie de H. Puisque $|B_d| = 2k + 1$ et $|F| \leqslant k$, il existe un sommet $b_F \in B_d$ non affecté par F. Nous dénotons N_D l'ensemble des voisins de b_F qui sont *après* b_F dans σ_B, $B_{b_F} := \{w \in B : b_1 \leqslant_{\sigma_B} w \leqslant_{\sigma_B} N_G[b_F]\}$ et $C := V \setminus B_{b_F}$ (voir Figure 4.12). Observons en particulier que, par définition de la clique d'attachement A_d (qui correspond à $N_B[b_{|B|}]$), nous avons $A_d \not\subset B_{b_F}$, et ainsi $A_d \cap C \neq \emptyset$.

Affirmation 4.25. *(i) $G[C]$ est un graphe connexe et (ii) $b_F <_{\sigma_H} C$ ou $C <_{\sigma_H} b_F$ est vérifié.*

Preuve. La propriété (i) vient du fait que G est connexe et que, par construction, $A_d \cap C \neq \emptyset$ et $G[A_d \cap C]$ est connexe. Pour prouver (ii), supposons par contradiction qu'il existe deux sommets $c, c' \in C$ tels que, sans perte de généralité, $c <_{\sigma_H} b_F <_{\sigma_H} c'$. Puisque $G[C]$ est connexe par (i), il existe un chemin entre c et c' qui évite les sommets de $N_G[b_F]$. Remarquons que $N_G[b_F]$ est égal à $N_H[b_F]$ puisque b_F est un sommet non affecté par F. Il suit qu'il existe deux sommets $c_1, c_2 \in C$ tels que $c_1 <_{\sigma_H} b_F <_{\sigma_H} c_2$ et $c_1c_2 \in E(G)$, $c_1b_F, c_2b_F \notin E(H)$, contredisant le fait que σ_H est un ordre parapluie. ◇

Dans la suite de la preuve, nous supposons sans perte de généralité que $b_F <_{\sigma_H} C$ est vérifié. Nous considérons maintenant l'ordre σ sur H défini comme suit : nous plaçons en premier les sommets de B_{b_F} en respectant l'ordre induit par σ_B, et plaçons ensuite les sommets de C en respectant l'ordre induit par σ_H (voir Figure 4.12). De plus, nous construisons une complétion $F' \subseteq F$ comme suit : nous supprimons de F toutes les paires ayant leurs deux éléments contenus dans B_{b_F}, et les paires contenant un sommet de $B_{b_F} \setminus N_D$ et un sommet de C. En d'autres termes, nous posons :

$$F' := F \setminus (F[B] \cup F[(B_{b_F} \setminus N_D) \times C])$$

Enfin, nous supprimons de F' (de manière itérative) toutes les arêtes extrémales de σ qui appartiennent à F. Dans un abus de notation, nous appelons F' la complétion ainsi obtenue.

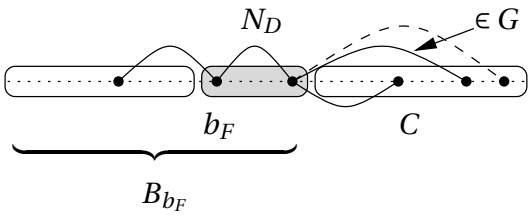
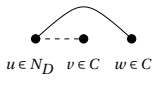


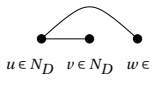
FIGURE 4.12 : Illustration de la construction de l'ordre σ à partir de l'ordre parapluie σ_H.

Affirmation 4.26. *L'ensemble F' est une k-complétion de G respectant les propriétés du Lemme 4.27.*

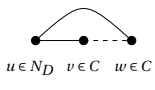
Preuve. Nous prouvons que l'ordre σ est un ordre parapluie du graphe $H' := G + F'$. Comme $F' \subseteq F$ par construction, le résultat suivra. Nous supposons par contradiction que σ n'est pas un ordre parapluie pour H. Par définition de F', les graphes $H'[B_{b_F}]$ et $H'[C]$ sont des graphes d'intervalle propre. Cela signifie qu'il existe un ensemble de sommets $S := \{u, v, w\}$, $u <_\sigma v <_\sigma w$ intersectant à la fois B_{b_F} et C et ne respectant pas la propriété parapluie. Deux cas peuvent se produire : (1) $uw \in E(H'), uv \notin E(H')$ ou (2) $uw \in E(H')$, $vw \notin E(H')$. Puisque ni H' ni G ne contiennent d'arête entre $B_{b_F} \setminus N_D$ et C (rappelons que $(B_{b_F} \setminus N_D) \subseteq B^R$), il suit que S intersecte N_D et C. Nous étudions les différentes configurations possibles :



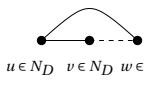
(i) (1) est vérifié et $u \in N_D$, $v, w \in C$: puisque l'ensemble $E(N_D, C)$ des arêtes entre N_D et C est le même dans H et dans H', il suit que $uv \notin E(H)$. Puisque σ_H est un ordre parapluie de H, nous avons $v <_{\sigma_H} u <_{\sigma_H} w$ ou $v <_{\sigma_H} w <_{\sigma_H} u$ (rappelons que les sommets de C sont ordonnés de la même manière dans σ et σ_H). Maintenant, par l'Affirmation 4.25, nous savons que $b_F <_{\sigma_H} \{v, w\}$. En particulier, puisque $b_F u \in E(G)$ et $b_F v, b_F w \notin E(G)$, cela implique que σ_H n'est pas un ordre parapluie dans les deux cas : contradiction.



(ii) (1) est vérifié et $u, v \in N_D$, $w \in C$: ce cas est impossible puisque N_D est une clique de G et donc de H'.



(iii) (2) est vérifié et $u \in N_D$, $v, w \in C$: ce cas est similaire à la configuration (i). Observons que nous pouvons supposer $uv \in E(H')$ (sinon (i) est vérifié). Par construction, $vw \notin E(H)$ et ainsi $v <_{\sigma_H} w <_{\sigma_H} u$ ou $v <_{\sigma_H} u <_{\sigma_H} w$. Le premier cas contredit le fait que σ_H est un ordre parapluie puisque $uv \in E(H)$. Dans le second cas, puisque σ_H est un ordre parapluie, cela signifie que $b_F v \in E(H)$. Comme b_F n'est pas affecté par F et $b_F v \notin E(G)$, il en résulte une contradiction.



(iv) (2) est vérifié et $u, v \in N_D$, $w \in C$: premièrement, si $uw \in E(G)$, alors nous obtenons une contradiction puisque $N_C(u) \subseteq N_C(v)$ dans G. Nous pouvons donc supposer $uw \in F'$. Par construction de F', nous savons que uw n'est pas une arête extrémale de σ. Ainsi, il existe une arête extrémale e dans σ appartenant à $E(G)$ et contenant uw. Plusieurs cas sont possibles : $e = uw'$ avec $w <_\sigma w$', $e = u'w$ avec $u' <_\sigma u$ ou $e = u'w'$ avec $u' <_\sigma u <_\sigma w <_\sigma w'$. Les trois configurations sont représentées dans la Figure 4.13.

Dans le premier cas, nous avons $vw' \in E(G)$ (puisque $N_C(u) \subseteq N_C(v)$ dans G) et nous sommes dans la configuration (i) avec l'ensemble de sommets $\{v, w, w'\}$. Dans le second cas, puisque $u'w \in E(G)$, nous obtenons une contradiction puisque $N_C(u') \subseteq N_C(v)$ dans G (observons que $u' \in B$ par construction). Finalement, dans le dernier cas, $uw', vw' \in E(G)$ puisque $N_C(u') \subseteq$

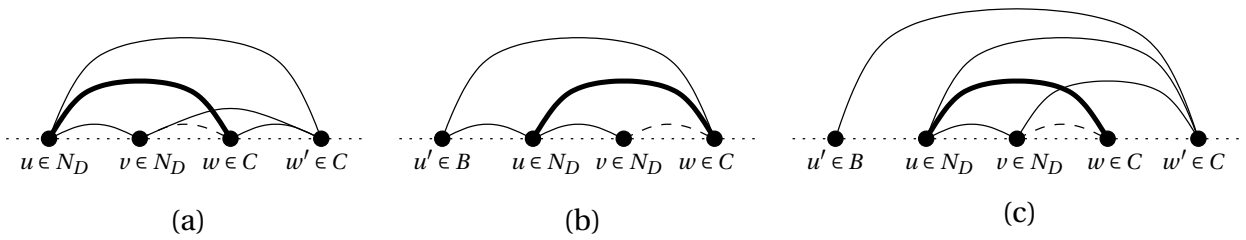


FIGURE 4.13 : Illustration des différents cas de la configuration (iv).

$N_C(u) \subseteq N_C(v)$ dans G, et nous sommes une nouvelle fois dans la configuration (i) avec l'ensemble de sommets $\{v, w, w'\}$.

◇

Nous avons donc prouvé qu'il existe une k-complétion F' et un sommet $b_{F'} = b_F$ telle que l'ordre parapluie σ de $G + F'$ respecte l'ordre des sommets B_{b_F}, qui sont les premiers de l'ordre σ. Ceci conclut la preuve du Lemme 4.27. □

Le Lemme 4.27 nous permet de concevoir la règle de réduction suivante. À nouveau, nous démontrons uniquement la validité de cette règle, et démontrerons qu'elle peut être appliquée en temps polynomial dans la Section 4.4.

Règle 4.5 (1-branche)**.** *Soient* (G, k) *une instance de* PROPER INTERVAL COMPLETION *et* $B \subseteq V$ *une* 1*-branche de* G *telle que* B^R *contient au moins* $2k+1$ *sommets. Supprimer* $B^R \setminus B_d$ *de* G*, où* B_d *dénote les* $2k+1$ *derniers sommets de* B^R*.*

Lemme 4.27. *La Règle 4.5 est valide.*

Preuve. Nous dénotons par $G' := G \setminus (B^R \setminus B_d)$ le graphe réduit. Observons que toute k-complétion de G est une k-complétion de G' puisque la classe des graphes d'intervalle propre est héréditaire (Observation 4.11). Inversement, soit F' une k-complétion de G'. Sans perte de généralité, nous supposons que F' respecte les conditions du Lemme 4.27. Nous posons $H := G' + F'$ pour dénoter le graphe d'intervalle propre résultant, et dénotons σ_H l'ordre parapluie associé. Par le Lemme 4.27, nous savons qu'il existe un sommet $b_{F'} \in B_d$ et tel que l'ordre de $B_{b_{F'}} := \{v \in B_d \,:\, v \leqslant_{\sigma_B} N_G[b_{F'}]\}$ induit par σ_H est le même que l'ordre de $B_{b_{F'}}$ dans σ_B. De plus, les sommets de $B_{b_{F'}}$ sont les premiers de l'ordre σ_H. Puisque $N_{G\setminus(B^R\setminus B_d)}(B^R \setminus B_d) \subseteq N_G[b_{F'}]$, il suit que les sommets de $B^R \setminus B_d$ peuvent être insérés au début de σ_H tout en respectant la propriété parapluie, et donc F' est une k-complétion de G. □

Observation 4.28. *Soit* (G, k) *une instance positive de* PROPER INTERVAL COMPLETION *réduite par les Règles 4.1 à 4.5. Les* 1*-branches de* G *contiennent au plus* $k^3 + 4k^2 + 9k + 4$ *sommets.*

Preuve. Soit $B \subseteq V$ une 1-branche de G. Supposons par contradiction que $|B| > k^3 + 4k^2 + 9k + 4$. Puisque G est réduit par la Règle 4.4, nous savons par l'Observation 4.17 que la clique d'attachement A_d de B contient au plus $k^3 + 4k^2 + 7k + 3$ sommets. Cela implique donc que $|B^R| > 2k + 1$, contredisant le fait que G est réduit par la Règle 4.5. □

4.3.5 Couper les 2-branches

Nous démontrons à présent comment borner la taille d'une 2-branche $B \subseteq V$ d'une instance (G,k) de PROPER INTERVAL COMPLETION. Cette dernière règle de réduction nous permettra d'établir un algorithme de noyau contenant $O(k^5)$ sommets dans la Section 4.5. Pour ce faire, nous utilisons la notion de *décomposition en K-joins* définie dans la Section 4.1.1.

Lemme 4.29. *Soient (G,k) une instance de* PROPER INTERVAL COMPLETION *et $B \subseteq V$ une 2-branche de G ayant une décomposition en K-joins $\mathscr{B} := \{B_1,\dots,B_p\}$, $p \geqslant k+4$. Si les cliques d'attachement A_g et A_d appartiennent à la même composante connexe de $G \setminus B^R$, alors G n'admet pas de k-complétion.*

Preuve. Puisque A_g et A_d appartiennent à la même composante connexe de $G \setminus B^R$, nous dénotons par π un plus court chemin entre A_g et A_d dans $G \setminus B^R$. Comme B contient au moins 3 K-joins dans sa décomposition par hypothèse, aucun sommet de A_g n'est adjacent à un sommet de A_d et π a une longueur supérieure ou égale à 2. Nous dénotons par $u \in A_g$ et $v \in A_d$ les extrémités du chemin π. Nous construisons maintenant un chemin induit P_{uv} de longueur au moins $p-1$ entre u et v n'utilisant que des sommets de B. Pour ce faire, remarquons que nous avons $u \in B_1$ et $v \in B_{p-1} \cup B_p$. Nous posons $u_1 := u$ et tant que $v \notin N_B[u_i]$, nous choisissons u_{i+1} comme le voisin de u_i d'indice maximal dans l'ordre parapluie σ_B. Dans ce cas, observons que $u_i \in \cup_{j=1}^{i} B_j$ pour tout $i \in [p]$: en effet, le voisin d'un sommet de $\cup_{j=1}^{i-1} B_j$ d'indice maximal dans σ_B appartient à $\cup_{j=1}^{i} B_j$. En dernier lieu, lorsque $v \in N_B[u_i]$, nous posons $u_{i+1} := v$. Soit l la longueur du chemin ainsi obtenu. Par construction, le chemin $P_{uv} := \{u_1,\dots,u_l\}$ est un chemin induit de $G[B]$ de longueur au moins $p-1$, avec $u_1 = u$, $u_l = v$ et dont les seuls sommets avec des (possibles) voisins dans $G \setminus B$ sont u_1, u_{l-1} et u_l (car $u_1 \in A_g, u_l \in A_d$ et u_{l-1} peut appartenir à A_d). En utilisant π, nous pouvons alors former un cycle induit $C := P_{uv} \cup \pi$ de longueur supérieure ou égale à $k+4$ dans G. Puisque au moins $q-3$ complétions sont nécessaires pour trianguler un cycle de longueur q [53], il suit qu'il n'existe aucune k-complétion pour G. □

L'observation suivante est une implication directe du Lemme 4.29.

Observation 4.30. *Soient (G,k) une instance positive connexe de* PROPER INTERVAL COMPLETION *réduite par les Règles 4.1 à 4.6 et $B \subseteq V$ une 2-branche de G telle que $G \setminus B^R$ est connexe. Alors B a une décomposition en K-joins de taille inférieure ou égale à $k+4$, et contient donc au plus $(k+4)\cdot(k^3+4k^2+5k+1)$ sommets.*

Nous définissons maintenant la règle de réduction permettant de réduire la taille des 2-branches d'une instance (G,k) de PROPER INTERVAL COMPLETION. Pour ce faire, étant donnée une 2-branche $B \subseteq V$, nous posons $A_g := \{b_1,\dots,b_{l'}\}$, où $b_{l'}$ est le voisin de b_1 d'indice maximal dans B, et $A_d := \{b_l,\dots,b_{|B|}\}$, où b_l est le voisin de $b_{|B|}$ d'indice minimal dans B. De manière similaire, nous définissons B_1 et B_2 comme les ensembles $\{b_{l'+1},\dots,b_i\}$ et $\{b_j,\dots,b_{l-1}\}$, où b_i (resp. b_j) est le voisin de $b_{l'+1}$ (resp. b_l) d'indice maximal (resp. minimal) dans B. De plus, nous posons $B_M := B \setminus (A_g \cup B_1 \cup B_2 \cup A_d)$ (voir Figure 4.14).

Règle 4.6 (2-branche)**.** *Soient (G,k) une instance connexe de* PROPER INTERVAL COMPLETION *et $B \subseteq V$ une 2-branche de G telle que $G \setminus B^R$ est non-connexe et $|B| \geqslant 4(k^3+4k^2+5k+1)+4(k+1)$. Supprimer $B_M \setminus (B_M^p \cup B_M^d)$ de G, où B_M^p et B_M^d dénotent les $2k+1$ premiers et derniers sommets de B^M, respectivement. De plus, supprimer toutes les arêtes $E(B_1 \cup B_p^M, B_2 \cup B_d^M)$.*

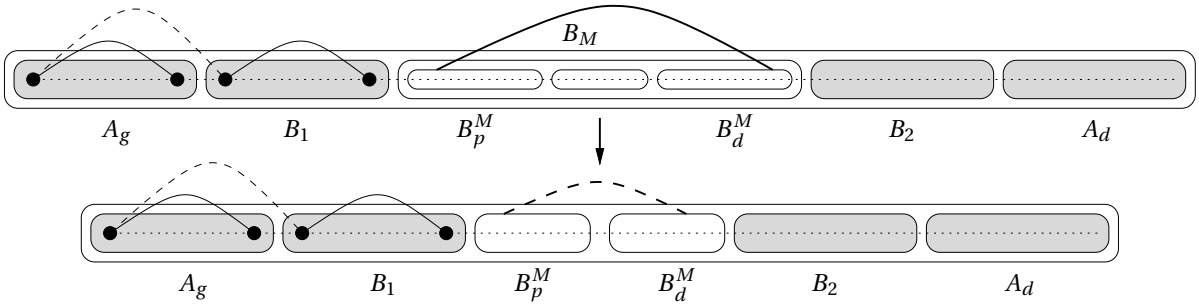


FIGURE 4.14 : Illustration de l'Application de la Règle 4.6.

Lemme 4.31. *La Règle 4.6 est valide.*

Preuve. Nous dénotons par $G' := G \setminus (B^M \setminus (B_p^M \cup B_d^M))$ le graphe réduit. Remarquons premièrement que toute k-complétion de G est une k-complétion de G' puisque la classe des graphes d'intervalle propre est héréditaire (Observation 4.11).

Inversement, soit F' une k-complétion de G'. Observons que $B_1 \cap B_2 = \emptyset$ par choix de B. Nous posons $B' := B_p^M \cup B_1$ et $B'' := B_d^M \cup B_2$. Puisque $|B| \geqslant 4(k^3+4k^2+5k+1)+4(k+1)$ et vu que B_1 et B_2 contiennent au plus (k^3+4k^2+5k+1) sommets (Observation 4.17), nous avons $|B'| \geqslant 2k+1$ et $|B''| \geqslant 2k+1$. Puisque les arêtes $E(B',B'')$ entre B' et B'' ont été supprimées, G' possède deux composantes connexes C_1 et C_2 : en effet, rappelons qu'il n'y a pas d'arêtes entre A_g et B'' (resp. A_d et B') par définition de B_1 (resp. B_2), et que B' et B'' n'ont aucun voisin à l'extérieur de B. Ainsi F' se décompose en deux complétions F'_1 et F'_2, respectivement de $G'_1 := G[C_1]$ et $G'_2 := G[C_2]$, telles que $|F'_1|+|F'_2| \leqslant k$. Maintenant, remarquons que $B'_1 := B' \cup A_g$ définit une 1-branche de G'_1 telle que $B'_1 \setminus A_g$ contient au moins $2k+1$ sommets et ayant A_g comme clique d'attachement. De même, $B''_2 := B'' \cup A_d$ définit une 1-branche de G'_2 telle que $B''_2 \setminus A_d$ contient au moins $2k+1$ sommets et ayant A_d comme clique d'attachement. Les conditions du Lemme 4.27 sont donc vérifiées par G'_1 et G'_2, et nous pouvons supposer que F'_1 et F'_2 respectent les propriétés du Lemme 4.27. Nous considérons les graphes $H_1 := G'_1 + F'_1$ et $H_2 := G'_2 + F'_2$, ainsi que les ordres parapluies σ_{H_1} et σ_{H_2}. Par le Lemme 4.27, nous savons qu'il existe un sommet $b_{F_1} \in B'_d$ (où B'_d dénote les $2k+1$ derniers sommets de $B'_1 \setminus A_g$ dans G'_1) tel que les sommets $\{v \in B'_1 \ : \ v \leqslant_{\sigma_B} N_{G'_1}[b_{F_1}]\}$ sont au début de σ_{H_1} et dans le même ordre que celui induit par σ_B. Ainsi, il est possible d'*insérer* les sommets de $(B^M \setminus (B_p^M \cup B_d^M))$ au début de σ_{H_1} tout en préservant un ordre parapluie. Soit H'_1 le graphe d'intervalle propre ainsi obtenu. De manière similaire, il existe un sommet $b_{F_2} \in B''_d$ (où B''_d dénote les $2k+1$ derniers sommets de $B''_2 \setminus A_d$ dans G'_2) tel que les sommets $B''_{b_{F_2}} := \{v \in B \ : \ v \leqslant_{\sigma_B} N_{G'_2}[b_{F_2}]\}$ sont au début de σ_{H_2} et dans le même ordre que celui induit par σ_B. À nouveau, les sommets de H_2 peuvent être ajoutés au graphe H'_1 tout en préservant un ordre parapluie en insérant les sommets de σ_{H_2} *dans l'ordre inverse* au début de $\sigma_{H'_1}$. Puisque les sommets de $B''_{b_{F_2}}$ sont dans le même ordre dans σ_{H_2} et dans σ_B, il suit que l'ordre ainsi obtenu est un ordre parapluie. Soit H' le graphe d'intervalle propre ainsi obtenu. Pour conclure, observons qu'il est possible de réinsérer les arêtes $E(B',B'')$ dans H' tout en préservant un ordre parapluie, vu que les sommets correspondant aux extrémités des arêtes de $E(B',B'')$ sont ordonnés de la même manière dans $\sigma_{H'}$ et σ_B. Il suit que F' est une k-complétion de G, ce qui conclut la preuve du Lemme 4.31. □

Observation 4.32. *Soit* (G,k) *une instance de* PROPER INTERVAL COMPLETION *réduite par les Règles 4.1 à 4.6. Les* 2*-branches de* G *contiennent au plus* $(k+4)\cdot(k^3+4k^2+5k+1)$ *sommets.*

Preuve. Soit $B \subseteq V$ une 2-branche de G contenue dans une composante connexe C. Si le graphe $G[C \setminus B^R]$ est connexe, alors l'Observation 4.30 implique le résultat. Dans le cas contraire, comme G est réduit par les Règles 4.1 à 4.6, nous savons que $|B| \leqslant 4(k^3+4k^2+5k+1)+4(k+1)$, ce qui est inférieur à $(k+4)\cdot(k^3+4k^2+5k+1)$ dès lors que $k \geqslant 1$ (ce qui peut être supposé sans perte de généralité). □

4.4 Détecter les branches en temps polynomial

Dans les Sections 4.3.4 et 4.3.5, nous avons démontré que les 1-branches et 2-branches d'une instance (G,k) du problème PROPER INTERVAL COMPLETION peuvent être réduites sous certaines conditions. Nous devons donc maintenant démontrer que les structures sur lesquelles s'appliquent ces règles de réduction peuvent être calculées en temps polynomial. La détection d'une branche peut se faire de manière gloutonne, sauf au niveau des cliques d'attachement, ou plusieurs choix peuvent être possibles. Nous allons détecter des branches de cardinalité maximum afin d'assurer une application correcte des règles de réduction. Pour ce faire, nous allons *choisir* la plus grande clique d'attachement possible.

Détecter les 1-branches. Nous détectons dans un premier temps les 1-branches de G. Remarquons que pour tout sommet u de G, l'ensemble $\{u\}$ est une 1-branche de G.

Lemme 4.33. *Soient* (G,k) *une instance de* PROPER INTERVAL COMPLETION *réduite par la Règle 4.1 et* $u \in V$ *un sommet de* G*. Il existe un algorithme qui, en temps* $O(n^2)$*, détecte une* 1*-branche de* G *de taille maximum contenant* u *comme premier sommet.*

Preuve. Pour détecter une 1-branche B de G contenant u comme premier sommet, nous construisons un algorithme qui contient deux parties. Informellement, nous allons dans un premier temps détecter l'ensemble B^R de la 1-branche B contenant u. Nous allons également utiliser le fait qu'une 1-branche admette une décomposition en K-joins, comme défini dans la Section 4.3.5. Nous posons $B_0^R := \{u\}$ et $\sigma_0 := \{u\}$, et allons détecter les K-joins de B^R de manière gloutonne. Une fois que l'ensemble B_{i-1}^R a été défini, nous construisons l'ensemble C_i de sommets de $G \setminus (\cup_{l=1}^{i-1} B_l^R)$ adjacents à au moins un sommet de B_{i-1}^R. Observons en particulier qu'aucun sommet de C_i n'est adjacent à $\cup_{l=1}^{i-2} B_l^R$, puisque, dans le cas contraire, il appartiendrait à C_{i-1}. Deux cas peuvent se produire. Dans un premier temps, supposons que C_i soit une clique et qu'il soit possible d'ordonner les sommets de C_i de telle sorte que pour tout $1 \leqslant j < |C_i|$, $N_{B_{i-1}^R}(c_{j+1}) \subseteq N_{B_{i-1}^R}(c_j)$ et $N_{G \setminus B_{i-1}^R}(c_j) \subseteq N_{G \setminus B_{i-1}^R}(c_{j+1})$, où c_j et c_{j+1} sont deux sommets consécutifs de C_i. Dans ce cas, les sommets de C_i correspondent à un nouveau K-join de la 1-branche en cours de détection. De ce fait, nous posons $B_i^R := C_i$ et nous définissons σ_i comme la concaténation de σ_{i-1} et de l'ordre défini sur C_i. Dans le cas contraire, *i.e.* lorsque les sommets de C_i ne peuvent pas être ordonnés de la sorte ou lorsque C_i n'est pas une clique, cela signifie que les sommets B^R de la 1-branche B en cours de détection ont déjà été considérés, ainsi qu'au moins un sommet de la clique d'attachement A_d ayant des voisins dans B^R. Supposons que le processus s'arrête à l'étape p, et considérons C l'ensemble de sommets de

$G \setminus \cup_{l=1}^{p} B_l^R$ ayant des voisins dans B_p^R. Remarquons que $C \neq \emptyset$ puisque G est réduit par la Règle 4.1. De plus, nous considérons l'ensemble A'_d des sommets de B_p^R adjacents à C. Nous dénotons par B^R l'ensemble $(\cup_{l=1}^{p} B_l^R) \setminus A'_d$ et allons maintenant construire un K-join de taille maximum dans $G \setminus B^R$ contenant A'_d étant compatible avec l'ordre σ_p. Cela nous permettra de déterminer la clique d'attachement A_d de taille maximum de la 1-branche détectée. Par définition, les sommets de C sont les candidats pour compléter la clique d'attachement. Pour détecter cette dernière, nous construisons sur C un graphe orienté $D_C := (V_C, A_C)$ comme suit : il existe un arc de u vers v, avec $u, v \in C$ si (i) $N_{B^R}[v] \subseteq N_{B^R}[u]$ et (ii) $N_{G \setminus B^R}[u] \subseteq N_{G \setminus B^R}[v]$. Ce graphe peut être calculé en temps $O(n^2)$: il suffit de considérer toutes les paires de C, et de vérifier en temps $O(m)$ les conditions sur le voisinage.

Affirmation 4.34. *Le graphe orienté D_C est transitif.*

Preuve. Soient $u, v, w \in V_C$ tels que, sans perte de généralité, $uv, vw \in A_C$. Par définition, cela signifie que $N_{B^R}[v] \subseteq N_{B^R}[u]$ et $N_{B^R}[w] \subseteq N_{B^R}[v]$. De manière similaire, nous avons $N_{G \setminus B^R}[u] \subseteq N_{G \setminus B^R}[v]$ et $N_{G \setminus B^R}[v] \subseteq N_{G \setminus B^R}[w]$. Il suit que $N_{G \setminus B^R}[u] \subseteq N_{G \setminus B^R}[w]$ et $N_{B^R}[w] \subseteq N_{B^R}[u]$, ce qui implique que $uw \in A_C$. ◇

Par construction, un chemin orienté dans D_C correspond à un K-join contenant A'_d et étant compatible avec l'ordre σ_p. Puisqu'il est possible de calculer un chemin de longueur maximale en temps linéaire dans D_C, il suit qu'une 1-branche maximum de G contenant u comme premier sommet peut être calculée en temps $O(n^2)$. □

Détecter les 2-branches. Nous allons maintenant prouver comment les 2-branches d'une instance (G, k) de PROPER INTERVAL COMPLETION peuvent être détectées en temps polynomial. Pour cela, pour toute paire de sommets adjacents, nous détectons dans un premier temps un K-join maximum contenant les sommets de la paire considérée comme premier et dernier sommets. Plus précisément, si $\{u, v\}$ sont deux sommets de G tels que $uv \in E(G)$, alors l'ensemble $\{u, v\}$ est un K-join de G, avec $N := N_G(u) \cap N_G(v)$, $L := N_G(u) \setminus N_G[v]$ et $R := N_G(v) \setminus N_G[u]$. Il suit qu'il existe toujours un K-join ayant u et v comme premier et dernier sommets, et nous allons prouver comment calculer de tels K-joins de taille maximum.

Lemme 4.35. *Soient (G, k) une instance de* PROPER INTERVAL COMPLETION*, et $u, v \in V$ deux sommets de G tels que $uv \in E$. Il existe un algorithme qui, en temps $O(n^3)$ détecte un K-join de G de taille maximum admettant u et v comme premier et dernier sommets.*

Preuve. Nous posons $N := N_G(u) \cap N_G(v)$, $L := N_G(u) \setminus N_G[v]$ et $R := N_G(v) \setminus N_G[u]$. De plus, nous dénotons par $N' \subseteq N$ l'ensemble de sommets contenant N dans leur voisinage clos. Les sommets de cet ensemble sont les candidats pouvant appartenir au K-join recherché. Comme dans la preuve du Lemme 4.33, nous construisons un graphe orienté $D_{N'}$ sur N' de la manière suivante : pour tous sommets $w, w' \in N'$, il existe un arc de w à w' dans $D_{N'}$ si (i) $N_L[w'] \subseteq N_L[w]$ et (ii) $N_R[w] \subseteq N_R[w']$. Observons que décider s'il existe un tel arc requiert un temps $O(n)$, et donc la construction totale du graphe $D_{N'}$ se fait en temps $O(n^3)$. En utilisant les mêmes arguments que dans la preuve de l'Affirmation 4.34, nous montrons que $D_{N'}$ est un graphe orienté transitif. De même, il est possible de calculer un chemin de longueur maximale dans $D_{N'}$ en temps linéaire. Par construction, un tel chemin correspond à un K-join de G admettant u et v comme premier et dernier sommets. □

Références aux K-joins. Maintenant, pour toute arête uv de G, nous calculons un K-join de taille maximum contenant u et v comme extrémités. De plus, nous mémorisons tous les sommets que ce K-join contient. Cela peut se faire en temps $O(n^3m)$ et permet d'obtenir, pour tout sommet $u \in V$, un K-join de taille maximum contenant u. Ces références seront utiles pour calculer les 2-branches.

Lemme 4.36. *Soient (G,k) une instance de* PROPER INTERVAL COMPLETION, *$B \subseteq V$ une 2-branche de G telle que $B^R \neq \emptyset$ et $u \in B^R$. Pour tout K-join B_u maximal pour l'inclusion et contenant u, il existe une arête extrémale uv de σ_B telle que :*

$$B_u = \{w \in B \,:\, u \leq_{\sigma_B} w \leq_{\sigma_B} v\}$$

Preuve. Conformément aux notations de la Définition 4.8, nous utilisons L, R et C pour dénoter la partition de $G \setminus B$ associée à B, et σ_B pour représenter l'ordre parapluie de B. Soit B_u un K-join maximal contenant u. Nous définissons b_p (resp. b_d) comme le premier (resp. dernier) sommet de B_u apparaissant dans B. Comme B_u est maximal et $u \in B^R$, il n'y a aucune arête entre $\{v \in B \,:\, v <_{\sigma_B} b_p\} \cup L \cup C$ et b_d et aucune arête entre $\{v \in B \,:\, b_d <_{\sigma_B} v\} \cup R \cup C$ et b_p. Nous déduisons donc que $B_u \subseteq \{v \in B \,:\, b_p \leqslant_{\sigma_B} v \leqslant_{\sigma_B} b_d\}$. De plus, comme $\{v \in B \,:\, b_p \leqslant_{\sigma_B} v \leqslant_{\sigma_B} b_d\}$ est un K-join et comme B_u est maximal, il suit que $B_u = \{v \in B \,:\, b_p \leqslant_{\sigma_B} v \leqslant_{\sigma_B} b_d\}$. Pour conclure, si $b_p b_d$ n'est pas une arête extrémale de σ_B, alors B_u peut être étendu et n'est donc pas maximal, une contradiction. □

Nous allons maintenant prouver comment détecter les 2-branches B vérifiant $B^R \neq \emptyset$. Observons que détecter de telles branches est suffisant pour notre algorithme de noyau : en effet, réduire uniquement les 2-branches contenant au moins $2 \cdot (k^3 + 4k^2 + 7k + 3)$ sommets nous permet d'obtenir un graphe réduit respectant l'Observation 4.32. De telles branches B vérifient nécessairement $B^R \neq \emptyset$ puisque leurs cliques d'attachement contiennent au plus $2 \cdot (k^3 + 4k^2 + 7k + 3)$ sommets par l'Observation 4.17 (en supposant sans perte de généralité que le graphe est réduit par la Règle 4.4).

Lemme 4.37. *Soient (G,k) une instance de* PROPER INTERVAL COMPLETION, *u un sommet de G et B_u un K-join maximum contenant u. Il existe un algorithme qui, en temps $O(n^2)$, décide si une 2-branche B de G telle que $u \in B^R$ existe et, si oui, qui détecte une telle 2-branche de taille maximum.*

Preuve. Par le Lemme 4.36, nous savons que s'il existe une 2-branche B de G contenant u comme sommet de B^R, alors B_u correspond à un ensemble $\{v \in B \,:\, b_p \leqslant_{\sigma_B} v \leqslant_{\sigma_B} b_d\}$, où $b_p b_d$ est une arête extrémale de σ_B. Conformément aux notations de la Définition 4.8, nous utilisons L_u, R_u et C_u pour dénoter la partition de $G \setminus B_u$ associée à B_u, et σ_{B_u} pour représenter l'ordre parapluie de B_u. Dans G, nous supprimons les sommets de l'ensemble $\{v \in B_u \,:\, v <_{\sigma_B} u\}$, ainsi que les arêtes entre L_u et $\{v \in B_u \,:\, u \leq_{\sigma_{B_u}} v\}$. Nous notons H_1 le graphe ainsi obtenu. Par définition de la 2-branche B en cours de détection, $\{v \in B \,:\, u \leqslant_{\sigma_B} v\}$ correspond à une 1-branche de H_1 contenant u comme premier sommet. De ce fait, en utilisant le Lemme 4.33, nous détectons une 1-branche B_1 de taille maximum contenant u comme premier sommet. Remarquons que les sommets de $\{v \in B \,:\, u \leqslant_{\sigma_B} v\} \cap B_1^R$ doivent être placés au début de l'ordre parapluie de B_1. De manière similaire, nous définissons le graphe H_2 obtenu à partir de G en supprimant l'ensemble de sommets $\{v \in B_u \,:\, u <_{\sigma_B} v\}$, ainsi que les arêtes entre R_u et $\{v \in B \,:\, v \leq_{\sigma_{B_u}} u\}$. De même, nous détectons dans H_2 une 1-branche B_2 contenant u comme premier sommet. Comme précédemment, les sommets de $\{v \in B \,:\, v \leqslant_{\sigma_B} u\} \cap B_2^R$ doivent se trouver au début de l'ordre parapluie de B_2. En réinsérant les arêtes précédemment supprimées, nous obtenons que $B_1 \cup B_2$ est une 2-branche de G de taille maximum contenant u. □

Corollaire 4.38. *Soit* (G,k) *une instance de* PROPER INTERVAL COMPLETION. *Un graphe réduit par les Règles 4.1 à 4.6 peut être calculé en temps polynomial.*

4.5 Borner la taille d'une instance réduite

Théorème 4.39. *Le problème* PROPER INTERVAL COMPLETION *admet un noyau avec* $O(k^5)$ *sommets.*

Preuve. Soit (G,k) une instance positive de PROPER INTERVAL COMPLETION réduite par les Règles 4.1 à 4.6. Soient F une k-complétion de G, $H := G+F$ le graphe d'intervalle propre correspondant et σ_H son ordre parapluie. Comme $|F| \leqslant k$ par définition, G contient au plus $p \leqslant 2k$ sommets affectés. Nous dénotons par $A := \{a_1 \dots a_p\}$ l'ensemble de ces sommets, et supposons sans perte de généralité que $a_1 <_{\sigma_H} \dots <_{\sigma_H} a_p$. La taille du noyau est obtenue grâce aux observations suivantes :

- Soient $L_0 := \{l \in V \ : \ l <_{\sigma_H} a_1\}$ et $R_{p+1} := \{r \in V \ : \ a_p <_{\sigma_H} r\}$. Puisque les sommets de L_0 et R_{p+1} ne sont pas affectés par construction, les graphes $G[L_0]$ et $G[R_{p+1}]$ sont des graphes d'intervalle propre. Comme la Règle 4.1 a été appliquée au graphe G, ces deux graphes sont connexes et, de ce fait, L_0 et R_{p+1} définissent des 1-branches de G. Par l'Observation 4.28, il suit que L_0 et R_{p+1} contiennent chacun au plus k^3+4k^2+9k+4 sommets.
- Pour tout $i \in [p]$, nous définissons l'ensemble $S_i := \{s \in V \ : \ a_i <_{\sigma_H} b <_{\sigma_H} a_{i+1}\}$. Comme précédemment, puisque les sommets de S_i ne sont pas affectés, le graphe $G[S_i]$ est un graphe d'intervalle propre. Comme G est réduit par la Règle 4.1, $G[S_i]$ contient au plus 2 composantes connexes. Si $G[S_i]$ est connexe, alors S_i définit une 2-branche de G et contient au plus $(k+3)\cdot(k^3+4k^2+5k+1)$ sommets (Observation 4.32). Dans le cas contraire, si $G[S_i]$ contient deux composantes connexes C et C', ces dernières correspondent à des 1-branches de G et par l'Observation 4.28 $G[S_i]$ contient au plus $2\cdot(k^3+4k^2+9k+4)$ sommets. Dans les deux cas, nous bornons le nombre de sommets de S_i par $(k+3)\cdot(k^3+4k^2+5k+1)$, à condition que $k \geqslant 1$, ce qui peut être supposé sans perte de généralité.

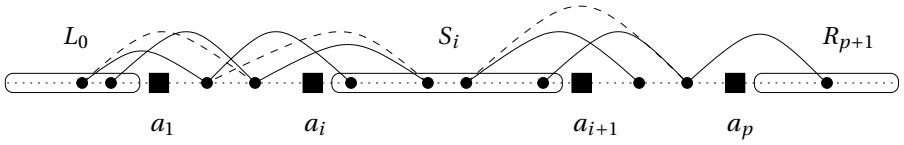


FIGURE 4.15 : Illustration du graphe d'intervalle propre H. Les sommets carrés représentent les sommets affectés par F .

En réunissant ces observations, nous obtenons que le graphe H (et donc G) contient au plus :

$$2\cdot(k^3+4k^2+9k+4)+(2k-1)\cdot[(k+4)\cdot(k^3+4k^2+5k+1)] = 2k^5+15k^4+36k^3+29k^2+5k+4$$

sommets. La complexité nécessaire pour construire le noyau provient du Corollaire 4.38. □

4.6 Un cas particulier : BIPARTITE CHAIN DELETION

graphes bipartis-chaînés

Les *graphes bipartis-chaînés* sont définis comme les graphes bipartis dont les parties sont connectées par un join. De manière équivalente, ces graphes sont les graphes n'admettant pas de $\{2K_2, C_5, K_3\}$ comme sous-graphes induits [157] (voir Figure 4.16). Guo a démontré que le problème BIPARTITE CHAIN DELETION WITH FIXED BIPARTITION [84], où nous sommes donnés un graphe *biparti* $G := (V,E)$ et un entier k et cherchons un sous-ensemble de E de taille au plus k dont la suppression dans E résulte en un graphe biparti-chaîné, admet un noyau avec $O(k^2)$ sommets. Puisque le complémentaire biparti d'un graphe biparti-chaîné est un graphe biparti-chaîné, le résultat précédent est également valable pour le problème BIPARTITE CHAIN COMPLETION WITH FIXED BIPARTITION. Cette classe de graphes a été introduite par Yannakakis [157], qui a prouvé que les problèmes précédemment cités étaient *NP*-Complets. Nous définissons les graphes *bi-clique-chaînés* comme les graphes composés de deux cliques disjointes connectées par un join. En utilisant des techniques similaires à celles de la Section 4.3, nous démontrons que les problèmes BIPARTITE CHAIN DELETION et BI-CLIQUE CHAIN COMPLETION admettent un noyau avec $O(k^2)$ sommets. En d'autres termes, nous étudions les problèmes précédents lorsque la bipartition n'est pas fixée. Par souci de simplicité, nous considérons le problème de complétion, défini formellement comme suit :

graphes bi-clique-chaînés

BI-CLIQUE CHAIN COMPLETION :
Entrée : Un graphe $G := (V,E)$, et un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq (V \times V) \setminus E$ de taille au plus k tel que le graphe $H := (V, E \cup F)$ est un graphe bi-clique-chaîné ?

Nous démontrons que ce problème admet un noyau avec $O(k^2)$ sommets[3]. Puisque le complémentaire d'un graphe bi-clique-chaîné est un graphe biparti-chaîné, ce résultat sera également valable pour le problème BIPARTITE CHAIN DELETION. Observons que, par définition, les graphes bi-clique-chaînés n'admettent pas de $\{C_4, C_5, 3K_1\}$ comme sous-graphe induit, où $3K_1$ dénote un ensemble indépendant de taille 3 (voir Figure 4.16). Observons de plus que les graphes bi-clique-chaînés sont des graphes d'intervalle propre, et admettent ainsi un ordre parapluie. Soit $G := (V,E)$ un graphe bi-clique-chaîné et $\sigma_G := v_1 \dots v_n$ un ordre parapluie de G. Par définition de G, il existe $i \in [n]$ tel que les ensembles $\{v_1, \dots, v_i\}$ et $\{v_{i+1}, \dots, v_n\}$ sont des cliques de G connectées par un join.

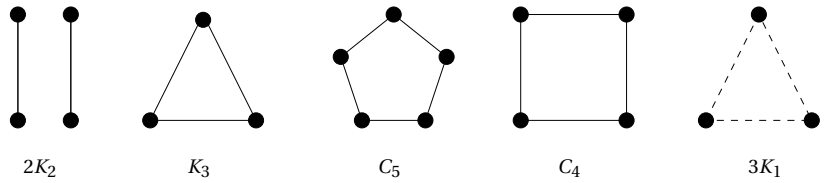


FIGURE 4.16 : Les sous-graphes induits interdits caractérisant la classe des graphes bipartis-chaînés et bi-clique-chaînés.

3. Nous ne donnons pas les preuves des résultats pouvant être prouvés de la même manière que dans la Section 4.3.

Règle 4.7 (Sunflower)**.** *Soient* (G,k) *une instance de* BI-CLIQUE CHAIN COMPLETION *et* $\mathscr{S} := \{C_1,\dots,C_m\}$, $m > k$ *un ensemble de* $3K_1$ *ayant deux sommets* u, v *en commun mais des troisièmes sommets différents. Ajouter* uv *à* F *et décrémenter* k *de* 1. *Soit* $\mathscr{S} := \{C_1,\dots,C_m\}$, $m > k$ *un ensemble de* C_4 *distincts ayant une non-arête* $uv \in \bar{E}$ *en commun. Ajouter* uv *à* F *et décrémenter* k *de* 1. Règle 2.6

Le résultat suivant est l'analogue du Lemme 4.21.

Lemme 4.40. *La Règle 4.7 est valide et peut être appliquée en temps polynomial.* Lemme 2.48

Lemme 4.41 ([13])**.** *Soit* (G,k) *une instance positive de* BI-CLIQUE CHAIN COMPLETION *réduite par la Règle 4.7. Il existe au plus* k^2+2k *sommets de* G *contenus dans des* $3K_1$, *et au plus* $2k^2+2k$ *sommets de* G *contenus dans des* C_4.

Nous définissons un K-join $B \subseteq V$ comme *simple* lorsque B est une 1-branche et $G[B]$ est une clique. En d'autres termes, un K-join simple consiste en une clique connectée au reste du graphe par un join. De plus, nous définissons un *K-join propre* comme un K-join dont les sommets n'appartiennent à aucun $3K_1$ ou C_4. La règle de réduction suivante est similaire à la Règle 4.4 dans l'intuition, et diffère seulement par les arguments techniques utilisés dans la preuve.

Règle 4.8. *Soient* (G,k) *une instance de* BI-CLIQUE CHAIN COMPLETION *et* $B \subseteq V$ *un K-join simple et propre contenant au moins* $2k+2$ *sommets. Soient* B_L *les 2k+1 premiers de* B, *et* B_R *les* $2k+1$ *derniers sommets de* B, *et* $M := B \setminus (B_L \cup B_R)$. *Supprimer l'ensemble de sommets* M *de* G.

Lemme 4.42 ([13])**.** *La Règle 4.8 est valide et peut être appliquée en temps polynomial.*

Observation 4.43. *Soit* (G,k) *une instance positive de* BI-CLIQUE CHAIN COMPLETION *réduite par les Règles 4.1,4.7 et 4.8. Tout K-join simple de* G *contient au plus* $3k^2+6k+2$ *sommets.*

Preuve. Par le Lemme 4.41, nous savons qu'au plus $3k^2+4k$ sommets de B sont contenus dans un $3K_1$ ou un C_4. Ainsi, B contient un sous-ensemble B' d'au moins $2k+3$ sommets non contenus dans un $3K_1$ ou un C_4. Puisque tout sous-ensemble d'un K-join simple est un K-join simple (Observation 4.10), il suit que B' est un K-join simple *propre* de G. Comme G est réduit par la Règle 4.8, nous savons que $|B'| \leqslant 2k+1$: contradiction. □

Théorème 4.44. *Le problème* BI-CLIQUE CHAIN COMPLETION *admet un noyau avec* $O(k^2)$ *sommets.*

Preuve. Soient (G,k) une instance positive de BI-CLIQUE CHAIN COMPLETION réduite par les Règles 4.1, 4.7 et 4.8, et F une k-complétion de G. Nous posons $H := G+F$ pour dénoter le graphe bi-clique-chaîné associé, et $\{H_1,H_2\}$ les deux cliques de H connectées par un join. Remarquons en particulier que H_1 et H_2 définissent tous deux des K-joins simples de H. Soit A l'ensemble des sommets affectés de G. Puisque $|F| \leqslant k$, nous avons $|A| \leqslant 2k$. Nous considérons maintenant les ensembles de sommets suivants : $A_1 := A \cap H_1$, $A'_1 := H_1 \setminus A_1$, $A_2 := A \cap H_2$ et $A'_2 := H_2 \setminus A_2$.

Comme H_1 est un K-join simple de H, il suit que $A'_1 \subseteq H_1$ est un K-join simple de H, et donc de G puisque les sommets de A'_1 ne sont pas affectés par construction. Par l'Observation 4.43, il suit que $|A'_1| \leqslant 3k^2+6k+2$. La même propriété est vérifiée par A'_2, et ainsi H contient au plus $2 \cdot (3k^2+6k+2)+2k$ sommets. □

Corollaire 4.45. *Le problème* BIPARTITE CHAIN DELETION *admet un noyau avec* $O(k^2)$ *sommets.*

CHAPITRE

5 Noyau cubique pour COGRAPH EDITION

Dans ce chapitre, nous établissons un noyau cubique pour le problème COGRAPH EDITION, les **cographes** étant les graphes n'admettant pas de P_4 comme sous-graphe induit. Nous établissons également que ce résultat est valable pour les problèmes COGRAPH EDGE-DELETION et COGRAPH COMPLETION. Ces problèmes sont NP-Complets [60, 117] mais admettent des algorithmes FPT [31, 117], notamment de complexité $O^*(4^k)$ par le résultat de Cai [31] présenté dans la **Section 2.1.2**. Une autre particularité des cographes est que ces derniers représentent les graphes **totalement décomposables pour la décomposition modulaire** (présentée **Chapitre 1**). Afin d'élaborer les règles de réduction pour ce problème, nous utilisons les modules et l'arbre de décomposition modulaire d'une instance réduite. Formellement, nous considérons le problème suivant.

COGRAPH EDITION :
Entrée : Un graphe $G := (V, E)$, et un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq (V \times V)$ de taille au plus k tel que le graphe $H := (V, E \triangle F)$ est un cographe ?

Théorème 5.1. *Les problèmes* COGRAPH EDITION, COGRAPH COMPLETION *et* COGRAPH EDGE-DELETION *admettent un noyau avec* $O(k^3)$ *sommets.*

5.1 Cographes et décomposition modulaire

Pour réaliser notre algorithme de noyau, nous utilisons diverses caractérisations connues pour les cographes. En particulier, nous avons besoin des deux définitions suivantes, qui sont équivalentes.

Définition 5.2 (Cographe)**.** *Un graphe $G := (V, E)$ est un cographe si et seulement si il ne contient pas de P_4 induit.* cographe

Définition 5.3 ([25])**.** *Un graphe $G := (V, E)$ est un cographe si et seulement si il peut être construit à partir d'un graphe à un seul sommet par une séquence de compositions Séries et Parallèles.*

Observation 5.4 (Classique)**.** *Soient $G := (V, E)$ un graphe, $M \subseteq V$ un module de G et $\{a, b, c, d\} \subseteq V$ un ensemble de sommets induisant un P_4 dans G. Alors $|M \cap \{a, b, c, d\}| \leqslant 1$ ou $\{a, b, c, d\} \subseteq M$.*

Modules et éditions. Nous utilisons l'Observation 5.4 pour prouver qu'il existe toujours une solution optimale pour les problèmes considérés préservant les modules.

Lemme 5.5. *Soit (G, k) une instance de* Cograph Edition *(resp.* Cograph Edge-Deletion, Cograph Completion*). Il existe une édition (resp. arête-suppression, complétion) optimale F telle que :*

(1) F ne contient aucune paire entre deux modules correspondant à deux fils d'un même noeud de l'arbre de décomposition modulaire de G.

(2) Tout module de G est un module de $H = G \triangle F$.

Preuve. Nous prouvons le résultat pour Cograph Edition uniquement (les preuves pour Cograph Edge-Deletion et Cograph Completion sont similaires). Nous prouvons dans un premier temps qu'il existe une édition optimale préservant les modules *forts* de G, et étendrons par la suite ce résultat à tous les modules de G.

Affirmation 5.6. *Il existe une édition optimale F telle que tout module* fort *de G est un module de $H := G \triangle F$.*

Preuve. Soient $M_1, \ldots, M_p$ les modules forts de G ordonnés selon un parcours postfixe de l'arbre de décomposition modulaire de G. Remarquons que nous avons alors $M_j \nsubseteq M_i$ pour tout $1 \leqslant i < j \leqslant p$. Nous prouvons par induction sur $0 \leqslant i \leqslant p$ qu'il existe une édition optimale F telle que $M_1, \ldots, M_i$ sont modules de $H := G \triangle F$. Le cas de base $i = 0$ est trivialement vérifié. Supposons maintenant que $i < p$, et considérons une édition optimale F telle que $M_1, \ldots, M_{i-1}$ sont modules de $H := G \triangle F$. Soit u un sommet de M_i tel que $|\{\{u, v\} \in F \ : \ v \notin M_i\}|$ est minimum. Nous prétendons que l'ensemble F' décrit ci-après est une édition optimale telle que $M_1, \ldots, M_i$ sont modules de $H' := G \triangle F'$:

$$F' := F[M] \cup F[V \setminus M] \cup \{\{v, w\} \ : \ v \in M, \ w \notin M, \ \{u, w\} \in F\}$$

Informellement, nous éditons tous les sommets de M_i comme le sommet u, qui a un *coût d'édition* minimum. Dans un premier temps, observons que, par construction, M_i est un module du graphe H'. De plus, par choix de u, $|F'| \leqslant |F|$. Nous prouvons maintenant que H' est un cographe. Comme $H'[M_i]$ et $H'[V \setminus M_i]$ sont isomorphiques à $H[M_i]$ et $H[V \setminus M_i]$, ils ne contiennent pas de P_4 comme sous-graphe induit. De ce fait, si H' contient un P_4 induit $P := \{a, b, c, d\}$, ce dernier doit

intersecter M_i et $V \setminus M_i$. Par l'Observation 5.4, il suit que $|M_i \cap \{a,b,c,d\}| = 1$: nous supposons $M_i \cap \{a,b,c,d\} = \{a\}$, les autres cas se prouvant de manière similaire. Par construction de F', cela implique que l'ensemble $\{u,b,c,d\}$ induit un P_4 dans H' et donc dans H, contredisant le fait que F est une édition de G. Il suit que F' est une édition optimale de G respectant la propriété désirée.

Nous prouvons maintenant que $M_1,\dots,M_i$ sont des modules de H'. Soit M_j un module fort de G avec $1 \leqslant j < i$. Par définition, nous savons que $M_j \cap M_i = \emptyset$ ou $M_i \subset M_j$ ou $M_j \subset M_i$. Dans le premier cas, M_j est un module de H' puisque M_j est un module de H par hypothèse d'induction (et ainsi $u \in V \setminus M_j$ est adjacent à tout ou aucun sommet de M_j). Le second cas ne peut pas arriver puisque nous avons considéré $M_1,\dots,M_p$ selon un ordre postixe de l'arbre de décomposition modulaire de G. Finalement, supposons que $M_j \subset M_i$. Dans ce cas, puisque M_i est un module de H' par construction, il suit que le voisinage de M_j est *uniforme* par rapport à $V \setminus M_i$. Maintenant, dans la mesure où M_j est un module de H', il suit que les arêtes entre M_j et $M_i \setminus M_j$ sont les mêmes dans H et dans H', impliquant que M_j est un module de H'. ⋄

Soit F une édition optimale. Par l'Affirmation 5.6, nous pouvons supposer que tout module fort de G est un module de $H := G \triangle F$. Pour conclure la preuve, nous prouvons que les points (1) et (2) sont vérifiés. Nous démontrons d'abord (1). Nous supposons qu'il existe un noeud Série ou Parallèle N de l'arbre de décomposition modulaire de G tel que F contient une paire entre deux modules correspondant à deux fils de N. Nous supposons également que N correspond à un module M dans G, et que ses fils correspondent aux modules $M_1,\dots,M_r$. Soit F' l'ensemble obtenu à partir de F en supprimant toute paire $\{u,v\}$ avec $u \in M_i$, $v \in M_j$, $1 \leqslant i < j \leqslant r$. Par hypothèse, nous avons $|F'| < |F|$. Nous prétendons que $H' := G \triangle F'$ est un cographe, contredisant l'optimalité de F. Nous supposons par contradiction que H' contient un P_4 induit P. Puisque M est un module de H, il suit par l'Affirmation 5.6 que M et $M_1,\dots,M_r$ sont des modules de H' (rappelons qu'un noeud de l'arbre de décomposition modulaire de G correspond à un module fort de G). Par l'Observation 5.4, nous avons alors $|P \cap M| \leqslant 1$ ou $P \subseteq M$. Si $|P \cap M| \leqslant 1$, alors P est un P_4 induit dans H, une contradiction. Maintenant, si $P \subseteq M$, alors l'Observation 5.4 implique $P \subseteq M_i$ pour un certain $1 \leqslant i \leqslant r$ ou $|P \cap M_i| \leqslant 1$ pour tout $1 \leqslant i \leqslant r$. Le premier cas est impossible puisque P serait alors un P_4 induit de H, amenant à nouveau une contradiction. Le second cas est également impossible puisque H' contient toute ou aucune arête entre M_i et M_j, $1 \leqslant i < j \leqslant r$.

Pour conclure, nous prouvons que (2) est vérifié. Pour ce faire, considérons un module non-trivial M de G. Si M est un module fort, alors le résultat est vérifié par définition de F. Supposons donc que M n'est pas un module fort. Nous utilisons la propriété suivante sur les modules forts d'un graphe : il existe un noeud Série ou Parallèle N de l'arbre de décomposition modulaire de G tel que M est l'union de modules forts représentés par un sous-ensemble des fils de N. Nous supposons que N correspond à un module fort M'. Soit $u \in V \setminus M$. Nous prétendons que u est adjacent à tout ou aucun sommet de M. En effet, si $u \in M' \setminus M$ alors le résultat est vérifié par (1). Maintenant, si $u \in V \setminus M'$, le résultat est vérifié car M' est un module de H. Nous déduisons donc que M est un module de H, ce qui implique le résultat. □

Un corollaire direct des précédents résultats est l'existence d'une solution optimale (quelle que

soit la version du problème considérée) qui édite toute ou aucune arête entre deux modules disjoints.

Corollaire 5.7. *Soit* (G,k) *une instance de* COGRAPH EDITION *(resp.* COGRAPH EDGE-DELETION, COGRAPH COMPLETION*). Il existe une arête-suppression (resp. complétion, édition) optimale* F *telle que pour toute paire de modules disjoints* M *et* M', $(M \times M') \subseteq F$ *ou* $(M \times M') \cap F = \emptyset$.

Dans la suite, nous considérerons uniquement des éditions (resp. arêtes-suppressions, complétions) vérifiant cette propriété.

5.2 Noyaux cubiques pour COGRAPH EDGE-DELETION et COMPLETION

Dans cette section, nous prouvons que les problèmes COGRAPH EDGE-DELETION et COGRAPH COMPLETION admettent tous deux des noyaux contenant $O(k^3)$ sommets. Puisque le complémentaire d'un P_4 est un P_4, il suit que le complémentaire d'un cographe est un cographe. Ainsi, les problèmes COGRAPH EDGE-DELETION et COGRAPH COMPLETION sont équivalents. Nous démontrons donc le résultat pour le problème COGRAPH EDGE-DELETION, et le résultat pour COGRAPH COMPLETION suivra directement.

5.2.1 Règles de réduction basées sur la décomposition modulaire

Nous énonçons maintenant plusieurs règles de réduction nous permettant de structurer l'arbre de décomposition modulaire d'une instance réduite. En particulier, les règles de réduction nous permettent de *borner la hauteur* de l'arbre de décomposition modulaire d'une instance réduite, ainsi que de *contrôler la structure* de ses modules. Comme dans les Chapitres 3 et 4, nous utilisons la Règle 2.5, qui est valide puisque les cographes sont stables par union disjointe (un P_4 étant obligatoirement inclus dans une composante connexe).

Règle 5.1 (Composante connexe)**.** *Soit* (G,k) *une instance de* COGRAPH EDGE-DELETION. *Supprimer toute composante connexe* C *de* G *telle que* $G[C]$ *est un cographe.* Règle 2.5

Lemme 5.8. *La Règle 5.1 est valide et peut être appliquée en temps* $O(n+m)$. Lemme 2.46

Règle 5.2 (Composition Série)**.** *Soient* (G,k) *une instance de* COGRAPH EDGE-DELETION*, et* $C := C_1 \otimes C_2$ *une composante connexe de* G. *Remplacer* C *par* $C_1 \oplus C_2$.

Lemme 5.9. *La Règle 5.2 est valide et peut être appliquée en temps* $O(n+m)$.

Preuve. Soit G' le graphe où C a été remplacée par $C_1 \oplus C_2$. Soit F une arête-suppression optimale de G.

Affirmation 5.10. *L'ensemble* F *ne contient aucune arête de* $(C_1 \times C_2)$.

Preuve. Supposons que $F \cap (C_1 \times C_2) \neq \emptyset$. Comme C_1 et C_2 sont des modules de G, le Lemme 5.5 implique que C_1 et C_2 sont des modules de $H := G - F$ et donc F contient $(C_1 \times C_2)$. Ainsi, C_1 et C_2 sont des unions de composantes connexes de H induisant des cographes. Par la Définition 5.3, le graphe H' obtenu en faisant une composition série entre C_1 et C_2 dans H est un cographe. Il suit

que $F' := F \setminus (C_1 \times C_2)$ est une arête-suppression de G vérifiant $|F'| < |F|$, ce qui est impossible. ⋄

Soit F une k-arête-suppression de G. Par l'Affirmation 5.10, nous savons que F ne contient aucune arête de $(C_1 \times C_2)$, et ainsi F est une k-suppression de G'. Inversement, soit F' une k-suppression de G'. Alors C_1 et C_2 induisent des cographes disjoints dans $G' - F'$, et la Définition 5.3 implique que le graphe H obtenu en faisant une composition série entre C_1 et C_2 dans $G' - F'$ est un cographe. Il suit que F' est une k-arête-suppression de G. Cette règle de réduction peut être exécutée en temps $O(n+m)$, qui est le temps nécessaire pour générer l'arbre de décomposition modulaire de G. □

Règle 5.3 (Modules)**.** *Soient* (G,k) *une instance de* COGRAPH EDGE-DELETION*, et M un module de G différent d'un ensemble indépendant de taille au plus $k+1$ et strictement contenu dans une composante connexe. Retourner le graphe $G' \oplus G[M]$, où G' est obtenu à partir de G en supprimant M et en ajoutant un ensemble indépendant de taille* $\min\{k+1, |M|\}$ *ayant le même voisinage que M.*

Lemme 5.11. *La Règle 5.3 est valide et peut être appliquée en temps linéaire.*

Preuve. Soient $G'' := G' \oplus G[M]$ et M' le module remplaçant M dans G''. Soit F une k-arête-suppression de G. Par le Corollaire 5.7, nous savons que pour tout sommet $v \in V \setminus M$ nous avons $(\{v\} \times M) \subseteq F$ ou $(\{v\} \times M) \cap F = \emptyset$. En particulier, cette dernière condition est vérifiée dès lors que $|M| > k$. Ainsi, G'' admet une k-complétion F'' définie comme suit : les suppressions $F[M]$ sont réalisées sur le graphe $G''[M]$, et les suppressions $F \setminus F[M]$ sont réalisées sur le graphe G'. Puisque M' contient $\min\{k+1, |M|\}$ sommets et a le même voisinage que M, ces suppressions sont bien définies dans G''. Il suit que $G''[M] - F[M]$ et $G'' - F$ sont tous deux des cographes, d'où le résultat. Inversement, soit F' une k-arête-suppression de G''. Par le Lemme 5.5, aucune modification n'est réalisée sur $G[M']$. Ainsi, comme précédemment, l'édition F' est bien définie sur G', qui admet ainsi une k-arête-suppression. Remarquons enfin que cette règle peut être appliquée en temps linéaire en parcourant l'arbre de décomposition modulaire de G. □

Remarque.
- Lors de l'application de la Règle 5.3, observons que si $G[M]$ est un cographe alors ajouter une copie de M au graphe G n'est pas utile puisque cette dernière sera supprimée par une application de la Règle 5.1.
- La Règle 5.3 rejoint de ce fait la notion de *branches* : en effet, un module M tel que $G[M]$ est un cographe est une branche pour ce problème, et peut être remplacé de manière valide par un ensemble indépendant de taille $k+1$.

L'observation suivante est une conséquence directe de la Règle 5.3.

Observation 5.12. *Soient* (G,k) *une instance de* COGRAPH EDGE-DELETION *réduite par les Règles 5.1, 5.2 et 5.3, et $C \subseteq V$ une composante connexe non première de G. Les modules de $G[C]$ sont des ensembles indépendants de taille au plus $k+1$.*

Remarquons qu'une application de la Règle 5.3 pourra éventuellement augmenter le nombre de sommets de l'instance réduite. Cependant, nous démontrerons par la suite que ces règles nous permettent de borner la taille d'une instance réduite positive. Avant cela, il nous faut démontrer

qu'une instance réduite par les Règles 5.1, 5.2 et 5.3 peut être générée en temps polynomial, *i.e.* qu'un nombre polynomial d'applications de ces règles permet de réduire l'instance considérée.

Remarque. Bien que nos règles de réduction soient définies sur des modules quelconques du graphe, notre algorithme de noyau ne les applique que sur les modules *forts* d'une instance (G,k). Nous verrons par la suite que cela est suffisant afin d'obtenir un noyau cubique pour ces problèmes.

Lemme 5.13. *Soit (G,k) une instance de* COGRAPH EDGE-DELETION. *Une instance réduite par les Règles 5.1 à 5.3 peut être calculée en temps polynomial.*

Preuve. Soit M un module de G. Nous disons que M est *réduit* si M est un ensemble indépendant de taille au plus $k+1$ ou l'union disjointe de composantes connexes de G. Par l'Observation 5.12, si G est réduit par les Règles 5.1, 5.2 et 5.3, alors tout module de G est réduit. De plus, observons que si tout module fort de G est réduit, alors tout module de G est réduit. Nous prouvons donc le résultat en comptant le nombre de modules forts (qui correspondent exactement aux noeuds de l'arbre de décomposition modulaire de G) qui ne sont pas réduits.

Remarquons que si une composante connexe C de G induit un cographe avec au moins deux sommets, une série d'applications de la Règle 5.2 transforme $G[C]$ en un ensemble indépendant. Cela implique que nous pouvons supposer que la Règle 5.1 est appliquée en dernier lors du processus de réduction. Cette remarque permet de simplifier les arguments qui suivent.

Lorsque la Règle 5.3 est appliquée sur un module fort (différent d'un ensemble indépendant par hypothèse), alors par définition le nombre de modules forts non réduits de G est décrémenté de 1. De même, lorsque la Règle 5.2 est appliquée (*i.e.* il existe une composante connexe $C := C_1 \otimes C_2$ et C_1 est un module fort), alors le nombre de modules forts non-réduits de G décroît de 1, sauf si C_1 est un ensemble indépendant de taille au plus $k+1$. Cependant, dans ce cas, les sommets de C_1 seront supprimés lors de l'application de la Règle 5.1, puisqu'ils seront alors des sommets isolés. Dans la mesure où le nombre de modules forts d'un graphe est borné par n, cela implique qu'une série d'au plus n applications des Règles 5.2 et 5.3 est suffisante pour construire un graphe réduit. □

5.2.2 Borner la taille d'une instance réduite

Nous utilisons maintenant les règles précédentes afin d'obtenir un algorithme de noyau pour les problèmes COGRAPH EDGE-DELETION et COGRAPH COMPLETION. Pour ce faire, nous avons besoin de la règle de réduction de type Sunflower (Règle 2.6 et Lemme 2.48).

Règle 5.4 (Sunflower)**.** *Soient (G,k) une instance de* COGRAPH EDGE-DELETION, *et $e \in E$ une arête appartenant à un ensemble $\mathscr{P} := \{P_1,\ldots,P_m\}$ de P_4 induits s'intersectant 2-à-2 sur e, $m > k$. Supprimer e de G et décrémenter k de 1.* Règle 2.6

Lemme 5.14. *La Règle 5.4 est valide et peut être appliquée en temps polynomial.* Lemme 2.6

Dans la suite, l'arbre de décomposition modulaire associé à un cographe est appelé *coarbre*.

Théorème 5.15. *Le problème* COGRAPH EDGE-DELETION *admet un noyau avec $O(k^3)$ sommets.*

Preuve. Soit (G,k) une instance positive de COGRAPH EDGE-DELETION réduite par les Règles 5.1 à 5.4. Soient F une k-arête-suppression de G, $H := G - F$ le cographe correspondant et T son co-arbre. Nous dénombrons le nombre de feuilles de T, qui est égal au nombre de sommets de H (et donc de G).

Dans un premier temps, remarquons que puisque $|F| \leqslant k$, il existe au plus $2k$ sommets de H incident à des arêtes de F, et donc T contient au plus $2k$ feuilles affectées. Nous définissons un noeud interne x de T comme *affecté* si x est le plus petit ancêtre commun de deux feuilles affectées. Observons que T contient au plus $2k$ noeuds affectés.

Affirmation 5.16. *La racine r de T est un noeud Parallèle et affecté.*

Preuve. Nous prouvons l'Affirmation 5.16 par contradiction. Premièrement, nous supposons que la racine r de T est un noeud Série. Les arêtes présentes dans H étant également présentes dans G, cela implique que le graphe G n'est pas réduit par la Règle 5.2 : contradiction. Il suit que r est un noeud Parallèle. De plus, puisque G est réduit par la Règle 5.1, aucune de ses composantes connexes n'est un cographe. Cela implique que toute composante connexe de G contient (au moins) un sommet incident à une arête de F. Ainsi, tout sous-arbre attaché à la racine r de T contient une feuille affectée comme descendant. Il suit par définition que r est un noeud affecté de T. ◇

Nous bornons maintenant la longueur d'un chemin reliant deux noeuds affectés consécutifs dans T, ce qui nous permettra de borner la taille de l'instance.

Affirmation 5.17. *Soient $p \neq r$ un noeud (ou une feuille) de T affecté(e), et q le plus petit ancêtre affecté de p dans T. Le chemin reliant p à q dans T a une longueur inférieure ou égale à $2k+3$.*

Preuve. Observons tout d'abord que le résultat est trivial si p est un des fils de la racine r (*i.e.* $q = r$). Dans tous les autres cas, nous dénotons par M_p l'ensemble de feuilles descendant de p dans T. Nous prouvons maintenant que M_p contient une feuille u incidente à une arête supprimée uv de F avec $v \notin M_p$. Premièrement, si p est une feuille affectée, alors le résultat est vrai par définition. Sinon, supposons par contradiction que toutes les arêtes de F incidentes à un sommet de M_p soient de la forme uv, avec $u, v \in M_p$. En particulier, cela implique que M_p est un module entièrement contenu dans une composante connexe de G : en effet, puisque p n'est pas un fils de r, il suit que M_p est entièrement contenu dans une composante connexe de H, et donc entièrement contenu dans une composante connexe de G. Par l'Observation 5.12, M_p est un ensemble indépendant de taille $k+1$ et ne contient donc aucune arête : contradiction. Soit t le plus petit ancêtre commun de u et v. Le noeud t est un noeud Parallèle ($uv \notin E(H)$ par définition) et un ancêtre de p et q (le cas $t = q$ étant possible). Supposons par contradiction que le chemin reliant u à t dans T contienne une séquence de $2k+3$ sommets consécutifs non-affectés. Le type de ces noeuds est alternativement Série et Parallèle. De ce fait, nous pouvons construire une séquence $\{s_1, p_1, \ldots, s_{k+1}, p_{k+1}\}$ de noeuds non-affectés consécutifs, s_i (resp. p_i) étant le père de p_i (resp. s_{i+1}). De plus, les noeuds s_i (resp. p_i), $i \in [k+1]$, correspondent à des noeuds Série (resp. Parallèle). Par construction, chaque noeud s_i (resp. p_i) est parent d'une feuille non-affectée u_i (resp. v_i) qui n'est pas descendante de p_i (resp. s_{i+1}). Il suit que pour tout $i \in [k+1]$ l'ensemble $\{v_i, u_i, u, v\}$ induit un P_4 de G. Nous avons ainsi construit un ensemble de P_4 induits de G s'intersectant 2-à-2 sur l'arête uv, et contenant $k+1$

éléments. Cela implique que G n'est pas réduit par la Règle 5.4 : contradiction. Pour conclure, observons que les noeuds se trouvant sur le chemin reliant p à q dans T sont non-affectés et contenus dans le chemin reliant u à t, impliquant que ce chemin a une longueur inférieure ou égale à $2k+3$. ⋄

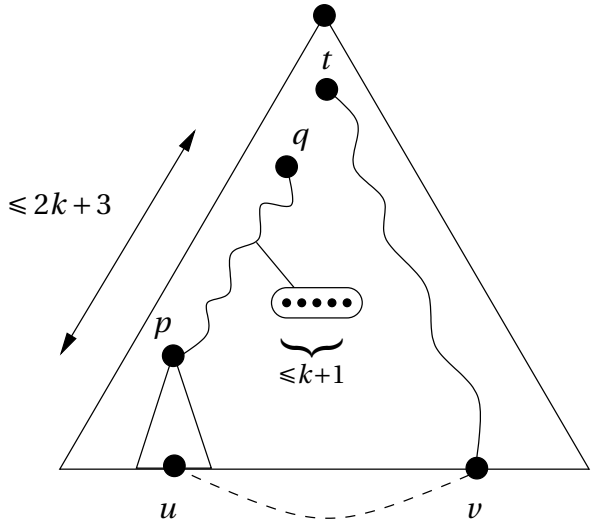


FIGURE 5.1 : Illustration de l'Affirmation 5.17.

Soit U le sous-arbre minimal de T reliant les feuilles affectées. Par l'Affirmation 5.17, U contient au plus $(4k-1)\cdot(2k+3)+2k)$ noeuds internes. Vu que G est réduit, l'Observation 5.12 implique que chacun de ces noeuds est parent d'un ensemble d'au plus $k+1$ feuilles ou d'un noeud parallèle étant lui-même parent d'au plus $k+1$ feuilles. Cela implique que T contient au plus $2k+(k+1)\cdot[(4k-1)\cdot(2k+3)+2k] \leqslant 8k^3+20k^2+11k$ feuilles.

Nous concluons la preuve en établissant la complexité nécessaire pour calculer le noyau. Puisque la Règle 5.4 décrémente la valeur du paramètre de 1 à chaque exécution, cette dernière est appliquée au plus k fois. Le Lemme 5.13 permet alors de conclure qu'une instance réduite peut être générée en temps polynomial. □

Corollaire 5.18. *Le problème* COGRAPH COMPLETION *admet un noyau avec* $O(k^3)$ *sommets.*

5.3 Noyau cubique pour COGRAPH EDITION

Nous démontrons maintenant que le problème COGRAPH EDITION admet un noyau avec $O(k^3)$ sommets. Les principales différences se trouvent au niveau de la règle de réduction *sunflower*, ainsi que dans la preuve du Théorème 5.21. En effet, nous avons le résultat suivant.

Lemme 5.19. *Les Règles 5.1 à 5.3 sont valides et peuvent être appliquées en temps polynomial pour le problème* COGRAPH EDITION.

Règle 2.6 **Règle 5.5** (Sunflower)**.** *Soient* (G,k) *une instance de* COGRAPH EDITION *et* $\{u,v\}$ *une paire de sommets de* G *appartenant à un ensemble* $\mathscr{P} := \{P_1,\ldots,P_m\}$ *de* P_4 *induits s'intersectant* 2*-à-*2 *sur la paire* $\{u,v\}$*,* $m > k$*. Remplacer* E *par* $E \triangle \{\{u,v\}\}$ *et décrémenter* k *de* 1*.*

Lemme 2.6 **Lemme 5.20.** *La Règle 5.5 est valide et peut être appliquée en temps polynomial.*

Théorème 5.21. *Le problème* COGRAPH EDITION *admet un noyau avec* $O(k^3)$ *sommets.*

Preuve. Soient (G,k) une instance positive de COGRAPH EDITION réduite par les Règles 5.1, 5.2, 5.3 et 5.5 et F une k-édition de G. Comme dans la preuve du Théorème 5.15, nous dénotons par $H := G \triangle F$ le cographe correspondant et par T son coarbre. Contrairement à la preuve du Théorème 5.15, la racine de T n'est plus nécessairement un noeud Parallèle. En revanche, le résultat suivant est toujours vérifié (la notion de noeud *affecté* étant la même que dans la preuve du Théorème 5.15).

Affirmation 5.22. *La racine* r *de* T *est affectée.*

Preuve. Supposons dans un premier temps que la racine r de T soit un noeud Série. Dans ce cas, r doit être un noeud affecté sans quoi le graphe G ne serait pas réduit par la Règle 5.2. Supposons maintenant que la racine r soit un noeud Parallèle non-affecté. Par définition, cela signifie qu'au plus un de ses fils contient une feuille affectée dans son sous-arbre, et donc G n'est pas réduit par la Règle 5.1 (les autres fils de r définissent des composantes connexes ne contenant aucun noeud affecté, *i.e.* des cographes). ◇

Dans la suite de la preuve nous supposons sans perte de généralité que la racine r de T est un noeud parallèle. Le nombre de sommets de l'instance réduite se calcule ensuite de la même manière que dans la preuve du Théorème 5.15.

Affirmation 5.23. *Soient* $p \neq r$ *un noeud (ou une feuille) de* T *affecté(e), et* q *le plus petit ancêtre affecté de* p *dans* T*. Le chemin reliant* p *à* q *dans* T *a une longueur d'au plus* $2k+3$*.*

Preuve. Observons tout d'abord que le résultat est trivial si p est un des fils de la racine r (*i.e.* $q = r$). Dans tous les autres cas, nous dénotons par M_p l'ensemble de feuilles descendant de p dans T. Nous prouvons maintenant que M_p contient une feuille u incidente à une paire éditée $\{u,v\}$ avec $v \notin M_p$. L'argument est similaire à celui de la preuve de l'Affirmation 5.16 : supposons par contradiction que toutes les éditions incidentes à un sommet de M_p soient de la forme $\{u,v\}$, avec $u,v \in M_p$. Dans ce cas, M_p est un module strictement contenu dans une composante connexe de G. Par l'Observation 5.12, il suit que M_p est un ensemble indépendant de taille $k+1$, et le Lemme 5.5 implique quant à lui qu'aucune édition n'est nécessaire dans $G[M_p]$. Nous concluons la preuve de la même manière que la preuve de l'Affirmation 5.16 : s'il existe dans T un chemin constitué de $2k+3$ noeuds non-affectés reliant p à q, alors la paire $\{u,v\}$ est contenue dans un ensemble $\mathscr{P}$ de P_4 induits s'intersectant 2-à-2 sur $\{u,v\}$, et de taille $k+1$. Ainsi, le graphe G n'est pas réduit par la Règle 5.5 : contradiction. ◇

La complexité nécessaire pour obtenir ce noyau découle du Lemme 5.13 et du fait que chaque application de la Règle 5.5 décrémente k de 1 et se réalise en temps polynomial. □

Partie II. Edition d'autres structures - Conflict Packing

Contexte général. Dans la seconde partie de cette thèse, nous nous intéressons principalement à des problèmes d'édition pour des structures différentes des graphes non-orientés. De manière informelle, les problèmes étudiés peuvent être considérés comme des problèmes d'**édition de relations** : étant donnés un ensemble V, une collection $\mathscr{R}$ de **relations** de taille $p \geqslant 2$ définies sur V et un entier k, l'objectif est de déterminer s'il existe une **structure** (e.g. un ordre linéaire ou un arbre binaire) **satisfaisant** chaque relation de $\mathscr{R}$. De manière équivalente, cela revient à décider s'il est possible d'**éditer** au plus k relations de $\mathscr{R}$ afin que la nouvelle collection admette une structure satisfaisant toutes ses relations. De nombreux problèmes peuvent se formuler de cette manière, comme par exemple FEEDBACK ARC SET IN TOURNAMENTS, DENSE ROOTED TRIPLET INCONSISTENCY, et DENSE BETWEENNESS et DENSE CIRCULAR ORDERING. Une particularité structurelle de ces problèmes est qu'ils considèrent des instances **denses**, *i.e.* où la collection de relations $\mathscr{R}$ contient **exactement** une relation pour tout sous-ensemble de taille p de V. Nous verrons que cette hypothèse permet d'avoir dans plusieurs cas une **caractérisation locale** de la consistence de $\mathscr{R}$, que nous utiliserons afin de concevoir nos algorithmes de noyau.

Π-EDITION [1] : **Entrée** : Une collection dense $\mathscr{R}$ de relations de taille $p \geqslant 2$ définies sur un univers V, un entier k. **Paramètre** : k. **Question** : Existe-t-il un ensemble $\mathscr{F} \subseteq \mathscr{R}$ de taille au plus k dont l'édition dans $\mathscr{R}$ permet de satisfaire la propriété Π ?

Résultats connus. Une première question soulevée par le problème Π-EDITION est de déterminer si une instance donnée est **consistente**, *i.e.* si elle respecte la propriété Π. Remarquons que cela correspond au cas où $k = 0$. Pour les problèmes cités précédemment, il est possible de décider en temps polynomial si une instance donnée est consistente. Pour le problème FEEDBACK ARC SET IN

1. Nous appliquerons également cette méthode sur les problèmes $\mathscr{G}$-EDGE DELETION et $\mathscr{G}$-VERTEX DELETION.

TOURNAMENTS, la notion de **consistence** est liée au fait d'être **acyclique**, ce qui peut clairement être vérifié en temps polynomial. Lorsque la collection de relations $\mathscr{R}$ n'est pas consistente, plusieurs problèmes d'optimisation sont envisageables. En particulier, **supprimer des éléments de** V ou **éditer des relations de** $\mathscr{R}$ peut être considéré. Comme mentionné précédemment, nous nous intéresserons principalement au second problème. Notons cependant que la première version de ces problèmes a également été largement étudiée. Ainsi, le problème FEEDBACK VERTEX SET (qui est l'analogue de FEEDBACK ARC SET) admet un algorithme FPT [99] ainsi qu'un algorithme de noyau polynomial [151]. Le problème DENSE ROOTED TRIPLET INCONSISTENCY paramétré par le nombre de sommets à supprimer peut quant à lui être résolu en temps $O^*(4^k)$ [81]. Remarquons cependant que, lorsque l'hypothèse de densité n'est plus supposée, le problème DENSE ROOTED TRIPLET INCONSISTENCY devient $W[1]$-Difficile [8]. En ce qui concerne le problème Π-EDITION paramétré par le nombre de **relations à éditer**, de nombreux résultats sont également connus. Le problème FEEDBACK ARC SET IN TOURNAMENTS admet ainsi plusieurs algorithmes FPT [5, 105], la meille complexité connue étant $O^*(2^{\sqrt{k}})$ [105]. De plus, un algorithme de noyau quadratique est connu [5]. Il est intéressant de noter que ce problème admet des algorithmes d'approximation à facteur constant [41], et plus particulièrement un PTAS [104, 106]. Le problème DENSE ROOTED TRIPLET INCONSISTENCY se comporte de manière similaire au niveau de la complexité paramétrée, puisqu'une technique analogue à celle utilisée pour FEEDBACK ARC SET IN TOURNAMENTS permet de le résoudre en temps $2^{O(k^{1/3}\log k)}$ [82]. De plus, ce problème admet également un algorithme de noyau quadratique [82]. Cependant, aucun algorithme d'approximation à facteur constant n'est à ce jour connu pour DENSE ROOTED TRIPLET INCONSISTENCY [82].

Résultats présentés. Nous établissons plusieurs noyaux **linéaires** pour les problèmes d'édition de relations précédemment cités. Nous utilisons en particulier le fait que la **consistence** de $\mathscr{R}$ puisse être caractérisée par des **conflits finis**, *i.e.* des ensembles de sommets de V de taille minimum induisant un ensemble de relations inconsistent. Ces caractérisations sont en particulier rendues possibles par l'hypothèse de densité. Pour obtenir ces résultats, nous introduisons deux notions indépendantes : **Partition Sûre** et **Conflict Packing**, que nous présentons **Chapitre 6**. Dans un premier temps, la notion de Partition Sûre nous permet d'obtenir un noyau contenant $(2+\epsilon)k$ sommets pour le problème FEEDBACK ARC SET IN TOURNAMENTS pour tout $\epsilon > 0$. Ce résultat améliore la meilleure borne connue jusqu'alors et contenant $O(k^2)$ sommets [5], et est présenté dans l'Annexe B. Cet algorithme de noyau utilise le PTAS existant pour pour ce problème comme routine. Par la suite, nous présentons la technique dite de **Conflict Packing** (**Chapitre 6**), déjà utilisée dans l'élaboration d'algorithmes de noyau pour quelques problèmes paramétrés [27, 63, 155]. Nous développons cette technique, obtenant **un noyau contenant au plus** $4k$ **sommets** pour FEEDBACK ARC SET IN TOURNAMENTS, ne faisant plus appel à l'algorithme d'approximation à facteur constant (**Section 7.1**). Cette nouvelle technique nous permet ainsi d'obtenir un noyau linéaire pour le problème DENSE ROOTED TRIPLET INCONSISTENCY (**Section 7.2**), qui n'admet pas à ce jour d'algorithme d'approximation à facteur constant [82]. Ce résultat améliore la meilleure borne connue pour le problème, consistant en un noyau contenant $O(k^2)$ sommets [82]. Finalement, nous démontrons comment utiliser cette méthode pour obtenir des noyaux linéaires pour les problèmes DENSE BETWEENNESS et DENSE CIRCULAR ORDERING, établissant ainsi les premiers algorithmes de noyaux pour ces problèmes [105].

CHAPITRE

6

Conflict Packing : un outil de conception de noyaux

Dans ce chapitre, nous présentons un outil de conception de noyaux que nous appelons **Conflict Packing**, et qui peut s'appliquer sur des problèmes d'édition de relations aussi bien que d'édition de graphes. Si le principe de base de cette méthode a déjà été utilisé dans quelques algorithmes de noyau [27, 63, 155], nous proposons une généralisation de cette idée nous permettant d'obtenir des algorithmes de noyau polynomiaux pour plusieurs problèmes [134]. Dans un premier temps, nous donnons un aperçu général de cette méthode, que nous illustrons en décrivant un noyau existant pour le problème TRIANGLE EDGE DELETION [27]. Nous mentionnons également la notion de **Partition Sûre**, une technique indépendante qui peut être combinée au Conflict Packing dans plusieurs cas. Dans un second temps, nous décrivons une méthode similaire intitulée **Algorithme de noyau via ajustement**. Introduite parallèlement au Conflict Packing par van Bevern et al. [152], cette dernière s'applique cependant principalement à des problèmes de **suppression de sommets**. Finalement, nous démontrons que ces deux méthodes permettent de retrouver un noyau quadratique pour le problème CLUSTER VERTEX DELETION, un résultat qui était déjà connu [1]. Des **applications** permettant d'**obtenir de nouveaux résultats** seront présentées dans le **Chapitre 7**.

CLUSTER VERTEX DELETION :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $H := G \setminus F$ est une union disjointe de cliques ?

Théorème 6.1. *Le problème* CLUSTER VERTEX DELETION *admet un noyau avec* $O(k^2)$ *sommets.*

6.1 Principe général et exemples

Dans cette section, nous nous intéressons principalement à des problèmes d'édition de relations, dont nous rappelons la définition :

Π-EDITION [1] :
Entrée : Une collection dense $\mathscr{R}$ de relations de taille $p \geqslant 2$ définies sur un univers V, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $\mathscr{F} \subseteq \mathscr{R}$ de taille au plus k dont l'édition dans $\mathscr{R}$ permet de satisfaire la propriété Π ?

Préliminaires. Rappelons que le terme *dense* signifie que $\mathscr{R}$ contient *exactement* une relation pour tout p-uplet de V. Dans la suite, nous considérons donc des instances $R := (V, \mathscr{R})$. Lorsque la collection $\mathscr{R}$ satisfait la propriété Π, nous disons que $\mathscr{R}$ est *consistente*. Dans le cas contraire, nous disons que $\mathscr{R}$ est *inconsistente*. Étant donnée une relation $c \in \mathscr{R}$, $V(c)$ dénote les sommets de V contenus dans c. Étant donné $S \subseteq V$, l'instance *induite par S* correspond à l'instance $(S, \mathscr{R}|S])$, où $\mathscr{R}[S] := \{c \in \mathscr{R} \ : \ V(c) \subseteq S\}$. Lorsque $\mathscr{R}$ est inconsistente, nous définissons un *conflit* comme un ensemble de sommets $C \subseteq V$ tel que $\mathscr{R}[C]$ est inconsistente. *Éditer* une relation c de $\mathscr{R}$ signifie supprimer c de $\mathscr{R}$ et ajouter $c' \neq c$ à $\mathscr{R}$. Étant donné un conflit C, *résoudre* C signifie éditer les relations de $\mathscr{R}[C]$ de sorte à ce que $\mathscr{R}[C]$ soit consistente. Nous considérons plus particulièrement des problèmes où la propriété Π est caractérisée par un ensemble fini de *conflits*. En termes d'édition de graphes, cela se traduit par le fait que la classe de graphes cible est caractérisée par un ensemble fini de sous-graphes induits interdits. Cela est par exemple le cas du problème CLUSTER VERTEX DELETION, un graphe étant une union disjointe de cliques *si et seulement si* il ne contient pas de P_3 induit. Nous illustrons d'ailleurs cette méthode sur ce problème dans la Section 6.2.

conflit

6.1.1 Partition Sûre et Conflict Packing

Dans un premier temps, nous définissons la notion de *Partition Sûre* pour des problèmes d'édition de relations. Nous verrons que cette notion permet d'obtenir des noyaux linéaires pour plusieurs problèmes [9, 134].

Partition Sûre. Soit (R, k) une instance du problème Π-EDITION dont la consistence peut être caractérisée par des *conflits finis*. L'intuition de la notion de *Partition Sûre* est la suivante : supposons que la collection $\mathscr{R}$ puisse être *partitionnée* en deux collections $\mathscr{R}_I$ et $\mathscr{R}_B$ telles que les conflits contenus dans $\mathscr{R}_B$ puissent être *résolus* de telle sorte que les conflits restants sont contenus dans $\mathscr{R}_I$. De ce fait, il est possible de montrer que les éditions des conflits de $\mathscr{R}_I$ et $\mathscr{R}_B$ sont *indépendantes*. Nous définissons une telle partition comme *sûre* dès lors que nous pouvons *certifier* que l'édition d'un ensemble $\mathscr{F}$ de relations de $\mathscr{R}_B$ résolvant *tous les conflits existant dans* $\mathscr{R}_B$ peut être effectuée de manière *valide*.

partition sûre

1. Nous appliquerons également cette méthode sur les problèmes $\mathscr{G}$-EDGE DELETION et $\mathscr{G}$-VERTEX DELETION.

Trouver une partition sûre. Afin d'obtenir un algorithme de noyau, il reste donc à déterminer comment trouver une telle partition en temps polynomial. Dans un travail réalisé avec Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH et Stéphan THOMASSÉ [9] (Annexe B), nous avons utilisé cette notion afin d'obtenir un noyau contenant au plus $(2+\epsilon)k$ sommets pour le problème FEEDBACK ARC SET IN TOURNAMENTS, pour tout $\epsilon > 0$. Nous donnons maintenant une intuition de notre algorithme de noyau. Un *tournoi* est un graphe orienté obtenu à partir d'une orientation arbitraire d'une clique. De plus, un *circuit orienté* est une séquence de sommets $\{v_1, \ldots, v_l\} \subseteq V$ telle que $v_i v_{i+1} \in A$ [2] pour tout $1 \leqslant i < l$ et $v_l v_1 \in A$.

FEEDBACK ARC SET IN TOURNAMENTS :
Entrée : Un tournoi $T := (V, A)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq A$ de taille au plus k tel que le tournoi obtenu en renversant tous les arcs de F dans T est acyclique ?

Afin de concevoir notre algorithme de noyau linéaire, nous avons utilisé la notion de *Partition Sûre*. Étant donné un tournoi dont les sommets sont ordonnés suivant un ordre $\sigma := v_1 \ldots v_n$,

partition ordonnée

une *partition ordonnée* est une partition $\mathscr{P} := \{V_1, \ldots, V_l\}$ de V telle que pour tout $i \in [l]$, V_i est un ensemble de sommets consécutifs de σ. Par convention, nous supposons que $u <_\sigma v$ pour tout $u \in V_i, v \in V_j$, $1 \leqslant i < j \leqslant l$. Observons qu'une partition ordonnée définit naturellement deux ensembles d'arcs disjoints : $A_B := \{uv \in A : u \in V_i,\ v \in V_j,\ i \neq j\}$ et $A_I := A \setminus A_B$. En d'autres termes, A_B correspond à l'ensemble d'arcs *traversant* $\mathscr{P}$, dont les extrémités appartiennent à deux parties différentes, et A_I à l'ensemble des arcs contenus dans une partie de $\mathscr{P}$. Nous dénotons par $\mathscr{B}(A_B)$

arc retour

l'ensemble des arcs retour (*i.e.* des arcs $v_i v_j$ avec $j < i$) contenus dans A_B.

Définition 6.2 (Partition Sûre)**.** *Soient $T_\sigma := (V, A, \sigma)$ un tournoi dont les sommets sont ordonnés suivant un ordre σ, et $\mathscr{P} := \{V_1, \ldots, V_l\}$ une partition ordonnée de T_σ telle que $\mathscr{B}(A_B) \neq \emptyset$. S'il existe une collection de $|\mathscr{B}(A_B)|$ circuits orientés sommets-disjoints contenus dans A_B, alors $\mathscr{P}$ est appelée* partition sûre.

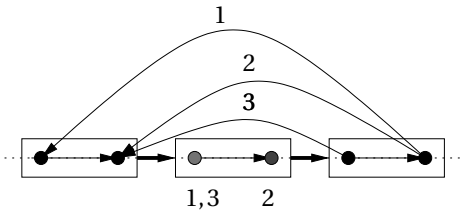


FIGURE 6.1 : Une partition sûre pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Les arcs non représentés sont des arcs *avant, i.e.* $v_i v_j$ avec $i < j$.

La principale intuition derrière cette notion est que renverser les arcs de $\mathscr{B}(A_B)$ permet d'obtenir une instance où les circuits orientés restants sont contenus dans A_I. Le fait qu'il existe $|\mathscr{B}(A_B)|$

2. Nous utilisons $v_i v_j$ pour dénoter l'orientation de l'arc de v_i vers v_j

circuits orientés disjoints permet de justifier la validité du renversement des arcs de $\mathscr{B}(A_B)$. Ainsi, cette notion permet d'obtenir une règle de réduction pour le problème [9]. Pour calculer une partition sûre en temps polynomial, nous avons en particulier utilisé un algorithme d' approximation à facteur constant [104, 106]. Par définition, un tel algorithme permet d'obtenir un ordre $v_1 \dots v_n$ sur les sommets contenant un nombre d'arcs retour linéaire en k. En se basant sur cet ordre, il est alors possible de calculer une partition sûre en temps polynomial dès lors que l'instance possède un nombre suffisant de sommets. Remarquons que les relations entre algorithmes d'approximation et algorithmes de noyau ne sont pas nouvelles, et ont déjà été utilisées à plusieurs reprises [110, 152].

Remarque. Comme nous le verrons dans le Chapitre 7, la notion de Partition Sûre peut également être définie sur des problèmes n'admettant pas d'algorithme d'approximation. Dans de tels cas, il n'est donc pas possible d'utiliser une solution approchée afin de calculer une partition sûre, et une stratégie différente est alors nécessaire.

Conflict Packing. Dans un travail réalisé avec Christophe PAUL et Stéphan THOMASSÉ [134], nous avons développé une technique utilisée dans quelques problèmes paramétrés [27, 63, 155], que nous avons appelée *Conflict Packing*, et qui permet de contourner ce problème. Informellement, cette dernière permet d'obtenir une solution particulière (sans garantie de taille), servant de point de départ pour calculer une partition sûre en temps polynomial. Ainsi, le Conflict Packing permet en quelque sorte de *remplacer* le rôle de l'algorithme d'approximation à facteur constant. Le principe de la méthode *Conflict Packing* est le suivant.

conflict packing

(i) **Conflict Packing** : dans un premier temps, calculer de manière gloutonne un *conflict packing* pour le problème considéré. De manière informelle, un conflict packing est une collection maximale de conflits relations (ou arêtes) disjoints. Dans la mesure où nous considérons des problèmes caractérisés par des conflits finis, un algorithme glouton permet bien d'obtenir une telle collection en temps polynomial. Comme nous le verrons par la suite, cette définition peut être raffinée pour plusieurs problèmes, en particulier lorsque deux conflits peuvent *partager* une relation mais requérir deux éditions distinctes afin d'être résolus. Dans tous les cas, l'objectif du Conflict Packing est de fournir une *borne inférieure sur le nombre d'éditions requises* pour obtenir une instance consistente.

(ii) **Taille** : étant donnée une instance *positive* du problème considéré, montrer qu'un conflict packing $\mathscr{C}$ contient $O(k^c)$ sommets, pour une certaine constante $c > 0$. Lorsque le conflict packing est défini comme une famille de conflits arêtes ou relations disjoints, le résultat est immédiat. Dans le cas où la définition du Conflict Packing $\mathscr{C}$ doit être raffinée, nous verrons comment borner la taille de $\mathscr{C}$. De plus, lorsque le Conflict Packing est utilisé pour le problème $\mathscr{G}$-VERTEX DELETION, il est nécessaire d'utiliser une règle de type *sunflower* pour obtenir un tel résultat.

(iii) **Réduction** : pour obtenir un noyau polynomial, il reste maintenant à borner la taille de $V \setminus V(\mathscr{C})$, où $V(\mathscr{C})$ dénote l'ensemble des sommets contenus dans le conflict packing $\mathscr{C}$. Par maximalité de $\mathscr{C}$, nous savons que l'instance induite par $G_{\mathscr{C}} := G \setminus V(\mathscr{C})$ (ou $R_{\mathscr{C}} := R \setminus V(\mathscr{C})$ obtenue en supprimant les sommets et les relations contenus dans $V(\mathscr{C})$) respecte la propriété Π. De plus, pour tout sommet $u \in V(\mathscr{C})$, l'instance $G_{\mathscr{C}} \cup \{u\}$ (ou $R_{\mathscr{C}} \cup \{u\}$) respecte également Π. Borner la taille de $V \setminus V(\mathscr{C})$ peut dont se faire en utilisant ces propriétés. Cette partie est *spécifique* au problème étudié, et ne peut se faire de manière générique.

Afin de pouvoir réaliser l'étape de **Réduction**, nous utiliserons notamment la notion de Partition Sûre. Nous donnons maintenant un exemple d'utilisation de cette méthode en considérant le problème TRIANGLE EDGE DELETION [27]. Nous verrons que la technique utilisée pour concevoir un algorithme de noyau linéaire pour ce problème d'édition de graphes rejoint la notion de Partition Sûre définie pour FEEDBACK ARC SET IN TOURNAMENTS.

6.1.2 TRIANGLE EDGE DELETION

Dans cette section, nous considérons le problème de suppression d'arête suivant :

TRIANGLE EDGE DELETION :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est un graphe sans triangle ?

Nous démontrons comment obtenir un noyau contenant au plus $4k$ sommets pour ce problème, améliorant la meilleure borne connue de $6k$ [27]. La première étape de l'algorithme de noyau consiste en une application exhaustive de la règle de réduction suivante, dont nous admettons la validité.

Règle 6.1. *Soit* (G, k) *une instance de* TRIANGLE EDGE DELETION. *Supprimer les sommets et arêtes n'appartenant à aucun triangle de* G.

Lemme 6.3 ([27])**.** *La Règle 6.1 est valide et peut être appliquée en temps polynomial.*

Nous définissons maintenant un conflict packing pour le problème TRIANGLE EDGE DELETION. Cette définition est similaire à celle introduite par Brügmann et al. [27].

Définition 6.4 (Conflict Packing)**.** *Étant donnée une instance* (G, k) *de* TRIANGLE EDGE DELETION, *un* conflict packing $\mathscr{C}$ *de* G *est une* collection maximale de triangles arête-disjoints.

Remarquons en particulier qu'un conflict packing conforme à la Définition 6.4 peut être calculé de manière gloutonne (*i.e.* en temps polynomial). Comme expliqué précédemment, nous savons que $|\mathscr{C}| \leqslant k$: en effet, au moins une suppression est nécessaire pour tout triangle de $\mathscr{C}$. Ainsi, si $|\mathscr{C}| > k$, nous retournons une instance négative triviale et de taille constante. Il suit alors que $|V(\mathscr{C})| \leqslant 3k$. Pour conclure, il reste donc à réduire le graphe $G \setminus V(\mathscr{C})$, qui est un graphe sans triangle par construction de $\mathscr{C}$. Observons que les graphes sans-triangles ne possèdent pas de réelles propriétés structurelles, et qu'il semble donc difficile de réaliser des règles de réduction sans avoir d'hypothèses supplémentaires. Ce sont de telles hypothèses que nous offre la Règle 6.1. En effet, pour toute instance réduite (G, k), en posant $I := V \setminus V(\mathscr{C})$, nous avons :

– I est un ensemble indépendant.
– tout triangle contenant un sommet $u \in I$ partage *exactement* une arête avec un triangle de $\mathscr{C}$.

L'idée de la prochaine règle de réduction rejoint la notion de *Partition Sûre* mentionnée précédemment pour FEEDBACK ARC SET IN TOURNAMENTS. Supposons qu'il existe $I' \subseteq I$ et $E' \subseteq$

$E(G[V(\mathscr{C})])$ tels que les triangles contenant des sommets de I' n'utilisent que des arêtes de E'. Supposons de plus que, pour toute arête $e \in E'$, il existe un sommet $v_e \in I'$ tel que $e \cup \{v_e\}$ induise un triangle de G. Dans ce cas, nous avons identifié un ensemble de $|E'|$ triangles arêtes-disjoints, et il est possible de démontrer que la suppression de toutes les arêtes de E' permet *toujours* d'obtenir une *suppression optimale* [27]. Formellement, nous admettons la règle de réduction suivante, qui est basée sur des propriétés de couplage d'un graphe biparti particulier (le *graphe des conflits*). Nous verrons plus tard que ces mêmes propriétés peuvent être utilisées pour FEEDBACK ARC SET IN TOURNAMENTS et DENSE ROOTED TRIPLET INCONSISTENCY (Chapitre 7). Nous définissons un graphe biparti $B := (I \cup J, E_B)$ comme suit :

graphe des conflits

- $J := \{v_e : e \in E(G[V(\mathscr{C})])\}$ et,
- il existe une arête uv_e, si $u \in I$ et l'ensemble $\{u\} \cup e$ induit un triangle de G.

Règle 6.2. *Soient (G,k) une instance de* TRIANGLE EDGE DELETION *et M un couplage maximum de B. Supprimer tous les sommets de I non saturés par M.*

Lemme 6.5 ([27])**.** *La Règle 6.2 est valide et peut être appliquée en temps polynomial.*

Théorème 6.6. *Le problème* TRIANGLE EDGE DELETION *admet un noyau avec au plus $4k$ sommets.*

Preuve. Soit (G,k) une instance positive de TRIANGLE EDGE DELETION réduite par les Règles 6.1 et 6.2. Soit B le graphe des conflits défini précédemment. Observons dans un premier temps que B ne peut pas contenir de couplage de taille $k+1$: en effet, cela impliquerait par définition l'existence de $k+1$ triangles arêtes-disjoints dans G, qui serait donc une instance négative. Soit M un couplage maximum de B : il suit que $|M| \leqslant k$. Maintenant, par la Règle 6.2, nous savons que I ne contient que des sommets saturés par M, et donc $|I| \leqslant k$. Comme $|V(\mathscr{C})| \leqslant 3k$ par définition du conflict packing $\mathscr{C}$, nous concluons que G contient au plus $4k$ sommets. □

6.1.3 Algorithme de noyau via ajustement

Nous décrivons maintenant un outil similaire à la notion de Conflict Packing. Introduit de manière parallèle au Conflict Packing, le concept d'*Algorithme de noyau via ajustement* [152] permet d'obtenir des noyaux polynomiaux pour des problèmes de *suppression de sommets*, qui peuvent se définir comme suit.

$\mathscr{G}$-VERTEX DELETION :
Entrée : Un graphe $G := (V,E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ appartient à $\mathscr{G}$?

Dans [152], van Bevern et al décrivent une méthode générale permettant d'obtenir des noyaux polynomiaux pour ce problème lorsque la classe $\mathscr{G}$ peut être caractérisée par une famille finie de sous-graphes induits interdits $\mathscr{F}$. Cela leur permet en particulier d'obtenir un noyau polynomial pour le problème s-PLEX VERTEX DELETION, un *s-plex* étant un graphe où tout sommet est adjacent à tous les autres sommets sauf au plus $s-1$ (les cliques sont ainsi des 1-plexes). Nous décrivons

tout d'abord leur méthode, puis démontrons comment leur technique peut être retrouvée en utilisant la notion de Conflict Packing. Dans la Section 6.2, nous illustrons ces résultats sur le problème CLUSTER VERTEX DELETION, qui correspond à 1-PLEX VERTEX DELETION par définition. La méthode d'Algorithme de noyau via ajustement se déroule en trois étapes :

- **Approximation** : dans un premier temps, calculer de manière gloutonne une solution approchée S pour le problème. Cela peut se faire en en temps $O(n^h)$ en calculant une famille de sous-graphes induits interdits sommets-disjoints, où h est le nombre maximum de sommets contenus dans un sous-graphe induit interdit de $\mathscr{F}$. Pour toute instance positive du problème considéré, S contient ainsi au plus hk sommets. L'objectif est maintenant de borner le nombre de sommets contenus dans le graphe $G \setminus S$, ce qui peut-être fait en exploitant le fait que $G \setminus S$ est $\mathscr{F}$-free (par définition de S) et via l'étape d'*ajustement*.
- **Ajustement** : dans un second temps, calculer un *ensemble d'ajustement* $T \subseteq V \setminus S$ en utilisant la règle de réduction sunflower. Par définition de cette règle de réduction, et comme $|S| \in O(k)$, nous déduisons que $|T| \in O(k^2)$. En particulier, cela implique que le graphe $(G \setminus (S \cup T)) \cup \{u\}$ est $\mathscr{F}$-free pour tout sommet $u \in S$ (*propriété locale d'ajustement*).
- **Réduction** : la dernière étape est spécifique au problème, et consiste à réduire le graphe $G \setminus (S \cup T)$ en utilisant des règles de réduction spécifiques au problème considéré ainsi que la propriété locale d'ajustement. Si le nombre de sommets de ce graphe peut être réduit à $p(k)$ pour un certain polynôme p, nous obtenons alors un noyau contenant $O(h^2k^2 + p(k))$ sommets.

Observons maintenant que la notion de Conflict Packing peut être adaptée pour obtenir des résultats similaires. Dans le cas du problème $\mathscr{G}$-VERTEX DELETION, un conflict packing $\mathscr{C}$ est une *famille maximale de sous-graphes induits interdits s'intersectant deux-à-deux en au plus un sommet*. Par définition, l'ensemble $V(\mathscr{C})$ est une approximation puisque le graphe $G \setminus V(\mathscr{C})$ est $\mathscr{F}$-free. En utilisant une règle de réduction de type *sunflower*, il est ensuite possible d'observer que $|V(\mathscr{C})| \leqslant h^2k^2$. Maintenant, le graphe $(G \setminus V(\mathscr{C})) \cup \{u\}$ est $\mathscr{F}$-free pour tout $u \in V(\mathscr{C})$, ce qui correspond à la propriété locale d'ajustement. Finalement, l'étape de réduction est à nouveau spécifique au problème, et correspond donc à l'étape de réduction de la technique d'Algorithme de noyau via ajustement.

6.2 Noyau quadratique pour CLUSTER VERTEX DELETION

Nous donnons un autre exemple d'application de la méthode Conflict Packing sur le problème CLUSTER VERTEX DELETION. Ce problème admet un noyau quadratique [1], mais nous présentons une manière alternative afin d'obtenir un tel résultat. Ce résultat peut également être obtenu de manière similaire en utilisant un algorithme de noyau via ajustement [152]. Formellement, le problème se définit comme suit.

CLUSTER VERTEX DELETION :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ est une union disjointe de cliques ?

Afin d'obtenir un algorithme de noyau pour le problème, nous utilisons les règles de réduction générique réduisant les composantes connexes appartenant à la classe cible, ainsi que les sunflowers. Leur validité découle des Lemmes 2.46 et 2.48.

Règle 6.3 (Composante connexe)**.** *Soit* (G,k) *une instance de* CLUSTER VERTEX DELETION*, et* C *une composante connexe de* G *telle que* $G[C]$ *est une clique. Supprimer* C *de* G*.* Règle 2.5

Lemme 6.7. *La Règle 6.3 est valide et peut être appliquée en temps polynomial.* Lemme 2.46

Règle 6.4 (Sunflower)**.** *Soient* (G,k) *une instance de* CLUSTER VERTEX DELETION*, et* $\mathscr{S} := \{P_1,\dots,P_m\}$ *un ensemble de* P_3 *induits s'intersectant* 2*-à-*2 *en un sommet* u*,* $m > k$*. Supprimer* u *de* G*, ajouter* u *à la solution et décrémenter* k *de* 1*.* Règle 2.6

Lemme 6.8. *La Règle 6.4 est valide et peut être appliquée en temps polynomial.* Lemme 2.48

Nous donnons maintenant la définition d'un conflict packing pour ce problème.

Définition 6.9. *Soit* (G,k) *une instance de* CLUSTER VERTEX DELETION*. Un* conflict packing *de* G *est une collection maximale* $\mathscr{C}$ *de* P_3 induits s'intersectant 2-à-2 en au plus un sommet.

Observons qu'un conflict packing peut être calculé en temps polynomial en énumérant tous les P_3 du graphe G, ce qui peut trivialement être réalisé en $O(n^3)$. Comme expliqué précédemment, le nombre de sommets contenus dans le conflict packing d'une instance positive peut être borné par $O(k^2)$.

Lemme 6.10. *Soient* (G,k) *une instance positive de* CLUSTER VERTEX DELETION *réduite par les Règles 6.3 et 6.4 et* $\mathscr{C}$ *un conflict packing de* G*. Alors* $|\mathscr{C}| \leqslant k^2$ *et* $|V(\mathscr{C})| \leqslant 6k^2+3k$*.*

Preuve. Puisque G est une instance positive, nous savons que $\mathscr{C}$ contient un ensemble $\mathscr{P}$ de P_3 induits *sommets-disjoints* de taille $p \leqslant k$, soit $\mathscr{P} := \{P_1,\dots,P_p\}$. De plus, comme G est réduit par la Règle 6.4, pour chaque sommet u contenu dans P_i, $i \in [p]$, il existe un ensemble de taille au plus k de P_3 induits s'intersectant 2-à-2 sur le sommet u. Ainsi, comme $|V(\mathscr{P})| \leqslant 3k$, il existe au plus $6k^2$ sommets dans $V(\mathscr{C}) \setminus V(\mathscr{P})$, impliquant au final que $|V(\mathscr{C})| \leqslant 6k^2+3k$. □

Remarque. En termes d'algorithme de noyau via ajustement, le Lemme 6.10 combine les étapes d'approximation et d'ajustement. En effet, l'ensemble $\mathscr{P}$ est une approximation de l'instance, et la Règle 6.4 permet elle de borner le nombre de sommets lors de l'étape d'ajustement.

Nous présentons maintenant la règle de réduction permettant de borner le nombre de sommets contenus dans $V \setminus V(\mathscr{C})$, et permettant donc d'obtenir un algorithme de noyau pour le problème. Étant donnés (G,k) une instance de CLUSTER VERTEX DELETION et $\mathscr{C}$ un conflict packing de G, le graphe $H := G \setminus V(\mathscr{C})$ est une union disjointe de cliques par définition de $\mathscr{C}$. Nous posons $H := \{C_0, C_1, \dots, C_l\}$ pour dénoter cette union disjointe de cliques, où C_0 dénote le graphe vide. Par maximalité de $\mathscr{C}$, nous avons maintenant la propriété suivante : pour tout sommet $u \in V(\mathscr{C})$, le graphe $H \cup \{u\}$ est une union disjointe de cliques. Ainsi, il existe une *unique* clique C_i, $0 \leqslant i \leqslant l$, telle que $C_i \cup \{u\}$ est une clique de $H \cup \{u\}$. Pour tout $0 \leqslant i \leqslant l$, nous définissons $S_i \subseteq V(\mathscr{C})$ comme

l'ensemble des sommets de $V(\mathscr{C})$ *s'insérant* dans la clique C_i dans H. Observons en particulier que les cliques C_i, $i \in [l]$ forment des ensembles de vrais jumeaux dans G.

Règle 6.5. *Soient (G,k) une instance de* CLUSTER VERTEX DELETION *et $\mathscr{C}$ un conflict packing de G. Pour tout $i \in [l]$ tel que $|S_i| < |C_i|$, supprimer un sommet arbitraire $u \in C_i$ de G.*

Lemme 6.11. *La Règle 6.5 est valide et peut être appliquée en temps polynomial.*

Preuve. Nous dénotons par G' le graphe réduit par la Règle 6.5. Observons tout d'abord que toute k-édition de G est une k-édition de G' puisque les unions disjointes de cliques sont stables par sous-graphe induits. Inversement, nous démontrons l'affirmation suivante, qui nous permettra de conclure.

Affirmation 6.12. *Pour tout $i \in [l]$ tel que $|S_i| < |C_i|$, il existe une édition optimale de G' ne supprimant pas les sommets de $C_i \setminus \{u\}$.*

Preuve. La preuve se fait par contradiction. Soit F' une édition optimale de G' contenant des sommets de $C_i \setminus \{u\}$. Comme $C_i \setminus \{u\}$ définit un ensemble de vrais jumeaux de G', observons que $C_i \setminus \{u\} \subseteq F'$: en effet, supposons que ce ne soit pas le cas, et considérons v un sommet de $(C_i \setminus \{u\}) \setminus F'$. Le graphe obtenu à partir de $G' - F'$ en réinsérant les sommets de $(C_i \setminus \{u\}) \cap F'$ (qui sont des vrais jumeaux de v dans $G' - F'$) est une union disjointe de cliques, et donc $F' \setminus C_i$ est une édition de plus petite taille que F' : contradiction. Maintenant, comme $|S_i| < |C_i|$, l'édition $F' := (F \setminus (C_i \setminus \{u\})) \cup S_i$ est une édition de G vérifiant $|F'| \leqslant |F|$. ◇

Soit F' une k-édition de G'. Par l'Affirmation 6.12, nous pouvons supposer que $F' \cap (C_i \setminus \{u\}) = \emptyset$. Ainsi, le graphe $G' - F'$ contient toujours les sommets de $C_i \setminus \{u\}$, qui sont des vrais jumeaux du sommet u. Il suit que le graphe $(G \setminus F') \cup \{u\}$ est une union disjointe de cliques, et ainsi F' est une k-édition de G. Observons finalement que cette règle de réduction peut être appliquée en temps polynomial dans la mesure où $\mathscr{C}$ peut être calculé en temps polynomial. □

Théorème 6.13. *Le problème* CLUSTER VERTEX DELETION *admet un noyau avec $O(k^2)$ sommets.*

Preuve. Soient (G,k) une instance positive de CLUSTER VERTEX DELETION réduite par les Règles 6.3 à 6.5, et $\mathscr{C}$ un conflict packing de G. Par le Lemme 6.10, nous savons que $\sum_{i=0}^{l} |S_i| = |V(\mathscr{C})| \leqslant 6k^2 + 3k$. De plus, comme G est réduit par la Règle 6.5, nous savons que $|C_i| \leqslant |S_i|$ pour tout $0 \leqslant i \leqslant l$. Il suit alors que :

$$\sum_{i=0}^{l} |C_i| \leqslant \sum_{i=0}^{l} |S_i| \leqslant 6k^2 + 3k$$

et donc le graphe G contient au plus $12k^2 + 6k$ sommets. □

CHAPITRE

7 Applications de la méthode Conflict Packing

Dans ce chapitre, nous présentons de nouvelles applications de la méthode **Conflict Packing**, décrite dans le **Chapitre 6**. Dans un premier temps, nous utilisons cette dernière pour obtenir un noyau linéaire (contenant au plus $4k$ sommets) pour le problème FEEDBACK ARC SET IN TOURNAMENTS (**Section 7.1**). Si un noyau contenant au plus $(2+\epsilon)k$ sommets existe pour tout $\epsilon > 0$ (Annexe B), les preuves pour obtenir le noyau contenant au plus $4k$ sommets sont plus simples et permettent de s'affranchir de l'utilisation d'un algorithme d'approximation à facteur constant. Cela sera en particulier utile pour concevoir des algorithmes de noyau pour des problèmes n'admettant pas de tel algorithme. En ce sens, nous étudions le problème DENSE ROOTED TRIPLET INCONSISTENCY, qui trouve des applications dans le domaine de la phylogénie [4, 30, 146]. Ce problème NP-Complet [30] a été largement étudié dans la littérature [30, 77, 82, 145]. Des algorithmes FPT sont connus [82, 145], la meilleure complexité étant $O^*(2^{k^{1/3}\log k})$ [82]. De plus, un noyau quadratique a été établi par Guillemot et Mnich [82]. Nous utilisons les notions de **Partition Sûre** et de **Conflict Packing** pour établir le premier noyau linéaire pour DENSE ROOTED TRIPLET INCONSISTENCY (**Section 7.2**).

DENSE ROOTED TRIPLET INCONSISTENCY :
Entrée : Un ensemble de feuilles V, une collection dense $\mathscr{R}$ d'arbres binaires enracinés (**triplets**) définis sur trois feuilles de V, un entier k.
Paramètre : k.
Question : Existe-t-il un arbre binaire enraciné T défini sur V contenant tous les triplets de $\mathscr{R}$ sauf au plus k ?

Théorème 7.1. *Le problème* DENSE ROOTED TRIPLET INCONSISTENCY *admet un noyau avec au plus* $5k$ *feuilles.*

Finalement, nous prouvons le théorème suivant, en utilisant des arguments similaires mais quelque peu simplifiés, dû à la nature des problèmes (**Section 7.3**).

Theorem 7.2. *Les problèmes* DENSE BETWEENNESS *et* DENSE CIRCULAR ORDERING *admettent des noyaux contenant au plus* $5k$ *feuilles.*

7.1 Noyau linéaire pour FEEDBACK ARC SET IN TOURNAMENTS

Dans cette section, nous présentons un algorithme de noyau *linéaire* pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Nous rappelons la définition de ce problème, un *tournoi* étant obtenu à partir d'une clique en orientant arbitrairement chacune de ses arêtes. Dans ce qui suit, nous utilisons des notations similaires aux notations pour les graphes non-orientés introduites Chapitre 1.

FEEDBACK ARC SET IN TOURNAMENTS :
Entrée : Un tournoi $T := (V, A)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq A$ de taille au plus k tel que le tournoi obtenu en renversant tous les arcs de F dans T est acyclique ?

Pour obtenir notre algorithme de noyau, nous utilisons les notions de Conflict Packing et de Partition Sûre décrites dans le Chapitre 6.

7.1.1 Tournois et acyclicité

Nous présentons quelques résultats classiques pour le problème FEEDBACK ARC SET IN TOURNAMENTS, ainsi que des définitions liées à la notion de Partition Sûre. Le résultat suivant permet de caractériser l'acyclicité d'un tournoi. En particulier, il permet d'établir que la propriété *être un tournoi acyclique* peut être caractérisée par un conflit fini. Étant donné un tournoi acyclique $T := (V, A)$, un *ordre transitif* de T est un ordre $\sigma := v_1 \dots v_n$ sur V tel que $v_i v_j \in A$ pour tout $i < j$.

ordre transitif

Lemme 7.3 (Classique)**.** *Soit $T := (V, A)$ un tournoi. Les propriétés suivantes sont équivalentes : (i) T est acyclique ; (ii) T ne contient pas de circuit orienté de taille* 3 *(*triangle *orienté) ; (iii) T admet un ordre transitif.*

triangle orienté

Dans la suite de cette section, les *triangles orientés* représentent donc les *conflits* du problème FEEDBACK ARC SET IN TOURNAMENTS.

Remarque. Dans la littérature, le problème du FEEDBACK ARC SET correspond à la recherche d'un ensemble d'arcs de taille au plus k dont la *suppression* permet d'obtenir un graphe orienté acyclique [144]. Par définition des noyaux, les règles de réduction ne doivent pas *sortir* du problème considéré, et l'instance réduite doit donc être un tournoi. En ce sens, nous avons défini FEEDBACK ARC SET IN TOURNAMENTS comme la recherche d'un ensemble de taille au plus k dont le *renversement* permet d'obtenir un tournoi acyclique. En fait, les deux problèmes sont équivalents, comme indiqué par le résultat suivant.

Lemme 7.4 ([137])**.** *Soient $D := (V, A)$ un graphe orienté et F un feedback arc set*[1] *de D. Le graphe D' obtenu à partir de D en retournant tous les arcs de F dans D est acyclique.*

1. Un feedback arc set d'un graphe orienté $D := (V, A)$ est un ensemble d'arcs dont la suppression permet d'obtenir un graphe orienté acyclique.

Tournoi ordonné. Dans ce qui suit, nous considérons des *tournois ordonnés* $T_\sigma := (V, A, \sigma)$, où
arc retour $\sigma := v_1 \dots v_n$ est un ordre fixé sur les sommets V. Nous définissons un *arc retour* de T_σ comme un arc $v_i v_j$ avec $j < i$. Étant donné un arc retour $f := vu$ de T_σ, nous définissons $span(f)$ comme l'ensemble de sommets $\{w \in V \ : \ u <_\sigma w <_\sigma v\}$. Étant donné $V' \subseteq V$, nous définissons $T_\sigma[V'] := (V', A', \sigma_{V'})$ comme le tournoi ordonné induit par V', où A' contient les arcs uv tels que $u, v \in V'$. De manière similaire, étant donné $A' \subseteq A$, nous définissons $T_\sigma[A']$ comme le tournoi $(V', A', \sigma_{V'})$ où V' correspond aux extrémités des arcs de A'. Nous donnons maintenant deux définitions qui seront les outils centraux de notre algorithme de noyau.

certificat **Définition 7.5** (Certificat)**.** *Soient $T_\sigma := (V, A, \sigma)$ un tournoi ordonné et $f := vu$ un arc retour de T_σ. Soit $w \in span(f)$ un sommet incident à aucun arc retour de T_σ. L'ensemble $c(f) := \{u, v, w\}$ est appelé un* certificat *pour f.*

Remarque. Pour tout arc retour f de T_σ, l'ensembe $c(f)$ forme un triangle orienté (*i.e.* un conflit) de T.

certifier **Définition 7.6** (Certifier)**.** *Soient $T_\sigma := (V, A, \sigma)$ un tournoi ordonné et $F \subseteq A$ un ensemble d'arcs retour de T_σ. S'il existe un ensemble $\mathscr{F} := \{c(f) \ : \ f \in F\}$ de certificats* arc-disjoints *pour les arcs de F, nous disons que l'ensemble F peut être* certifié.

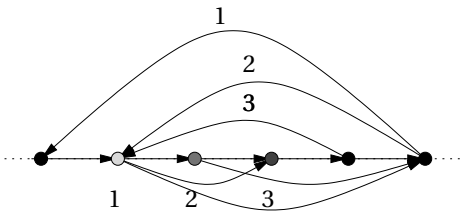


FIGURE 7.1 : Illustration de la Définition 7.6 : les sommets 1, 2 et 3 permettent de certifier la famille d'arcs $\{1,2,3\}$.

Afin de concevoir notre algorithme de noyau, nous admettons la validité de la règle de réduction suivante, qui permet de supposer que tout sommet d'une instance (T, k) de FEEDBACK ARC SET IN TOURNAMENTS est contenu dans un triangle.

Règle 7.1 (Sommet inutile)**.** *Soit (T, k) une instance de* FEEDBACK ARC SET IN TOURNAMENTS*. Supprimer de T tous les sommets n'appartenant à aucun triangle.*

Lemme 7.7 ([5])**.** *La Règle 7.1 est valide et peut être appliquée en temps polynomial.*

7.1.2 Conflict Packing et Partition Sûre

Partition sûre. Nous rappelons maintenant les définitions et notations fondamentales pour établir la notion de Partition Sûre pour FEEDBACK ARC SET IN TOURNAMENTS. Ces notations ont déjà été présentées dans le Chapitre 6, mais nous les rappelons par souci de simplicité. Étant donné un
partition ordonnée tournoi ordonné $T_\sigma := (V, A, \sigma)$, une *partition ordonnée* de T_σ est une partition $\mathscr{P} := \{V_1, \dots, V_l\}$ de

V telle que pour tout $i \in [l]$, V_i est un ensemble de sommets consécutifs de σ. Par convention, nous supposons que $u <_\sigma v$ pour tout $u \in V_i, v \in V_j$, $1 \leqslant i < j \leqslant l$. Observons qu'une partition ordonnée définit naturellement deux ensembles d'arcs disjoints : $A_B := \{uv \in A : u \in V_i,\ v \in V_j,\ i \neq j\}$ et $A_I := A \setminus A_B$. En d'autres termes, A_B correspond à l'ensemble d'arcs *traversant* $\mathscr{P}$, dont les extrémités appartiennent à deux parties différentes de $\mathscr{P}$, et A_I l'ensemble des arcs contenus dans une partie de $\mathscr{P}$. Nous dénotons par $\mathscr{B}(A_B)$ l'ensemble des arcs retour contenus dans A_B. Nous redéfinissons à présent la notion de Partition Sûre (Définition 6.2) en utilisant les terminologies nouvellement introduites.

partition sûre

Définition 7.8 (Partition sûre)**.** *Soient $T_\sigma := (V, A, \sigma)$ un tournoi ordonné et $\mathscr{P} := \{V_1, \ldots, V_l\}$ une partition ordonnée de T_σ telle que $\mathscr{B}(A_B) \neq \emptyset$. Si les arcs de $\mathscr{B}(A_B)$ peuvent être* certifiés *en n'utilisant que des arcs de A_B, alors $\mathscr{P}$ est une* partition sûre *de T_σ.*

Remarque. En d'autres termes, une partition ordonnée est sûre s'il est possible de trouver une collection de $|\mathscr{B}(A_B)|$ conflits arc-disjoints n'utilisant que des arcs de A_B, ce qui correspond à la Définition 6.2.

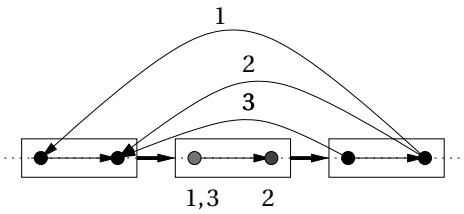


FIGURE 7.2 : Une partition sûre pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Les arcs non représentés sont des arcs *avant, i.e.* $v_i v_j$ avec $i < j$. Ainsi, les arcs 1 et 3 peuvent être certifiés avec le même sommet, l'arc 2 étant certifié avec un sommet différent.

Règle 7.2 (Partition sûre)**.** *Soient (T_σ, k) une instance ordonnée de* FEEDBACK ARC SET IN TOURNAMENTS *et $\mathscr{P} := \{V_1, \ldots, V_l\}$ une partition sûre de T_σ. Retourner tous les arcs de $\mathscr{B}(A_B)$ et décrémenter k de $|\mathscr{B}(A_B)|$.*

Nous démontrons uniquement la validité de la Règle 7.2. Nous verrons par la suite comment calculer une telle partition en temps polynomial.

Lemme 7.9. *La Règle 7.2 est valide.*

Preuve. Nous dénotons par T'_σ le tournoi obtenu à partir de T_σ en retournant les arcs de $\mathscr{B}(A_B)$. Supposons dans un premier temps que T'_σ admette une $(k - |\mathscr{B}(A_B)|)$-édition F'. Par définition de T'_σ, tous les triangles orientés sont contenus *dans* les parties de $\mathscr{P}$. Comme le renversement d'un arc de A_I dans T'_σ ne peut pas créer de triangle orienté contenant un arc de A_B, il suit que l'ensemble d'arcs $F' \cup \mathscr{B}(A_B)$ est une k-édition de T_σ. Inversement, afin de démontrer la validité de la règle de réduction, nous prouvons le résultat suivant.

Affirmation 7.10. *Il existe une k-édition de T_σ contenant $\mathscr{B}(A_B)$.*

Preuve. Nous démontrons que les éditions de $T_\sigma[A_I]$ et $T_\sigma[A_B]$ sont indépendantes. En d'autres termes, nous prouvons $fas(T_\sigma) = fas(T_\sigma[A_I]) + fas(T_\sigma[A_B])$, où $fas(T)$ dénote la taille d'une édition pour le tournoi T. Dans un premier temps, remarquons que $fas(T_\sigma) \geqslant fas(T_\sigma[A_1]) + fas(T_\sigma[A_2])$ pour toute bipartition $\{A_1, A_2\}$ de A. Il reste donc à montrer que $fas(T_\sigma) \leqslant fas(T_\sigma[A_I]) + fas(T_\sigma[A_B])$. Cela vient du fait que lorsque tous les arcs de $\mathscr{B}(A_B)$ ont été retournés (ce qui est une édition optimale pour $T_\sigma[A_B]$ par définition d'une partition sûre), les seuls triangles orientés restants sont inclus dans A_I. Dans le tournoi T'_σ ainsi obtenu, aucune édition réalisée dans $T'_\sigma[V_i]$, $i \in [l]$, ne peut créer de triangles orientés contenant un arc de A_B. Il suit que les éditions dans $T_\sigma[A_I]$ et $T_\sigma[A_B]$ sont indépendantes. Ainsi, $fas(T_\sigma) = fas(T_\sigma[A_I]) + fas(T_\sigma[A_B])$. En particulier, cela implique qu'il existe une k-édition de T contenant les arcs de $\mathscr{B}(A_B)$. ⋄

Soit F une k-édition de T. Par l'Affirmation 7.10, nous pouvons supposer que F contient les arcs $\mathscr{B}(A_B)$. Il suit que le tournoi obtenu T' à partir de T_σ en retournant tous les arcs de $\mathscr{B}(A_B)$ peut être rendu acyclique en au plus $k - |\mathscr{B}(A_B)|$ éditions. □

Conflict Packing. Nous illustrons maintenant le principe du Conflict Packing sur le problème FEEDBACK ARC SET IN TOURNAMENTS, obtenant un noyau contenant au plus $4k$ sommets. Comme nous l'avons mentionné précédemment, un noyau linéaire est connu pour ce problème (Annexe B). Ce dernier fait appel à la notion de Partition Sûre ainsi qu'à un algorithme d'approximation à facteur constant. Plus précisément, dans un travail réalisé avec Stéphane BESSY, Fedor V. FOMIN, Serge GASPERS, Christophe PAUL, Saket SAURABH et Stéphan THOMASSÉ [9], nous avons utilisé un PTAS existant pour le problème [104, 106] afin d'obtenir un noyau contenant $(2+\epsilon)k$ sommets pour tout $\epsilon > 0$. Les techniques que nous présentons dans la suite de cette section sont plus simples, et permettent également de s'affranchir de l'utilisation d'un algorithme d'approximation à facteur constant. Ainsi, cette méthode est plus générale et pourra être réutilisée pour d'autres problèmes, n'admettant par exemple pas d'algorithme d'approximation à facteur constant (Section 7.2).

conflict packing

Définition 7.11 (Conflict Packing)**.** *Soit (T,k) une instance de* FEEDBACK ARC SET IN TOURNAMENTS. *Un* conflict packing *$\mathscr{C}$ de T est une collection maximale de triangles orientés arc-disjoints de T.*

Remarque. Il est possible de calculer de manière gloutonne (*i.e.* en temps $O(n^3)$) un conflict packing $\mathscr{C}$ d'une instance (T,k) de FEEDBACK ARC SET IN TOURNAMENTS.

Lemme 7.12. *Soient (T,k) une instance positive de* FEEDBACK ARC SET IN TOURNAMENTS *et $\mathscr{C}$ un conflict packing de T. Alors $|V(\mathscr{C})| \leqslant 3k$.*

Preuve. Dans un premier temps, observons que $|\mathscr{C}| \leqslant k$: en effet, dans le cas contraire, puisque les triangles orientés contenus dans $\mathscr{C}$ sont arc-disjoints, un renversement serait requis pour chacun de ces triangles, et donc T ne serait pas une instance positive. Comme chaque élément de $\mathscr{C}$ comporte au plus trois sommets, il suit que $|V(\mathscr{C})| \leqslant 3k$. □

Étant donnés (T,k) une instance de FEEDBACK ARC SET IN TOURNAMENTS et $\mathscr{C}$ un conflict packing de T, nous démontrons maintenant comment ordonner les sommets de T selon un ordre σ en garantissant qu'aucun sommet de $V \setminus V(\mathscr{C})$ n'est incident à un arc retour de T_σ. Un tel ordre nous sera par la suite utile afin de calculer une partition sûre en temps polynomial.

Lemme 7.13 (Conflict Packing)**.** *Soient* (T,k) *une instance de* FEEDBACK ARC SET IN TOURNAMENTS *et* $\mathscr{C}$ *un conflict packing de* T*. Il existe un ordre* σ *sur les sommets* V *de* T *tel que tout arc retour* vu *de* T_σ *vérifie* $u,v \in V(\mathscr{C})$*. De plus, un tel ordre peut être calculé en temps polynomial.*

Preuve. Par définition de $\mathscr{C}$, observons tout d'abord que le tournoi T' induit par $V' := V \setminus V(\mathscr{C})$ est acyclique. En effet, si T' contient un triangle orienté alors il est possible d'augmenter le conflict packing $\mathscr{C}$, contredisant sa maximalité. Nous considérons l'unique ordre transitif σ' associé à T', qui peut être calculé en temps polynomial. En utilisant encore une fois la maximalité de $\mathscr{C}$, remarquons que pour tout sommet $u \in V(\mathscr{C})$, le tournoi $T_u := T[V' \cup \{u\}]$ est acyclique. Puisque l'ordre transitif σ' associé à T' est unique, il existe une *unique* paire de sommets $\{v,w\}$ telle que : (i) v et w sont consécutifs dans σ' et (ii) l'ordre σ_u obtenu en insérant u entre v et w dans σ' définit l'ordre transitif de T_u. Nous appelons la paire $\{v,w\}$ le *locus* de u. Nous considérons un ordre σ sur les sommets V de T vérifiant :

(i) pour tout $v,w \in V'$, $v <_\sigma w$ si $v <_{\sigma'} w$ et,

(ii) si $u \in V(\mathscr{C})$ alors $v <_\sigma u <_\sigma w$, où $\{v,w\}$ est le locus de u.

Comme σ' et σ_u sont transitifs pour tout $u \in V(\mathscr{C})$, il suit par construction que les arcs retour vu de σ vérifient $u,v \in V(\mathscr{C})$. □

Théorème 7.14. *Le problème* FEEDBACK ARC SET IN TOURNAMENTS *admet un noyau avec au plus* $4k$ *sommets.*

Preuve. Soit (T,k) une instance positive de FEEDBACK ARC SET IN TOURNAMENTS réduite par la Règle 7.1. Nous calculons de manière gloutonne (*i.e.* en temps polynomial) un conflict packing $\mathscr{C}$ pour T et dénotons par σ l'ordre sur V obtenu via le Lemme 7.13. Nous considérons maintenant le graphe biparti $B := (R \cup V', E)$, où :

(i) $V' := V \setminus V(\mathscr{C})$,

(ii) R contient un sommet r_{vu} pour tout arc retour vu de T_σ et,

(iii) $r_{vu}w \in E$ si $w \in V'$ et l'ensemble $\{u,v,w\}$ est un certificat pour l'arc vu.

Observons qu'un couplage de taille $k+1$ dans B correspond à une collection de $k+1$ triangles orientés arc-disjoints dans T_σ, ce qui ne peut pas exister puisque nous avons supposé que T était une instance positive. Par le Théorème de König-Egerváry [59, 108], nous savons que la taille minimum d'un vertex cover dans un graphe biparti est égale à la taille maximum d'un couplage. Il suit que B admet un vertex cover D de taille au plus k. Nous posons $D_1 := D \cap R$ et $D_2 := D \cap V'$. Supposons maintenant que $|V| > 4k$. Par le Lemme 7.12, nous savons que $|V(\mathscr{C})| \leqslant 3k$, impliquant $|V \setminus V(\mathscr{C})| > k$. De plus, comme $|D_2| \leqslant k$, il suit que $V' \setminus D_2 \neq \emptyset$. Soit $\mathscr{P} := \{V_1, \ldots, V_l\}$ la partition ordonnée de T_σ où tout V_i, $i \in [l]$, est soit un sommet de $V' \setminus D_2$, soit un sous-ensemble maximal de sommets de $V \setminus (V' \setminus D_2)$, consécutifs dans σ.

Affirmation 7.15. *La partition* $\mathscr{P}$ *est une partition sûre de* T_σ*.*

Preuve. Soit w un sommet de $V' \setminus D_2$. Par le Lemme 7.13, w n'est incident à aucun arc retour dans T_σ. De plus, comme T est réduit par la Règle 7.1, il existe un arc retour $f := vu$ tel que $w \in span(f)$: en effet, dans le contraire, w ne serait contenu dans aucun triangle orienté. Comme w est un élément isolé de $\mathscr{P}$, cela implique que A_B contient au moins un arc retour et donc $\mathscr{B}(A_B) \neq \emptyset$.

Soit $f := vu$ un arc retour de T_σ contenu dans A_B. Par construction de $\mathscr{P}$, il existe un sommet $w \in (V' \setminus D_2) \cap span(f)$. De ce fait, $\{u, v, w\}$ définit un certificat pour l'arc f et donc $r_{vu}w$ est une arête du graphe B. Observons que puisque D est un vertex cover de B et $w \notin D_2$, le sommet r_{vu} doit appartenir à D_1. Ainsi l'ensemble $R' \subseteq R$ correspondant aux arcs retour de A_B est inclus dans D_1.

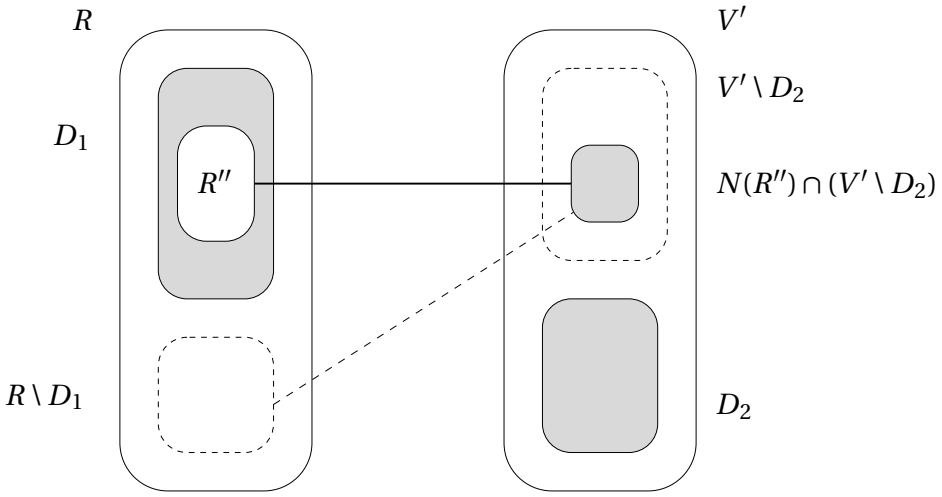


FIGURE 7.3 : Illustration de la preuve de l'Affirmation 7.15.

Pour conclure, nous prouvons que R' peut être couplé dans $V' \setminus D_2$. Pour cela, nous utilisons le théorème de Hall [93], qui énonce que R' peut être matché dans $V' \setminus D_2$ si et seulement si $|R''| \leqslant |N_{V' \setminus D_2}(R'')|$ est vérifié pour tout $R'' \subseteq R'$. Par contradiction, supposons qu'il existe $R'' \subseteq R' \subseteq D_1$ tel que $|R''| > |N_{V' \setminus D_2}(R'')|$. Par construction, l'ensemble $D' := (D \setminus R'') \cup N_{V' \setminus D_2}(R'')$ est un vertex cover de B vérifiant $|D'| < |D|$, contredisant l'optimalité de D (voir Figure 7.3). Il suit que R' peut être couplé dans $V' \setminus D_2$. Finalement, comme tout sommet de $V' \setminus D_2$ est un élément isolé dans $\mathscr{P}$, l'existence du couplage entre R' et $V' \setminus D_2$ implique que les arcs retour de A_B peuvent être certifiés en n'utilisant que des arcs de A_B, et ainsi $\mathscr{P}$ est une partition sûre. ◇

L'Affirmation 7.15 implique donc que, si $|V| > 4k$, il existe une partition sûre qui peut être calculée en temps polynomial, et le tournoi peut être réduit par la Règle 7.2. Nous appliquons alors la Règle 7.1 et répétons les étapes précédentes jusqu'à ce que nous ne trouvions plus de partition sûre ou $k < 0$. Dans le premier cas nous savons que $|V| \leqslant 4k$; dans le second cas, nous retournons une instance triviale négative et de taille constante. □

7.2 Noyau linéaire pour DENSE ROOTED TRIPLET INCONSISTENCY

Préliminaires. Un arbre binaire enraciné T est *défini sur* un ensemble V si les éléments de V sont en bijection avec les feuilles (*i.e.* sommets de degré 1) de T. Dans la suite, les éléments de V seront appelés *feuilles*, et le terme *noeuds* désignera les sommets internes de T. Un *triplet enraciné* (ou *triplet*) t est un arbre binaire enraciné défini sur un ensemble de trois feuilles $V(t) := \{a, b, c\}$. Nous écrivons $t := ab|c$ si a et b sont descendants de la racine de t, l'autre fils de la racine étant c.

triplet (enraciné)

Nous disons également que t *choisit* c. Nous considérons des paires $R := (V, \mathscr{R})$, ou $\mathscr{R}$ dénote une collection *dense* de triplets définis sur V. Rappelons que *dense* signifie que $\mathscr{R}$ contient *exactement* un triplet pour tout triple de V. Pour tout $S \subseteq V$, nous définissons $\mathscr{R}[S] := \{t \in \mathscr{R} \ : \ V(t) \subseteq S\}$ et $R[S] = (S, \mathscr{R}[S])$. Par $T_{|S}$, avec $S \subseteq V$, nous dénotons l'arbre binaire enraciné défini sur S homéomorphique au sous-arbre de T couvrant les feuilles de S. De plus, étant donné un noeud x de T, T_x dénote le sous-arbre de T enraciné en x. Soient $t \in \mathscr{R}$ un triplet et T un arbre défini sur V. Le triplet t est *consistent avec* T si $T_{|V(t)} = t$, et *inconsistent* sinon (voir Figure 7.4). La collection de triplets $\mathscr{R}$ est *consistente* s'il existe un arbre binaire enraciné T défini sur V tel que tout triplet $t \in \mathscr{R}$ est consistent avec T. Dans le cas contraire, $\mathscr{R}$ est *inconsistente*. Nous disons de manière similaire que *l'instance* R est consistente ou inconsistente. Un *conflit* $C \subseteq V$ est un ensemble de feuilles tel que $R[C]$ est inconsistente. Dans ce qui suit, nous utiliserons le terme conflit pour déterminer à la fois C et $\mathscr{R}[C]$. Finalement, remarquons que si une collection dense $\mathscr{R}$ est consistente, il existe un *unique* arbre binaire enraciné T (à isomorphisme près) dans lequel tout triplet de $\mathscr{R}$ est consistent [82]. Le problème considéré se formule comme suit : consistence

DENSE ROOTED TRIPLET INCONSISTENCY :
Entrée : Un ensemble de feuilles V, une collection dense $\mathscr{R}$ de triplets définis sur V, un entier k.
Paramètre : k.
Question : Existe-t-il un arbre enraciné T défini sur V tel que tous sauf au plus k triplets de $\mathscr{R}$ sont consistents avec T ?

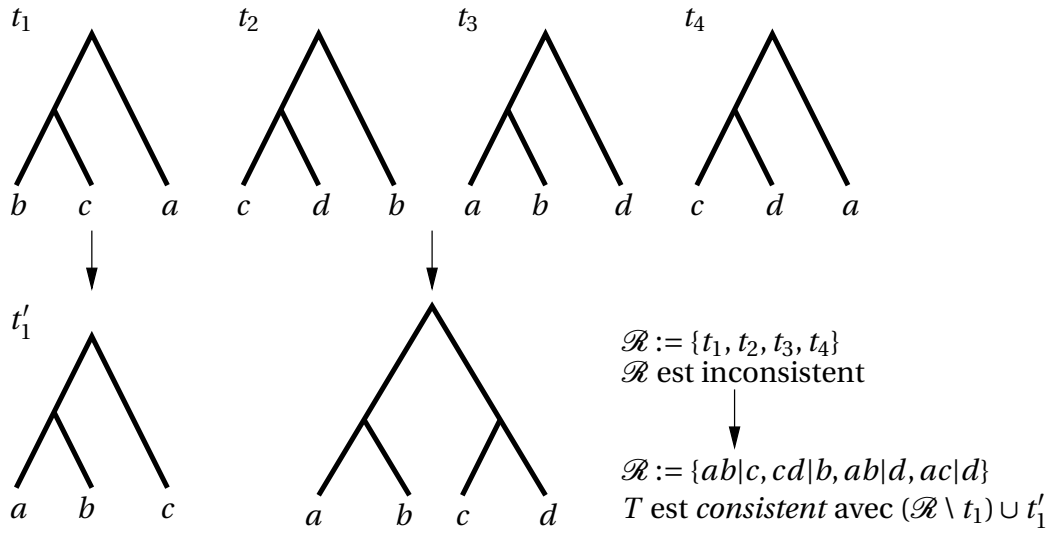


FIGURE 7.4 : Un exemple du problème DENSE ROOTED TRIPLET INCONSISTENCY : la collection $\mathscr{R}$ n'est pas consistente, mais le devient lorsque nous remplaçons le triplet t_1 par t_1'.

Le problème DENSE ROOTED TRIPLET INCONSISTENCY peut se formuler comme un problème d'édition de la manière suivante : étant donnés un ensemble de feuilles V, une collection dense $\mathscr{R}$ de triplets définis sur V et un entier k, peut-on *éditer* au plus k triplets de $\mathscr{R}$ afin d'obtenir une collection consistente ? Dans le cas du problème DENSE ROOTED TRIPLET INCONSISTENCY, *éditer un*

triplet signifie modifier le sommet choisi par ce triplet. Comme dans le cas de FEEDBACK ARC SET IN TOURNAMENTS, DENSE ROOTED TRIPLET INCONSISTENCY peut-être caractérisé par des conflits finis.

Lemme 7.16 ([82])**.** *Soit $R := (V, \mathscr{R})$ une instance de* DENSE ROOTED TRIPLET INCONSISTENCY. *Les propriétés suivantes sont équivalentes :* (i) *R est consistente ;* (ii) *R ne contient aucun conflit sur* 4 *feuilles ;* (iii) *R ne contient aucun conflit de la forme $\{ab|c, cd|b, bd|a\}$ ou $\{ab|c, cd|b, ad|b\}$.*

Dans la suite, nous considérons donc uniquement des conflits définis sur quatre feuilles.

Intuition. L'algorithme de noyau pour le problème DENSE ROOTED TRIPLET INCONSISTENCY suit des idées similaires au noyau pour FEEDBACK ARC SET IN TOURNAMENTS présenté dans la Section 7.1. Il fait appel à deux principales règles de réduction, la première permettant de supprimer les sommets n'appartenant à aucun conflit, la seconde considérant des partitions sûres de l'instance donnée. La première règle a été démontrée valide par Guillemot et Mnich [82]. La seconde règle est similaire à la Règle 7.2. Cependant, comme une instance de DENSE ROOTED TRIPLET INCONSISTENCY est constituée de triplets qui *choisissent* un sommet (remarquons qu'une instance de FEEDBACK ARC SET IN TOURNAMENTS peut être vue comme des *couples* choisissant un sommet), nous devons adapter les notions de *conflit, certificat* et *partition sûre*.

Remarque. En utilisant un raisonnement similaire à celui de la Section 7.1, un algorithme d'approximation à facteur constant couplé avec la règle de réduction Partition Sûre permettrait d'obtenir un noyau linéaire pour ce problème. Cependant, le problème DENSE ROOTED TRIPLET INCONSISTENCY n'admet à ce jour pas d'algorithme d'approximation à facteur constant [82].

Nous utilisons ainsi le Conflict Packing pour *remplacer* l'algorithme d'approximation à facteur constant, et calculer une partition sûre en temps polynomial sous certaines conditions de cardinalité.

7.2.1 Adaptation de la notion de Partition Sûre

Certificat. Nous adaptons maintenant la notion de Partition Sûre présentée dans la Section 7.1 pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Pour cela, nous avons besoin des notations et définitions suivantes. Une *instance enracinée* de DENSE ROOTED TRIPLET INCONSISTENCY est un triple $R_T := (V, \mathscr{R}, T)$, tel que $\mathscr{R}$ est une collection dense de triplets définis sur V, et T est un arbre binaire enraciné défini sur V. Lorsque nous considérons des instances enracinées R_T, l'inconsistence d'un triplet $t \in \mathscr{R}$ est toujours considérée *par rapport à* T. Étant donné un triplet $t \in \mathscr{R}$ tel que $V(t) := \{a, b, c\}$, nous définissons $span(t)$ comme l'ensemble de feuilles de V contenues dans $T_{lca(\{a,b,c\})}$, où $lca(S)$, $S \subseteq V$, dénote le *plus petit ancêtre commun* des sommets de S. Étant donné $S \subseteq V$, nous définissons $R_T[S] := (S, \mathscr{R}[S], T_{|S})$ comme l'instance enracinée induite par S. *Éditer* un triplet inconsistent $t \in \mathscr{R}$ *par rapport à* T signifie *remplacer* t dans $\mathscr{R}$ par le triplet défini sur $V(t)$ consistent avec T. Nous admettons la validité de la règle de réduction suivante, qui nous permettra de concevoir notre algorithme de noyau. Observons que cette règle est l'analogue de la Règle 7.1 définie pour le problème FEEDBACK ARC SET IN TOURNAMENTS.

instance enracinée

Règle 7.3. *Soit (R, k) une instance de* DENSE ROOTED TRIPLET INCONSISTENCY. *Supprimer toute feuille $v \in V$ n'appartenant à aucun conflit.*

Lemme 7.17 ([82])**.** *La Règle 7.3 est valide et peut être appliquée en temps polynomial.*

Comme mentionné précédemment, notre algorithme de noyau utilise des conflits définis sur quatre feuilles de V. Nous définissons plus particulièrement la topologie des conflits que nous allons considérer.

Lemme 7.18 (Conflit simple)**.** *Soient (R_T, k) une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY*, et $\{a,b,c,d\} \subseteq V$ un ensemble de feuilles tel que $t := bc|a$ est le seul triplet inconsistent de $R_T[\{a,b,c,d\}]$. L'ensemble $\{a,b,c,d\}$ est un conflit si et seulement si $d \in span(t)$.* conflit simple

Preuve. Puisque $t := bc|a$ est inconsistent avec T par hypothèse, il suit que $T_{|\{a,b,c\}}$ est homéomorphique à $ab|c$ ou $ac|b$. Les deux cas sont symétriques, et nous supposons ainsi sans perte de généralité que le premier est vérifié. Supposons que $d \in span(t)$. Par hypothèse, le triplet $t' \in \mathscr{R}$ défini sur $\{b,c,d\}$ est consistent avec T. Nous considérons les deux cas possibles pour t' (remarquons que $t' := bc|d$ n'est pas possible puisque, dans ce cas, $d \notin span(t)$) :

(i) $t' := bd|c$: puisque $ab|c$ est consistent avec T par hypothèse, et comme t est le seul triplet inconsistent avec T, il suit que $ad|c \in \mathscr{R}$. Ainsi, l'ensemble $\{bc|a, bd|c, ad|c\}$ définit un conflit de R_T.

(ii) $t' := cd|b$: comme précédemment, nous savons que $ab|d \in \mathscr{R}$. Ainsi, l'ensemble $\{bc|a, cd|b, ab|d\}$ définit un conflit de R_T.

Ainsi, si $d \in span(t)$, alors l'ensemble $\{a,b,c,d\}$ est un conflit quel que soit le choix du triplet défini sur $\{b,c,d\}$. Supposons maintenant que $d \notin span(t)$. À nouveau, comme $bc|a$ est l'unique triplet de $R_T[\{a,b,c,d\}]$ inconsistent avec T, il suit que tout triplet contenant d choisit d. Cela implique donc que $\{a,b,c,d\}$ n'est pas un conflit. □

Remarque. Observons que le Lemme 7.18 est analogue aux conflits définis sur une instance ordonnée de FEEDBACK ARC SET IN TOURNAMENTS : en effet, étant donnés un tournoi ordonné $T_\sigma := (V, A, \sigma)$ et $\{u,v,w\}$ un ensemble de sommets tel que vu est le seul arc retour de $T_\sigma[\{u,v,w\}]$ l'ensemble $\{u,v,w\}$ définit un triangle orienté si et seulement si $w \in span(f)$.

Nous définissons maintenant de manière formelle la notion de *certificat* pour une instance de DENSE ROOTED TRIPLET INCONSISTENCY.

Définition 7.19 (Certificat)**.** *Soient (R_T, k) une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY*, $t \in \mathscr{R}$ un triplet inconsistent avec T et $d \in span(t)$ un sommet n'appartenant à aucun triplet inconsistent. L'ensemble $C(t) := V(t) \cup \{d\}$ est appelé* certificat *de t.* certificat

Remarquons que, comme pour le problème FEEDBACK ARC SET IN TOURNAMENTS, l'ensemble $C(t)$ induit un conflit de R_T (Lemme 7.18). Par convention, lorsque nous parlons d'un triplet t d'un certificat C, nous sous-entendons que t appartient à $\mathscr{R}_T[C]$.

Définition 7.20 (Certifier)**.** *Soient (R_T, k) une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY*, et $\mathscr{B} \subseteq \mathscr{R}$ un ensemble de triplets inconsistents avec T. Nous disons que $\mathscr{B}$ peut être* certifié *lorsqu'il existe une collection $C(\mathscr{B}) := \{C(t) : t \in \mathscr{B}\}$ de* certificats triplets-disjoints *pour les triplets de $\mathscr{B}$.* certifier

Partition sûre. Nous décrivons maintenant la notion de *Partition Sûre* pour le problème DENSE ROOTED TRIPLET INCONSISTENCY. Nous allons voir que les notions définies pour le problème FEEDBACK ARC SET IN TOURNAMENTS dans la Section 7.1 peuvent se retrouver pour DENSE ROOTED TRIPLET INCONSISTENCY. Cependant, puisque la structure consiste en un arbre binaire enraciné, il est nécessaire d'adapter ces dernières. Étant donnée une instance enracinée (R_T, k) de DENSE ROOTED TRIPLET INCONSISTENCY, nous disons que $\mathscr{P} := \{T_1, \ldots, T_l\}$ est une *partition arborée* de R_T s'il existe l noeuds et feuilles $\{x_1, \ldots, x_l\}$ de T tels que :

partition arborée

(i) pour tout $i \in [l]$, $T_i = T_{x_i}$ et,

(ii) l'ensemble de feuilles contenues dans $\cup_{i=1}^{l} T_{x_i}$ partitionne V.

Observons en particulier qu'une partition ordonnée définit deux ensembles de triplets : $\mathscr{R}_I := \{t \in \mathscr{R} : \exists\, i \in [l]\ V(t) \subseteq V(T_i)\}$ et $\mathscr{R}_B := \mathscr{R} \setminus \mathscr{R}_I$. Nous dénotons par $\mathscr{B}(\mathscr{R}_B)$ l'ensemble des triplets de $\mathscr{R}_B$ inconsistents avec T.

partition sûre

Définition 7.21 (Partition sûre)**.** *Soient (R_T, k) une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY *et $\mathscr{P}$ une partition arborée de R_T telle que $\mathscr{B}(\mathscr{R}_B) \neq \emptyset$. Nous disons que $\mathscr{P}$ est une* partition sûre *s'il est possible de certifier les triplets de $\mathscr{B}(\mathscr{R}_B)$ en n'utilisant que des triplets de $\mathscr{R}_B$.*

Remarque. Une partition arborée est *sûre* lorsqu'il existe $|\mathscr{B}(\mathscr{R}_B)|$ conflits triplet-disjoints n'utilisant que des triplets de $\mathscr{R}_B$.

Nous prouvons maintenant que l'existence d'une partition sûre permet d'obtenir une règle de réduction. Nous verrons dans la section suivante comment une telle partition peut être obtenue en temps polynomial en utilisant la notion de Conflict Packing.

Règle 7.4. *Soient (R_T, k) une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY *et $\mathscr{P}$ une partition sûre de R_T. Éditer tout triplet $t \in \mathscr{B}(\mathscr{R}_B)$ par rapport à T et décrémenter k de $|\mathscr{B}(\mathscr{R}_B)|$.*

Lemme 7.22. *La Règle 7.4 est valide.*

Preuve. Nous utilisons l'observation suivante, qui suit par définition d'une partition arborée.

Observation 7.23. *Soit $\mathscr{P} := \{T_1, \ldots, T_l\}$ une partition arborée d'une instance enracinée de* DENSE ROOTED TRIPLET INCONSISTENCY. *Soit t un triplet tel que $V(t) \subseteq V(T_i)$ pour un certain $i \in [l]$, et $u \in V \setminus V(T_i)$. Alors $u \notin span(t)$.*

Comme $\mathscr{P}$ est une partition sûre, il existe un ensemble $C_{\mathscr{B}}$ de certificats triplets-disjoints pour $\mathscr{B}(\mathscr{R}_B)$. Par construction de $C_{\mathscr{B}}$, une édition (au moins) doit être effectuée pour tout certificat. Nous démontrons le Lemme 7.22 en prouvant que R_T admet une k-édition si et seulement si l'instance R'_T obtenue à partir de R_T en éditant tous les triplets de $\mathscr{B}(\mathscr{R}_B)$ par rapport à T admet une $(k - |\mathscr{B}(\mathscr{R}_B)|)$ édition. Pour ce faire, nous démontrons l'affirmation suivante.

Affirmation 7.24. *Il existe une k-édition de R_T qui contient les triplets de $\mathscr{B}(\mathscr{R}_B)$.*

Preuve. Nous démontrons le résultat en prouvant que R'_T admet une $(k - |\mathscr{B}(\mathscr{R}_B)|)$-édition. Soient $C := \{a, b, c, d\}$ un conflit de R'_T, et $t \in \mathscr{R}[C]$ un triplet de $R[C]$ inconsistent avec T. Par définition, comme $t \notin \mathscr{B}(\mathscr{R}_B)$, nous avons $t \in \mathscr{R}_I$ et donc $V(t) \subseteq V(T_i)$ pour un certain $i \in [l]$. De plus, par le Lemme 7.18, nous savons que $d \in span(t)$. Il suit que $d \in V(T_i)$, et ainsi tout conflit de R'_T est contenu dans $V(T_i)$ pour un certain $i \in [l]$.

Pour conclure, observons qu'aucune édition (dans R'_T) d'un triplet de $\mathscr{R}_I$ ne peut créer de conflit impliquant un triplet de $\mathscr{R}_B$. En effet, soit $t \in \mathscr{R}$ un triplet tel que $V(t) \subseteq V(T_i)$ pour un certain $i \in [l]$. Soit $d \notin V(T_i)$. Par l'Observation 7.23, $d \notin span(t)$, et les triplets de $R'_T[\{a,b,c,d\}]$ contenant d sont consistents avec T (rappelons que tout triplet de $\mathscr{R}_B$ est consistent avec T dans R'_T). Par le Lemme 7.18, $\{a,b,c,d\}$ n'est donc pas un conflit, et ce quel que soit le choix de t. Soit F' une $(k-|\mathscr{B}(\mathscr{R}_B)|)$-édition de R'_T. Par les arguments précédents, il suit que $F' \cup \mathscr{B}(\mathscr{R}_B)$ est une k-édition de R_T contenant $\mathscr{B}(\mathscr{R}_B)$. ◇

L'Affirmation 7.24 implique donc que R_T a une k-édition si et seulement si R'_T a une $(k-|\mathscr{B}(\mathscr{R}_B)|)$-édition, ce qui conclut la preuve du Lemme 7.22. □

7.2.2 Définition et conséquences du Conflict Packing

Comme évoqué dans le Chapitre 6, l'objectif du Conflict Packing est de fournir une borne inférieure sur le nombre d'éditions à réaliser afin d'obtenir une instance consistente. Cela peut être réalisé en considérant une famille maximale de conflits relations-disjoints, en particulier lorsque deux conflits peuvent partager une relation mais requérir deux éditions distinctes afin d'être résolus. Observons cependant que ce n'est pas le cas pour le problème DENSE ROOTED TRIPLET INCONSISTENCY : en effet, deux conflits peuvent partager un triplet mais requérir deux éditions distinctes. Pour voir cela, considérons un ensemble de feuilles $\{a,b,c,d,e\}$ ainsi que les conflits suivants : $\mathscr{C}_1 := \{ab|c, ac|d, ad|b, cd|b\}$ et $\mathscr{C}_2 := \{ed|c, ed|b, bc|e, bd|c\}$. Observons tout d'abord que $\mathscr{C}_1$ est un conflit quel que soit le choix de $\{b,c,d\}$. Puisque $\{b,c,d\}$ est le seul triplet que partagent $\mathscr{C}_1$ et $\mathscr{C}_2$, aucune édition réalisée sur $\mathscr{C}_2$ ne peut résoudre $\mathscr{C}_1$. De ce fait, (au moins) deux éditions distinctes sont nécessaires afin de résoudre $\mathscr{C}_1$ et $\mathscr{C}_2$. Afin de prendre en considération cette remarque, nous définissons la notion de *graine*, qui permet de formaliser l'exemple précédent.

Définition 7.25 (Graine)**.** *Soient* (R,k) *une instance de* DENSE ROOTED TRIPLET INCONSISTENCY*, et* $C := \{a,b,c,d\} \subseteq V$ *un conflit de* R*. La feuille* a *est une* graine *de* C *si* C *est un conflit quel que soit le sommet choisi dans le triplet défini sur* $\{b,c,d\}$*.* graine

Cette définition nous permet de raffiner la notion de Conflict Packing, basée sur l'observation suivante : si deux conflits C et C' partagent un triplet t mais que $C \setminus \{t\}$ est toujours un conflit, alors deux éditions distinctes sont nécessaires pour résoudre C et C'.

Définition 7.26 (Conflict Packing)**.** *Soit* (R,k) *une instance de* DENSE ROOTED TRIPLET INCONSISTENCY*. Un* conflict packing *de* R *est une collection maximale de conflits* $\mathscr{C} := \{C_1,\dots,C_m\}$ *telle que, pour tout* $2 \leqslant i \leqslant m$ *:* conflict packing

- C_i *intersecte* $\cup_{j=1}^{i-1} C_j$ *en au plus deux feuilles ou,*
- C_i *a une unique feuille n'appartenant pas à* $\cup_{j=1}^{i-1} C_j$*, et cette feuille est une graine de* C_i*.*

Lemme 7.27. *Soient* (R,k) *une instance positive de* DENSE ROOTED TRIPLET INCONSISTENCY *et* $\mathscr{C} := \{C_1,\dots,C_m\}$ *un conflict packing de* R*. Alors* $m \leqslant k$ *et* $|V(\mathscr{C})| \leqslant 4k$*.*

Preuve. Nous prouvons par induction sur le nombre de conflits m contenus dans $\mathscr{C}$ qu'au moins m éditions sont nécessaires pour résoudre tous les conflits de $\mathscr{C}$. Si $m = 1$ alors le résultat est trivialement vérifié. Sinon, nous savons par hypothèse d'induction qu'au moins $m-1$ éditions sont

nécessaires pour résoudre les conflits $\{C_1, \ldots, C_{m-1}\}$. Par la Définition 7.26, deux configurations sont possibles : si C_m intersecte $\cup_{j=1}^{m-1} C_j$ en au plus deux feuilles, une édition supplémentaire est alors nécessaire pour résoudre C_m. Dans le cas contraire, nous savons par définition que l'unique feuille de C_m non contenue dans $\cup_{j=1}^{m-1} C_j$ est une graine de C_m. De ce fait, aucune édition réalisée pour résoudre les conflits de $\{C_1, \ldots, C_{m-1}\}$ ne peut résoudre le conflit C_m. Il suit donc que m éditions sont nécessaires pour résoudre les conflits de $\mathscr{C}$, et ainsi que $m \leqslant k$; puisque chaque conflit de $\mathscr{C}$ contient au plus 4 feuilles, nous avons également $|V(\mathscr{C})| \leqslant 4k$. □

Étant donné un conflict packing $\mathscr{C}$ d'une instance (R, k) de DENSE ROOTED TRIPLET INCONSISTENCY, nous prouvons maintenant qu'il existe un arbre binaire enraciné T défini sur V tel que tout triplet $t \in \mathscr{R}$ inconsistent avec T vérifie $V(t) \subseteq V(\mathscr{C})$. Cet arbre sera le point de départ l'algorithme permettant de calculer une partition sûre en temps polynomial.

Lemme 7.28 (Conflict Packing)**.** *Soient (R, k) une instance de* DENSE ROOTED TRIPLET INCONSISTENCY *et $\mathscr{C}$ un conflict packing de R. Il existe un arbre binaire enraciné T défini sur V tel que tout triplet t inconsistent avec T vérifie $V(t) \subseteq V(\mathscr{C})$. De plus, T peut être calculé en temps polynomial.*

Preuve. Les arguments de la preuve sont similaires à ceux de la preuve du Lemme 7.13. Par maximalité du conflict packing $\mathscr{C}$, observons que l'instance induite par $V' := V \setminus V(\mathscr{C})$ est consistente avec un unique arbre binaire enraciné T', qui peut être calculé en temps polynomial. En utilisant à nouveau la maximalité du conflict packing, nous savons également que l'instance $R_a := R[V' \cup \{a\}]$ est consistente pour toute feuille $a \in V(\mathscr{C})$. Ainsi, il existe un unique arbre binaire enraciné T_a tel que tout triplet t de R_a est consistent avec T_a. En d'autres termes, cela signifie que l'arbre T' contient une *unique* arête $e := xy$ (avec x un ancêtre de y) qui peut être divisée en deux arêtes xz et zy pour attacher la feuille a au noeud z et obtenir l'arbre T_a consistent avec R_a. Dans la suite de cette preuve, l'arête e est appelée le *locus* de a. La maximalité de $\mathscr{C}$ implique également que pour toutes feuilles a et b de $V(\mathscr{C})$, l'instance $R_{ab} := R[V' \cup \{a, b\}]$ est consistente. Si a et b ont des loci différents, alors l'arbre T_{ab} obtenu à partir de T' en insérant a et b dans leurs loci respectifs est consistent avec R_{ab}. Le cas à considérer est donc lorsque a et b ont le même locus. Nous dénotons par $B_e \subseteq V(\mathscr{C})$ l'ensemble des feuilles ayant locus $e := xy$. Nous prouvons maintenant comment insérer de telles feuilles dans T' pour obtenir un arbre respectant la propriété désirée. Étant données deux feuilles $a, b \in B_e$, nous définissons les relations binaires $<_e$ et $\sim_e$ comme suit :

$<_e$ et $\sim_e$

$$a <_e b \text{ s'il existe } c \in V' \text{ tel que } ac|b \in \mathscr{R}$$

$$a \sim_e b \text{ si } a \not<_e b \text{ et } b \not<_e a$$

Remarque. Étant données deux feuilles a, b de B_e, $ab|c \in \mathscr{R}$ est vérifié pour tout $c \notin T'_y$.

Nous démontrons que la relation $<_e$ est un ordre strict faible, *i.e.* la relation $<_e$ est transitive et asymétrique, et $\sim_e$ est transitive.

Affirmation 7.29. *La relation $<_e$ est un ordre strict faible.*

Preuve. Premièrement, observons que si $a <_e b$ alors le sommet c appartient à T'_y : en effet, dans le cas contraire, nous aurions $ab|c \in \mathscr{R}$ par la remarque précédente. Supposons que $<_e$ ne soit pas asymétrique. Cela signifie qu'il existe deux feuilles $c \in T'_y$ et $d \in T'_y$, $c \neq d$, telles que $\mathscr{R}[\{a, b, c, d\}] :=$

$\{ac|b, bd|a, cd|a, cd|b\}$. Cela implique que $\{a,b,c,d\}$ est un conflit, contredisant le fait que R_{ab} est consistente pour tous $a,b \in B_e$. Il suit que $<_e$ est asymétrique.

Nous démontrons maintenant la transitivité de $<_e$. Soient $a,b,c \in B_e$ tels que (sans perte de généralité) $a <_e b$ et $b <_e c$. Par définition, cela signifie qu'il existe $d \in V'$ tel que $ad|b \in \mathscr{R}$. Par les arguments précédents, nous savons que $d \in T'_y$ et il suit donc que $bd|c \in \mathscr{R}$. Supposons maintenant que $dc|a \in \mathscr{R}$ (le cas $ac|d \in \mathscr{R}$ est similaire). Dans ce cas, quel que soit le choix pour le triplet défini sur $\{a,b,c\}$, l'ensemble $\{a,b,c,d\}$ est un conflit, impliquant que d est une graine du conflit $\{a,b,c,d\}$. En particulier, cela signifie que le conflict packing $\mathscr{C}$ n'est pas maximal : contradiction. Ainsi $da|c \in \mathscr{R}$ et donc $a <_e c$ et $<_e$ est transitive.

Pour conclure, nous prouvons maintenant que $\sim_e$ est transitive. Soient $a,b,c \in B_e$ tels que (sans perte de généralité) $a \sim_e b$ et $b \sim_e c$. Par définition, pour tout $d \in V'$ les triplets $ab|d$ et $bc|d$ appartiennent à $\mathscr{R}$. Supposons maintenant que $ad|c \in \mathscr{R}$ (le cas $dc|a \in \mathscr{R}$ est similaire). Dans ce cas, quel que soit le choix pour le triplet $\{a,b,c\}$, l'ensemble $\{a,b,c,d\}$ est un conflit, impliquant que d est une graine du conflit $\{a,b,c,d\}$. Ici encore, cela signifie que $\mathscr{C}$ n'est pas maximal : contradiction. Nous concluons donc que $ac|d$ appartient à $\mathscr{R}$, et donc $a \sim_e c$ et $\sim_e$ est transitive. ◇

Nous pouvons désormais décrire comment insérer les feuilles partageant le même locus dans T' pour obtenir un arbre T respectant les propriétés désirées. Pour toute arête $e := xy$ de T' telle que $B_e \neq \emptyset$, nous procédons comme suit. Soient $\{B_1, \dots, B_q\}$ les classes d'équivalence de la relation $\sim_e$ telles que $B_i <_e B_j$ pour $1 \leqslant i < j \leqslant q$ (rappelons que les relations $<_e$ et $\sim_e$ sont transitives par l'Affirmation 7.29). Nous remplaçons l'arête e par un chemin $\{x, z_q, \dots, z_1, y\}$. Pour tout $i \in [q]$, si B_i contient une unique feuille a nous attachons a au noeud z_i. Dans le cas contraire, nous attachons un noeud w_i à z_i et ajoutons un arbre binaire enraciné en w_i et défini sur les feuilles de B_i. Soit T l'arbre ainsi obtenu.

Affirmation 7.30. *L'arbre T respecte les propriétés du Lemme 7.28.*

Preuve. Soit $t \in \mathscr{R}$ un triplet tel que $V(t) := \{a,b,c\}$. Nous supposons dans un premier temps que $V(t) \subseteq V'$. Dans ce cas, t est consistent avec T puisque, par construction, t est consistent avec T'. Ensuite, supposons sans perte de généralité que $V(t) \cap V(\mathscr{C}) = \{a\}$. Alors t est consistent avec T puisque R_a est consistent et a a été inséré dans son locus dans T'. Finalement, supposons $V(t) \cap V(\mathscr{C}) = \{a,b\}$. Si a et b ont des loci différents, alors t est consistent avec T' par les arguments précédents. Dans le cas contraire, t est consistent avec T puisque a et b ont été insérés en fonction de l'ordre strict faible $<_e$. Nous avons donc montré que tout triplet t tel que $V(t) \cap V' \neq \emptyset$ était consistent avec T, impliquant le résultat. ◇

Cela conclut la preuve du Lemme 7.28. □

7.2.3 Algorithme de noyau

Nous présentons maintenant l'algorithme de noyau pour le problème DENSE ROOTED TRIPLET INCONSISTENCY. La preuve permettant de borner la taille d'une instance positive est similaire à celle du Théorème 7.14, et utilise notamment des arguments de couplage sur le *graphe des conflits*.

Théorème 7.31. *Le problème* DENSE ROOTED TRIPLET INCONSISTENCY *admet un noyau avec au plus* $5k$ *feuilles.*

Preuve. Soit (R,k) une instance positive de DENSE ROOTED TRIPLET INCONSISTENCY réduite par la Règle 7.3. Nous construisons de manière gloutonne (*i.e.* en temps polynomial) un conflict packing $\mathscr{C}$ de R et dénotons par T un arbre obtenu via le Lemme 7.28. Nous considérons le graphe biparti $B := (I \cup V', E)$ construit comme suit :

- $V' := V \setminus V(\mathscr{C})$,
- il existe un sommet i_t dans I pour triplet t inconsistent avec T et,
- $i_t a \in E$ si $a \in V'$ et $V(t) \cup \{a\}$ définit un certificat pour t.

Dans un premier temps, observons que B ne contient pas de couplage de taille $k+1$: en effet, un tel couplage dans B correspondrait à un conflict packing de R de taille $k+1$, ce qui est impossible par hypothèse (Lemme 7.27). Il suit alors que B admet un vertex cover D de taille au plus k. Nous posons $D_1 := D \cap I$ et $D_2 := V' \cap D$. Supposons maintenant que $|V| > 5k$. Par le Lemme 7.27, nous savons que $|V(\mathscr{C})| \leqslant 4k$. Comme $|D_2| \leqslant k$ par hypothèse, nous avons alors $V' \setminus D_2 \neq \emptyset$. Soit $\mathscr{P} := \{T_1, \ldots, T_l\}$ la partition arborée de R_T ou T_i, $i \in [l]$, est soit une feuille de $V' \setminus D_2$, soit une composante connexe de $T \setminus S$, où S dénote le plus petit sous-arbre couvrant de $(V' \setminus D_2) \cup \{r\}$ (r étant la racine de T, voir Figure 7.5).

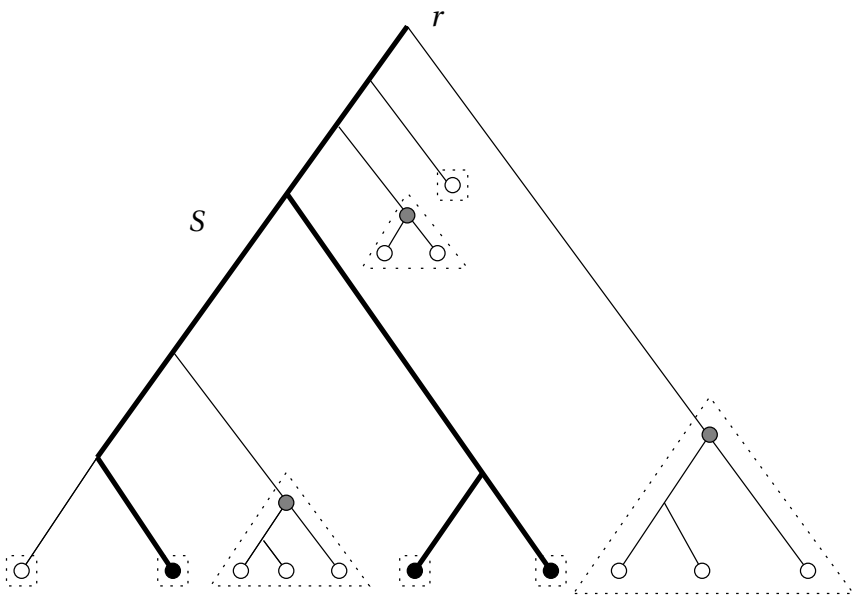


FIGURE 7.5 : Illustration de la construction de la partition arborée $\mathscr{P}$. Les sommets noirs correspondent aux sommets de $V' \setminus D_2$.

Affirmation 7.32. *La partition $\mathscr{P}$ est sûre.*

Preuve. Soit a une feuille de $V' \setminus D_2$. Par le Lemme 7.28, a n'est contenue dans aucun triplet inconsistent de T. De plus, comme R est réduite par la Règle 7.3, il existe un triplet t inconsistent avec T tel que $a \in span(t)$. En effet, dans le cas contraire, a n'appartiendrait à aucun conflit (Lemme 7.18) et R ne serait pas réduite par la Règle 7.3. Il suit que $\mathscr{R}_B$ contient au moins un triplet inconsistent avec T.

Soit $t \in \mathscr{R}_B$ un triplet inconsistent avec T. Par construction de $\mathscr{P}$, il existe une feuille $a \in (V' \setminus D_2) \cap span(t)$. Par définition, l'ensemble $V(t) \cup \{a\}$ est un certificat pour t et $i_t a$ est une arête de B. Puisque D est un vertex cover de B et $a \notin D_2$, il suit que le sommet i_t appartient D_1. Ainsi l'ensemble $I' \subseteq I$ correspondant aux triplets de $\mathscr{R}_B$ inconsistents avec T est inclus dans D_1.

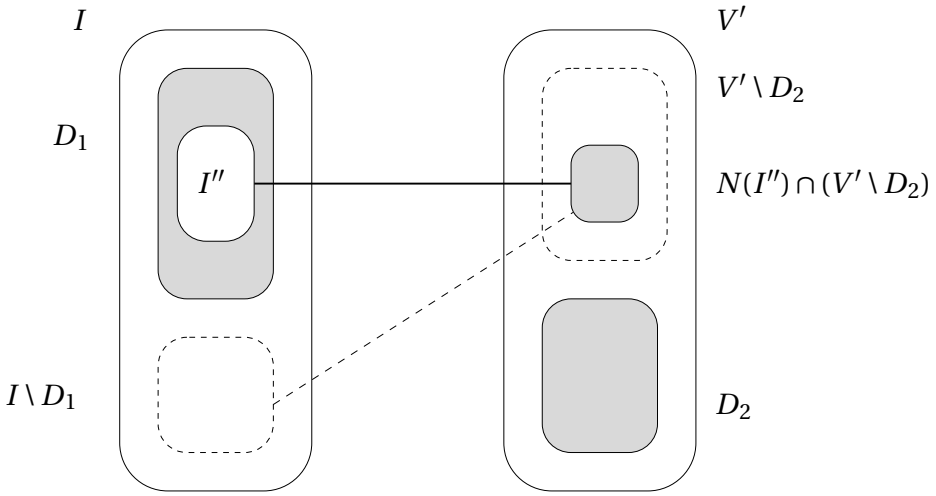


FIGURE 7.6 : Illustration de la preuve de l'Affirmation 7.32.

Pour conclure, nous prouvons que I' peut être matché dans $V' \setminus D_2$. Pour cela, nous utilisons à nouveau le théorème de Hall [93] : I' peut être matché dans $V' \setminus D_2$ si et seulement si $|I''| \leqslant |N_{V' \setminus D_2}(I'')|$ pour tout $I'' \subseteq I'$. Par contradiction, nous supposons donc qu'il existe $I'' \subseteq I'$ tel que $|I''| > |N_{V' \setminus D_2}(I'')|$. Par construction, l'ensemble $D' := (D \setminus I'') \cup N_{V' \setminus D_2}(I'')$ est un vertex cover de B vérifiant $|D'| < |D|$. Cela contredit la minimalité de D (voir Figure 7.6). Ainsi I' peut être matché dans $V' \setminus D_2$. Comme toute feuille de $V' \setminus D_2$ est un singleton de $\mathscr{P}$, l'existence du couplage implique que l'ensemble de triplets de $\mathscr{R}_B$ inconsistents avec T peut être certifié en n'utilisant que des triplets enracinés de $\mathscr{R}_B$. ◇

L'Affirmation 7.32 implique donc que si $|V| > 5k$, il existe une partition sûre qui peut être construite en temps polynomial, et l'instance peut être réduite par la Règle 7.4. Nous appliquons alors la Règle 7.3 et répétons les étapes précédentes jusqu'à ce que nous ne trouvions plus de partition sûre ou $k < 0$. Dans le premier cas, nous avons $|V| \leqslant 5k$; dans le second cas, nous retournons une instance négative triviale de taille constante. □

7.3 DENSE BETWEENNESS et DENSE CIRCULAR ORDERING

Nous présentons à présent des algorithmes de noyau linéaires pour les problèmes DENSE BETWEENNESS et DENSE CIRCULAR ORDERING. Ces derniers utilisent la méthode Conflict Packing, et sont les premiers algorithmes de noyau polynomiaux connus pour ces problèmes.

7.3.1 DENSE BETWEENNESS

Préliminaires. Étant donné un univers V, un *triplet betweenness* (ou *triplet*) t est une paire $\{\{a,b,c\},m\}$ où $\{a,b,c\} \subseteq V$ et où $m \in \{a,b,c\}$ est le sommet *choisi* par t. Nous écrivons $t = abc$ pour illustrer le fait que le triplet t choisit le sommet b. Nous utilisons $V(t)$ pour dénoter l'ensemble de sommets contenus dans le triplet t. Nous considérons des paires $R := (V, \mathscr{R})$, où $\mathscr{R}$ est une collection de triplets définis sur l'ensemble V. Nous considérons uniquement des instances *denses*, *i.e.* pour lesquelles $\mathscr{R}$ contient *exactement* un triplet pour tout triple de V. Soient $t := abc \in \mathscr{R}$ un triplet et σ un ordre sur les sommets V. Le triplet t est *consistent* avec σ si $a <_\sigma b <_\sigma c$ ou $c <_\sigma b <_\sigma a$ est vérifié. Une collection de triplets $\mathscr{R}$ est *consistente* s'il existe un ordre σ sur V tel que tout triplet $t \in \mathscr{R}$ est consistent avec σ. Si un tel ordre n'existe pas, nous disons alors que $\mathscr{R}$ (ou, de manière équivalente, R), est *inconsistente*. Formellement, le problème considéré se définit comme suit.

DENSE BETWEENNESS :
Entrée : Un ensemble V, une collection dense $\mathscr{R}$ de triplets définis sur V, un entier k.
Paramètre : k.
Question : Existe-t-il un ordre sur V satisfaisant tous sauf au plus k triplets de $\mathscr{R}$?

Étant donné $S \subseteq V$, nous définissons $\mathscr{R}[S]$ comme l'ensemble des triplets de $\mathscr{R}$ vérifiant $V(t) \subseteq S$. Nous définissons ainsi $R[S] := (S, \mathscr{R}[S])$. Un *conflit* C est un ensemble de sommets de V tel que $R[C]$ est inconsistent. Étant donnée une instance (R,k) de DENSE BETWEENNESS, une *édition* pour un triplet $t \in \mathscr{R}$ est une modification de son sommet choisi. Un ensemble $\mathscr{F}$ de triplets édités est une *édition de* R si l'édition de tout triplet $t \in \mathscr{F}$ résulte en une instance consistente. Nous utilisons $R_\sigma := (V, \mathscr{R}, \sigma)$ pour dénoter une instance de DENSE BETWEENNESS dont les sommets V sont ordonnés suivant un certain ordre σ. *Éditer* un triplet t *par rapport à* σ signifie supprimer t de $\mathscr{R}$ et le remplacer par le triplet consistent avec σ.

Nous donnons maintenant une caractérisation de la consistence d'une instance de DENSE BETWEENNESS en termes de conflits finis.

Lemme 7.33. *Soit* $R := (V, \mathscr{R})$ *une instance de* DENSE BETWEENNESS. *L'instance* B *est consistente si et seulement si* B *ne contient pas de conflits sur* 4 *sommets.*

Preuve. $\Rightarrow$ Soit $R := (V, \mathscr{R})$ une instance consistente de DENSE BETWEENNESS. Alors R ne contient aucun conflit par définition.
$\Leftarrow$ Soit $R := (V, \mathscr{R})$ une instance de DENSE BETWEENNESS ne contenant pas de conflits sur 4 sommets. Nous prouvons par induction sur le nombre de sommets que R est consistente. Remarquons que la propriété est trivialement vérifiée pour $n = 4$. Supposons maintenant que la propriété est vérifiée pour les instances à $n-1$ sommets. Soit $d \in V$ un sommet de R. Par hypothèse d'induction, nous savons que l'instance $R_d := R[V \setminus \{d\}]$ admet un (unique à renversement près) ordre consistent σ. Nous démontrons que d peut être *inséré* dans σ afin d'obtenir un ordre consistent de R.

Soient a, b, c les premier, deuxième et dernier sommets de σ, respectivement. Puisque R_d est consistent, nous savons que $abc \in \mathscr{R}$. De plus, puisque R ne contient pas de conflits sur 4 sommets par hypothèse, il existe une *unique* manière d'insérer d dans σ afin d'obtenir un ordre σ_d tel que $R[\{a,b,c,d\}]$ est consistente avec σ_d. Plusieurs cas sont à considérer : (i) $d <_{\sigma_d} a$ (ou $c <_{\sigma_d} d$) ou

(ii) $a <_{\sigma_d} d <_{\sigma_d} b$ ou (iii) $b <_{\sigma_d} d <_{\sigma_d} c$. Dans le cas (iii), nous répétons les étapes précédentes sur l'ordre $\sigma_{V\setminus\{a\}}$ (*i.e.* l'ordre σ restreint à $V \setminus \{a\}$).

Soient a', b', c' les derniers sommets considérés dans le processus. Nous supposons que $a' <_{\sigma_d} d <_{\sigma_d} b'$ (les autres cas sont similaires). Nous prétendons que σ_d est un ordre consistent de R. Soit t un triplet de $\mathscr{R}$ contenant d (observons que les autres triplets sont consistents avec σ_d par hypothèse d'induction). Supposons tout d'abord sans perte de généralité que $V(t) := \{a', d, e\}$ avec $e \notin \{a', b', c'\}$. Remarquons que les triplets $\{a', b', d\}$ et $\{a', b', e\}$ sont consistents avec σ_d. Cela signifie $\{a'db', \{ea'b' \vee a'b'e\}\} \in \mathscr{R}$: il suit que t est consistent avec σ_d (sinon l'ensemble $\{a', b', d, e\}$ serait un conflit sur 4 sommets). Maintenant, supposons sans perte de généralité $V(t) := \{d, e, f\}$ avec $\{e, f\} \notin \{a', b', c'\}$. Par les arguments précédents, nous savons que les triplets $\{a', d, e\}$, $\{a', d, f\}$ et (par l'hypothèse d'induction) $\{a', e, f\}$ sont consistents avec σ_d. Ainsi, t est consistent avec σ_d : autrement, $\{a', d, e, f\}$ serait un conflit sur 4 sommets. Dans tous les cas, nous avons prouvé que σ_d ne contient aucun triplet inconsistent, impliquant que R est consistent. □

Intuition. L'algorithme de noyau pour le problème DENSE BETWEENNESS utilise la notion de Conflict Packing mais ne nécessite en revanche pas une règle de réduction basée sur la notion de partition sûre. En effet, une règle de réduction de type sunflower peut être utilisée dans le cadre de ce problème. Cela provient du fait que, contrairement aux problèmes FEEDBACK ARC SET IN TOURNAMENTS et DENSE ROOTED TRIPLET INCONSISTENCY, étant donnée une instance ordonnée R_σ de DENSE BETWEENNESS, *tout* ensemble de sommets $\{a, b, c, d\} \subseteq V$ tel que $R_\sigma[\{a, b, c, d\}]$ contient un unique triplet inconsistent t est un conflit. Ainsi, il n'est plus nécessaire de requérir que le quatrième sommet appartienne au *span* du triplet t, ce qui permet de simplifier l'algorithme de noyau pour le problème.

Sunflower. Rappelons qu'une *sunflower* $\mathscr{S}$ est un ensemble de conflits $\{C_1, \ldots, C_m\}$ s'intersectant 2-à-2 en *exactement* un triplet t. Nous définissons t comme le *centre* de $\mathscr{S}$.

Lemme 7.34. *Soient (R, k) une instance de* DENSE BETWEENNESS*, et $\mathscr{S} := \{C_1, \ldots, C_m\}$, $m > k$ une sunflower de centre t. Toute k-édition $\mathscr{F}$ doit éditer le triplet t.* Lemme 2.48

Observons qu'il existe deux possibilités pour éditer le centre t. En posant $m > 2k$, il est possible de déterminer quelle édition réaliser sur t, obtenant ainsi un noyau quadratique pour le problème. Cela découle en fait directement des techniques utilisées dans [82] pour obtenir un noyau quadratique pour le problème DENSE ROOTED TRIPLET INCONSISTENCY. Cependant, ces techniques ne permettent pas d'obtenir un noyau *linéaire* pour ce problème. Pour construire un tel algorithme, nous combinons la technique Conflict Packing à des sunflowers définis sur des *conflits simples*.

Lemme 7.35. *Soient $R_\sigma := (V, \mathscr{R}, \sigma)$ une instance ordonnée de* DENSE BETWEENNESS*, et $\{a, b, c, d\} \subseteq V$ tel que $bca \in \mathscr{R}$ est le seul triplet inconsistent de $R_\sigma[\{a, b, c, d\}]$. L'ensemble $\{a, b, c, d\}$ est un conflit.*

Preuve. Puisque $bca \in \mathscr{R}$ est inconsistent avec σ, nous savons que c n'est *pas* entre a et b dans σ. Ainsi, nous avons soit b entre a et c soit a entre b et c. Supposons sans perte de généralité $a <_\sigma b <_\sigma c$ ou $b <_\sigma a <_\sigma c$. Les deux cas sont similaires, et nous supposons donc que le premier est vérifié. Par hypothèse, le triplet $t \in \mathscr{R}$ défini sur $\{b, c, d\}$ est consistent avec σ. Nous considérons les trois cas possibles pour t :

- $t := bcd$: par hypothèse, comme $a <_\sigma b <_\sigma c$, il suit que $b <_\sigma c <_\sigma d$ et donc $acd \in \mathscr{R}$ (rappelons que bca est le seul triplet inconsistent). Ainsi l'ensemble $\{bca, bcd, acd\}$ est un conflit.
- $t := cbd$: par hypothèse, nous avons dans ce cas $d <_\sigma a <_\sigma c$ ou $a <_\sigma d <_\sigma c$. Ainsi $\{dac \vee adc\} \in \mathscr{R}$. Dans les deux cas, l'ensemble $\{dac \vee adc, cbd, bca\}$ est un conflit.
- $t := bdc$: par hypothèse, nous avons $a <_\sigma d <_\sigma c$ ou $a <_\sigma c <_\sigma d$. Comme précédemment, l'ensemble $\{adc \vee acd, bdc, bca\}$ définit alors un conflit.

Nous avons donc prouvé que, pour toute instance ordonnée comportant 4 sommets $\{a,b,c,d\}$ et un *unique* triplet inconsistent, l'ensemble $\{a,b,c,d\}$ est un conflit. □

Définition 7.36 (Conflit simple)**.** *Soient $R_\sigma := (V, \mathscr{R}, \sigma)$ une instance ordonnée de* DENSE BETWEENNESS*, $t \in \mathscr{R}$ un triplet inconsistent et $d \in V$ un sommet n'appartenant à aucun triplet inconsistent. L'ensemble $V(t) \cup \{d\}$ est appelé un* conflit simple *de B.*

Nous décrivons maintenant la principale règle de réduction de cette section. Soit $R_\sigma = (V, \mathscr{R}, \sigma)$ une instance ordonnée de DENSE BETWEENNESS. Une sunflower $\mathscr{S} := \{C_1, \ldots, C_m\}$ de R_σ est *simple* si les C_i sont des conflits simples, $i \in [m]$, et si le centre de $\mathscr{S}$ est le seul triplet inconsistent de $R_\sigma[C_i]$, $1 \leqslant i \leqslant m$.

sunflower simple

Règle 7.5 (Sunflower simple)**.** *Soient $R_\sigma := (V, \mathscr{R}, \sigma)$ une instance ordonnée de* DENSE BETWEENNESS *et $\mathscr{S} := \{C_1, \ldots, C_m\}$ une sunflower simple de R_σ de centre t, $m > k$. Éditer t par rapport à σ et décrémenter k de* 1.

Lemme 7.37. *La Règle 7.5 est valide.*

Preuve. Soient $R_\sigma := (V, \mathscr{R}, \sigma)$ une instance ordonnée de DENSE BETWEENNESS et $\mathscr{S} := \{C_1, \ldots, C_m\}$ une sunflower simple de R_σ de centre t, avec $m > k$. Soit $\mathscr{F}$ une k-édition de B. Par le Lemme 7.34, nous savons que $\mathscr{F}$ contient t. Maintenant, comme $|\mathscr{F}| \leqslant k$, il existe $i \in [m]$ tel que le seul triplet édité dans C_i est t. Nous affirmons que t *doit* être édité par rapport à σ. Supposons par contradiction que ce ne soit pas le cas : comme t est le seul triplet édité de C_i, il suit que $R_\sigma[C_i]$ contient toujours un unique triplet inconsistent. Par le Lemme 7.35, il suit que C_i est un conflit, contredisant le fait que $\mathscr{F}$ est une k-édition. □

Remarque. Le problème DENSE BETWEENNESS admet un algorithme d'approximation à facteur constant, et plus particulièrement un PTAS [104]. Ainsi, en combinant les règles de réduction présentées précédemment avec un tel algorithme, il est possible d'obtenir un noyau contenant $(4+\epsilon)k$ sommets pour tout $\epsilon > 0$ [135]. Par souci de cohérence, nous présentons cependant un algorithme de noyau utilisant la notion de Conflict Packing.

Conflict Packing. Soit (R, k) une instance de DENSE BETWEENNESS. Afin de pouvoir appliquer la Règle 7.5, nous avons besoin d'ordonner R suivant un ordre σ respectant certaines propriétés, nous permettant en particulier d'avoir un contrôle sur les triplets inconsistents. Pour ce faire, nous allons utiliser la technique Conflict Packing. Plus précisément, nous allons démontrer que DENSE BETWEENNESS peut être résolu en temps polynomial pour les instances dont le paramètre vérifie $k < |V|/5$. L'existence d'un tel algorithme impliquera directement l'existence d'un noyau linéaire pour DENSE BETWEENNESS, comme précisé dans le Corollaire 7.45.

Définition 7.38 (Graine)**.** *Soient* (R,k) *une instance de* DENSE BETWEENNESS*, et* $C := \{a,b,c,d\} \subseteq V$ *un conflit de* R*. Le sommet* a *est une* graine *de* C *si* C *est un conflit quel que soit le sommet choisi dans le triplet* $\{b,c,d\}$*.*

Définition 7.39 (Conflict Packing)**.** *Soit* (R,k) *une instance de* DENSE BETWEENNESS*. Un* conflict packing *de* R *est une collection maximale de conflits* $\mathscr{C} := \{C_1,\dots,C_m\}$ *telle que, pour tout* $2 \leqslant i \leqslant m$ *:*

- C_i *intersecte* $\cup_{j=i}^{i-1} C_j$ *en au plus deux feuilles ou,*
- C_i *a une unique feuille n'appartenant pas à* $\cup_{j=i}^{i-1} C_j$*, et cette feuille est une graine de* C_i*.*

Lemme 7.40. *Soient* (R,k) *une instance positive de* DENSE BETWEENNESS *et* $\mathscr{C}$ *un conflict packing de* B*. Alors* $|V(\mathscr{C})| \leqslant 4k$*.*

Preuve. Nous prouvons par induction sur le nombre de conflits m contenus dans $\mathscr{C}$ qu'au moins m éditions sont nécessaires pour résoudre tous les conflits de $\mathscr{C}$. Si $m = 1$ alors le résultat est trivialement vérifié. Sinon, nous savons par hypothèse d'induction qu'au moins $m-1$ éditions sont nécessaires pour résoudre les conflits $\{C_1,\dots,C_{m-1}\}$. Par définition d'un conflict packing, deux configurations sont possibles : si C_m intersecte $\cup_{j=1}^{m-1} C_j$ en au plus deux sommets, une édition supplémentaire est alors nécessaire pour résoudre C_m. Dans le cas contraire, nous savons par définition que l'unique sommet de C_m non contenu dans $\cup_{j=1}^{m-1} C_j$ est une graine de C_m. De ce fait, aucune édition réalisée pour résoudre les conflits de $\{C_1,\dots,C_{m-1}\}$ ne peut résoudre le conflit C_m. Il suit que $m \leqslant k$; puisque chaque conflit de $\mathscr{C}$ contient au plus 4 sommets, nous avons également $|V(C)| \leqslant 4k$. □

Lemme 7.41. *Soient* (R,k) *une instance positive de* DENSE BETWEENNESS*, et* $\mathscr{C}$ *un conflict packing de* R*. Il existe un ordre* σ *sur* V *tel que tout triplet* $t \in \mathscr{R}$ *inconsistent avec* σ *vérifie* $V(t) \subseteq V(\mathscr{C})$*. De plus, un tel ordre peut être calculé en temps polynomial.*

Preuve. Nous posons $V' := V \setminus V(\mathscr{C})$. Remarquons que $R' := R[V']$ est consistente et admet un (unique) ordre consistent σ, qui peut être calculé en temps polynomial. De plus, observons que par maximalité du conflict packing $\mathscr{C}$, $R_a := R[V' \cup \{a\}$ est consistente pour tout sommet $a \in V(\mathscr{C})$. Il suit qu'il existe un unique ordre σ_a tel que tout triplet de R_a est consistent avec σ_a. En d'autres termes, σ contient une unique paire de sommets consécutifs $\{u,w\}$ telle que $u <_{\sigma_a} a <_{\sigma_a} w$. Nous appelons une telle paire $\{u,w\}$ le *locus* de a. En utilisant à nouveau la maximalité de $\mathscr{C}$, nous savons également que pour toute paire de sommets $a,b \in V(\mathscr{C})$, l'instance $R_{ab} := R[V' \cup \{a,b\}]$ est consistente. Ainsi, si a et b ont des loci différents, alors R_{ab} est clairement consistente avec l'ordre obtenu à partir de σ en insérant a et b à leurs loci respectifs. Il reste donc à considérer le cas où a et b ont le même locus. Étant donnés deux sommets consécutifs u et w de σ, nous définissons $B_{uw} \subseteq V(\mathscr{C})$ comme l'ensemble des sommets de $V(\mathscr{C})$ dont le locus est uw. Étant donnés $a,b \in B_{uw}$, nous définissons également la relation binaire $<_{uw}$ sur B_{uw} comme suit (observons qu'il y a exactement deux possibilités pour insérer a et b dans leur locus uw) : $<_{uw}$

$$a <_{uw} b \text{ si } abw \in \mathscr{R}$$

Affirmation 7.42. *La relation* $<_{uw}$ *est un ordre total strict (*i.e. *asymétrique et transitive).*

Preuve. Observons dans un premier temps que $<_{uw}$ est asymétrique par définition. Ainsi, il reste à prouver que $<_{uw}$ est transitive. Soient $a,b,c \in B_{uw}$ tels que $a <_{uw} b$ et $b <_{uw} c$. Cela signifie que $abw \in \mathscr{R}$ et $bcw \in \mathscr{R}$. Supposons par contradiction que $acw \notin \mathscr{R}$. Nous avons alors

$\{awc \vee wac\} \in \mathscr{R}$. Dans les deux cas, il suit que $\{a,b,c,w\}$ est un conflit quel que soit le choix pour le triplet $\{a,b,c\}$. De ce fait, w est une graine du conflit $\{a,b,c,w\}$, impliquant que le conflict packing $\mathscr{C}$ n'est pas maximal : contradiction. Il suit que $acw \in \mathscr{R}$ et donc $a <_{uw} c$, impliquant que $<_{uw}$ est transitive. ◇

Nous décrivons maintenant comment obtenir l'ordre σ' respectant les conditions du Lemme 7.41 à partir de σ. Pour toute paire de sommets consécutifs (dans σ) u et w telle que $B_{uw} \neq \emptyset$, nous insérons les sommets de B_{uw} en respectant l'ordre induit par l'ordre total strict $<_{uw}$. Nous affirmons que σ' respecte la propriété désirée. Soit $t := \{a,b,c\}$ un triplet de $\mathscr{R}$ tel que $V(t) \cap V' \neq \emptyset$. Dans un premier temps, si $V(t) \subseteq V'$, alors t est consistent avec σ' par construction de σ. Maintenant, si $|V(t) \cap V'| = 2$, t est encore une fois consistent par construction. Finalement, si $|V(t) \cap V'| = 1$ avec, par exemple, $a,b \in V(\mathscr{C})$, alors t est consistent car R_{ab} est consistent et les sommets a et b ont été insérés dans leur locus en respectant l'ordre $<_{uw}$. □

Théorème 7.43. *Soit (R,k) une instance de* DENSE BETWEENNESS *telle que $k < |V|/5$. Il existe un algorithme décidant si R est une instance positive en temps polynomial.*

Preuve. Nous considérons l'ordre σ de R obtenu via le Lemme 7.41, et calculons de manière gloutonne (*i.e.* en temps polynomial) un conflict packing $\mathscr{C}$ de R. Remarquons que, puisque $k < |V|/5$ par hypothèse, $|V| > 5k$ est vérifié. Par le Lemme 7.40, si $|V(\mathscr{C})| > 4k$, nous retournons NON. Nous pouvons donc supposer que $|V(\mathscr{C})| \leqslant 4k$ est vérifié. De ce fait, $V' := V \setminus V(\mathscr{C})$ contient au moins $k+1$ sommets n'appartenant à aucun triplet inconsistent par définition de σ.

Affirmation 7.44. *Il existe une sunflower simple $\mathscr{S} := \{C_1,\ldots,C_m\}$, $m > k$, qui peut être calculée en temps polynomial.*

Preuve. Soient $t \in \mathscr{R}$ un triplet inconsistent avec σ et $V'' \subseteq V'$ un ensemble de $k+1$ sommets de V'. Par le Lemme 7.35, nous savons que $C_d := V(t) \cup \{d\}$ définit un conflit simple pour tout sommet $d \in V''$. En supposant que $V'' := \{d_1,\ldots,d_{k+1}\}$, il suit que $C_i := V(t) \cup \{d_i\}$ est un conflit simple dont le seul triplet inconsistent est t, pour tout $i \in [k+1]$. Ainsi $\mathscr{S} := \{C_1,\ldots,C_{k+1}\}$ est une sunflower simple de centre t. ◇

La Règle 7.5 s'applique, et nous devons éditer le centre t de $\mathscr{S}$ par rapport à σ. Puisque σ contient toujours au moins $k+1$ sommets non contenus dans un triplet inconsistent après l'application de la règle de réduction, tout triplet inconsistent doit être édité par rapport à σ. Ainsi, si σ contient plus de k triplets inconsistents nous retournons NON ; dans le cas contraire, nous retournons OUI. □

Corollaire 7.45. *Le problème* DENSE BETWEENNESS *admet un noyau avec au plus $5k$ sommets.*

Preuve. Par le Théorème 7.43, nous savons que le problème peut se résoudre en temps polynomial si $k < |V|/5$. Ainsi, nous pouvons supposer sans perte de généralité que $k \geqslant |V|/5$, impliquant $|V| \leqslant 5k$. □

7.3.2 DENSE CIRCULAR ORDERING

Le problème DENSE CIRCULAR ORDERING est similaire au problème DENSE BETWEENNESS, et consiste à décider, étant donnée une collection dense $\mathscr{R}$ de *triplets circulaires* (qui sont des ordres cycliques définis sur 3 sommets) définis sur un univers V, s'il existe un ordre cyclique contenant tous sauf au plus k triplets de $\mathscr{R}$. La plupart des résultats évoqués dans la suite de cette section se prouvent de manière similaire aux résultats de la Section 7.3.1, et nous ne détaillons pas ces preuves. A l'instar des problèmes évoqués précédemment, le problème DENSE CIRCULAR ORDERING peut être caractérisé par des conflits finis.

Lemme 7.46 ([135])**.** *Une instance du problème* DENSE CIRCULAR ORDERING *est consistente si et seulement si elle ne contient pas de conflits définis sur quatre feuilles de V.*

Comme mentionné en début de section, l'algorithme de noyau pour le problème DENSE CIRCULAR ORDERING est similaire à l'algorithme de noyau du problème DENSE BETWEENNESS. Ainsi, les notions de Conflict Packing et de sunflower sont utilisées. En particulier, il est possible d'établir des analogues aux Lemmes 7.34, 7.35 et 7.37. En effet, le principe de *sunflower simple* défini pour le problème DENSE BETWEENNESS peut également s'exprimer sur le problème DENSE CIRCULAR ORDERING. Notons cependant que, contrairement au problème DENSE BETWEENNESS, il existe une *unique* manière d'éditer un triplet circulaire t. Ainsi, la règle de sunflower *classique* permet directement d'obtenir un algorithme de noyau quadratique pour ce problème. Cependant, cette observation n'est pas suffisante pour obtenir un algorithme de noyau linéaire. Pour ce faire, il est nécessaire d'utiliser la notion de Conflict Packing. À nouveau, la définition d'un conflict packing pour le problème DENSE CIRCULAR ORDERING est *identique* à la Définition 7.39, et le Lemme 7.40 est également vérifié pour ce problème. Nous énonçons maintenant les principaux résultats de cette section.

Lemme 7.47 ([135])**.** *Soient (R,k) une instance positive de* DENSE CIRCULAR ORDERING*, et $\mathscr{C}$ un conflict packing de B. Il existe un ordre σ sur V tel que tout triplet $t \in \mathscr{R}$ inconsistent avec σ vérifie $V(t) \subseteq V(\mathscr{C})$.*

Théorème 7.48 ([135])**.** *Soit (R,k) une instance de* DENSE CIRCULAR ORDERING *telle que $k < |V|/5$. Il existe un algorithme décidant si R est une instance positive en temps polynomial.*

Corollaire 7.49 ([135])**.** *Le problème* DENSE CIRCULAR ORDERING *admet un noyau contenant au plus $5k$ sommets.*

Partie III. Techniques alternatives et Conclusion

Dans cette partie, nous décrivons plusieurs techniques d'algorithmique paramétrée différentes des algorithmes de noyau. Si la recherche d'algorithmes de noyaux polynomiaux représente un enjeu théorique important (rappelons qu'il est conjecturé que certains problèmes paramétrés n'admettent pas de noyaux polynomiaux [16]), leur existence permet également d'obtenir des algorithmes FPT **efficaces** (e.g. simplement exponentiels en l) dans de nombreux cas. En effet, par le Théorème 2.10, nous savons qu'un algorithme de noyau couplé à n'importe quel algorithme résolvant le problème de manière exacte permet d'obtenir un algorithme FPT. Lorsque l'algorithme de noyau est **polynomial**, cette technique permet d'obtenir des algorithmes FPT efficaces, comme par exemple pour les problèmes VERTEX COVER ou MAXIMUM INTERNAL SPANNING TREE [65]. Cependant, il existe des problèmes paramétrés pour lesquels l'utilisation d'un algorithme de noyau et d'un algorithme exponentiel exact ne permet pas d'obtenir un algorithme FPT efficace. A titre d'exemple, nous évoquons le problème k-PATH, où l'instance est un graphe $G := (V, E)$ et un entier k et où l'objectif est de décider s'il existe un chemin de longueur k dans G.

Exemple. Le problème k-PATH admet plusieurs algorithmes FPT. En particulier, il peut être résolu de manière déterministe en temps $O^*(2^k)$ en utilisant la notion de Color Coding [6], la meilleure complexité connue ($O^*(1.66^k)$) étant elle obtenue via un algorithme randomisé. Maintenant, un algorithme de complexité $O^*(2^{k^c})$ (pour une certaine constante $c > 0$) utilisant des algorithmes de noyau et exact peut-il être obtenu pour ce problème ? En supposant que l'**Exponential Time Hypothesis (ETH)** est vérifiée [96], le problème k-PATH n'admet pas d'algorithme exponentiel exact de complexité $2^{o(n)}$. Ainsi, pour obtenir un algorithme FPT de complexité $O^*(2^{k^c})$ utilisant un algorithme de noyau, il est **nécessaire** d'avoir un noyau polynomial pour le problème k-PATH. Comme nous l'avons mentionné dans la **Section 2.3**, un tel algorithme ne peut pas exister [16] (sauf si $NP \subseteq coNP/poly$). Ainsi, l'approche via algorithme de noyau et algorithme exact ne peut pas permettre de résoudre **efficacement** ce problème.

De plus, il existe certains problèmes pour lesquels déterminer l'existence d'un algorithme de noyau polynomial semble être un problème difficile, ce qui est par exemple le cas des problèmes MULTICUT ou INTERVAL COMPLETION. Ainsi, il est nécessaire de développer des techniques d'algorithmique paramétrée alternatives pour résoudre de tels problèmes. Suivant cette ligne de

recherche, nous présentons trois méthodes **classiques** permettant d'obtenir des algorithmes FPT. Dans un premier temps, nous revenons sur la notion d'algorithme de branchement (**Section 2.1.2**). De plus, nous décrivons la notion de **sommet/arête inutile** et de **réduction à treewidth bornée**, qui permet d'obtenir des algorithmes FPT pour de nombreux problèmes [44, 123]. En particulier, nous mentionnons un travail réalisé avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ [44], dans lequel nous présentons un algorithme FPT permettant de réduire le problème MULTICUT à des instances de treewidth bornée par une fonction de k. Déterminer la complexité paramétrée de ce problème constituait un problème majeur en théorie de la complexité, et nos travaux ont permis à Bousquet et al. [22] d'obtenir un algorithme FPT pour ce problème. Finalement, nous présentons la technique de **Compression Itérative** introduite par Reed et al. [141] pour obtenir un algorithme FPT pour le problème ODD CYCLE TRANSVERSAL. Cette technique a par la suite été réutilisée dans de nombreux problèmes paramétrés, tels que CLUSTER VERTEX DELETION [95] ou encore FEEDBACK VERTEX SET [86].

Nous résumons également les principaux résultats connus en complexité paramétrée, ainsi que les travaux réalisés durant cette thèse (**Chapitre 9**). Nous rappelons notamment les deux principales techniques que nous avons introduites, à savoir la notion de **branches** pour le problème $\mathscr{G}$-EDITION et **Conflict Packing** pour le problème Π-EDITION. Pour conclure, nous mentionnons un certain nombre de problèmes ouvert, en relation avec les travaux présentés dans cette thèse.

CHAPITRE

8 Techniques alternatives

Dans ce chapitre, nous présentons diverses techniques permettant d'obtenir des algorithmes FPT sans utiliser la notion d'algorithme de noyau. Nous présentons trois méthodes : les **algorithmes de branchement** (déjà évoqués dans le **Chapitre 2**), la dichotomie **petite/grande treewidth** permettant de résoudre plusieurs problèmes paramétrés [123], ainsi que la technique de **Compression Itérative** introduite par Reed et al. [141] pour résoudre le problème ODD CYCLE TRANSVERSAL. Nous illustrons les deux dernières techniques avec des exemples concrets d'application sur les problèmes MULTICUT [44] et FEEDBACK VERTEX SET [95]. Le premier résultat est extrait de travaux réalisés en collaboration avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ.

MULTICUT :
Entrée : Un graphe $G := (V, E)$, un ensemble de requêtes (*i.e.* de paires de sommets de V) R, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k dont la suppression *déconnecte* toute requête de R ?

Theorem 8.1 ([44])**.** *Étant donnée une instance* (G, R, k) *de* MULTICUT*, il existe un algorithme FPT qui retourne une instance équivalente* (G', R', k') *telle que la treewidth de* G' *est bornée par une fonction de* k*.*

8.1 Règles de branchement

Dans cette section, nous développons la technique d'*algorithme de branchement* présentée dans la Section 2.1.2. En particulier, nous établissons certaines conditions pour déterminer quels types de problèmes paramétrés peuvent être résolus par algorithme de branchement. La plupart des résultats présentés dans cette section sont issus d'une présentation réalisée par *Daniel Marx* à *Worker 2010* [124].

8.1.1 Sur quels problèmes brancher ?

Nous présentons dans un premier temps une condition générale permettant de déterminer si un algorithme de branchement peut être efficace pour un problème paramétré. Comme nous l'avons vu dans la Section 2.1.2, un algorithme de branchement peut en particulier être utilisé lorsqu'il est possible d'établir que certains éléments *doivent* appartenir à une solution. Cela se traduit en particulier par la notion d'*ensemble nécessaire* [124]. Un *ensemble nécessaire N* pour un problème paramétré est un ensemble d'éléments tel que toute solution *doit contenir* au moins un élément de N. Rappelons que cela correspond aux arêtes du graphe pour le problème VERTEX COVER, ou encore aux sous-graphes induits interdits pour le problème $\mathcal{G}$-VERTEX DELETION où $\mathcal{G}$ est héréditaire et caractérisée par une famille finie de sous-graphes induits interdits. En fonction de la taille de N, nous obtenons alors des algorithmes de branchement ayant des complexités différentes :

(i) $|N|$ est constante : algorithme ayant une complexité $O^*(c^k)$ pour une certaine constante $c > 0$;

(ii) $|N| \in O(k)$: algorithme ayant une complexité $O^*(k^k) = O^*(2^{k \log k})$;

(iii) $|N| \in O(\log n)$: algorithme ayant une complexité $O^*(\log^k n)$.

Nous nous concentrons essentiellement sur les problèmes ayant une complexité $O^*(c^k)$. Nous formalisons à présent la notion d'algorithme de branchement en introduisant le concept de *règle de branchement*.

Règle de branchement ([124])**.** *Une* règle de branchement *pour un problème paramétré L est un algorithme* polynomial *qui, étant donnée une instance (I,k) de L, $k > 1$, produit c instances $(I_1,k_1),\ldots,(I_c,k_c)$ (pour une certaine constante $c > 0$) telles que :*

(i) $|I_i| \leqslant |I|$ *pour tout* $i \in [c]$,

(ii) $k_i < k$ *pour tout* $i \in [c]$ *et,*

(iii) (I,k) *est une instance positive si et seulement s'il existe* $i \in [c]$ *tel que* (I_i,k_i) *est une instance positive.*

Remarque. Un problème pour lequel une règle de branchement *devrait* diminuer le paramètre de moitié à chaque étape n'admet pas d'algorithme de branchement. En effet, cela impliquerait que l'arbre de recherche ait une profondeur bornée par $\log k$, et ainsi une taille polynomiale en k, ce qui ne peut pas arriver à moins que $P = NP$.

En utilisant la notion de *transformation polynomiale en temps et paramètre* définie dans la Section 2.3. il est possible de *porter* l'existence d'un algorithme de branchement pour un problème L_2 vers un problème L_1. En particulier, en imposant que le nouveau paramètre soit *linéaire* en k, il

est également possible de conserver l'ordre de grandeur de la complexité. Autrement dit, si L_1 se réduit à L_2 avec une transformation polynomiale en temps et linéaire en paramètre, et si L_2 admet un algorithme de branchement de complexité $O^*(c^k)$, alors L_1 admet un algorithme de branchement de complexité $O^*(c^k)$. Il est ainsi possible de définir la classe *branchFPT* [124], qui contient un problème paramétré L s'il existe une transformation polynomiale en temps et *linéaire en paramètre* depuis L vers un problème L' admettant une règle de branchement. En particulier, cette classe et ces transformations particulières peuvent être liées à la notion d'algorithme de noyau via l'observation suivante.

Observation 8.2 ([124])**.** *Soit L un problème paramétré (défini sur un univers V) admettant un noyau linéaire (en nombre d'éléments de V). Si L paramétré par le* nombre d'éléments de V *peut être résolu avec une règle de branchement, alors L appartient à $branchFPT$.*

En effet, dans un tel cas, il existe une transformation polynomiale en temps et linéaire en paramètre de L vers un problème (le noyau paramétré par le nombre de sommets) qui peut être résolu par branchement. Nous verrons dans la Section 8.3 que la notion de *Compression Itérative* couplée à un algorithme d'approximation à facteur constant permet également d'établir un algorithme FPT via arbre de recherche.

Pour conclure sur cette partie, nous mentionnons un dernier résultat permettant d'établir quels types de problèmes paramétrés admettent des algorithmes via règle de branchement.

Observation 8.3 ([124])**.** *Un problème paramétré L appartient à la classe $branchFPT$ si et seulement si il a une caractérisation pour la classe NP avec des certificats de taille $O(k)$.*

Résultats. Comme nous l'avons vu dans la Section 2.1.2, le problème VERTEX COVER peut se résoudre avec un algorithme de branchement. Plusieurs autres problèmes admettent également des algorithmes de branchement, comme par exemple les problèmes d'édition de graphes où la classe cible $\mathcal{G}$ est caractérisée par un ensemble fini d'obstructions [31]. Des algorithmes plus techniques ont également été développés pour d'autres problèmes, comme par exemple INTERVAL COMPLETION [94] ou encore FEEDBACK VERTEX SET [86].

8.1.2 Questions ouvertes

Nous terminons cette section en présentant quelques questions ouvertes relatives à la notion de règle de branchement.

Problème 2. *Le problème* MAXIMUM INTERNAL SPANNING TREE *paramétré par le nombre de sommets peut-il être résolu via règle de branchement ?*

Comme nous l'avons mentionné dans la Section 2.2.1, ce problème (paramétré par le nombre de noeuds internes) admet un noyau contenant au plus $3k$ sommets [65]. Ainsi, s'il existe un algorithme de branchement lorsque nous paramétrons par le nombre de sommets de l'instance, alors l'Observation 8.2 implique que ce problème appartient à $branchFPT$. Nous concluons finalement cette section avec le problème suivant.

Problème 3. *Le problème k-*PATH *admet-il un algorithme de branchement ?*

8.2 Réduction à treewidth bornée

8.2.1 Dichotomie petite/grande treewidth

Nous mentionnons maintenant une technique classique en algorithmique paramétrée, qui consiste à réduire (en temps FPT) une instance d'un problème donné vers une instance équivalente et ayant une treewidth [143] *bornée* en fonction du paramètre considéré k. Comme nous l'avons mentionné dans la Section 2.1.3, de nombreux problèmes paramétrés par la *treewidth de l'instance* admettent des algorithmes FPT (Théorème de Courcelle [42]). Ainsi, une telle réduction couplée au Théorème 2.7 permet d'obtenir un algorithme FPT pour le problème considéré paramétré par k.

Observation 8.4. *Soit L un problème (défini sur un graphe $G := (V,E)$) paramétré par k. Supposons que le problème L paramétré par la treewidth de G puisse être résolu en temps FPT. Si une instance (G,k) de L peut être réduite en temps FPT vers une instance* équivalente *de treewidth bornée par une fonction de k, alors le problème L admet un algorithme FPT.*

Applications. A titre d'exemple, remarquons que le problème CHORDAL DELETION, dont la complexité paramétrée a été ouverte pendant de nombreuses années, a récemment été résolu par Daniel Marx [123], qui a fourni un algorithme en utilisant la notion de réduction à treewidth bornée. Le problème étant expressible en Logique Monadique du Second Ordre, le Théorème de Courcelle [42] permet alors de conclure. Afin d'obtenir cette réduction, Daniel Marx a en particulier utilisé la notion d'*inutilité*, qui est fortement liée à la théorie de la treewidth et des mineurs [143]. Dans la section suivante, nous donnons un exemple d'application de ces notions sur le problème MULTICUT [44], la totalité de l'algorithme étant présenté dans l'Annexe C. Réalisé avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ [44], ce travail a servi de base pour l'élaboration d'un algorithme FPT résolvant le problème MULTICUT [22].

8.2.2 Application au problème MULTICUT

Nous présentons maintenant une application des idées présentées Section 8.2.1 pour le problème MULTICUT, défini comme suit.

MULTICUT :
Entrée : Un graphe $G := (V,E)$, un ensemble de requêtes (*i.e.* de paires de sommets de V) R, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k dont la suppression *déconnecte* toute requête de R ?

Résultat. Dans un travail réalisé avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ [44], nous utilisons l'idée d'*inutilité* [123] et la théorie des mineurs [142, 143] afin de démontrer que le problème MULTICUT peut être réduit en temps FPT à des instances dont la treewidth est bornée par une fonction de k. Déterminer la complexité paramétrée de ce problème était une question majeure en complexité paramétrée [49], et nos résultats ont été utilisés par Bousquet et

al. [22] afin d'obtenir un algorithme FPT pour MULTICUT. Le principal résultat évoqué dans cette section est le suivant.

Theorem 8.5 ([44])**.** *Étant donnée une instance* (G,R,k) *de* MULTICUT*, il existe un algorithme FPT qui retourne une instance équivalente* (G',R',k') *telle que la treewidth de* G' *est bornée par une fonction de* k*.*

Remarque. Ce résultat ne suffit cependant pas à obtenir un algorithme FPT pour MULTICUT. En effet, MULTICUT n'est pas expressible en Logique Monadique du Second Ordre, et le Théorème de Courcelle [42] (Théorème 2.7) ne peut donc pas s'appliquer. Néanmoins, une partie de nos résultats a été utilisée par Bousquet et al. [22] afin d'obtenir un algorithme FPT pour MULTICUT.

Si leur algorithme ne se sert pas directement de notre réduction à treewidth bornée, il utilise en revanche le résultat suivant, qui est l'une des principales conséquences de notre réduction. Le *graphe des requêtes* G_R d'une instance (G,R,k) de MULTICUT contient les sommets V de G comme sommets, et il existe une arête entre deux sommets $u,v \in V(G_R)$ si et seulement si $\{u,v\} \in R$.

Theorem 8.6 ([44])**.** *Soit* (G,R,k) *une instance de* MULTICUT*. Il existe une fonction* f *telle que le degré du graphe des requêtes* G_R *est borné par* $f(k)$*.*

Réduire le graphe. Soit (G,R,k) une instance du problème MULTICUT. Nous établissons maintenant des règles de réduction pour le problème MULTICUT. Nous verrons par la suite comment utiliser la notion de treewidth afin d'obtenir un algorithme FPT utilisant ces règles de réduction.

Les outils principaux de notre approche résident dans l'étude de problèmes de connectivité d'intérêt indépendant. En particulier, nous prouvons que le problème TRIPLE SEPARATION admet un algorithme FPT. Étant donnés trois sommets $u,v,w \in V(G)$, nous définissons une $(uv|w)$*-coupe de taille* k comme un ensemble de k arêtes de G dont la suppression déconnecte w de u mais ne déconnecte pas u de v. Formellement, ce problème s'exprime comme suit.

TRIPLE SEPARATION :
Entrée : Un graphe $G := (V,E)$, trois sommets u,v,w, un entier k.
Paramètre : k.
Question : Existe-t-il une $(uv|w)$-coupe de taille au plus k ?

Théorème 8.7. *Le problème* TRIPLE SEPARATION *admet un algorithme FPT*[1].

Le deuxième problème que nous considérons est basé sur la notion de *connectivité forte*. Étant donnés un graphe $G := (V,E)$, un ensemble $T \subseteq V$, deux entiers k et l et un sommet u de G, nous disons que $v \notin T$ est l*-fortement* $(u|T)$*-* k*-connecté* si pour tout ensemble $S \subseteq T$ tel que $|S| \geqslant |T| - l$, il n'existe pas de $(uv|S)$-coupe de taille k. En d'autres termes, pour tout tel sous-ensemble S, toute coupe de taille au plus k entre u et S est une coupe entre v et S. Nous prouvons que pour tout sommet $u \in V$ et tout ensemble T de taille suffisamment grande (par rapport au paramètre k), il existe un sommet $v \in T$ qui est l-fortement $(u|T \setminus \{v\})$-k- connecté et qui peut être calculé en temps

1. Un résultat similaire a été obtenu par Daniel Marx et Igor Razgon [125].

FPT. Ce résultat nous permet en particulier d'obtenir la règle de réduction suivante. Nous disons qu'une requête $r \in R$ est *inutile* si les instances (G, R, k) et $(G, R \setminus \{r\}, k)$ sont équivalentes.

> *S'il existe un sommet $u \in V$ incident à au moins $k^{O(k)}$ requêtes, alors il existe une requête inutile contenant u qui peut être identifiée en temps FPT.*

Ces deux résultats nous permettent donc d'obtenir les règles de réduction suivantes pour MULTICUT. Un ensemble $T \subseteq V$ est *groupé* si pour tout ensemble $F \subseteq E$ de taille au plus k, il existe au plus une composante connexe de $G \setminus F$ contenant plus d'un sommet de T.

Règle 8.1 ([44])**.** *Soit (G, R, k) une instance de* MULTICUT.
- *S'il existe un sommet $u \in V$ incident à au moins $k^{O(k)}$ requêtes, alors il existe une requête inutile contenant u qui peut être identifiée en temps FPT.*
- *Si G est réduit par la règle de réduction précédente, et s'il existe un ensemble groupé de taille suffisamment grande, alors il existe une requête inutile qui peut être identifiée en temps FPT.*

Borner la treewidth. Nous décrivons maintenant comment ces règles de réduction peuvent être appliquées en temps FPT. Par un résultat majeur de Robertson et Seymour [142, 143], il est connu que tout graphe ayant une grande treewidth admet une grande grille comme mineur. Nous utilisons cette propriété structurelle des graphes ayant une grande treewidth afin d'obtenir des informations sur les solutions possibles pour le problème. Nous considérons tout d'abord le cas où le graphe G contient une grande *clique* comme mineur, utilisant les règles de réduction précédentes pour réduire le graphe. De manière similaire, nous considérons ensuite le cas où le graphe admet une grande grille comme mineur mais n'admet pas de grande clique mineur. Si les idées sont similaires dans les deux cas, le second requiert une analyse technique plus approfondie. Quelle que soit l'hypothèse supposée, nous démontrons que le graphe admet soit un ensemble groupé de terminaux (*i.e.* extrémités de requêtes) de taille suffisante, soit qu'il existe une arête qui peut être contractée de manière valide. Dans le premier cas, la Règle 8.1 peut donc s'appliquer. Ainsi, tant que la treewidth du graphe est suffisamment grande, nous pouvons le réduire via un algorithme FPT et ainsi obtenir le Théorème 8.5.

8.3 Compression itérative

8.3.1 Description générale

Pour conclure ce chapitre, nous décrivons la technique dite de *Compression Itérative*, qui a permis d'établir de nombreux algorithmes FPT [95, 120, 123, 140]. Cette technique a été introduite par Reed et al. afin d'établir un algorithme FPT pour le problème ODD CYCLE TRANSVERSAL[2] [141]. Cette méthode peut s'appliquer dans de nombreux problèmes, aussi bien d'édition de graphes [95, 123] que de relations [140]. De manière générale, la Compression Itérative considère une version *compressée* du problème, pour laquelle chaque instance est accompagnée d'une solution S et où l'objectif est de déterminer si une solution de taille au plus k existe. Le problème revient alors à résoudre cette version compressée avec un algorithme ayant une complexité $f(k, |S|) \cdot n^{O(1)}$.

2. Étant donné un graphe $G := (V, E)$ et un entier k, existe-t-il un ensemble d'au plus k sommets dont la suppression dans G permet d'obtenir un graphe biparti ?

Informellement, en utilisant la phase *itérative*, il est alors possible de construire une solution de taille $k+1$ pour le problème original, et d'utiliser l'algorithme de compression afin de le résoudre. De manière plus générale, dès lors que la solution S a une taille bornée par une fonction de k, l'algorithme de compression permet d'obtenir un algorithme FPT pour le problème considéré. Cela permet en particulier de lier cette notion aux algorithmes d'approximation, comme nous le verrons Section 8.3.2.

Compression Itérative pour le problème $\mathcal{G}$-VERTEX DELETION. Dans la suite de cette section, nous illustrons cette technique sur le problème $\mathcal{G}$-VERTEX DELETION, où la propriété $\mathcal{G}$ est héréditaire.

$\mathcal{G}$-VERTEX DELETION :
Entrée : Un graphe $G := (V, E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ appartient à $\mathcal{G}$?

Le principe de la compression itérative consiste à considérer une *version compressée* du problème, définie comme suit :

$\mathcal{G}$-VERTEX DELETION COMPRESSION :
Entrée : Un graphe $G := (V, E)$, un ensemble $S \subseteq V$ tel que $G \setminus S$ appartient à $\mathcal{G}$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ appartient à $\mathcal{G}$?

Le principal intérêt de cette méthode réside dans le résultat suivant.

Lemme 8.8. *Si le problème $\mathcal{G}$-VERTEX DELETION COMPRESSION admet un algorithme FPT*[3]*, alors le problème $\mathcal{G}$-VERTEX DELETION admet un algorithme FPT.*

La raison de ce résultat est la suivante : soit (G, k) une instance du problème $\mathcal{G}$-VERTEX DELETION. Nous supposons que $\mathcal{G}$-VERTEX DELETION COMPRESSION admet un algorithme FPT. Nous construisons une solution de manière *itérative*, en ajoutant des sommets au graphe au fur et à mesure. Plus précisément, considérons la séquence de graphes $\{G_1, \ldots, G_{k+1}\}$, où $G_1 := \{u\}$ pour un sommet $u \in V(G)$ et où chaque graphe $G_i := (V_i, E_i)$, $2 \leqslant i \leqslant k+1$, est obtenu à partir de G_{i-1} en ajoutant un sommet $v \notin V_{i-1}$. De manière évidente, l'ensemble V_{k+1} est une solution de taille $k+1$ pour le graphe G_{k+1}. Nous effectuons maintenant l'étape de *compression* : soit G_{k+2} le graphe obtenu suivant l'étape définie précédemment. Nous appliquons maintenant l'*algorithme de compression* sur l'instance (G_{k+2}, V_{k+1}, k), et obtenons soit une réponse négative, soit une solution de taille au plus k. Dans le premier cas, nous pouvons directement répondre NON : en effet, la propriété $\mathcal{G}$ étant héréditaire, le fait que (G_{k+2}, V_{k+1}, k) soit une instance négative implique directement que

3. Nous supposons dans la suite de cette section que les algorithmes FPT considérés sont *constructifs, i.e.* fournissent effectivement une solution au problème.

(G,k) est une instance négative. Dans le second cas, soit S_{k+2} la solution de taille au plus k obtenue. Nous réitérons les étapes précédentes sur le graphe G_{k+3} en considérant la solution (de taille au plus $k+1$) $S_{k+2} \cup \{v\}$. Nous répétons ces étapes jusqu'au graphe G_n, obtenant une solution (s'il en existe une) pour le graphe G. Ainsi, si la version compressée du problème peut être résolue en temps $f(k) \cdot n^c$ pour une certaine fonction calculable f et une certaine constante $c > 0$, alors le problème original peut être résolu en temps $f(k) \cdot n^{c+1}$.

Algorithme de compression. Nous décrivons maintenant brièvement le principe de l'algorithme de compression résolvant le problème $\mathscr{G}$-VERTEX DELETION COMPRESSION en temps FPT. Soit (G,S,k) une instance de ce problème, telle que S a taille au plus $k+1$. L'idée est d'utiliser l'information structurelle apportée par la solution S afin de décider si une solution de plus petite taille existe. L'algorithme de compression va donc considérer *toutes les bipartitions possibles* de l'ensemble S. Chaque partie correspondra aux sommets *forcés à appartenir à la solution*, et aux sommets *dont l'appartenance à la solution est interdite*. Soient $X \subseteq S$ le premier ensemble, et $Y := X \setminus S$. Observons dans un premier temps que $G[Y]$ doit appartenir à la classe $\mathscr{G}$, sans quoi il ne sera pas possible d'obtenir une solution. L'objectif est alors de concevoir un algorithme polynomial ou paramétré permettant de déterminer s'il existe une solution S' de taille au plus k, vérifiant de plus $X \subseteq S'$ et $S' \cap Y = \emptyset$. Dans la mesure où il existe 2^{k+1} bipartitions possibles pour l'ensemble S, l'étape de compression peut alors s'effectuer en temps FPT.

Remarques :

- Dans notre description, nous avons supposé que $|S| = k+1$. Comme mentionné précédemment, ce principe peut également être appliqué si l'instance est accompagnée d'une solution ayant une taille bornée par une fonction de k, qui peut notamment être obtenue en utilisant un algorithme d'approximation.
- En particulier, cela permet de lier la notion de compression itérative à celle d'algorithme via règle de branchement. En effet, tout problème paramétré admettant un algorithme d'approximation *à facteur constant* et dont la version compressée peut être résolue via un algorithme de branchement admet un algorithme FPT via règle de branchement. Observons de plus que, dans ce cas, *une unique* application de l'algorithme de compression est suffisante, dans la mesure où l'algorithme d'approximation permet d'obtenir une solution pour le graphe G directement. Les étapes itératives de l'algorithme de compression (permettant de construire une solution de taille $k+1$) ne sont donc plus nécessaires, et permettent d'améliorer la complexité d'un facteur n.

8.3.2 Application sur FEEDBACK VERTEX SET

Nous donnons à présent un exemple d'application de la technique de *Compression Itérative* sur le problème FEEDBACK VERTEX SET.

FEEDBACK VERTEX SET :
Entrée : Un graphe $G := (V,E)$, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ est une forêt ?

Comme décrit dans la Section 8.3, nous considérons une *version compressée* de ce problème, définie comme suit.

FEEDBACK VERTEX SET COMPRESSION :
Entrée : Un graphe $G := (V, E)$, un feedback vertex set S, un entier k.
Paramètre : k.
Question : Existe-t-il un ensemble $F \subseteq V$ de taille au plus k tel que le graphe $G \setminus F$ est une forêt ?

Nous allons voir que ce problème admet un algorithme FPT ayant une complexité $O(c^k \cdot m)$, ce qui impliquera que la version non compressée du problème peut se résoudre en temps $O(c^k \cdot mn)$ (Lemme 8.8). Pour ce faire, l'algorithme reprend les étapes décrites dans la Section 8.3.1 : nous construisons dans un premier temps une solution S de taille $k+1$ de manière itérative, puis considérons les 2^{k+1} bipartitions possibles de S en deux ensembles X et Y, correspondant respectivement aux sommets qui *doivent* appartenir à la solution, et à ceux qui ne *doivent pas* appartenir à la solution. Sans perte de généralité, nous supprimons X de l'instance considérée à chaque étape, obtenant ainsi un graphe $G \setminus X$ et un feedback vertex set Y pour ce graphe. L'objectif est alors de décider s'il existe un feedback vertex set S' de taille inférieure ou égale à $k - |X|$ vérifiant $S' \cap Y = \emptyset$. Pour cela, l'algorithme décrit par Guo et al. [86] utilise deux règles de réduction.

Règle 8.2. *Soit* (G, S, k) *une instance de* FEEDBACK VERTEX SET COMPRESSION. *Supprimer de* G *tout sommet de degré* 1.

Règle 8.3. *Soit* (G, S, k) *une instance de* FEEDBACK VERTEX SET COMPRESSION. *S'il existe un sommet* $v \in S$ *de degré* 2 *ayant pour voisins* v_1 *et* v_2 *avec* $v_1 \notin S$ *ou* $v_2 \notin S$, *alors supprimer* v *de* G *et connecter* v_1 *et* v_2 *par une arête. Si une arête parallèle est créée entre* v_1 *et* v_2, *alors supprimer le sommet de* $\{v_1, v_2\}$ *n'appartenant pas à* S *et l'ajouter à la solution.*

Dans la suite de cette section, nous admettons la validité de ces règles de réduction.

Lemme 8.9 ([86])**.** *Les Règles 8.2 et 8.3 sont valides et peuvent être appliquées en temps polynomial.*

L'utilisation de ces règles de réduction permet en particulier d'obtenir le résultat suivant.

Lemme 8.10 ([86])**.** *Étant donnée une instance* (G, S, k) *de* FEEDBACK VERTEX SET COMPRESSION *telle que* $|S| = k+1$, *il existe un algorithme FPT de complexité* $O^*(c^k)$ *retournant une solution de taille au plus* k *(s'il en existe une).*

Le résultat suivant est alors vérifié, et provient du Lemme 8.8.

Théorème 8.11. *Le problème* FEEDBACK VERTEX SET *admet un algorithme FPT de complexité* $O(c^k \cdot mn)$.

Remarque. Il est possible d'améliorer la partie *polynomiale* de cette complexité (au détriment de la constante c) en utilisant la dernière remarque expliquée dans la Section 8.3.1. En effet, le problème FEEDBACK VERTEX SET admet une 4-approximation [7] qui peut être obtenue en temps $O(m)$. Ainsi, nous pouvons supposer que S a taille $4k$ dans l'énoncé du Lemme 8.10, et considérer les 2^{4k} bi-partitions possibles de S. Il suit que le problème FEEDBACK VERTEX SET admet un algorithme de paramétré ayant une complexité $O(c^k \cdot m)$ [86].

CHAPITRE

9 Conclusion et perspectives de recherche

Dans le cadre de cette thèse, nous avons établi plusieurs résultats théoriques concernant l'existence de noyaux polynomiaux pour des problèmes d'édition de graphes (et autres structures). Deux principales techniques se dégagent de ces résultats, à savoir les notions de **branches** et de **Conflict Packing**. Ces techniques nous ont notamment permis d'obtenir des algorithmes de noyau polynomiaux pour les problèmes CLOSEST 3-LEAF POWER, PROPER INTERVAL COMPLETION, COGRAPH EDITION, FEEDBACK ARC SET IN TOURNAMENTS et DENSE ROOTED TRIPLET INCONSISTENCY. Avant de **résumer ces deux méthodes** et de **rappeler les résultats qu'elles nous ont permis d'obtenir**, nous dressons **un état de l'art de la théorie de la complexité paramétrée**. Par la suite, nous énonçons un certain nombre de **problèmes ouverts**, découlant pour la plupart des travaux présentés dans cette thèse.

9.1 État de l'art

Complexité paramétrée. Depuis sa formalisation dans la fin des années 1990 par Downey et Fellows [57], la théorie de la complexité paramétrée a reçu une attention considérable. En particulier, de nombreuses techniques ont été développées afin de permettre de concevoir des algorithmes FPT de manière *générique*. Comme nous l'avons évoqué dans cette thèse, ces techniques comprennent le *Color Coding* [6], la *Compression Itérative* [141] (présentée Section 8.3), les *Règles de Branchement* [124] (présentées Section 8.1), ou encore le principe de *Réduction à treewidth bornée* [143, 123] (présenté Section 8.2.1).

Noyaux. Parmi les nombreuses techniques existantes, nous nous sommes principalement intéressés au principe d'*Algorithme de Noyau*, qui est considéré comme l'une des techniques les plus puissantes permettant de concevoir des algorithmes FPT [14]. Comme nous l'avons énoncé dans le Théorème 2.10, l'existence d'un algorithme de noyau est équivalente à l'existence d'un algorithme FPT. Cependant, lorsque le problème considéré est NP-Difficile, ce résultat permet uniquement d'obtenir des noyaux de taille *super-polynomial*. Déterminer l'existence d'un algorithme de noyau *polynomial* pour un problème paramétré donné constitue donc un axe de recherche important, notamment car cela permet d'en améliorer la complexité dans plusieurs cas (voir par exemple [65]). L'un des premiers algorithmes de noyau polynomial a été établi par Buss et Goldsmith [29] pour le problème VERTEX COVER. Depuis lors, de nombreux algorithmes de noyau ont été proposés pour le problème VERTEX COVER, culminant en un noyau contenant au plus $2k$ sommets obtenus via diverses méthodes : (réduction des couronnes et programmation linéaire [3], par exemple). Récemment, un noyau contenant au plus $2k-c$ sommets pour toute constante $c>0$ a été établi [149]. Une question naturelle qui découlait du Théorème 2.10 était de déterminer si *tous* les problèmes paramétrés admettaient des algorithmes de noyau *polynomiaux*. En 2008, Bodlaender et al. [16] et Fortnow et Santhanam [70] ont répondu par la négative à cette question, en démontrant qu'il existait des problèmes paramétrés (dont notamment k-PATH et k-TREEWIDTH) qui n'admettaient pas d'algorithme de noyau polynomial, sauf si certaines hypothèses de complexité n'étaient pas vérifiées. Ce résultat majeur a ainsi permis d'ouvrir de nouveaux axes de recherche, et d'établir que de nombreux problèmes paramétrés *n'admettent (probablement) pas* d'algorithme de noyau polynomial. En se basant sur ce résultat, de nombreuses techniques ont été développées, et en particulier les *transformations polynomiales en temps et paramètre* [20] et la *cross-composition* [18], qui permet d'unifier les deux techniques précédemment citées.

Édition de graphes et de relations. En ce qui concerne l'existence de noyaux polynomiaux pour les problèmes d'édition de graphes et de relations, de nombreux résultats ont été démontrés. Les principaux consistent en un noyau quadratique pour le problème FEEDBACK VERTEX SET obtenu par Stéphan Thomassé [151], un noyau polynomial pour MULTICUT IN TREES obtenu par Bousquet et al. [23], ou bien encore un résultat récent de Kratsch et Wahlström proposant un noyau polynomial pour ODD CYCLE TRANSVERSAL [112]. De plus, ces dernières années ont également vu apparaître des *méta-théorèmes*, qui permettent d'établir l'existence de noyaux polynomiaux pour des problèmes paramétrés sous des hypothèses *générales*. Comme nous l'avons mentionné dans l'introduction de la Partie I, cela est en particulier le cas pour les problèmes paramétrés *compacts* définis sur des graphes de *genre borné* via la notion de *protrusions* [17]. Ces problèmes admettent

des algorithmes de noyaux polynomiaux, qui deviennent linéaires dès lors que le problème considéré a un *index entier fini* [17]. En utilisant à nouveau ce concept de protrusions, de récents travaux menés par Fomin et al. [67] ont également permis d'établir l'existence de noyaux polynomiaux pour des problèmes de *suppression de sommets*, où l'objectif est d'obtenir un graphe ne contenant pas une liste de graphes données comme *mineur*. En particulier, il a été prouvé que ce problème admettait un noyau polynomial dès lors que la liste d'*obstructions* comprenait un graphe planaire. En ce qui concerne la non-existence de noyaux polynomiaux pour ces problèmes, Krastch et Wahlström [111] ont récemment établi que certains problèmes de modification de graphes n'admettaient pas de noyaux polynomiaux, répondant ainsi par la négative à une conjecture posée par Cai [15]. Ces résultats se basent notamment sur les notions de *ou-composition* et de *transformation polynomiale en temps et paramètre* introduites par Bodlaender et al. [16, 20]. Observons pour conclure que de nouvelles techniques (notamment via le concept de Sparsification [48]) ont permis d'établir des *bornes inférieures pour des noyaux polynomiaux*. Il a en particulier été démontré que le problème VERTEX COVER n'admettait pas de noyau contenant $O(k^{2-\epsilon})$ *arêtes*, pour toute constante $\epsilon > 0$. Un résultat similaire est vérifié pour le problème FEEDBACK VERTEX SET, impliquant que le noyau quadratique établi par Stéphan Thomassé [151] est *optimal*.

9.2 Techniques introduites et résultats obtenus

Nous résumons maintenant les techniques que nous avons introduites dans le cadre de cette thèse afin d'obtenir des algorithmes de noyau polynomial pour plusieurs problèmes d'édition. Nous verrons dans la Section 9.3 que nous espérons que ces dernières vont permettre d'obtenir des algorithmes de noyau pour d'autres problèmes.

Branches. Dans le cadre de l'étude des problèmes d'édition de graphes, nous avons tout d'abord établi des noyaux polynomiaux pour les problèmes CLOSEST 3-LEAF POWER et PROPER INTERVAL COMPLETION. Pour ce faire, nous avons introduit la notion de *branches* pour le problème $\mathcal{G}$-EDITION, lorsque la classe $\mathcal{G}$ admet une certaine *décomposition d'adjacence*. Le principe de cette technique est d'identifier en temps polynomial des ensembles de sommets induisant un graphe appartenant à $\mathcal{G}$ et étant connecté au reste du graphe en respectant la décomposition d'adjacence de $\mathcal{G}$. Une fois identifiés, le travail restant est de montrer que ces ensembles peuvent être réduits à des ensembles de taille bornée par une fonction de k, tout en préservant la structure d'une solution. Cette partie de l'algorithme de noyau est spécifique au problème. Nous avons ainsi obtenu un noyau contenant $O(k^3)$ sommets pour le problème CLOSEST 3-LEAF POWER, et un noyau contenant $O(k^5)$ sommets pour PROPER INTERVAL COMPLETION. Ces résultats constituent les premiers algorithmes de noyau polynomial connus pour ces problèmes. Finalement, en utilisant la décomposition modulaire et des notions similaires aux *branches*, nous obtenons un noyau cubique pour les problèmes COGRAPH EDITION, COGRAPH COMPLETION et COGRAPH EDGE-DELETION.

Bornes inférieures. En utilisant les algorithmes de ou-composition [16] et les transformations polynomiales en temps et paramètre [20], nous avons établi des bornes inférieures pour les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION, pour $l \geqslant 7$. De récents travaux nous ont permis d'améliorer ces bornes : en utilisant la technique de cross-composition [18], nous avons

pu démontrer que le problème VERTEX COVER DANS LES GRAPHES DE MAILLE $\geqslant 9$ [1] se cross-compose dans le problème C_l-FREE EDGE DELETION pour $l \geqslant 4$. Cette réduction utilise des techniques similaires à celles présentées dans la Section 2.3.5. En particulier, le *graphe de sélection* est également présent dans cette réduction.

Conflict Packing. Dans un second temps, nous avons formalisé et développé une technique déjà utilisée dans le cadre de quelques problèmes paramétrés [27, 63, 155], que nous avons appelée *Conflict Packing*. Cette méthode considère principalement des problèmes d'édition de relations, mais peut également s'appliquer à des problèmes d'édition de graphes. En utilisant la méthode *classique* déjà utilisée pour d'autres problèmes, nous avons ainsi pu retrouver un noyau quadratique pour le problème CLUSTER VERTEX DELETION. De plus, nous avons également présenté un algorithme de noyau linéaire pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Par la suite, nous avons étendu et raffiné cette technique afin d'obtenir des noyaux linéaires pour les problèmes DENSE ROOTED TRIPLET INCONSISTENCY, DENSE BETWEENNESS et DENSE CIRCULAR ORDERING. En ce qui concerne le problème DENSE ROOTED TRIPLET INCONSISTENCY, nous avons amélioré un noyau existant contenant $O(k^2)$ sommets [82]. En ce qui concerne DENSE BETWEENNESS et DENSE CIRCULAR ORDERING, notre résultat constitue le premier noyau polynomial connu.

9.3 Perspectives de recherche et problèmes ouverts

9.3.1 Edition de graphes

Branches. L'une des premières question soulevée par les méthodes introduites dans le cadre de cette thèse est de savoir si la notion de branches peut-être utilisée afin d'établir l'existence d'algorithmes de noyau polynomiaux pour d'autres problèmes. En particulier, il existe plusieurs problèmes d'édition de graphes pour lesquels cette notion pourrait s'avérer utile. Cependant, les règles de réduction qui devront être établies afin de réduire ces dernières seront encore une fois **locales** au problème, et demanderont donc une analyse différente des résultats présentés dans cette thèse. Deux problèmes majeurs se dégagent notamment de cette analyse, à savoir CHORDAL DELETION et INTERVAL COMPLETION, demandant respectivement de supprimer k arêtes d'un graphe pour obtenir un graphe triangulé, et d'ajouter k arêtes à un graphe afin d'obtenir un graphe d'intervalle. Ces deux problèmes admettent des algorithmes FPT [94, 123], mais l'existence de noyaux polynomiaux est à ce jour ouverte.

Problème 4. *Les problèmes* CHORDAL DELETION *et* INTERVAL COMPLETION *admettent-ils des algorithmes de noyaux polynomiaux ?*

Ordres. Dans l'élaboration de notre algorithme de noyau polynomial pour le problème PROPER INTERVAL COMPLETION, nous avons utilisé la notion de branches et le fait que les graphes d'intervalle propre admettent un ordre particulier. Cette observation nous a également permis d'obtenir un algorithme de noyau pour le problème FEEDBACK ARC SET IN TOURNAMENTS. En ce sens, il semble naturel de se demander si cette notion d'ordre (couplée ou non au principe de branches)

1. La *maille* d'un graphe $G := (V, E)$ correspond à la taille minimum d'un cycle de G.

peut être utilisée pour obtenir des algorithmes de noyau polynomiaux pour d'autres problèmes d'édition de graphes. En particulier, nous mentionnons deux classes de graphes : les graphes de *comparabilité* (qui sont les graphes dont les arêtes admettent une orientation transitive) et les graphes de *permutation* (qui sont les graphes de comparabilité dont le complémentaire est également un graphe de comparabilité). Ces orientations définissant des ordres particuliers, peuvent-elles être utilisées pour concevoir des algorithmes de noyau ?

Problème 5. *Les problèmes* COMPARABILITY EDITION *et* PERMUTATION EDITION *admettent-ils des algorithmes FPT ? Si oui, admettent-ils des noyaux polynomiaux ?*

Nous mentionnons maintenant deux problèmes ouverts liés aux bornes inférieures pour les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION.

Problème 6. *Les problèmes* P_5-FREE EDGE DELETION *et* P_6-FREE EDGE-DELETION *admettent-ils des noyaux polynomiaux ?*

Bornes inférieures. Dans le Chapitre 2, nous avons établi des bornes inférieures pour des problèmes de *suppression d'arêtes*. En considérant les versions complémentaires des problèmes étudiés, nos résultats impliquent en particulier qu'il existe des problèmes de *complétion d'arêtes* n'admettant pas de noyaux polynomiaux, même si la classe de graphes cibles est très restrictive. Par exemple, le fait que le problème C_4-EDGE DELETION n'admette pas de noyau polynomial est équivalent au fait que le problème $2K_2$-COMPLETION n'admette pas de noyau polynomial. Ainsi, l'existence d'un algorithme de noyau polynomial pour le problème $\mathscr{G}$-COMPLETION ne peut pas être due au fait que $\mathscr{G}$ soit caractérisée par un ensemble fini d'obstructions (une telle observation avait déjà été rendue possible par le résultat de Kratsch et Wahlström [111]). De ce fait, il serait intéressant de déterminer quels types de problèmes de complétion d'arêtes admettent des noyaux polynomiaux. En particulier, nous avons établi dans les Chapitres 3 et 4 que les problèmes 3-LEAF POWER COMPLETION et PROPER INTERVAL COMPLETION admettaient des noyaux polynomiaux, ces problèmes étant tous deux caractérisés par une famille de sous-graphes induits interdits comprenant les cycles de longueur supérieure ou égale à 4 et un ensemble fini d'autres graphes.

Problème 7. *Le problème* $\mathscr{G}$-COMPLETION, *où* $\mathscr{G}$ *est caractérisée par une famille de sous-graphes induits interdits comprenant les cycles de longueur supérieure ou égale à* 4 *et un ensemble fini d'autres graphes admet-il un noyau polynomial ?*

Décomposition. Dans le Chapitre 5, nous avons utilisé le fait que la classe des cographes est *totalement décomposable pour la décomposition modulaire* afin d'élaborer notre algorithme de noyau. En ce sens, il serait intéressant de déterminer si cette caractéristique des cographes est la raison de l'existence d'un tel noyau, ou si cette dernière provient de la caractérisation par sous-graphe induit interdit (rappelons que les cographes sont les graphes n'admettant pas de P_4 comme sous-graphe induit) ? Les résultats négatifs obtenus pour le problème P_l-FREE EDGE DELETION (Section 2.3.5, Chapitre 2) laissent supposer que le second cas n'est pas la raison de l'existence d'un algorithme de noyau polynomial. Ainsi, un problème important est de savoir si d'autres décompositions peuvent être utilisées dans la conception d'algorithmes de noyau. Cela peut en particulier se traduire avec le

problème suivant, les graphes *distances héréditaires* étant *totalement décomposables pour la split-décomposition* [75].

Problème 8. *Le problème* DISTANCE HEREDITARY EDITION *admet-il un noyau polynomial ?*

Divers. Comme mentionné dans la Section 8.2.2, le problème MULTICUT admet un algorithme FPT. Qu'en est-il de l'existence d'un noyau polynomial ?

Problème 9. *Le problème* MULTICUT *admet-il un algorithme de noyau polynomial ?*

Cette question semble relativement difficile, et il semblerait de plus surprenant que ce problème admette un algorithme de noyau polynomial. Ainsi, une autre question est de déterminer les cas pour lesquels le problème admet un noyau polynomial, notamment en fonction de la *treewidth* de l'instance considérée. Le cas $treewidth(G) = 1$, correspondant à MULTICUT IN TREES, admet un noyau polynomial [23].

Problème 10. *Le problème* MULTICUT *admet-il un algorithme de noyau polynomial sur les graphes de treewidth* 2 *? Plus généralement, le problème admet-il un noyau polynomial pour les graphes de treewidth t, pour une certaine constante $t > 1$?*

Problème 11. *Peut-on améliorer les tailles de noyau obtenues pour les problèmes* CLOSEST 3-LEAF POWER, PROPER INTERVAL COMPLETION *et* COGRAPH EDITION *?*

$\mathcal{G}$-Vertex Deletion. Pour conclure avec les problèmes ouverts d'édition de graphes, nous mentionnons le cas des problèmes FEEDBACK VERTEX SET IN TOURNAMENTS et CLUSTER VERTEX DELETION, pour lesquels la méthode Conflict Packing pourrait s'appliquer. Ces problèmes admettent des algorithmes de noyau quadratiques [1], et déterminer si ces résultats peuvent être améliorés est une question importante en théorie de la complexité paramétrée.

Problème 12. *Les problèmes* FEEDBACK VERTEX SET IN TOURNAMENTS *et* CLUSTER VERTEX DELETION *admettent-ils des noyaux linéaires (voire sous-quadratiques) ?*

9.3.2 Edition de relations

Conflict Packing. En ce qui concerne les problèmes d'édition de relations, il existe de nombreux problèmes NP-Complets dans le cas dense pour lesquels l'existence ou non de noyaux polynomiaux n'est pas encore établie. En particulier, il serait intéressant de déterminer si la technique Conflict Packing peut être utilisée (couplée ou non à la notion de Partition Sûre) afin d'obtenir des algorithmes de noyau polynomiaux pour ces problèmes. Les problèmes d'édition de relations de taille 3 où la structure cible est une permutation ont notamment été étudiés dans le contexte de la complexité paramétrée [91].

Problème 13. *Les problèmes (denses) d'édition de relations de taille* 3 *où la structure cible est un ordre admettent-ils des noyaux polynomiaux ?*

Comme nous l'avons mentionné, l'existence d'un algorithme d'approximation à facteur constant pour le problème DENSE ROOTED TRIPLET INCONSISTENCY permettrait également d'obtenir un algorithme de noyau linéaire, dont la taille dépendrait du facteur d'approximation.

Problème 14. *Le problème* DENSE ROOTED TRIPLET INCONSISTENCY *admet-il un algorithme d'approximation à facteur constant ?*

Pour conclure cette section, nous mentionnons des problèmes d'édition de relation particuliers, où la structure cible est un ordre et où les relations considérées sont *(faiblement-)fragiles*. Une relation est *fragile* si tout déplacement d'un sommet sur un ordre consistent résulte en un ordre inconsistent. Observons que cela est par exemple le cas pour le problème FEEDBACK ARC SET IN TOURNAMENTS, dans la mesure où il existe une unique manière d'ordonner un arc afin que ce ne soit pas un arc retour. De manière similaire, une relation est *(faiblement-)fragile* si un de ces déplacements rend inconsistent tout ordre consistent : inverser les deux premiers sommets, les deux derniers sommets, ou bien faire un déplacement cyclique du premier ou dernier sommet. Il a été démontré que ces problèmes admettent des algorithmes FPT ainsi que des PTAS [104].

Problème 15. *Les problèmes d'édition de relations où la structure cible est un ordre et où les relations sont (faiblement-)fragiles admettent-ils des noyaux polynomiaux ?*

Bibliographie

[1] F. N. Abu-Khzam. A kernelization algorithm for d-hitting set. *Journal of Computer and System Sciences*, 76(7) :524 – 531, 2010. Cité pages 48, 117, 124, and 171.

[2] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the vertex cover problem : Theory and experiments. In *Workshop on Algorithm Engineering and Experiments and Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALC)*, pages 62–69, 2004. Cité page 21.

[3] F. N. Abu-Khzam, M. R. Fellows, M. A. Langston, and W. H. S. Crown structures for vertex cover kernelization. *Theory of Computing Systems*, 41(3) :411–430, 2007. Cité pages 21 and 167.

[4] A. Aho, Y. Sagiv, T. Szymansk, and J. Ullman. Inferring a tree from lowest common ancestor with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3) :405–421, 1981. Cité page 127.

[5] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *Automata, Languages and Programming (ICALP)*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009. Cité pages 33, 36, 116, 130, and 201.

[6] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the Association for Computing Machinery*, 42(4) :844–856, 1995. Cité pages 8, 36, 151, and 167.

[7] R. Bar-Yehuda, D. Geiger, J. S. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27(4) :942–959, 1998. Cité page 163.

[8] V. Berry and F. Nicolas. Maximum agreement and compatible supertrees. *Journal of Discrete Algorithms*, 5(3) :564–591, 2007. Cité page 116.

[9] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 4 of *LIPIcs*, pages 37–47, 2009. Cité pages 11, 119, 120, 121, and 132.

[10] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for feedback arc set in tournaments. *Journal of Computer and System Sciences*, In Press, Corrected Proof :–, 2010. Cité page 11.

[11] S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. In *International Workshop On Combinatorial Algorithms (IWOCA)*, volume 5874 of *Lecture Notes in Computer Science*, pages 72–82, 2009. Cité pages 10, 11, 49, 50, 52, and 57.

[12] S. Bessy, C. Paul, and A. Perez. Polynomial kernels for 3-leaf power graph modification problems. *Discrete Applied Mathematics*, 158(16) :1732–1744, 2010. Cité pages 10, 11, 49, 50, 52, 57, 61, and 74.

[13] S. Bessy and A. Perez. Polynomial kernels for proper interval completion and a related problem. In *International Symposium on Fundamentals of Computation Theory (FCT)*, number 6914, pages 1732–1744, 2011. Cité pages 10, 11, 49, 50, 77, and 103.

[14] H. L. Bodlaender. Kernelization : New upper and lower bound techniques. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 17–37, 2009. Cité pages 8, 29, 36, and 167.

[15] H. L. Bodlaender, L. Cai, J. Chen, M. R. Fellows, J. A. Telle, and D. Marx. Open problems in parameterized and exact computation. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, 2006. Cité pages 40, 49, 168, and 191.

[16] H. L. Bodlaender, R. Downey, M. Fellows, and D. Hermelin. On problems without polynomial kernels. In *Automata, Languages and Programming (ICALP)*, number 5125 in LNCS, pages 563–574, 2008. Cité pages 9, 34, 35, 36, 151, 167, and 168.

[17] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Symposium on Foundations of Computer Science (FOCS)*, pages 629–638, 2009. Cité pages 49, 54, 55, 167, and 168.

[18] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition : A new technique for kernelization lower bounds. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *Leibniz International Proceedings in Informatics*, pages 165–176, 2011. Cité pages 27, 39, 40, 45, 167, and 168.

[19] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel bounds for path and cycle problems. *CoRR*, abs/1106.4141, 2011. Cité page 27.

[20] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *European Symposium on Algorithms (ESA)*, volume 5757 of *LNCS*, pages 635–646, 2009. Cité pages 35, 37, 38, 167, and 168.

[21] H. L. Bodlaender and T. C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory of Computing Systems*, 46(3) :566–597, 2010. Cité page 50.

[22] N. Bousquet, J. Daligault, and S. Thomassé. Multicut is FPT. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 459–468. ACM, 2011. Cité pages 10, 47, 48, 152, 157, 158, and 215.

[23] N. Bousquet, J. Daligault, S. Thomassé, and A. Yeo. A polynomial kernel for multicut in trees. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, Leibniz International Proceedings in Informatics, pages 183–194, 2009. Cité pages 167 and 171.

[24] A. Brandstädt and V. B. Le. Structure and linear time recognition of 3-leaf powers. *Information Processing Letters*, 98(4) :133–138, 2006. Cité pages 57 and 60.

[25] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes : A Survey*. SIAM Monographs on Discrete Mathematics and Applications. 1999. Cité page 107.

[26] A. Brandstädt, V. B. Le, and R. Sritharan. Structure and linear time recognition of 4-leaf powers. *ACM Transactions on Algorithms (TALG)*, 5(1) :1–22, 2008. Cité pages 57 and 60.

[27] D. Brügmann, C. Komusiewicz, and H. Moser. On generating triangle-free graphs. *Electronic Notes in Discrete Mathematics*, 32 :51–58, 2009. Cité pages 10, 116, 117, 121, 122, 123, and 169.

[28] B.-M. Bui-Xuan, J.A. Telle, and M. Vatshelle. Boolean-width of graphs. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5917 of *Lecture Notes in Computer Science*, pages 61–74, 2009. Cité page 28.

[29] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, (3) :560–572, 1993. Cité pages 48 and 167.

[30] J. Byrka, S. Guillemot, and J. Jansson. New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11) :1136–1147, 2010. Cité page 127.

[31] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4) :171–176, 1996. Cité pages 24, 26, 40, 48, 105, 156, and 191.

[32] L. Cai and J. Chen. On fixed-parameter tractability and approximability of NP optimization problems. *Journal of Computer System and Sciences*, 54(3) :465–474, 1997. Cité page 8.

[33] L. S. Chandran and F. Grandoni. Refined memorization for vertex cover. *Information Processing Letters*, 93(3) :123–131, 2005. Cité pages 8, 9, 10, 24, and 27.

[34] M-S. Chang and M-T. Ko. The 3-steiner root problem. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, pages 109–120, 2007. Cité page 57.

[35] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combinatorics, Probability and Computing*, 16(1) :1–4, 2007. Cité pages 10 and 47.

[36] J. Chen, Y. Liu, and S. Lu. Directed feedback vertex set problem is FPT. In *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, number 07281 in Dagstuhl Seminar Proceedings, 2007. Cité page 10.

[37] J. Chen and J. Meng. A $2k$ kernel for the cluster editing problem. In *International Computing and Combinatorics Conference (COCOON)*, volume 6196 of *Lecture Notes in Computer Science*, pages 459–468, 2010. Cité pages 9, 21, 47, 48, and 52.

[38] B. Chor, M. R. Fellows, and D. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004. Cité page 34.

[39] N. Cohen, F. V. Fomin, G. Gutin, E.J. Kim, S. Saurabh, and A. Yeo. Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. *Journal of Computer and System Sciences*, 76(7) :650–662, 2010. Cité page 31.

[40] S. A. Cook. The complexity of theorem proving procedures. *ACM Symposium on Theory of Computing*, 1971. Cité pages 4, 5, 6, and 23.

[41] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Transactions on Algorithms*, 6(3), 2010. Cité page 116.

[42] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1) :12–75, 1990. Cité pages 10, 28, 157, and 158.

[43] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. In *International Symposium on Parameterized and Exact Computation (IPEC)*, 2011. Cité page 29.

[44] J. Daligault, C. Paul, A. Perez, and S. Thomassé. Treewidth reduction for the parameterized Multicut problem. *http ://www.lirmm.fr/~daligault/MulticutTreewidthReduction.pdf*, 2010. Cité pages 10, 11, 28, 47, 152, 153, 157, 158, and 159.

[45] P. Damaschke. Fixed-parameter enumerability of Cluster Editing and related problems. *Theory of Computing Systems*, 46(2) :261–283, 2010. Cité page 30.

[46] G. B. Dantzig, D. R. Fulkerson, and S. M. Iohnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2 :393, 1954. Cité page 3.

[47] F. Dehne, M. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang. The cluster editing problem : implementations and experiments. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 4169 of *Lecture Notes in Computer Science*, pages 13–24, 2006. Cité page 47.

[48] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *ACM Symposium on Theory of Computing (STOC)*, pages 251–260, 2010. Cité page 168.

[49] E. D. Demaine, M. Hajiaghayi, and D. Marx. Open problems – parameterized complexity and approximation algorithms. Number 09511 in Dagstuhl Seminar Proceedings, 2010. Cité pages 10 and 157.

[50] X. Deng, P. Hell, and J. Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing*, 25(2) :390–403, 1996. Cité page 80.

[51] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959. Cité page 3.

[52] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. In *International Symposium on Algorithms and Computation (ISAAC)*, number 3341 in Lecture Notes in Computer Science, pages 389–401, 2004. Cité page 10.

[53] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Error compensation in leaf root problems. *Algorithmica*, 44 :363–381, 2006. Cité pages 15, 49, 57, 74, and 96.

[54] M. Dom, J. Guo, F. Hüffner, and R. Niedermeier. Closest 4-leaf power is fixed-parameter tractable. *Discrete Applied Mathematics*, 156(18) :3345–3361, 2008. Cité pages 57 and 59.

[55] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Automata, Languages and Programming (ICALP)*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009. Cité pages 27, 37, 38, and 39.

[56] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II : On completeness for $W[1]$. *Theoretical Computer Science*, 141(1–2) :109–131, 1995. Cité pages 8 and 23.

[57] R.G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999. Cité pages 8, 23, 27, 48, and 167.

[58] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17 :449–467, 1965. Cité pages 3 and 28.

[59] J. Egerváry. Matrixok kombinatoricus tuladjdonsagairol. *Matematikai és Fizikai Lapok*, 38 :16–28, 1931. Cité page 133.

[60] E. S. El-Mallah and C. Colbourn. The complexity of some edge deletion problems. *IEEE Transactions on Circuits and Systems*, 35(3) :354–362, 1988. Cité page 105.

[61] M. R. Fellows, M. Langston, F. Rosamond, and P. Shaw. Efficient parameterized preprocessing for cluster editing. In *International Symposium on Fundamentals of Computation Theory (FCT)*, number 4639 in Lecture Notes in Computer Science, pages 312–321, 2007. Cité page 21.

[62] M. R. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. A. Rosamond, and S. Saurabh. The complexity ecology of parameters : An illustration using bounded max leaf number. *Theory of Computing Systems*, 45(4) :822–848, 2009. Cité pages 26 and 27.

[63] H. Fernau and D. Raible. A parameterized perspective on packing paths of length two. *Journal of Combinatorial Optimization*, 18(4) :319–341, 2009. Cité pages 10, 116, 117, 121, and 169.

[64] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Text in Theoretical Computer Science. Springer, 2006. Cité page 8.

[65] F. V. Fomin, S. Gaspers, S. Saurabh, and S. Thomassé. A linear vertex kernel for maximum internal spanning tree. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 5878 of *Lecture Notes in Computer Science*, pages 275–282, 2009. Cité pages 31, 151, 156, and 167.

[66] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Springer, 2010. Cité page 7.

[67] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh. Hitting forbidden minors : Approximation and kernelization. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *Leibniz International Proceedings in Informatics*, pages 189–200, 2011. Cité page 168.

[68] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Fast local search algorithm for weighted feedback arc set in tournaments. In *AAAI Conference on Artificial Intelligence*, pages 65–70, 2010. Cité page 47.

[69] L. R. Ford and D. R. Fulkerson. *Flows in networks*. Princeton University Press, Princeton, N.J.,, 1962. Cité page 3.

[70] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *ACM Symposium on Theory of Computing (STOC)*, pages 133–142, 2008. Cité pages 9, 34, 35, and 167.

[71] P. Galinier, M. Habib, and C. Paul. Chordal graphs and their clique graphs. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 1017 of *Lecture Notes in Computer Science*, pages 358–371, 1995. Cité page 53.

[72] M.R. Garey and D.S. Jonhson. *Computers and intractability : a guide to the theory on NP-Completeness*. W.H. Freeman and Co., 1979. Cité pages 4 and 36.

[73] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1) :3–20, 1997. Cité page 10.

[74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory*, 16 :47–56, 1974. Cité page 15.

[75] E. Gioan, C. Paul, M. Tedder, and D. G. Corneil. Practical and efficient split decomposition via graph-labelled trees. *CoRR*, abs/1104.3283, 2011. Cité page 171.

[76] M.C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15, 1994. Cité page 77.

[77] A.D. Gordon. Consensus super tree containing overlapping sets of labeled leaves. *Journal of Classification*, 3 :31–39, 1986. Cité page 127.

[78] G. Gottlob and S. T. Lee. A logical approach to multicut problems. *Information Processing Letters*, 103(4) :136–141, 2007. Cité pages 28 and 47.

[79] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering : Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4) :373–392, 2005. Cité page 47.

[80] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13(2.2), 2008. Cité page 31.

[81] S. Guillemot and V. Berry. Fixed-parameter tractability of the maximum agreement supertree problem. *IEEE/ACM Trans. Comput. Biology Bioinform*, 7(2), 2010. Cité page 116.

[82] S. Guillemot and M. Mnich. Kernel and fast algorithm for dense triplet inconsistency. In *Theory and Applications of Models of Computation (TAMC)*, volume 6108 of *Lecture Notes in Computer Science*, pages 247–257, 2010. Cité pages 33, 47, 50, 116, 127, 135, 136, 137, 145, and 169.

[83] S. Guillemot, C. Paul, and A. Perez. On the (non-)existence of polynomial kernels for p_l - free edge modification problems. In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 6478 of *Lecture Notes in Computer Science*, pages 147–157, 2010. Cité pages 11, 37, 49, and 50.

[84] J. Guo. A more effective linear kernelization for cluster editing. In *International Symposium Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE)*, volume 4614 of *Lecture Notes in Computer Science*, pages 36–47, 2007. Cité pages 48 and 102.

[85] J. Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10) :718–726, 2009. Cité pages 21 and 52.

[86] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72 :1386–1396, 2006. Cité pages 152, 156, 162, and 163.

[87] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1) :31–45, 2007. Cité pages 9 and 23.

[88] G. Gutin, E.J. Kim, M. Mnich, and A. Yeo. Betweenness parameterized above tight lower bound. *Journal of Computer and System Sciences*, 76(8) :872–878, 2010. Cité page 29.

[89] G. Gutin, E.J. Kim, S. Szeider, and A. Yeo. Solving MAX-2-SAT above a tight lower bound. *CoRR*, abs/0907.4573, 2009. Cité page 28.

[90] G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The linear arrangement problem parameterized above guaranteed value. *Mathematical Systems Theory*, 41, 2007. Cité page 28.

[91] G. Gutin, L. van Iersel, M. Mnich, and A. Yeo. All ternary permutation constraint satisfaction problems parameterized above average have kernels with quadratic numbers of variables. In *European Symposium on Algorithms (ESA)*, volume 6346 of *Lecture Notes in Computer Science*, pages 326–337, 2010. Cité page 171.

[92] M. Habib and C. Paul. A survey on algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1) :41–59, 2010. Cité pages 10 and 16.

[93] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10(37) :26–30, 1935. Cité pages 134 and 143.

[94] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval completion with few edges. In *ACM Symposium on Theory of Computing (STOC)*, pages 374–381, 2007. Cité pages 47, 48, 156, and 169.

[95] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory of Computing Systems*, 47(1) :196–217, 2010. Cité pages 152, 153, and 159.

[96] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity ? *Journal of Computer and System Sciences*, 63(4) :512–530, 2001. Cité page 151.

[97] B. M. P. Jansen and H. L. Bodlaender. Vertex cover kernelization revisited : Upper and lower bounds for a refined parameter. In *Symposium on Theoretical Aspects of Computer Science*

(STACS), volume 9 of *Leibniz International Proceedings in Informatics*, pages 177–188, 2011. Cité page 28.

[98] B. M. P. Jansen and S. Kratsch. Data reduction for graph coloring problems. *CoRR*, abs/1104.4229, 2011. Cité page 27.

[99] I. A. Kanj, M. J. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 3162 of *Lecture Notes in Computer Science*, pages 235–247, 2004. Cité page 116.

[100] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs : Minimum fill-in and physical mapping. In *Symposium on Foundations of Computer Science (FOCS)*, pages 780–791, 1994. Cité pages 10, 15, 47, 48, 49, and 77.

[101] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28(5) :1906–1922, 1999. Cité pages 10, 48, and 77.

[102] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of the Workshop on Complexity of Computations (Yorktown Heights, 1972)*, pages 85–103, 1972. Cité pages 7, 10, and 47.

[103] R. M. Karp. Mapping the genome : some combinatorial problems arising in molecular biology. In *ACM Symposium on Theory of Computing (STOC)*, pages 278–285, 1993. Cité page 77.

[104] M. Karpinski and W. Schudy. Approximation schemes for the betweenness problem in tournaments and related ranking problems. *CoRR*, abs/0911.2214, 2009. Cité pages 116, 121, 132, 146, and 172.

[105] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 3–14, 2010. Cité pages 10 and 116.

[106] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors : A PTAS for weighted feedback arc set on tournaments. In *ACM Symposium on Theory of Computing (STOC)*, pages 95–103, 2007. Cité pages 116, 121, 132, and 201.

[107] C. Komusiewicz and J. Uhlmann. A cubic-vertex kernel for flip consensus tree. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2 of *LIPIcs*, pages 280–291, 2008. Cité page 51.

[108] D. König. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38 :116–119, 1931. Cité page 133.

[109] J. Kratochvil, P. D. Manuel, and M. Miller. Generalized domination in chordal graphs. 2, 1995. Cité page 15.

[110] S. Kratsch. Polynomial kernelizations for MIN $F^{+}\mathrm{pi}_1$ and MAX NP. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 601–612, 2009. Cité page 121.

[111] S. Kratsch and M. Wahlström. Two edge modification problems without polynomial kernels. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5917 of *Lecture Notes in Computer Science*, pages 264–275, 2009. Cité pages 37, 41, 42, 49, 168, 170, 187, and 192.

[112] S. Kratsch and M. Wahlström. Compression via matroids : A randomized polynomial kernel for odd cycle transversal. *CoRR*, abs/1107.3068, 2011. Cité page 167.

[113] M. R. Krom. The decision problem for formulas in prenex conjunctive normal form with binary disjunctions. *Journal of Symbolic Logic*, 35(2) :210–216, 1970. Cité page 3.

[114] V. B. Le and N. N. Tuy. Hardness results and efficient algorithms for graph powers. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 5911 of *Lecture Notes in Computer Science*, pages 238–249, 2009. Cité page 57.

[115] L. Levin. Universal sequential search problems. *Problems of Information Transmission (translated from Problemy Peredachi Informatsii (Russian))*, 1973. Cité page 5.

[116] G. H. Lin, P. E. Kearney, and T. Jiang. Phylogenetic k-root and steiner k-root. In *International Symposium on Algorithms and Computation (ISAAC)*, number 1969 in Lecture Notes in Computer Science, pages 539–551, 2000. Cité pages 17 and 57.

[117] Y. Liu, J. Wang, J. Guo, and J. Chen. Cograph editing : Complexity and parameterized algorithms. 158(16) :1732–1744, 2010. Cité pages 47 and 105.

[118] D. Lokshtanov. Wheel-free deletion is W[2]-hard. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5018 of *Lecture Notes in Computer Science*, pages 141–147, 2008. Cité page 48.

[119] D. Lokshtanov and S. Saurabh. Kernel : Lower and Upper bounds, 2009. Cité pages 31 and 33.

[120] D. Lokshtanov, S. Saurabh, and S. Sikdar. Simpler parameterized algorithm for OCT. In *International Workshop On Combinatorial Algorithms, (IWOCA)*, volume 5874 of *Lecture Notes in Computer Science*, pages 380–384. Springer, 2009. Cité page 159.

[121] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications*, 25(7) :15 – 25, 1993. Cité pages 77 and 79.

[122] D. Marx. Chordal deletion is fixed-parameter tractable. In *International Workshop on Graph Theoretical Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 37–48, 2006. Cité page 15.

[123] D. Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4) :747–768, 2010. Cité pages 8, 10, 15, 28, 47, 48, 49, 152, 153, 157, 159, 167, and 169.

[124] D. Marx. What's next ? http://www.lorentzcenter.nl/lc/web/2010/418/presentations/Marx_What_s_next.pdf, 2010. Cité pages 155, 156, and 167.

[125] D. Marx, B. O'Sullivan, and I. Razgon. Treewidth reduction for constrained separation and bipartization problems. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, Leibniz International Proceedings in Informatics, 2010. Cité page 158.

[126] D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. In *European Symposium on Algorithms (ESA)*, volume 5757 of *Lecture Notes in Computer Science*, pages 647–658. Springer, 2009. Cité page 10.

[127] S. Mishra, V. Raman, S. Saurabh, and S. Sikdar. The complexity of könig subgraph problems and above-guarantee vertex cover. Cité page 28.

[128] J. Nederlof. Fast polynomial-space algorithms using Möbius inversion : Improving on Steiner tree and related problems. In *Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 713–725, 2009. Cité page 31.

[129] R. Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*. Oxford University Press, USA, March 2006. Cité pages 8, 9, 23, 24, and 48.

[130] R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73(3-4) :125–129, 2000. Cité pages 9 and 31.

[131] R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2) :63–77, 2003. Cité pages 8, 9, 10, 24, and 27.

[132] S. Oum and P. D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory*, 96(4) :514–528, 2006. Cité page 28.

[133] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. 1982. Cité page 7.

[134] C. Paul, A. Perez, and S. Thomassé. Conflict packing yields linear-vertex kernels for k-FAST, k-DENSE RTI and a related problem. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, number 6907 in Lecture Notes in Computer Science, pages 497–507, 2011. Cité pages 10, 11, 117, 119, and 121.

[135] C. Paul, A. Perez, and S. Thomassé. Conflict packing yields linear vertex-kernels for rooted triplet inconsistency and other problems. *CoRR*, abs/1101.4491, 2011. Cité pages 146 and 149.

[136] F. Protti, M. Dantas da Silva, and J.L. Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 2007. Cité pages 9, 17, 21, 48, 52, and 73.

[137] V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science*, 351(3) :446–458, 2006. Cité page 129.

[138] I. Rapaport, K. Suchan, and I. Todinca. Minimal proper interval completions. *Information Processing Letters*, 106(5) :195–202, 2008. Cité page 47.

[139] I. Razgon and B. O'Sullivan. Directed feedback vertex set is fixed-parameter tractable. In *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, number 07281 in Dagstuhl Seminar Proceedings, 2007. Cité pages 10 and 47.

[140] I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed-parameter tractable. *Journal of Computer and System Sciences*, 75(8) :435–450, 2009. Cité pages 28 and 159.

[141] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32 :299–301, 2004. Cité pages 8, 10, 48, 152, 153, 159, and 167.

[142] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory*, 63(1) :65–110, 1995. Cité pages 15, 28, 157, and 159.

[143] N. Robertson, P.D. Seymour, and R. Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory*, 62(2) :323–348, 1994. Cité pages 8, 15, 28, 157, 159, and 167.

[144] Y. Saab. A fast and effective algorithm for the feedback arc set problem. *Journal of Heuristics*, 7 :235–250, 2001. Cité page 129.

[145] S. Saurabh. Chromatic coding and universal (hyper-)graph coloring families. *Parameterized Complexity News*, pages 49–58, June 2009. Cité page 127.

[146] C. Semple and M. Steel. *Phylogenetics*, volume 24 of *Oxford Lecture Series in Mathematics and Its Applications*. Oxford University Press, 2003. Cité page 127.

[147] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2) :173–182, 2004. Cité pages 10 and 47.

[148] R. Sharan. *Graph modification problems and their applications to genomic research*. PhD thesis, Tel-Aviv University, 2002. Cité pages 10 and 47.

[149] A. Soleimanfallah and A. Yeo. A kernel of order $2k - c$ for vertex cover. *Discrete Mathematics*, 311(10-11) :892–895, 2011. Cité pages 21, 31, 48, and 167.

[150] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13 :566–579, 1984. Cité pages 15 and 47.

[151] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010. Cité pages 10, 47, 48, 116, 167, and 168.

[152] R. van Bevern, H. Moser, and R. Niedermeier. Kernelization through tidying. In *Latin American Theoretical Informatics Symposium (LATIN)*, volume 6034 of *Lecture Notes in Computer Science*, pages 527–538, 2010. Cité pages 10, 48, 117, 121, 123, and 124.

[153] A. van Zuylen, R. Hegde, K. Jain, and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 405–414, 2007. Cité page 47.

[154] V. V. Vazirani. *Approximation algorithms*. Springer, 2001. Cité page 7.

[155] J. Wang, D. Ning, Q. Feng, and J. Chen. An improved kernelization for P_2-packing. *Information Processing Letters*, 110(5) :188–192, 2010. Cité pages 10, 116, 117, 121, and 169.

[156] G. Wegner. *Eigenschaften der nerven homologische-einfactor familien in R^n*. PhD thesis, Universität Gottigen, Gottingen, Germany, 1967. Cité pages 77 and 79.

[157] M. Yannakakis. Computing the minimum fill-in is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1) :77–79, 1981. Cité page 102.

[158] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24, 1987. Cité page 15.

Index

Notations

$<_e$ et $\sim_e$ 138
$<_{uw}$ 145
$G+F$ 16
$G-F$ 16
$G \setminus S$ 11
$G \triangle F$ 16
$G_1 \oplus G_2$ 12
$N_G(S)$ 11
$N_G(v)$ 11
$N_G[v]$ 11
$N_S(v)$ 11
O^* 21
$\leqslant_{tptp}$ 35
$\mathscr{H}$-free 12
σ_S 12
$d_G(u,v)$ 12
$u <_\sigma v$ 12
$u \leqslant_\sigma v$ 12
uv 11

A

arbre de décomposition modulaire 14
arc retour 128

B

1-branche (3-leaf power) 60
1-branche (intervalle propre) 79
2-branche (3-leaf power) 60
2-branche (intervalle propre) 80
branche 51

C

certificat (FAST) 128
certificat (RTI) 135
certifier (FAST) 128
certifier (RTI) 135
chemin 12
clique critique 15
cographe 105
composante connexe 12
conflict packing 119
conflict packing (FAST) 130
conflict packing (RTI) 137
conflit 117
conflit simple 135
connexe 12
consistence (RTI) 133
consistente 117
cross-composition 37
cycle 12

D

degré 11
denses 113

E

k-édition 16
édition 16

F
FPT 21
frontière 11

G
graine 137
graphe d'intervalle propre 77
graphe des cliques critiques 15
graphe des conflits 121
graphes bi-clique-chaînés 100
graphes bipartis-chaînés 100

H
héréditaire 12

I
inconsistente 117
instance 21
instance enracinée 134
instance réduite 27

J
jumeaux 11

K
K-join 81

L
3-leaf power 58
p-leaf power 57

M
modules triviaux 14

O
ordre transitif 127
ou-composition 33

P
partition arborée 136
partition ordonnée 118, 128
Partition Sûre 117
partition sûre (FAST) 129
partition sûre (RTI) 136

R
règles de réduction 27

S
sommet affecté 16
sous-graphe induit 11
sunflower simple 144

T
tournoi ordonné 128
triangle orienté 127
triplet enraciné 132

V
validité 28

W
W-hiérarchie 21

Table des figures

Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial. 4
Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial car il n'existe pas de tel algorithme. 4
Je n'arrive pas à trouver d'algorithme résolvant ce problème en temps polynomial, mais je peux vous garantir qu'aucune de ces personnes n'y est arrivée. 5
Une illustration de la hiérarchie de complexité sous les hypothèses $P \neq NP$ et $P = NP$. . 6

1.1 Illustration des différentes classes de graphes présentées : (*a*) une clique ; (*b*) un ensemble indépendant ; (*c*) un graphe biparti ; (*d*) un arbre ; (*e*) une forêt ; (*f*) les sommets encerclés représentent un vertex cover ; (*g*) les arêtes épaisses représentent un couplage ; (*h*) un graphe triangulé. 15
1.2 Un graphe et son arbre de décomposition modulaire associé. 16
1.3 Un graphe et son graphe des cliques critiques $\mathscr{C}(G)$. 17

2.1 Bornes inférieures pour l'existence de noyaux polynomiaux obtenues via la notion de cross-composition. 40
2.2 Le graphe Γ pour lequel le problème Γ-FREE EDGE DELETION n'admet pas de noyau polynomial [111]. 41
2.3 Illustration de la construction du graphe $H := (V_H, E_H)$ pour le problème C_l-FREE EDGE DELETION. 44

3.1 Un 3-leaf power $G := (V, E)$ et son 3-leaf root T. 59
3.2 Les sous-graphes induits interdits caractérisant la classe des 3-leaf powers. 60
3.3 Illustration de la notion de *join-composition* entre deux graphes G_1 et G_2. 61
3.4 (*i*) Une 1-branche dans une instance (G, k) de CLOSEST 3-LEAF POWER ; (*ii*) Une 2-branche dans une instance (G, k) de CLOSEST 3-LEAF POWER. 63
3.5 Le graphe $H' := G_2 \oplus ((G_3, A_1'') \otimes (G_1, \{A_1, \ldots, C_c\}))$ est un 3-leaf power par le Lemme 3.5. . 65
3.6 Illustration de la Règle 3.3. 65

3.7 Illustration du **Cas** 2 lorsque A_1 et A_2 appartiennent à différentes composantes connexes CC_1 et CC_2. 68
3.8 Illustration du cas (a). L'ensemble d'arêtes en gras correspond aux arêtes que nous allons supprimer. 70
3.9 Illustration de la Règle 3.5. 71
3.10 Illustration des propriétés de l'Observation 3.21. Les sommets noirs représentent les cliques critiques de $\mathscr{A}$, les sommets ronds et gris les cliques critiques de $\mathscr{A}^{\geqslant 3}$ et les sommets carrés les 1-branches réduites. 73

4.1 Les sous-graphes induits interdits caractérisant la classe des graphes d'intervalle propre. 79
4.2 Une illustration de la propriété parapluie (Théorème 4.4). L'arête $v_i v_j$ représentée est extrémale (Définition 4.5).[2] 80
4.3 (a) Un graphe d'intervalle propre $G := (V,E)$ ainsi que sa représentation associée. (b) La décomposition en cliques de G et son ordre parapluie associé σ_G. 81
4.4 Illustration de la notion de 1-branche pour le problème PROPER INTERVAL COMPLETION. 82
4.5 Illustration de la notion de 2-branche pour le problème PROPER INTERVAL COMPLETION. 83
4.6 Une décomposition en K-joins d'une 2-branche. 84
4.7 Les propriétés structurelles du K-join B dans G. 86
4.8 Illustration du cas (i). 87
4.9 Cas (a) : u et v sont distingués par un sommet $w <_{\sigma_H} u$. Cas (b) : u et v sont distingués par un sommet $w \in B_H$. 88
4.10 Illustration du nouvel ordre appliqué sur σ_H. Sur la gauche, les sommets gris représentent les sommets de U_2, et le sommet blanc est un sommet de U_1. 90
4.11 Les différentes configurations possibles pour $c_{h_{i-1}} <_{\sigma_H} c_{h_{j+1}}$. Cas ($a$) : $c_{h_i} <_{\sigma_H} c_{h_j}$. Cas ($b$) : $c_{h_j} \leqslant_{\sigma_H} c_{h_i}$. 91
4.12 Illustration de la construction de l'ordre σ à partir de l'ordre parapluie σ_H. 94
4.13 Illustration des différents cas de la configuration (iv). 95
4.14 Illustration de l'Application de la Règle 4.6. 97
4.15 Illustration du graphe d'intervalle propre H. Les sommets carrés représentent les sommets affectés par F 101
4.16 Les sous-graphes induits interdits caractérisant la classe des graphes bipartis-chaînés et bi-clique-chaînés. 102

5.1 Illustration de l'Affirmation 5.17. 113

6.1 Une partition sûre pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Les arcs non représentés sont des arcs *avant*, *i.e.* $v_i v_j$ avec $i < j$. 120

7.1 Illustration de la Définition 7.6 : les sommets 1, 2 et 3 permettent de certifier la famille d'arcs $\{1,2,3\}$. 130
7.2 Une partition sûre pour le problème FEEDBACK ARC SET IN TOURNAMENTS. Les arcs non représentés sont des arcs *avant*, *i.e.* $v_i v_j$ avec $i < j$. Ainsi, les arcs 1 et 3 peuvent être certifiés avec le même sommet, l'arc 2 étant certifié avec un sommet différent. 131
7.3 Illustration de la preuve de l'Affirmation 7.15. 134

7.4 Un exemple du problème DENSE ROOTED TRIPLET INCONSISTENCY : la collection $\mathscr{R}$ n'est pas consistente, mais le devient lorsque nous remplaçons le triplet t_1 par t_1'. 135
7.5 Illustration de la construction de la partition arborée $\mathscr{P}$. Les sommets noirs correspondent aux sommets de $V' \setminus D_2$. 142
7.6 Illustration de la preuve de l'Affirmation 7.32. 143

A.1 Construction du gadget de propagation pour les instances G_3 et G_6. 194
A.2 Illustration de la construction du graphe $H := (V_H, E_H)$ pour le problème C_l-FREE EDGE DELETION. 197

ANNEXE

A

Bornes inférieures pour C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION

Travail réalisé avec Sylvain GUILLEMOT, Frédéric HAVET et Christophe PAUL

Nous présentons maintenant des travaux réalisés avec Sylvain GUILLEMOT, Frédéric HAVET et Christophe PAUL, dans lesquels nous obtenons des bornes inférieures sur l'existence de noyaux polynomiaux pour le problème Γ-FREE EDGE-DELETION, où Γ dénote un graphe fini. Par le résultat de Cai [31] présenté dans la Section 2.1.2, nous savons que ce problème admet un algorithme FPT. Mais qu'en est-il de l'existence d'un algorithme de noyau polynomial ? De manière plus générale, le résultat de Cai a donné lieu à la conjecture suivante, également proposée par Cai [15].

Conjecture 6 ([15])**.** *Le problème $\mathcal{G}$-EDITION admet un noyau polynomial lorsque $\mathcal{G}$ est héréditaire et caractérisée par un ensemble fini de sous-graphes induits interdits.*

En 2009, Krastch et Wahlström ont démontré par la négative la Conjecture 6. Pour ce faire, ils ont prouvé qu'une version particulière de SAT n'admettait pas de noyau polynomial, et ont ensuite réduit ce problème via une transformation polynomiale en temps et paramètre au problème Γ-FREE EDGE-DELETION, où Γ est le graphe représente Figure 2.2.

Avec Sylvain GUILLEMOT et Christophe PAUL, nous avons adapté l'algorithme de composition de Kratsch et Wahlström au problème NOT-1-IN-3-EDGE-TRIANGLE, et utilisé ce résultat pour démontrer que les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION n'admettaient pas de noyau polynomial pour $l \geqslant 12$ et $l \geqslant 13$, respectivement. Avec l'aide de Frédéric HAVET, nous avons par la suite démontré que ce résultat pouvait être amélioré jusqu'à $l \geqslant 7$ pour les deux problèmes, résultats que nous présentons dans ce chapitre.

Théorème A.1. *Les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION n'admettent pas de noyau polynomial pour $l \geqslant 7$.*

Les problèmes considérés sont définis comme suit :

C_l-FREE EDGE DELETION :
Entrée : Un graphe $G := (V, E)$, $S \subseteq V$, deux entiers k et l.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est C_l-free ?

P_l-FREE EDGE DELETION :
Entrée : Un graphe $G := (V, E)$, $S \subseteq V$, deux entiers k et l.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est P_l-free ?

Pour cela, nous proposons des transformations polynomiales en temps et paramètre depuis une version graphique du problème NOT-1-IN-3-SAT, défini et prouvé ou-composable par Kratsch et Wahlström [111]. Formellement, nous définissons le problème suivant. Étant donnés un graphe $G := (V, E)$ et une arête-bicoloration partielle $B : E' \to \{0, 1\}$, avec $E' \subseteq E$, une arête-bicoloration $B' : E \to \{0, 1\}$ *étend* B lorsque $B'(uv) = B(uv)$ pour toute arête $uv \in E(G)$ colorée par B. Nous disons qu'une arête-bicoloration B' étendant B est *valide* lorsqu'aucun triangle $\{u, v, w\}$ de G possède *exactement* une arête colorée 1.

NOT-1-IN-3-EDGE-TRIANGLE :
Entrée : Un graphe $G := (V, E)$, une arête-bicoloration partielle $B : E' \to \{0, 1\}$, avec $E' \subseteq E$, un entier k.
Paramètre : k
Question : Existe-t-il une arête-bicoloration *valide* B', de poids au plus k, qui étend B ?

Lemme A.2. *Le problème* NOT-1-IN-3-EDGE-TRIANGLE *est NP-Complet.*

Preuve. Nous prouvons la NP-Complétude en construisant une réduction polynomiale depuis le problème VERTEX COVER. Soit $G := (V, E)$ une instance de VERTEX COVER. Nous construisons une instance (G', B', k') du problème NOT-1-IN-3-EDGE-TRIANGLE comme suit. Le graphe $G' := (V', E')$ est obtenu à partir de G en ajoutant un sommet dominant q, l'arête-bicoloration partielle B étant telle que $B(e) = 1$ pour tout $e \in E$. De plus, nous posons $k' := |E| + k$. Puisque les arêtes des triangles de G sont monochromatiques, les contraintes pour obtenir une extension valide de B sont portées par les arêtes des triangles $\{u, v, q\}$, avec $uv \in E$. Par construction, il suit que (G', B', k') a une arête-bicoloration valide étendant B de poids au plus k' si et seulement si G a un vertex cover de taille au plus k. □

Lemme A.3. *Le problème* NOT-1-IN-3-EDGE-TRIANGLE *est ou-composable.*

Preuve. Soient $(G_1, B_1, k), \ldots, (G_t, B_t, k)$ une séquence d'instances du problème, et $E_1(i)$ les 1-arêtes de G_i, $i \in [t]$. Nous prouvons tout d'abord un résultat technique qui nous permettra de supposer que chaque instance possède le même nombre de 1-arêtes.

Affirmation A.4. *Soient* (G, B, k) *une instance du problème* NOT-1-IN-3-EDGE-TRIANGLE *et* r *et* p *deux entiers tels que* $p \geqslant r + k$. *En temps polynomial, il est possible de construire une instance* (G', B', k') *équivalente à* G *et vérifiant* $\omega(B') = \omega(B) + r$.

Preuve. Nous construisons l'instance (G', B', k') comme suit. Premièrement, nous ajoutons à $E(G)$ un ensemble d'arêtes isolées $\{e_1, \ldots, e_r\}$ telle que $B'(e_i) = 1$ pour tout $i \in [r]$. Ensuite nous ajoutons $k' - (k + r)$ gadgets au graphe ainsi obtenu. Soit $f_j := u_j v_j$ une 1-arête arbitraire du graphe G : nous ajoutons deux sommets w_j et z_j tels que $\{u_j, v_j, w_j\}$ et $\{v_j, w_j, z_j\}$ induisent des triangles. Les arêtes f_j choisies pour cette construction ne sont pas nécessairement distinctes. Maintenant, observons que toute arête-bicoloration valide de G' étendant B' doit assigner la couleur 0 à l'arête $v_j w_j$ et la couleur 1 à l'arête $u_j w_j$. Il suit que (G, B, k) est une instance positive si et seulement si (G', B', k') est une instance positive vu que l'ensemble F incrémente le poids de l'arête-bicoloration de r et les gadgets ajoutés de $k' - (k + r)$, exactement. ⋄

Nous prouvons désormais que le problème NOT-1-IN-3-EDGE-TRIANGLE est ou-composable. Par l'Affirmation A.4, nous pouvons supposer que $|E_1(i)| = s \leqslant k$ pour tout $i \in [t]$. En effet, soient $l := \max_{i \in [t]} |E_1(i)|$ et $r := \max_{i \in [t]} (l - |E_1(i)|)$ (remarquons que $\omega(B_i) \leqslant k$ pour $i \in [t]$). En posant $k' := k + r$, nous obtenons des instances de paramètre k' possédant toutes le même nombre de 1-arêtes. De plus, nous pouvons également supposer que $t \leqslant 3^k$, sans quoi un algorithme de branchement exact résoudrait le problème en temps polynomial en $\sum_{i=1}^{t} |G_i| + k$, et permettrait donc d'obtenir une ou-composition triviale.

Dans un premier temps, nous décrivons informellement la composition réalisée. Le graphe G de l'instance composée est constitué de l'union disjointe des graphes G_i, $i \in [t]$. Ensuite, nous ajoutons un nouveau graphe G_T, dit de *sélection* et obtenu à partir d'un arbre binaire complet défini sur t feuilles. Le graphe G_T connecte une *arête racine* e_r à des arêtes e_i pour $i \in [t]$. Enfin, pour tout $i \in [t]$, les 1-arêtes du graphe G_i sont reliées à l'arête e_i du graphe de sélection via des *gadgets de propagation*. L'arête racine de T est l'unique 1-arête de G et les copies des graphes G_i héritent des arêtes colorées 0 dans B_i. L'idée de cette réduction est que le graphe de sélection garantit qu'au moins une arête e_i est colorée 1 dans toute arête-bicoloration valide. Le gadget de propagation relié à cette arête e_i transmettra quant à lui la couleur 1 aux copies des 1-arêtes de G_i, *activant* ainsi l'instance G_i.

Nous décrivons maintenant cette transformation de manière formelle. Le graphe G de l'instance réduite (G, B, k') est obtenu en prenant l'union disjointe des graphes G_i, $i \in [t]$. De plus, G comporte plusieurs gadgets, dits de *sélection* et de *propagation*. Le graphe de sélection G_T est obtenu à partir d'un arbre binaire complet T contenant t feuilles $s_1, \ldots, s_t$. A chaque noeud u de T, nous associons une arête e_u de G_T de la manière suivante : si u est associé à l'arête xy et si u a deux fils v et v', nous créons un nouveau sommet z dans G_T et posons $e_v := xz$, $e_{v'} := yz$. Les feuilles de T sont ainsi associées à des arêtes $e_1, \ldots, e_t$. Maintenant, pour toute arête $e_i \in E(G)$, $i \in [t]$, le gadget de propagation S_i est composé de graphes sommets-disjoints $S_{i,e}$, $e \in E_1(i)$, connectant l'arête e_i à e. Supposons $e := uv$ et $e_i := xy$. Alors le graphe $S_{i,e}$ est constitué de quatre triangles $\{u, v, a\}$, $\{v, a, b\}$, $\{a, b, x\}$ et $\{b, x, y\}$, les arêtes ua, vb, ax et by étant colorées 0 par B (les autres arêtes ne sont pas colorées par l'arête-bicoloration partielle B). Finalement, l'unique 1-arête de B est l'arête racine e_r de T. En particulier, les arêtes de $E_1(i)$, $i \in [t]$ ne sont pas colorées par B. Les arêtes colorées 0 par

B_i, $i \in [t]$ sont cependant préservées dans B (voir Figure A.1). Pour terminer la construction, nous posons $k' := k + 3s + l$.

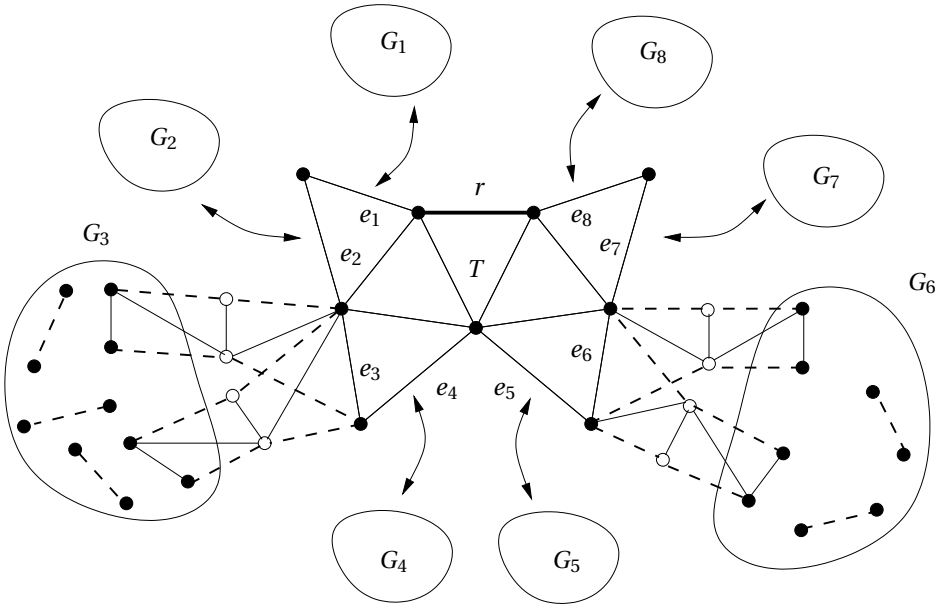


FIGURE A.1 : Construction du gadget de propagation pour les instances G_3 et G_6.

Affirmation A.5. *L'instance composée* (G, B, k') *est une instance positive si et seulement si il existe* $i \in [t]$ *tel que* (G_i, B_i, k) *est une instance positive.*

Preuve. Observons dans un premier temps que tout arête-bicoloration valide étendant B doit assigner la couleur 1 à *au moins* une arête e_i, $i \in [t]$, et aux l arêtes e_u correspondant aux sommets u du chemin reliant r à s_i dans T. De ce fait, les arêtes de $E_1(j)$ et les $3k$ arêtes non colorées de S_i doivent également recevoir la couleur 1. Cela implique en particulier que les 1-arêtes de G_i recoivent également la couleur 1, et donc que G_i admet une arête-bicoloration valide de poids inférieur ou égal à k. De manière similaire, pour tout $i \in [t]$ tel que G_i est positive, il est possible de construire une arête- bicoloration étendant B de telle sorte que les seules 1-arêtes de $G \setminus G_i$ soient les arêtes correspondant au chemin reliant r à s_i dans T, ainsi qu'aux arêtes non colorées de S_i. Ainsi, (G, B, k') est une instance positive si et seulement si il existe $i \in [t]$ tel que (G_i, B_i, k) est positive. ◇

Cela prouve donc la validité de l'algorithme de composition. Observons finalement que le temps d'exécution de cet algorithme est bien polynomial en $\Sigma_{i=1}^{t}|G_i| + \mathrm{k}$. □

Par le Corollaire 2.27, nous obtenons le résultat suivant.

Corollaire A.6. *Le problème* NOT-1-IN-3-EDGE-TRIANGLE *n'admet pas de noyau polynomial, à moins que* $NP \subseteq coNP/poly$.

Le problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE est la restriction du problème NOT-1-IN-3-EDGE-TRIANGLE où le graphe $G := (V,E)$ est triparti, *i.e.* V peut être partitionné en 3 ensembles indépendants V_1, V_2, V_3. Les résultats de complexité obtenus pour le problème NOT-1-IN-3-EDGE-TRIANGLE peuvent être appliqués à cette restriction, comme l'indique le résultat suivant.

Proposition A.7. *Le problème* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE *est NP-Complet et n'admet pas de noyau polynomial, sauf si* $NP \subseteq coNP/poly$.

Preuve. Nous proposons une réduction polynomiale en temps et paramètre depuis le problème NOT-1-IN-3-EDGE-TRIANGLE. La NP-Complétude et l'algorithme de Ou-composition pour TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE suivront directement de cette réduction (remarquons que le problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE appartient clairement à NP). Soit (G,B,k) une instance de NOT-1-IN-3-EDGE-TRIANGLE. Nous construisons une instance $(G',B',6k)$ de TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE comme suit. Suivant les notations classiques, nous supposons $G := (V,E)$. Le graphe $G' := (V',E')$ est obtenu en posant $V' := \{v_1, v_2, v_3 \ : \ v \in V\}$ et $E' := \{u_1v_2, u_1v_3, u_2v_3 \ : \ u = v \text{ ou } uv \in E\}$. L'arête-bicoloration partielle B' est obtenue en posant $B'(u_iu_j) = 0$ pour tout $1 \leqslant i < j \leqslant 3$. De plus, si l'arête uv de G est colorée par B, alors $B'(u_iv_j) = B(uv)$ pour tout $1 \leqslant i < j \leqslant 3$. Les autres arêtes de E' ne sont pas colorées par B'.

Affirmation A.8. *L'instance* (G,B,k) *est positive si et seulement si* $(G',B',6k)$ *est positive.*

Preuve. Observons que tout arête-bicoloration valide étendant B' colore de la même façon les six arêtes de G' associées avec la même arête uv de G. En effet, étant donnés u_iv_j, u_kv_l, $1 \leqslant i,j,k,l \leqslant 3$, si $i = j$ alors le résultat est vérifié car $B'(v_kv_l) = 0$; si $k = l$ alors le résultat est vérifié car $B'(u_iu_j) = 0$; et dans tous les autres cas le résultat suit par transitivité. ◇

Finalement, observons que la réduction peut s'effectuer en temps polynomial, ce qui conclut la preuve de la Proposition A.7. □

Nous décrivons maintenant les transformations polynomiales en temps et paramètre depuis le problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE. Pour des raisons techniques, le problème ne sera pas directement réduit vers les problèmes C_l-FREE EDGE DELETION et P_l-FREE EDGE DELETION mais vers une version *annotée* des problèmes, définie comme suit :

ANNOTATED Γ-FREE EDGE-DELETION :
Entrée : Un graphe $G := (V,E)$, $S \subseteq V$, un entier k.
Paramètre : k
Question : Existe-t-il un ensemble $F \subseteq E \cap (S \times S)$ de taille au plus k tel que le graphe $H := (V, E \setminus F)$ est Γ-free ?

Nous prouvons que la version annotée d'un problème se réduit vers sa version classique sous certaines hypothèses. Nous verrons plus tard que ces hypothèses sont vérifiées par les problèmes considérés, et ce résultat nous permettra donc de conclure nos preuves.

Observation A.9. *Soit* $\mathcal{G}$ *une classe de graphes héréditaire et close par ajout de vrai (resp. faux) jumeau. Il existe une transformation polynomiale en temps et paramètre du problème* ANNOTATED $\mathcal{G}$-EDGE-DELETION *vers le problème* $\mathcal{G}$-EDGE-DELETION.

Preuve. Soit (G,k) une instance du problème ANNOTATED $\mathscr{G}$-EDGE-DELETION. Nous supposons dans un premier temps que $\mathscr{G}$ est close par ajout de vrai jumeau. Nous construisons une instance (G',k') du problème $\mathscr{G}$-EDGE-DELETION comme suit : tout sommet u de $V(G)\setminus S$ est remplacé par une clique de taille $k+1$ ayant le même voisinage que u. Soit F une k-suppression de G. Par définition, $F \subseteq E(G)\cap(S\times S)$, et ainsi le graphe $G'-F$ est obtenu à partir de $G-F$ en ajoutant des vrais jumeaux aux sommets de $V(G)\setminus S$. Il suit que $G'-F$ appartient à $\mathscr{G}$ et ainsi F est une k-suppression de G'. Inversement, soit F' une k-suppression de G'. Puisque $\mathscr{G}$ est close par ajout de vrai jumeau et héréditaire, nous pouvons supposer (Lemme 2.49) que F' supprime toute ou aucune arête entre deux cliques critiques. Il suit que les cliques de G' correspondant aux sommets de $V(G)\setminus S$ ne sont pas affectées par F', et ainsi $F' \subseteq E(G')\cap(S\times S)$. Cela implique que F' est une k-suppression de G.

Dans le cas où $\mathscr{G}$ est close par ajout de faux jumeau, la preuve est similaire et consiste à remplacer les sommets de $V\setminus S$ par un ensemble indépendant de taille $k+1$. □

Nous démontrons maintenant les principaux résultats de ce chapitre.

Théorème A.10. *Le problème C_l-*FREE EDGE DELETION *n'admet pas de noyau polynomial pour tout $l \geqslant 7$, sauf si $NP \subseteq coNP/poly$.*

Preuve. Nous prouvons tout d'abord un résultat technique similaire à un résultat démontré par Kratsch et Wahlström pour le problème NOT-1-IN-3-SAT.

Affirmation A.11. *Soit (G,B,k) une instance de* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE. *Une instance équivalente (G',B',k) telle que toute arête $uv \in E(G')$ colorée par B' vérifie $B'(uv)=1$ peut être créée en temps polynomial.*

Nous décrivons une transformation polynomiale en temps et paramètre depuis la restriction du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE sans 0-arêtes vers le problème ANNOTATED C_l-FREE EDGE-DELETION. Le résultat suit alors du Corollaire 2.27 et du fait que ANNOTATED C_l-FREE EDGE-DELETION se réduise à C_l-FREE EDGE DELETION par l'Observation A.9 (puisque les graphes sans C_l sont clos par ajout de vrai jumeau pour $l \geqslant 4$).

Soit (G,B,k) une instance du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE, où V_1, V_2 et V_3 dénotent les trois ensembles indépendants de $G := (V,E)$. La construction de l'instance (H,S,k') de ANNOTATED C_l-FREE EDGE-DELETION est la suivante : premièrement, les ensembles V_1, V_2 et V_3 sont transformés en cliques et les 1-arêtes de G sont supprimées. En plus de V, le graphe H contient un ensemble U de nouveaux sommets. Pour toute paire $t := (e,v)$, avec $e := uw \in E(G)$ et $v \in V(G)$ tels que $\{u,v,w\}$ induit un triangle, nous créons dans H un chemin $P_t := \{p_t^1,\ldots,p_t^{l-3}\}$ de longueur $l-3$. Les sommets du chemin P_t sont ajoutés à l'ensemble de sommets U. Remarquons que chaque triangle de G génère trois chemins de la sorte. Pour terminer la construction, nous ajoutons des *arêtes de sécurité* incidentes aux sommets de U.

- pour tous sommets x et y n'appartenant pas au même chemin, xy est une arête de H.
- Dans tout chemin P_t, le sommet p_t^1 (resp. p_t^{l-3}) est adjacent à $V\setminus\{v,w\}$ (resp. $V\setminus\{u,v\}$).

Nous dénotons par $H := (V_H,E_H)$ le graphe résultant de cette réduction (voir Figure A.2). Pour terminer la description de l'instance (H,S), nous posons $S := V$ et $k' := k-k_1$, où k_1 correspond au nombre de 1-arêtes de G. L'intuition de cette construction est que tout cycle de longueur l dans H correspondra à un triangle de G ne contenant qu'une seule 1-arête. Comme ces triangles ne

peuvent plus exister dans une arête-bicoloration valide B' de G, une arête-suppression pour C_l pourra ainsi être déduite de B'. Formellement, nous obtenons les résultats suivants.

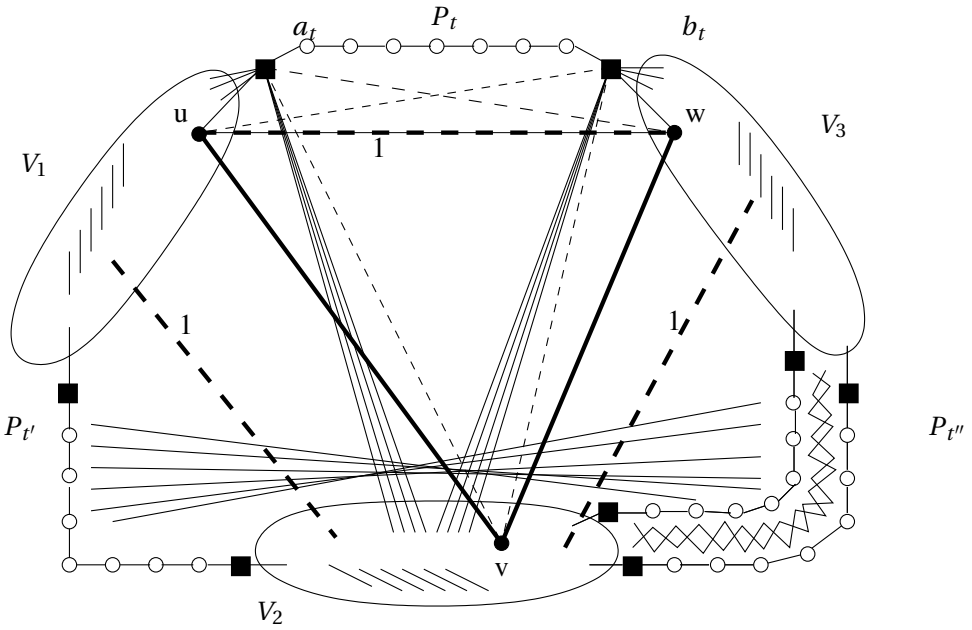


FIGURE A.2 : Illustration de la construction du graphe $H := (V_H, E_H)$ pour le problème C_l-FREE EDGE DELETION.

Affirmation A.12. *Un ensemble de sommets $C \in V_H$ induit un cycle de longueur l si et seulement si G contient un triangle $\{u, v, w\}$, avec $e := uw$ une 1-arête et uv, vw deux arêtes non colorées, telles que $C := P_t \cup \{v\}$, $t := (e, v)$.*

Preuve. Par construction, si G contient un triangle $\{u, v, w\}$ avec une unique 1-arête $e := uw$, alors $C := P_t \cup \{v\}$ (avec $t := (e, v)$) induit un cycle de longueur l dans H (rappelons que les 1-arêtes de G ont été supprimées dans H). Inversement, soit C un cycle induit de longueur l dans H. Puisque V_1, V_2 et V_3 sont des cliques de H, il suit que $|C \cap V| \leqslant 6$, et donc C doit intersecter l'ensemble de sommets U. Observons de plus que C doit également intersecter V : en effet, dans le cas contraire, nous aurions $|C| \leqslant 4$ (rappelons que les sommets de chemins distincts sont adjacents). Nous prouvons par contradiction que $C \cap U$ est contenu dans un *unique* chemin P_t. Dans un premier temps, remarquons que $C \cap U$ ne peut intersecter trois chemins $P_t, P_{t'}, P_{t''}$, puisque cela impliquerait l'existence de trois sommets induisant un C_3. Supposons maintenant que $C \cap U$ intersecte deux chemins distincts P_t, $P_{t'}$ ($t \neq t'$). Si $C \cap U$ contient deux sommets de P_t et deux sommets de $P_{t'}$, alors ces quatre sommets induisent un C_4, ce qui est impossible. De manière similaire, si $C \cap U$ contient trois sommets de P_t et un sommet de $P_{t'}$, alors ce sommet aurait degré au moins 3 dans C, ce qui est impossible (tout sommet d'un cycle a degré exactement 2). Il suit que $|C \cap P_t| \leqslant 2$ et $|C \cap P_{t'}| = 1$. Puisque $C \cap V \neq \emptyset$, les sommets de $C \cap P_t$ correspondent aux extrémités de P_t. Maintenant, par définition de C, un sommet de $C \cap P_t$ est non-incident à au moins $l - 3 \geqslant 3$ sommets de $C \cap V$, et ne peut donc correspondre ni à p_t^1 ni à p_t^{l-3} par définition des adjacences de ces sommets :

contradiction. Ainsi, il existe un chemin P_t, avec $t := (e, v)$ et $e := uv$, contenant les sommets de $C \cap U$. Deux cas sont alors possibles : (i) $C \cap P_t \subseteq \{p_t^1, p_t^{l-3}\}$ ou (ii) $P_t \subseteq C$. Dans le premier cas, p_t^1 ou p_t^{l-3} est non-adjacent à au moins $l-3 \geqslant 3$ sommets de $C \cap V$, ce qui contredit la définition des adjacences de ces sommets. Il suit que (ii) est vérifié, et C est constitué des sommets de P_t plus trois sommets $x, y, z \in V$, où x est (uniquement) adjacent à p_t^1 dans C, z adjacent uniquement à p_t^{l-3} et y non-adjacent à p_t^1 et p_t^{l-3}. Puisque v est le seul sommet de V non incident à la fois à p_t^1 et p_t^{l-3}, il suit que $y = v$. Cela implique que $x = u$ et $z = w$ (puisque ce sont les seuls sommets non-incidents à p_t^{l-3} et p_t^1, respectivement). Finalement, l'existence du chemin P_t implique l'existence du triangle $\{u, v, w\}$ dans G. Comme $uw, wv \in E_H$ et $uw \notin E_H$, uw est l'unique 1-arête du triangle $\{u, v, w\}$ dans G. Ceci conclut la preuve de l'Affirmation A.12. ◇

Affirmation A.13. *L'instance (G, B, k) est une instance positive de* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE *si et seulement si l'instance (H, S, k') est une instance positive de C_l-*FREE EDGE DELETION.

Preuve. Supposons qu'il existe un ensemble F d'au plus k' arêtes autorisées tel que $H' := (V_H, E_H \setminus F)$ est C_l-free. Nous définissons l'arête-bicoloration de E comme suit : pour toute arête $e \in E_H$, $B'(e) = 1$ si $e \in F$, et $B'(e) = 0$ sinon. De plus, puisque toute arête $e \in E(G) \setminus E_H$ est une 1-arête, nous posons $B'(e) = 1$ pour de telles arêtes. Comme B n'assigne aucune 0-arête par hypothèse, il suit que B' étend B et a un poids d'au plus $|F| + k_1 \leqslant k' + k_1 = k$. Nous prouvons maintenant que B' est une arête-bicoloration valide de G. Soit $t := (e, v)$, avec $e := uw \in E(G)$, une paire telle que $\{u, v, w\}$ induit un triangle de G. Supposons par contradiction et sans perte de généralité que $B(uw) = 1$, $B'(uv) = B'(vw) = 0$. Dans ce cas, $P_t \cup \{v\}$ induit un cycle de longueur dans H', ce qui est impossible. Inversement, supposons que B' soit une arête-bicoloration valide de G, de poids au plus k, qui étend B. Soit $F \subseteq E(G)$ l'ensemble d'arêtes non-colorées par B telles que $B'(e) = 1$. Par construction, F est un ensemble d'arêtes autorisées de H et a taille au plus $k - k_1$ (rappelons que les 1-arêtes de G ne sont pas ajoutées dans F). Comme B' est une arête-bicoloration valide de G, l'Affirmation A.12 implique que $H' := (V_H, E_H \setminus F)$ ne contient pas de cycle induit de longueur l. ◇

Cela conclut la preuve du Théorème A.10. □

En modifiant légèrement la construction décrite précédemment, nous obtenons le résultat suivant.

Théorème A.14. *Le problème P_l-*FREE EDGE DELETION *n'admet pas de noyau polynomial pour tout $l \geqslant 7$, sauf si $NP \subseteq coNP/poly$.*

Preuve. Nous décrivons une transformation polynomiale en temps et paramètre depuis la restriction du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE sans 0-arêtes vers le problème ANNOTATED P_l-FREE EDGE-DELETION. Le résultat suit alors du Corollaire 2.27 et du fait que ANNOTATED C_l-FREE EDGE-DELETION se réduise à P_l-FREE EDGE DELETION par l'Observation A.9 (puisque les graphes sans P_l sont clos par ajout de vrai jumeau pour $l \geqslant 3$).

Soit (G, B, k) une instance du problème TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE, où V_1, V_2 et V_3 dénotent les trois ensembles indépendants de $G := (V, E)$. Nous modifions la construction présentée dans la preuve du Théorème A.10 pour obtenir une instance (H, S, k') du problème ANNOTATED P_l-FREE EDGE-DELETION : ici encore, les ensembles V_1, V_2 et V_3 sont transformés en cliques

et les 1-arêtes de G sont supprimées. En plus de V, le graphe H contient un ensemble U de nouveaux sommets. Dans ce cas, pour toute paire $t := (e, v)$, avec $e := uw \in E(G)$ et $v \in V(G)$ tels que $\{u, v, w\}$ induit un triangle, le gadget associé P_t ne consiste plus en un chemin de longueur $l-1$. A la place, P_t est composé de deux chemins distincts P_t^u et P_t^w contenant u et w comme extrémités et dont les longueurs somment à $l-3$ (observons que cela est possible car $l \geqslant 7$). Nous considérons maintenant p_u et p_w deux extrémités de P_t^u et P_t^w, respectivement. Les sommets internes de P_t sont ajoutés à l'ensemble de sommets U. Pour terminer la construction, nous ajoutons des *arêtes de sécurité* incidentes aux sommets de U.

- pour tous sommets x et y n'appartenant pas au même gadget, xy est une arête de H.
- le sommet p_u est adjacent à $V \setminus \{v, w\}$, et le sommet p_w est adjacent à $V \setminus \{u, v\}$.

Nous dénotons par $H := (V_H, E_H)$ le graphe résultant de cette réduction. Pour terminer la description de l'instance (H, S), nous posons $S := V$ et $k' = k - k_1$, où k_1 correspond au nombre de 1-arêtes de G. Le résultat structurel sur lequel cette validation s'appuie est désormais le suivant, la validité de la transformation étant elle similaire à la preuve du Théorème A.10.

Affirmation A.15. *Un ensemble de sommets $P \in V_H$ induit un chemin de longueur l si et seulement si G contient un triangle $\{u, v, w\}$, avec $e := uw$ une 1-arête et uv, vw deux arêtes non colorées, telles que $P := P_t \cup \{v\}$, $t := (e, v)$.*

Preuve. Par construction, si G contient un triangle $\{u, v, w\}$ avec une unique 1-arête $e := uw$, alors $P := P_t \cup \{v\}$ (avec $t := (e, v)$) induit un chemin de longueur l dans h (rappelons que les 1-arêtes de G ont été supprimées dans H). Inversement, soit P un chemin induit de longueur l dans H. Comme dans la preuve de l'Affirmation A.13, remarquons que $|P \cap V| \leqslant 6$, et donc que P intersecte U et qu'il existe une unique paire $t := (e, v)$, avec $e := uw$, telle que P_t contient $P \cap U$. Nous avons alors plusieurs cas à considérer :

$$P \cap P_t \subseteq \{p_u, p_w\} \tag{A.1}$$

$$|P \cap P_t^u| \geqslant 2 \tag{A.2}$$

$$|P \cap P_t^w| \geqslant 2 \tag{A.3}$$

Si (A.1) est vérifié, alors p_u ou p_w est non-adjacent à au moins $l - 4 \geqslant 3$ sommets de V, ce qui contredit la définition des adjacences de ces sommets. Maintenant, si (A.2) est vérifié, alors p_u est adjacent à au plus un sommet de V dans P. Comme p_u est non-adjacent à seulement deux sommets de V, il suit que $|P \cap V| \leqslant 3$. Nous obtenons la même observation lorsque (A.3) est vérifié, en considérant p_w au lieu de p_u. Il suit que $|P \cap P_t| \geqslant l - 3$, et par construction cela est une égalité. Il suit que P est composé des sommets de P_t^u suivis par trois sommets $x, y, z \in V$, suivis par les sommets de P_t^w. Comme dans la preuve de l'Affirmation A.13, nous avons alors $x = u$, $y = v$ et $z = w$. Finalement, l'existence de P_t implique l'existence du triangle $\{u, v, w\}$ dans G. De plus, l'arête uw ne peut pas exister dans H, impliquant que uw est une 1-arête de (G, B, k). ◇

La preuve de l'affirmation suivante est similaire à la preuve de l'Affirmation A.13.

Affirmation A.16. *L'instance (G, B, k) est une instance positive de* TRIPARTITE-NOT-1-IN-3-EDGE-TRIANGLE *si et seulement si l'instance (H, S, k') est une instance positive de P_l-*FREE EDGE DELETION.

□

ANNEXE

B

Noyaux pour FEEDBACK ARC SET IN TOURNAMENTS

Travail réalisé en collaboration avec Stéphane BESSY, Fedor V. FOMIN et Serge GASPERS, Christophe PAUL, Saket SAURABH et Stéphan THOMASSÉ

Un tournoi $T := (V, A)$ est un graphe orienté contenant **exactement** un arc entre toute paire de sommets distincts. Étant donnés un graphe orienté à n sommets et un entier k, le problème FEEDBACK ARC SET consiste à décider s'il existe un ensemble de k arcs dont la suppression permet d'obtenir un **graphe orienté acyclique**. Ce problème restreint à la classe des tournois est appelé FEEDBACK ARC SET IN TOURNAMENTS. Dans cet article, nous obtenons un noyau avec un nombre linéaire de sommets pour FEEDBACK ARC SET IN TOURNAMENTS. En d'autres termes, nous donnons un algorithme polynomial qui, étant donnée une instance T de FEEDBACK ARC SET IN TOURNAMENTS, calcule une instance équivalente T' contenant $O(k)$ sommets. En fait, pour tout $\epsilon > 0$, l'instance retournée par l'algorithme de noyau possède au plus $(2+\epsilon)k$s sommets. Notre résultat améliore la précédente borne de $O(k^2)$ sommets connue pour ce problème [5]. Notre algorithme de noyau résoud le problème en temps polynomial sur une sous-classe des tournois, et utilise un algorithme d'approximation à facteur constant pour FEEDBACK ARC SET IN TOURNAMENTS [106].

Kernels for Feedback Arc Set In Tournaments

Stéphane Bessy Fedor V. Fomin Serge Gaspers Christophe Paul
Anthony Perez Saket Saurabh Stéphan Thomassé

July 1, 2011

Abstract

A tournament $T = (V, A)$ is a directed graph in which there is exactly one arc between every pair of distinct vertices. Given a digraph on n vertices and an integer parameter k, the Feedback Arc Set problem asks whether the given digraph has a set of k arcs whose removal results in an acyclic digraph. The Feedback Arc Set problem restricted to tournaments is known as the k-Feedback Arc Set in Tournaments (k-FAST) problem. In this paper we obtain a linear vertex kernel for k-FAST. That is, we give a polynomial time algorithm which given an input instance T to k-FAST obtains an equivalent instance T' on $O(k)$ vertices. In fact, given any fixed $\epsilon > 0$, the kernelized instance has at most $(2+\epsilon)k$ vertices. Our result improves the previous known bound of $O(k^2)$ on the kernel size for k-FAST. Our kernelization algorithm solves the problem on a subclass of tournaments in polynomial time and uses a known polynomial time approximation scheme for k-FAST.

1 Introduction

Given a directed graph $G = (V, A)$ on n vertices and an integer parameter k, the Feedback Arc Set problem asks whether the given digraph has a set of k arcs whose removal results in an acyclic directed graph. In this paper, we consider this problem in a special class of directed graphs, *tournaments*. A tournament $T = (V, A)$ is a directed graph in which there is exactly one directed arc between every pair of vertices. More formally the problem we consider is defined as follows.

> k-Feedback Arc Set in Tournaments (k-FAST): Given a tournament $T = (V, A)$ and a positive integer k, does there exist a subset $F \subseteq A$ of at most k arcs whose removal makes T acyclic.

In the weighted version of k-FAST, we are also given integer weights (each weight is at least one) on the arcs and the objective is to find a feedback arc set of weight at most k. This problem is called k-Weighted Feedback Arc Set in Tournaments (k-WFAST).

Feedback arc sets in tournaments are well studied from the combinatorial [20, 22, 29, 30, 33, 37], statistical [31] and algorithmic [2, 3, 14, 26, 35, 36] points of view. The problems k-FAST and k-WFAST have several applications. In *rank aggregation* we are given several rankings of a set of objects, and we wish to produce a single ranking that on average is as consistent as possible with the given ones, according to some chosen measure of consistency. This problem has been studied in the context of voting [8, 11, 13], machine learning [12], and search engine ranking [18, 19]. A natural consistency measure for rank aggregation is the number of pairs that occur in a different order in the two rankings. This leads to *Kemeny rank aggregation* [24, 25], a special case of k-WFAST.

The k-FAST problem is known to be NP-complete by recent results of Alon [3] and Charbit *et al.* [10] while k-WFAST is known to be NP-complete by Bartholdi III *et al.* [5]. From an approximation perspective, k-WFAST is APX-hard [32] but admits a polynomial time approximation scheme when the edge weights are bounded by a constant [26]. The problem is also well studied in parameterized complexity. In this

area, a problem with input size n and a parameter k is said to be fixed parameter tractable (FPT) if there exists an algorithm to solve this problem in time $f(k) \cdot n^{O(1)}$, where f is an arbitrary function of k. Raman and Saurabh [28] showed that k-FAST and k-WFAST are FPT by obtaining an algorithm running in time $O(2.415^k \cdot k^{4.752} + n^{O(1)})$. Recently, Alon *et al.* [4] have improved this result by giving an algorithm for k-WFAST running in time $O(2^{O(\sqrt{k}\log^2 k)} + n^{O(1)})$. This algorithm runs in sub-exponential time, a trait uncommon to parameterized algorithms. Moreover, a new algorithm due to Karpinsky and Schudy [23] with running time $O(2^{O(\sqrt{k})} + n^{O(1)})$ improves again the complexity of k-WFAST. Finally, Fomin *et al.* [21] provided a sub-exponential local search algorithm for k-WFAST. In this paper we investigate k-FAST from the view point of kernelization, currently one of the most active subfields of parameterized algorithms.

A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial (in n) time algorithm, called a *kernelization* algorithm, that reduces the input instance to an instance whose size is bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. When $p(k)$ is a linear function of k then the corresponding kernel is a linear kernel. Kernelization has been at the forefront of research in parameterized complexity in the last couple of years, leading to various new polynomial kernels as well as tools to show that several problems do not have a polynomial kernel under some complexity-theoretic assumptions [6, 7, 9, 15, 17, 34]. In this paper we continue the current theme of research on kernelization and obtain a *linear vertex* kernel for k-FAST. That is, we give a polynomial time algorithm which given an input instance T to k-FAST obtains an equivalent instance T' on $O(k)$ vertices. More precisely, given any fixed $\epsilon > 0$, we find a kernel with a most $(2+\epsilon)k$ vertices in polynomial time. The reason we call it a linear *vertex* kernel is that, even though the number of vertices in the reduced instance is at most $O(k)$, the number of arcs is still $O(k^2)$. Our result improves the previous known bound of $O(k^2)$ on the vertex kernel size for k-FAST [4, 16]. For our kernelization algorithm we find a subclass of tournaments where one can find a minimum sized feedback arc set in polynomial time (see Lemma 3.8) and use the known polynomial time approximation scheme for k-FAST by Kenyon-Mathieu and Schudy [26]. The polynomial time algorithm for a subclass of tournaments could be of independent interest.

The paper is organized as follows. In Section 2, we give some definition and preliminary results regarding feedback arc sets. In Section 3 we give a linear vertex kernel for k-FAST. Finally we conclude with some remarks in Section 4.

2 Preliminaries

Let $T = (V, A)$ be a tournament on n vertices. We use $T_\sigma = (V_\sigma, A)$ to denote a tournament whose vertices are ordered under a fixed ordering $\sigma = v_1, \ldots, v_n$ (we also use D_σ for an ordered directed graph). We say that an arc v_iv_j of T_σ is a *backward arc* if $i > j$, otherwise we call it a *forward arc*. Moreover, given any partition $\mathcal{P} := \{V_1, \ldots, V_l\}$ of V_σ, where every V_i is an interval according to the ordering of T_σ, we use A_B to denote all arcs between the intervals (having their endpoints in different intervals), and A_I for all arcs within the intervals. If T_σ contains no backward arc, then we say that it is *transitive*.

For a vertex $v \in V$ we denote its *in-neighborhood* by $N^-(v) := \{u \in V \mid uv \in A\}$ and its *out-neighborhood* by $N^+(v) := \{u \in V \mid vu \in A\}$. A set of vertices $M \subseteq V$ is a *module* if and only if $N^+(u) \setminus M = N^+(v) \setminus M$ for every $u, v \in M$. For a subset of arcs $A' \subseteq A$, we define $T[A']$ to be the digraph (V', A') where V' is the union of endpoints of the arcs in A'. Given an ordered digraph D_σ and an arc $e = v_iv_j$, $S(e) = \{v_i, \ldots, v_j\}$ denotes the *span* of e. The number of vertices in $S(e)$ is called the *length* of e and is denoted by $l(e)$. Thus, for every arc $e = v_iv_j$, $l(e) = |i - j| + 1$. Finally, for every vertex v in the span of e, we say that e is *above* v.

In this paper, we will use the well-known fact that every acyclic tournament admits a transitive ordering. In particular, we will consider *maximal transitive modules*. We also need the following result for our kernelization algorithm.

Lemma 2.1. *([28]) Let $D = (V, A)$ be a directed graph and F be a minimal feedback arc set of D. Let D' be the graph obtained from D by reversing the arcs of F in D, then D' is acyclic.*

We now introduce a definition which is useful for a lemma we prove later.

Definition 2.2. *Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph and let $f = vu$ be a backward arc of D_σ. We call* certificate *of f, and denote it by $c(f)$, any directed path from u to v using only forward arcs in the span of f in D_σ.*

Observe that such a directed path together with the backward arc f forms a directed cycle in D_σ whose only backward arc is f.

Definition 2.3. *Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph, and let $F \subseteq A$ be a set of backward arcs of D_σ. We say that we can* certify *F whenever it is possible to find a set $\mathcal{F} = \{c(f) : f \in F\}$ of arc-disjoint certificates for the arcs in F.*

Let $D_\sigma = (V_\sigma, A)$ be an ordered directed graph, and let $F \subseteq A$ be a subset of backward arcs of D_σ. We say that we can certify the set F using *only arcs from* $A' \subseteq A$ if F can be certified by a collection $\mathcal{F}$ such that the union of the arcs of the certificates in $\mathcal{F}$ is contained in A'. In the following, $fas(D)$ denotes the *size* of a minimum feedback arc set, that is, the cardinality of a minimum sized set F of arcs whose removal makes D acyclic.

Lemma 2.4. *Let D_σ be an ordered directed graph, and let $\mathcal{P} = \{V_1, \ldots, V_l\}$ be a partition of D_σ into intervals. Assume that the set F of all backward arcs of $D_\sigma[A_B]$ can be certified using only arcs from A_B. Then $fas(D_\sigma) = fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. Moreover, there exists a minimum sized feedback arc set of D_σ containing F.*

Proof. For any bipartition of the arc set A into A_1 and A_2, $fas(D_\sigma) \geq fas(D_\sigma[A_1]) + fas(D_\sigma[A_2])$. Hence, in particular for a partition of the arc set A into A_I and A_B we have that $fas(D_\sigma) \geq fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. Next, we show that $fas(D_\sigma) \leq fas(D_\sigma[A_I]) + fas(D_\sigma[A_B])$. This follows from the fact that once we reverse all the arcs in F, each remaining directed cycle lies in $D_\sigma[V_i]$ for some $i \in \{1, \ldots, l\}$. In other words once we reverse all the arcs in F, every cycle is completely contained in $D_\sigma[A_I]$. This concludes the proof of the first part of the lemma. In fact, what we have shown is that there exists a minimum sized feedback arc set of D_σ containing F. This concludes the proof of the lemma. □

3 Kernels for k-FAST

In this section we first give a subquadratic vertex kernel of size $O(k\sqrt{k})$ for k-FAST and then improve on it to get our final vertex kernel of size $O(k)$. We start by giving a few reduction rules that will be needed to bound the size of the kernels.

Rule 3.1. *If a vertex v is not contained in any triangle, delete v from T.*

Rule 3.2. *If there exists an arc uv that belongs to more than k distinct triangles, then reverse uv and decrease k by* 1.

We say that a reduction rule is *sound*, if whenever the rule is applied to an instance (T, k) to obtain an instance (T', k'), T has a feedback arc set of size at most k if and only if T' has a feedback arc set of size at most k'. Moreover, *applying* a reduction rule in polynomial time means that the structure sought by the reduction rule can be identified in polynomial time and the instance can be updated in polynomial time. Finally, we say that an instance (T, k) is *reduced* according to a set of reduction rules whenever none of the reduction rules can be applied to (T, k).

Lemma 3.1. *([4, 16]) Rules 3.1 and 3.2 are sound and can be applied in polynomial time.*

The Rules 3.1 and 3.2 together led to a quadratic kernel for k-WFAST [4]. Earlier, these rules were used by Dom *et al.* [16] to obtain a quadratic kernel for k-FAST. We now add a new reduction rule that will allow us to obtain the claimed bound on the kernel sizes for k-FAST. Given an ordered tournament $T_\sigma = (V_\sigma, A)$, we say that $\mathcal{P} = \{V_1, \ldots, V_l\}$ is a *safe partition* of V_σ into intervals whenever it is possible to certify the backward arcs of $T_\sigma[A_B]$ using only arcs from A_B.

Rule 3.3. *Let T_σ be an ordered tournament, and $\mathcal{P} = \{V_1, \ldots, V_l\}$ be a safe partition of V_σ into intervals such that $F \neq \varnothing$, where F denotes the set of backward arcs of $T_\sigma[A_B]$. Then reverse all the arcs of F and decrease k by $|F|$.*

Lemma 3.2. *Rule 3.3 is sound.*

Proof. Let $\mathcal{P}$ be a safe partition of T_σ. Observe that it is possible to certify all the backward arcs, that is F, using only arcs in A_B. Hence using Lemma 2.4 we have that $fas(T_\sigma) = fas(T_\sigma[A_I]) + fas(T_\sigma[A_B])$. Furthermore, by Lemma 2.4 we also know that there exists a minimum sized feedback arc set of D_σ containing F. Thus, T_σ has a feedback arc set of size at most k if and only if the tournament T'_σ obtained from T_σ by reversing all the arcs of F has a feedback arc set of size at most $k - |F|$. □

3.1 A subquadratic kernel for k-FAST

In this section, we show how to obtain an $O(k\sqrt{k})$ sized vertex kernel for k-FAST. To do so, we introduce the following reduction rule.

Rule 3.4. *Let T_m be a* maximal transitive module *of size p, and I and O be the set of in-neighbors and out-neighbors of the vertices of T_m in T, respectively. Let Z be the set of arcs uv such that $u \in O$ and $v \in I$. If $q = |Z| < p$ then reverse all the arcs in Z and decrease k by q.*

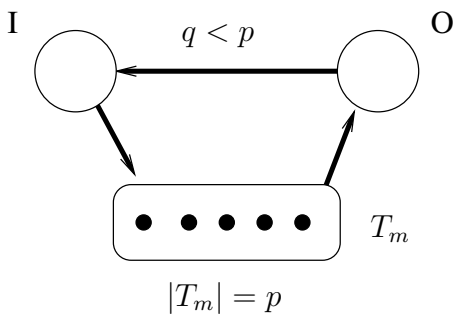


Figure 1: A transitive module on which Rule 3.4 applies.

Lemma 3.3. *Rule 3.4 is sound and can be applied in $O(n + m)$ time.*

Proof. We first prove that the partition $\mathcal{P} = \{I, T_m, O\}$ forms a safe partition of the input tournament. Let $T'_m = \{w_1, \ldots, w_q\} \subseteq T_m$ be an arbitrary subset of size q of T_m and let $Z = \{u_i v_i \mid 1 \leq i \leq q\}$. Consider the collection $\mathcal{F} = \{v_i w_i u_i \mid u_i v_i \in Z, w_i \in T'_m\}$ and notice that it certifies all the arcs in Z. In fact we have managed to certify all the backwards arcs of the partition using only arcs from A_B and hence $\mathcal{P}$ forms a safe partition. Thus, by Rule 3.3, it is safe to reverse all the arcs from O to I.

The time complexity follows from the fact that computing the modular decomposition tree can be done in $O(n+m)$ time on directed graphs [27]. It is well-known that the modular decomposition tree of a tournament has nodes labelled either *prime* or *transitive* and that each maximal transitive module corresponds to the set of leaves attached to some transitive node of the modular decomposition tree. □

We show that any YES-instance to which none of the Rules 3.1, 3.2 and 3.4 could be applied has at most $O(k\sqrt{k})$ vertices.

Theorem 3.4. *Let $(T = (V, A), k)$ be a* YES*-instance to* k-FAST *which has been reduced according to Rules 3.1, 3.2 and 3.4. Then T has at most $O(k\sqrt{k})$ vertices.*

Proof. Let S be a feedback arc set of size at most k of T and let T' be the tournament obtained from T by reversing all the arcs in S. Let σ be the transitive ordering of T' and $T_\sigma = (V_\sigma, A)$ be the ordered tournament corresponding to the ordering σ. We say that a vertex is *affected* if it is incident to some arc in S. Thus, the number of affected vertices is at most $2|S| \leq 2k$. The reduction Rule 3.1 ensures that the first and last vertex of T_σ are affected. To see this note that if the first vertex in V_σ is not affected then it is a source vertex (vertex with in-degree 0) and hence it is not part of any triangle and thus Rule 3.1 would have applied. We can similarly argue for the last vertex. Next we argue that there is no backward arc e of length greater than $2k+2$ in T_σ. Assume to the contrary that $e = uv$ is a backward arc with $S(e) = \{v, x_1, x_2, \ldots, x_{2k+1}, \ldots, u\}$ and hence $l(e) > 2k + 2$. Consider the collection $\mathcal{T} = \{vx_iu \mid 1 \leq i \leq 2k\}$ and observe that at most k of these triples can contain an arc from $S \setminus \{e\}$ and hence there exist at least $k + 1$ triplets in $\mathcal{T}$ which corresponds to distinct triangles all containing e. But then e would have been reversed by an application of Rule 3.2. Hence, we have shown that there is no backward arc e of length greater than $2k + 2$ in T_σ. Thus $\sum_{e \in S} l(e) \leq 2k^2 + 2k$.

We also know that between two consecutive affected vertices there is exactly one maximal transitive module. Let us denote by t_i the number of vertices in these modules, where $i \in \{1, \ldots, 2k-1\}$. The objective here is to bound the number of vertices in V_σ or V using $\sum_{i=1}^{2k-1} t_i$. To do so, observe that since T is reduced under the Rule 3.4, there are at least t_i backward arcs above every module with t_i vertices, each of length at least t_i. This implies that $\sum_{i=1}^{2k-1} t_i^2 \leq \sum_{e \in S} l(e) \leq 2k^2 + 2k$. Now, using the Cauchy-Schwarz inequality we can show the following.

$$\sum_{i=1}^{2k-1} t_i = \sum_{i=1}^{2k-1} t_i \cdot 1 \leq \sqrt{\sum_{i=1}^{2k-1} t_i^2 \cdot \sum_{i=1}^{2k-1} 1} \leq \sqrt{(2k^2 + 2k) \cdot (2k-1)} = \sqrt{4k^3 + 2k^2 - 2k}.$$

Thus every reduced YES-instance has at most $\sqrt{4k^3 + 2k^2 - 2k} + 2k = O(k\sqrt{k})$ vertices. □

3.2 A linear kernel for k-FAST

We begin this subsection by showing some general properties about tournaments which will be useful in obtaining a linear kernel for k-FAST.

3.2.1 Backward Weighted Tournaments

Let T_σ be an ordered tournament with weights on its backward arcs. We call such a tournament a *backward weighted tournament* and denote it by T_ω, and use $\omega(e)$ to denote the weight of a backward arc e. For every interval $I := [v_i, \ldots, v_j]$ we use $\omega(I)$ to denote the total weight of all backward arcs having both their endpoints in I, that is, $\omega(I) = \sum_{e=uv} w(e)$ where $u, v \in I$ and e is a backward arc.

Definition 3.5. (Contraction) *Let $T_\omega = (V_\sigma, A)$ be an ordered tournament with weights on its backward arcs and $I = [v_i, \ldots, v_j]$ be an interval. The contracted tournament is defined as $T_{\omega'} = (V_{\sigma'} = V_\sigma \setminus \{I\} \cup \{c_I\}, A')$. The arc set A' is defined as follows.*

- *It contains all the arcs $A_1 = \{uv \mid uv \in A, u \notin I, v \notin I\}$*
- *Add $A_2 = \{uc_I \mid uv \in A, u \notin I, v \in I\}$ and $A_3 = \{c_I v \mid uv \in A, u \in I, v \notin I\}$.*
- *Finally, we remove every forward arc involved in a 2-cycle after the addition of arcs in the previous step.*

The order σ' for $T_{\omega'}$ is provided by $\sigma' = v_1, \ldots, v_{i-1}, c_I, v_{j+1}, \ldots, v_n$. We define the weight of a backward arc $e = xy$ of A' as follows.

$$w'(xy) = \begin{cases} w(xy) & \text{if } xy \in A_1 \\ \sum_{\{xz \in A \mid z \in I\}} w(xz) & \text{if } xy \in A_2 \\ \sum_{\{zy \in A \mid z \in I\}} w(zy) & \text{if } xy \in A_3 \end{cases}$$

We refer to Figure 2 for an illustration.

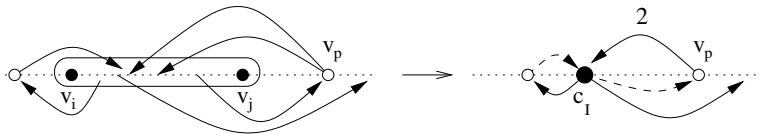


Figure 2: Illustration of the contraction step for the interval $I := [v_i, \ldots, v_j]$.

Next we generalize the notions of certificate and certification (Definitions 2.2 and 2.3) to backward weighted tournaments.

Definition 3.6. *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament, and let $f = vu \in A$ be a backward arc of T_ω. We call* ω-certificate *of f, and denote it by $\mathcal{C}(f)$, a collection of $\omega(f)$ arc-disjoint directed paths going from u to v and using only forward arcs in the span of f in T_ω.*

Definition 3.7. *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament, and let $F \subseteq A$ be a subset of backward arcs of T_ω. We say that we can* ω-certify *F whenever it is possible to find a set $\mathcal{F} = \{\mathcal{C}(f) : f \in F\}$ of arc-disjoint ω-certificates for the arcs in F.*

Lemma 3.8. *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament such that for every interval $I := [v_i, \ldots, v_j]$ the following holds:*

$$2 \cdot \omega(I) \leq |I| - 1 \tag{1}$$

Then it is possible to ω-certify the backward arcs of T_ω.

Proof. Let $V_\sigma = v_1, \ldots, v_n$. The proof is by induction on n, the number of vertices. Note that by applying (1) to the interval $I = [v_1, \ldots, v_n]$, we have that there exists a vertex v_i in T_ω that is not incident to any backward arc. Let $T'_\omega = (V'_\sigma, A')$ denote the tournament $T_\omega \setminus \{v_i\}$. We say that an interval I is *critical* whenever $|I| \geq 2$ and $2 \cdot \omega(I) = |I| - 1$. We now consider several cases, based on different types of critical intervals.

(i) Suppose that there are no critical intervals. Thus, in T'_ω, every interval satisfies (1), and hence by induction on n the result holds.

(ii) Suppose now that the only critical interval is $I = [v_1, \ldots, v_n]$, and let $e = vu$ be a backward arc above v_i with the maximum length. Note that since v_i does not belong to any backward arc, we can use it to form a directed path $c(e) = uv_iv$, which is a certificate for e. We now consider T'_ω where the weight of e has been decreased by 1. In this process if $\omega(e)$ becomes 0 then we reverse the arc e. We now show that every interval of T'_ω respects (1). If an interval $I' \in T'_\omega$ does not contain v_i in the corresponding interval in T_ω, then by our assumption we have that $2 \cdot \omega(I') \leq |I'| - 1$. Now we assume that the interval corresponding to I' in T_ω contains v_i but either $u \notin I' \cup \{v_i\}$ or $v \notin I' \cup \{v_i\}$. Then we have $2 \cdot \omega(I') = 2 \cdot \omega(I) < |I| - 1 = |I'|$ and hence we get that $2 \cdot \omega(I') \leq |I'| - 1$. Finally, we assume that the interval corresponding to I' in T_ω contains v_i and $u, v \in I' \cup \{v_i\}$. In this case, $2 \cdot \omega(I') = 2 \cdot (\omega(I) - 1) \leq |I| - 1 - 2 < |I'| - 1$. Thus, by the induction hypothesis, we obtain a family of arc-disjoint ω-certificates $\mathcal{F}'$ which ω-certify the backward arcs of T'_ω. Observe that the maximality of $l(e)$ ensures that if e is reversed then it will not be used in any ω-certificate of $\mathcal{F}'$, thus implying that $\mathcal{F}' \cup c(e)$ is a family ω-certifying the backward arcs of T_ω.

(iii) Finally, suppose that there exists a critical interval $I \subsetneq V_\sigma$. Roughly speaking, we will show that I and $V_\sigma \setminus I$ can be certified separately. To do so, we first show the following.

Claim 3.9. *Let $I \subset V_\sigma$ be a critical interval. Then the tournament $T_{\omega'} = (V_{\sigma'}, A')$ obtained from T_ω by contracting I satisfies the conditions of the lemma.*

Proof. Let H' be any interval of $T_{\omega'}$. As before if H' does not contain c_I then the result holds by hypothesis. Otherwise, let H be the interval corresponding to H' in T_ω. We will show that $2\omega(H') \leq |H'| - 1$. By hypothesis, we know that $2\omega(H) \leq |H| - 1$ and that $2\omega(I) = |I| - 1$. Thus we have the following.

$$2\omega(H') = 2 \cdot (\omega(H) - \omega(I)) \leq |H| - 1 - |I| + 1 = (|H| + 1 - |I|) - 1 = |H'| - 1$$

Thus, we have shown that the tournament $T_{\omega'}$ satisfies the conditions of the lemma. ◇

We now consider a minimal critical interval I. By induction, and using the claim, we know that we can obtain a family of arc-disjoint ω-certificates $\mathcal{F}'$ which ω-certifies the backward arcs of $T_{\omega'}$ without using any arc within I. Now, by minimality of I, we can use (ii) to obtain a family of arc-disjoint ω-certificates $\mathcal{F}''$ which ω-certifies the backward arcs of I using only arcs within I. Thus, $\mathcal{F}' \cup \mathcal{F}''$ is a family ω-certifying all backward arcs of T_ω.

This concludes the proof of the lemma. □

In the following, any interval that does not respect condition (1) is said to be a *dense interval*.

Lemma 3.10. *Let $T_\omega = (V_\sigma, A)$ be a backward weighted tournament reduced under Rule 3.1 with $|V_\sigma| \geq 2p+1$ and $\omega(V_\sigma) \leq p$. Then there exists a safe partition of V_σ with at least one backward arc between the intervals and it can be computed in polynomial time.*

Proof. The proof is by induction on $n = |V_\sigma|$. Observe that by hypothesis every vertex of T_ω belongs to the span of some backward arc (since otherwise it would not be contained in any triangle). It follows that the statement is true for $n = 3$: in such a case, T_ω is a triangle with exactly one backward arc, and hence the partition into singletons is a safe partition. This constitutes our base case.

For the inductive step, we assume first that there is no dense interval in T_ω. In this case Lemma 3.8 ensures that the partition of V_σ into singletons of vertices is a safe partition. So from now on we assume that there exists at least one dense interval.

Let I be a dense interval. By definition of I, we have that $\omega(I) \geq \frac{1}{2} \cdot |I|$. We now contract I and obtain the backward weighted tournament $T_{\omega'} = (V_{\sigma'}, A')$. In the contracted tournament $T_{\omega'}$, we have:

$$\begin{cases} |V_{\sigma'}| & \geq 2p + 1 - (|I| - 1) = 2p - |I| + 2; \\ \omega'(V_{\sigma'}) & \leq p - \frac{1}{2} \cdot |I|. \end{cases}$$

Thus, if we set $r := p - \frac{1}{2} \cdot |I|$, we get that $|V_{\sigma'}| \geq 2r + 1$ and $\omega'(V_{\sigma'}) \leq r$. Since $|V_{\sigma'}| < |V_\sigma|$, by the induction hypothesis we can find a safe partition $\mathcal{P}$ of $T_{\omega'}$, and thus obtain a family $\mathcal{F}_{\omega'}$ that ω-certifies the backward arcs of $T_{\omega'}[A_B]$ using only arcs in A_B. Observe that every vertex of $T_{\omega'}$ still belongs to the span of some backward arc, and hence it is safe to apply our induction hypothesis.

We claim that $\mathcal{P}'$ obtained from $\mathcal{P}$ by substituting c_I by its corresponding interval I is a safe partition in T_ω. To see this, first observe that if c_I has not been used to ω-certify the backward arcs in $T_{\omega'}[A_B]$, that is, if c_I is not an end point of any arc in the ω-certificates, then we are done. So from now on we assume that c_I has been part of a ω-certificate for some backward arc. Let $e = vu$ be such a backward arc in $T_{\omega'}[A_B]$, and let $c_{\omega'}(e) \in \mathcal{F}_{\omega'}$ be a ω-certificate of e. First we assume that c_I is neither the first nor the last vertex of the certificate $c_{\omega'}(e)$ (with respect to ordering σ'), and let c_1 and c_2 be the left (in-) and right (out-) neighbors of c_I in $c_{\omega'}(e)$. By definition of the contraction step together with the fact that there is a forward arc between c_1 and c_I and between c_I and c_2 in $T_{\omega'}$, we have that there were no backward arcs between any vertex in the interval corresponding to c_I and c_1 and c_2 in the original tournament T_ω. So we can always find a vertex in I to replace c_I in $c_{\omega'}(e)$, thus obtaining a certificate $c(e)$ for e in $T_\omega[A_B]$ (observe that e remains a backward arc even in T_ω). Now we assume that c_I is either the first or last vertex in the certificate $c_{\omega'}(e)$. Let e' be an arc in T_ω corresponding to e in $T_{\omega'}$ with one of its endpoints being $e_I \in I$. To certify e' in $T_\omega[A_B]$, we need to show that we can construct a certificate $c(e')$ using only arcs of $T_\omega[A_B]$. We have two cases to deal with.

(i) If c_I is the first vertex of $c_{\omega'}(e)$, then let c_1 be the right neighbor of c_I for any directed path P between $u = c_I$ and v in $c_{\omega'}(e)$. Using the same argument as before, there are only forward arcs between any vertex in I and c_1. In particular, there is a forward arc $e_I c_1$ in T_ω, meaning that we can construct a ω-certificate for e' in T_ω by setting $c(e') := (c_{\omega'}(e) \setminus \{c_I\}) \cup \{e_I\}$.

Figure 3: On the left, the ω-certificate $c_{\omega'}(e) \in \mathcal{F}_{\omega'}$. On the right, the corresponding ω-certificate obtained in T_ω by replacing c_I by the interval I.

(ii) If c_I is the last vertex of $c_{\omega'}(e)$, then let c_q be the left neighbor of c_I for any directed path P between u and $v = c_I$ in $c_{\omega'}(e)$. Once again, we have that there are only forward arcs between c_q and vertices in I, and thus between c_q and e_I. So using this we can construct a ω-certificate for e' in T_ω.

Notice that the fact that all ω-certificates are pairwise arc-disjoint in $T_{\omega'}[A_B]$ implies that the corresponding ω-certificates are arc-disjoint in $T_\omega[A_B]$, and so $\mathcal{P}'$ is indeed a safe partition of V_σ. □

We are now ready to give the linear size kernel for k-FAST. To do so, we make use of the fact that there exists a polynomial time approximation scheme for this problem [26]. The kernelization algorithm is depicted in Algorithm 1.

Theorem 3.11. *For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2+\epsilon)k$ vertices that can be computed in polynomial time.*

Proof. Let $(T = (V, A), k)$ be an instance of k-FAST. First, we reduce (T, k) according to Rule 3.1. Then, for a fixed $\epsilon > 0$, we compute a feedback arc set S using the known $(1 + \frac{\epsilon}{2})$-polynomial time approximation scheme for k-FAST [26]. If $|S| > (1 + \frac{\epsilon}{2})k$, then there is no feedback arc set of size at most k for T. Hence we return a trivial small No-instance. Otherwise, S has size at most $(1 + \frac{\epsilon}{2})k$. We then order T with the transitive ordering of the tournament obtained by reversing every arc of S in T. Let T_σ denote the resulting ordered tournament. By the upper bound on the size of S, we know that T_σ has at most $(1 + \frac{\epsilon}{2})k$ backward arcs. Thus, if T_σ has more than $(2+\epsilon)k$ vertices then Lemma 3.10 ensures that we can find a safe partition with at least one backward arc between the intervals in polynomial time. Hence we can reduce the tournament by applying Rule 3.3, decreasing the value of k accordingly. Finally, if $k \geqslant 0$, we reduce the tournament according to Rule 3.1; notice that if we get $V = \varnothing$ doing so, then we return a small trivial Yes-instance. We repeat the previous steps until we do not find a safe partition or $k \leqslant 0$. In the former case, we know by Lemma 3.10 that T can have at most $(2 + \epsilon)k$ vertices, thus implying the result. In the latter case we return a trivial small No-instance. □

4 Conclusion

In this paper we obtained linear vertex kernel for k-FAST, in fact, a vertex kernel of size $(2 + \epsilon)k$ for any fixed $\epsilon > 0$. The new bound on the kernel size improves the previous known bound of $O(k^2)$ on the vertex kernel size for k-FAST given in [4, 16]. Moreover, it would be interesting to see if one can obtain kernels for other problems using either polynomial time approximation schemes or a constant factor approximation algorithm for the corresponding problem. An interesting problem which remains unanswered is, whether there exists a linear or even a $o(k^2)$ vertex kernel for the k-Feedback Vertex Set in Tournaments (k-FVST) problem. In the k-FVST problem we are given a tournament T and a positive integer k and the aim is to find a set of at most k vertices whose deletion makes the input tournament acyclic. The smallest known kernel for k-FVST has size $O(k^2)$ [1].

Algorithm 1: Kernelization algorithm for the $(2+\epsilon)k$-vertex kernel of k-FAST.

Input : An instance $T = ((V, A), k)$ of k-FAST, where $k \in \mathbb{N}$, and a fixed $\epsilon > 0$.
Output: An equivalent instance $T' = ((V', A'), k')$ with $|V'| \leqslant (2+\epsilon)k$.

Reduce (T, k) according to Rule 3.1;
Compute a feedback arc set S using the $(1+\frac{\epsilon}{2})$-PTAS for k-FAST [26];
if $(|S| > (1+\frac{\epsilon}{2})k)$ **then**
    Return a small trivial No-instance;
else
    $T_\sigma \leftarrow$ the transitive ordering obtained by reversing every arc of S in T (observe that $\omega(T_\sigma) \leqslant p := (1+\frac{\epsilon}{2})k$);
    repeat
        if $|V_\sigma| \geqslant 2p+1 > (2+\epsilon)k$ **then**
            $\mathcal{P} \leftarrow$ the safe partition (with at least one backward arc between its intervals) obtained by Lemma 3.10;
            $T_\sigma \leftarrow$ reverse all backward arcs between intervals of $\mathcal{P}$ in T_σ (Rule 3.3);
            $k \leftarrow$ decrease the value of k accordingly;
            if $(k \geqslant 0)$ **then**
                Reduce (T_σ, k) according to Rule 3.1;
                if $(V_\sigma = \varnothing)$ **then**
                    Return a small trivial Yes-instance;
    until $(k \leqslant 0)$ *or* $(|V_\sigma| \leqslant (2+\epsilon)k)$;
    if $(k \leqslant 0)$ **then**
        Return a small trivial No-instance;
    else
        Return T_σ;

References

[1] F. N. Abu-Khzam. A kernelization algorithm for d-hitting set. *Journal of Computer and System Sciences*, 76(7):524 – 531, 2010.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *ACM Symposium on Theory of Computing (STOC)*, pages 684–693, 2005.

[3] N. Alon. Ranking tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006.

[4] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *ICALP*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009.

[5] J. Bartholdi III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[6] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci*, 75(8):423–434, 2009.

[7] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *FOCS*, pages 629–638, 2009.

[8] J. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[9] N. Bousquet, J. Daligault, S. Thomassé, and A. Yeo. A polynomial kernel for multicut in trees. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 183–194, 2009.

[10] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is NP-hard for tournaments. *Combin. Probab. Comput.*, 16(1):1–4, 2007.

[11] I. Charon and O. Hudry. A survey on the linear ordering problem for weighted or unweighted tournaments. *4OR*, 5(1):5–60, 2007.

[12] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *Advances in neural information processing systems (NIPS)*, pages 451–457, 1997.

[13] M. Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, 1785.

[14] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 776–782, 2006.

[15] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *STOC*, pages 251–260. ACM, 2010.

[16] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. Fixed-parameter tractability results for feedback set problems in tournaments. In *Conference on Algorithms and Complexity (CIAC)*, volume 3998 of *LNCS*, pages 320–331, 2006.

[17] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *ICALP*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.

[18] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation revisited. Technical report.

[19] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *World Wide Web Conference (WWW)*, 2001.

[20] P. Erdös and J. W. Moon. On sets on consistent arcs in tournaments. *Canad. Math. Bull.*, 8:269–271, 1965.

[21] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Fast local search algorithm for weighted feedback arc set in tournaments. In *AAAI*, pages 65–70, 2010.

[22] H. A. Jung. On subgraphs without cycles in tournaments. *Combinatorial Theory and its Applications II*, pages 675–677, 1970.

[23] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. *CoRR*, abs/1006.4396, 2010.

[24] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.

[25] J. Kemeny and J. Snell. *Mathematical models in the social sciences*. Blaisdell, 1962.

[26] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *ACM Symposium on Theory of Computing (STOC)*, pages 95–103, 2007.

[27] R. M. McConnell and F. de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Appl. Math.*, 145(2):198–209, 2005.

[28] V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theor. Comput. Sci*, 351(3):446–458, 2006.

[29] K. D. Reid and E. T. Parker. Disproof of a conjecture of Erdös and Moser on tournaments. *J. Combin. Theory*, 9:225–238, 1970.

[30] S. Seshu and M. B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley, 1961.

[31] P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48:303–312, 1961.

[32] E. Speckenmeyer. On feedback problems in digraphs. In *Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 411 of *Lecture Notes in Computer Science*, pages 218–231. Springer, 1989.

[33] J. Spencer. Optimal ranking of tournaments. *Networks*, 1:135–138, 1971.

[34] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

[35] A. van Zuylen. Deterministic approximation algorithms for ranking and clusterings. Technical Report 1431, Cornell ORIE, 2005.

[36] A. van Zuylen, R. Hegde, K. Jain, and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 405–414, 2007.

[37] D. H. Younger. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circuit Theory*, 10:238–245, 1963.

ANNEXE

C

Réduction du problème MULTICUT à treewidth bornée

Travail réalisé en collaboration avec Jean DALIGAULT, Christophe PAUL et Stéphan THOMASSÉ

Le problème MULTICUT consiste à décider, étant donné un graphe $G := (V, E)$, un ensemble de **requêtes** (*i.e.* des paires de sommets) et un entier k, s'il existe un ensemble de k arêtes dont la suppression dont G **déconnecte** les extrémités de chaque requête. Déterminer si MULTICUT est FPT constitue l'un des problèmes ouverts majeurs dans le domaine de la complexité paramétrée. Dans cet article, nous prouvons que le problème MULTICUT peut être réduit en temps FPT à des instances ayant une treewidth bornée par une fonction de k. Pour ce faire, nous établissons de nouvelles règles de réduction s'appliquant à des instances arbitraires de MULTICUT. Basées sur des propriétés de connexité des graphes, ces règles nous permettent d'identifier en temps FPT une requête qui peut être supprimée de l'instance de manière sûre. La conséquence principale de cette règle est que le degré du **graphe des requêtes** peut être borné par une fonction ne dépendant que du paramètre k. Pour réaliser notre réduction, nous prouvons que si le graphe a une grande treewidth (*i.e.* admet une grande clique ou grille comme mineur), alors nous pouvons identifier une requête inutile ou contracter une arête.

Remarque. Une partie des résultats présentés dans cet article à été réutilisée par Bousquet et al. pour montrer que le problème MULTICUT admet un algorithme FPT [22].

Treewidth reduction for the parameterized MULTICUT problem *

Jean Daligault, Christophe Paul, Anthony Perez, Stéphan Thomassé
LIRMM – Université de Montpellier 2, CNRS,
Montpellier, France.
{daligault|paul|perez|thomasse}@lirmm.fr

May 19, 2011

Abstract

The parameterized MULTICUT problem consists in deciding, given a graph, a set of requests (i.e. pairs of vertices) and an integer k, whether there exists a set of k edges which disconnects the two endpoints of each request. Determining whether MULTICUT is Fixed-Parameter Tractable with respect to k is one of the most important open question in parameterized complexity [5]. We show that MULTICUT reduces to instances of treewidth bounded in k. To that aim, we establish new reduction rules that apply to arbitrary instances of MULTICUT. Based on graph separability properties, these rules identify an irrelevant request that can be safely removed. As a main consequence, these rules imply that the degree of the request graph of any instance is bounded by a function of k. We prove that when the input graph has a large clique minor or a large grid minor, then we can remove an irrelevant request or contract an edge.

1 Introduction

Among the classical techniques to cope with NP-hard problems, parameterized algorithms have known a considerable development in last few years. A problem is *fixed parameter tractable* (FPT) with respect to parameter k (e.g. solution size, treewidth) if, for any instance of size n, it can be solved in time $O(f(k).n^d)$ for some fixed d. The reader is invited to refer to books [6], [7] or [19]. In this paper, we are interested in the MULTICUT problem parameterized by the solution size, which is considered as one of the main open problems of the fixed parameterized complexity theory [5]. Given a graph G and a set R of *requests* between pairs of vertices, called *terminals* (or *endpoints*), an (edge)-*multicut*[1] is a subset F of edges of G whose removal separates the two endpoints of every request in different connected components of $G \setminus F$:

Problem MULTICUT:
Input: A graph $G = (V, E)$, a set of requests R, an integer k.
Parameter: k.
Output: TRUE if there is a multicut of size at most k, otherwise FALSE.

Related results. The MULTICUT problem is already hard when restricted to a set of requests on a tree. Indeed, VERTEX COVER can be viewed as MULTICUT in stars, hence MULTICUT is NP-complete and Max-SNP hard. MULTICUT and its variants have raised an extensive literature. These problems play an important role in network issues, such as routing and telecommunication (see [4]).

MULTICUT IN TREES was already a challenging problem. Garg *et. al.* [9] proved that it admits a factor 2 approximation algorithm. Guo and Niedermeier [13] proved that MULTICUT IN TREES is FPT with respect to the solution size. And recently, Bousquet *et. al.* [1] provided a polynomial kernel. Another variant is the

*Research supported by the AGAPE project (ANR-09-BLAN-0159)
[1] As this paper never considers vertex multicut, the term *multicut* stands for edge-multicut.

MULTIWAYCUT problem in which a set of (non-paired) terminals has to be pairwise separated. Parameterized by the solution size MULTIWAYCUT has been proved FPT by Marx [15]. A faster $O^\star(4^k)$ algorithm is due to Chen *et. al.* [3].

On general instances, Garg *et. al.* gave an approximation algorithm for MULTICUT within a logarithmic factor in [8]. However MULTICUT has no constant factor approximation algorithm if Khot's Unique Games Conjecture holds [2]. This fact motivates the study of the fixed parameterized tractability of MULTICUT. Guo *et. al.* showed in [12] that MULTICUT is FPT when parameterized by both the treewidth of the graph and the number of requests. Gottlob and Lee in [10] proved a stronger result: MULTICUT is FPT when parameterized by the treewidth of the input structure, namely the input graph whose edge set is completed by the set of request pairs.

The graph minor theorem of Roberston and Seymour implies that MULTICUT is non-uniformly FPT when parameterized by the solution size and the number of requests. Marx proved that MULTICUT is (uniformly) FPT for this latter parameterization [15]. A faster algorithm running in time $O^*(8 \cdot l)^k$ was given by Guillemot [11]. Marx *et. al.* [16] obtained FPT results for more general types of constrained MULTICUT problems through treewidth reduction results. However their treewidth reduction techniques do not yield FPTness of MULTICUT when parameterized by the solution size only. Regarding this parameterization, Marx and Razgon recently obtained a factor 2 Fixed-Parameter-Approximation for MULTICUT in [17].

Our results. We show that MULTICUT parameterized by the solution size k can be reduced to graphs of treewidth bounded by a function of k. The key is to establish conditions under which we can identify an *irrelevant* request, that is a request the removal of which yields an equivalent instance. For example, it is proved that if a vertex is an endpoint of too many requests, then one of these requests is irrelevant. In particular, this implies that the request graph of any instance has degree bounded by a function of k. Likewise, if a so called *gathered* set of terminals exists (that is a set of terminals that satisfies some separability properties), then one of these terminals is incident to an irrelevant request. Both cases yield reduction rules as the irrelevant request can be identified in FPT time. We then prove that if the graph of the input instance has large treewidth (with respect to the solution size k), then either one of the first rules applies or we can identify an edge which contraction yields an equivalent instance. The cases where the input graph has a large clique minor or a large grid minor are proved separately. Therefore we prove that MULTICUT is FPT when parameterized by the solution size k if and only if MULTICUT is FPT when restricted to graphs of treewidth bounded in k. Besides we give an $O^*((2k+1)^k)$ algorithm for the INTEGER WEIGHTED MULTICUT IN TREES. This last problem, which was left open in [1] and [13], is one of the simplest subcases of the MULTICUT problem on bounded treewidth graphs.

2 Preliminaries

We consider undirected graphs $G = (V(G), E(G))$ together with a set of requests $R = \{(s_1, t_1), \ldots, (s_l, t_l)\}$, with $s_i, t_i \in V(G)$ for $i = 1, \ldots, l$. Given a vertex x of $V(G)$, $N_G(x)$ denotes the set of neighbours of x (we forget the subscript when the context is clear). When $X \subseteq V(G)$ we denote by $G[X]$ the subgraph of G induced by X.

Given two vertices $x, y \in V(G)$, *contracting* x and y in G means deleting x and y and adding a new vertex with neighbourhood $N(x) \cup N(y) \setminus \{x, y\}$. An (x, y)-*cut* is a set of edges of the graph G which removal disconnects x and y. We define similarly an (x, Y)-cut and an (X, Y)-cut for X, Y two sets of vertices of $V(G)$. We say that an edge xy *belongs* to a set $X \subseteq V(G)$ if $x, y \in X$. A set of edges F *touches* an induced subgraph G' of G if an edge in F belongs to $V(G')$. Given a graph J, let us denote by L_i and call *level* i *of* J *starting from* $L_0 \subseteq V(J)$ the subset of $V(J)$ containing vertices lying at distance exactly i from L_0 in J. The *level decomposition* of J starting with $L_0 \subseteq V(J)$ is the partition of $V(J)$ into levels L_i, $i \geq 0$.

Let $\mathcal{P}$ be a partition of $V(G)$, such that every part h of $\mathcal{P}$ induces a connected subgraph of G. Let H be the graph G quotiented by $\mathcal{P}$, i.e. the graph G where each part of $\mathcal{P}$ is contracted into a single vertex. If J is a subgraph of H, then J is said to be a *minor* of G. We say that the pair $(\mathcal{P}, H)$ is a J-*model* of G. In a slight abuse of notation, we write that H is a J-*model* of G (or simply a *model* of G), leaving partition

$\mathcal{P}$ implicit. We also abusively write that a part h of $\mathcal{P}$ is a *part* of H. We want to emphasize that edges of H are *all* pairs of parts of $\mathcal{P}$ between which G induces an edge. When H' is a subgraph of H, $V(H')$ naturally denotes the set of vertices of H', which are parts, and we write $V_G(H')$ to denote the set of vertices $\cup_{h \in H'}\{x \in h\} \subseteq V(G)$.

Given an instance (G, R, k) of the parameterized MULTICUT problem (or MULTICUT for short), a *k-multicut* is a multicut of size at most k. We say that a k-multicut is *optimal* when its size is minimum among all k-multicuts. We say that a request (s_i, t_i) is *irrelevant* whenever the instance $(G, R \setminus \{(s_i, t_i)\}, k)$ is equivalent to the instance (G, R, k).

Finally, we assume that the reader is familiar with the concept of treewidth. We refer the reader to Robertson and Seymour's Graph Minors Series or to [21] for definitions and results on treewidth.

3 General reduction rules for MULTICUT

Let (G, R, k) be an instance of MULTICUT. This section is devoted to show that the following set of reduction rules is correct and can be applied in FPT time.

Rule 1 *If there exist $k+1$ edge-disjoint paths between two vertices x and y, then contract x and y.*

Rule 2 *There exist an integer $g(k)$ such that if a vertex t is an endpoint of at least $g(k)$ requests, then we can find in FPT time an irrelevant request incident to t.*

By Rule 2 we may assume that the degree of the request graph is bounded by a function of k. We say that a set $T \subseteq V$ is *gathered* if for every $F \subseteq E$, $|F| \leq k$, there exists at most one connected component in $G \setminus F$ containing more than one vertex of T. We denote this connected component by C_F. Note that a subset of a gathered set is gathered.

Rule 3 *If the instance is reduced under Rule 2 and there exists a gathered set of terminals T of size at least $f_1(k) = 4g(k)^3$, then we can find in FPT time an irrelevant request incident to one of these terminals.*

Lemma 1 *Rule 1 is safe and can be applied in polynomial time.*

Proof: If there exist $k+1$ edge-disjoint paths between two vertices x and y, then x and y lie in the same connected component of $G \setminus F$ for any set F of k edges. Hence contracting x and y yields an equivalent instance. Testing whether two vertices are $(k+1)$-edge-connected is polynomial by flow. □

To prove the soundness of Rule 2 and Rule 3, we study two edge-connectivity problems of independent interest. We define the central notions of this section:

- A $(zy|x)$-cut is a (z, x)-cut which is not a (z, y)-cut. Similarly, given a subset of vertices T, a $(zy|T)$-cut is a (z, T)-cut which is not a (z, y)-cut.
- A vertex $y \notin T$ is *$(z|T)$-k-linked* if there is no $(zy|T)$-cut of size at most k, *i.e.* if every (z, T)-cut of size at most k is a (z, y)-cut.

Given a graph $G = (V, E)$, $x, y, z \in V(G)$ and a positive integer k, we call TRIPLE SEPARATION the problem of finding a $(zy|x)$-cut of size at most k (if one exists) parameterized by k.

Theorem 1 *The* TRIPLE SEPARATION *problem is FPT with respect to the solution size k.*

A stronger statement has recently been proved by Marx *et. al.* [16]. Their Theorem 3.4 states that deciding whether there exists a set of k vertices separating prescribed pairs and not separating other prescribed pairs is FPT with respect to k and the number of prescribed pairs. We give a different proof of our statement for the sake of completeness.

Proof: Let $G = (V, E)$ be a graph, and $X \subseteq V(G)$. We denote by $\delta(X)$ the *border* of the set of vertices X, i.e. the set of edges having exactly one endpoint in X. Let c be the size of a minimum cut between z and x, and X a minimal set of vertices of border of size c containing x. Observe first that if $c > k$ then the answer is No. So assume that $c \leqslant k$. Notice that a minimal edge-cut separates the graph into exactly 2 connected components. Hence if $y \notin X$ then $\delta(X)$ is a $(zy|x)$-cut and the answer is Yes. Otherwise, let G' be the multigraph obtained from G by contracting $V(G) \setminus X$ into a single vertex z', keeping multiple edges.

Claim 1 *There exists a $(zy|x)$-cut of size at most k in G if and only if there exists a $(z'y|x)$-cut of size at most k in G'.*

Proof. Assume first that there exists a $(z'y|x)$-cut F' of size at most k in G'. Let W be the connected component of $G' \setminus F'$ containing x. We have that $|\delta_G(W)| = |\delta_{G'}(W)|$, hence $\delta_G(W)$ is a $(zy|x)$-cut of size at most k in G.

Conversely, let F be a $(zy|x)$-cut of size at most k in G, and let Z be the connected component of $G \setminus F$ containing z and y. Let $Z' \subseteq V(G')$ be the set $Z \cap X \cup \{z'\}$. The set $\delta_{G'}(Z')$ is a $(z'y|x)$-cut in G'. By definition, we have $|\delta_{G'}(Z')| = |\delta_{G'}(Z \cap X \cup \{z'\})| = |\delta_G(Z \cup \overline{X})|$. By submodularity of the border, we know that $|\delta_G(Z)| + |\delta_G(\overline{X})| \geq |\delta_G(Z \cup \overline{X})| + |\delta_G(Z \cap \overline{X})|$. Thus, $|\delta_{G'}(Z')| \leq |\delta_G(Z)| + |\delta_G(X)| - |\delta_G(Z \cup \overline{X})|$. By minimality of X, we know that $|\delta(X)| \leq |\delta_G(Z \cup \overline{X})|$, which gives $|\delta_{G'}(Z')| \leq |\delta_G(Z)| \leq k$. ⋄

We are now looking for a $(z'y|x)$-cut of size at most k in G'. By minimality of X, we know that $\delta_{G'}(X)$ is the only minimum (z', x)-cut in G' (otherwise we would find a smaller set $X' \subsetneq X$ containing x of border of size c in G). We thus have to check the cuts of size l with $c + 1 \leq l \leq k$. By definition, such a cut does not contain all edges of $\delta_{G'}(X)$, because otherwise it would be a (z, y)-cut. We thus branch over the c edges adjacent to z', obtaining c new instances where the considered edge has been contracted to a single vertex $\tilde{z}$. Doing so, we strictly increase the connectivity between z' and x by minimality of X. We now have to decide if there exists a $(\tilde{z}y|x)$-cut in a graph where connectivity between $\tilde{z}$ and x is at least $c+1$. Note that if the contracted edge was $z'y$, we just need to decide whether there exists a cut of size at most k between $\tilde{z}$ and x. Since $c \leq k$ and since the connectivity between $\tilde{z}$ and x strictly increases at each step, the whole branching algorithm runs in time $O(k! \times poly(n))$. □

We say that a vertex x is *k'-strongly $(z|T)$-k-linked* if and only if for every $S \subseteq T$ such that $|S| \geq |T| - k'$, x is $(z|S)$-k-linked (*i.e.* if there is no $(zx|S)$-cut of size at most k). Note that when x is k'-strongly $(z|T')$-k-linked, x is a fortiori k'-strongly $(z|T)$-k-linked in G when $T' \subseteq T$. In particular, we get the following corollary by using Theorem 1 on every subset $S \subseteq T$ such that $|S| \geq |T| - k'$, contracting set S into a single vertex:

Corollary 1 *Deciding whether a vertex x is k'-strongly $(z|T)$-k-linked and producing a witness of non-strong-linkness, i.e. a $(zx|S)$-cut for S a subset of T of size at least $|T| - k'$, is FPT in k, k' and $|T|$.*

The following Theorem is a key result of this paper:

Theorem 2 *Let $G = (V, E)$ be a graph, z a vertex of $V(G)$ and T a subset of $V(G) \setminus \{z\}$ of size at least $f(k, k')$, for a large enough $f(k, k')$. There exists a vertex $x \in T$ which is k'-strongly $(z|T \setminus \{x\})$-k-linked and which can be found in FPT time in k, k' and $|T|$.*

Proof: Assume that T has size at least $f(k, k') = k^{k+k'+1}(\frac{k'+1+k^2}{k-1} + 1) - \frac{k'+1+k^2}{k-1}$.

We initiate our algorithm with the graph $f_6 := G$ and let $T_0 := T$. We repeat the following process $k + k' + 2$ times, selecting a vertex x_i in T_i for $i = 0, \ldots, k + k' + 1$. We first test whether x_i is k'-strongly $(z|T_i \setminus \{x_i\})$-$k$-linked in G_i as in Corollary 1. If this is the case, then we are done. If this is not the case, we obtain a subset S of T_i of size at least $|T_i| - k' - 1$ and a $(z|S)$-cut C_i of size at most k which is not a (z, x_i)-cut. Denote by L_i the connected component of $G_i \setminus C_i$ containing z, and consider the connected component V_{i+1} of $G_i \setminus C_i$ not containing z and containing the largest number of vertices of S. Let G_{i+1} be the graph obtained from $G[V_{i+1} \cup L_i]$ by contracting $L_i \cup V(C_i)$ into z, where $V(C_i)$ denotes the set of vertices incident

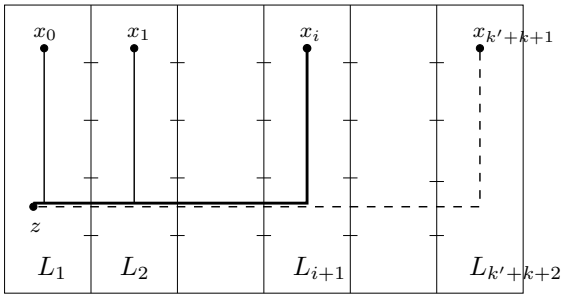


Figure 1: Vertex $x_{k'+k+1}$ is k'-strongly $(z|T \setminus \{x\})$-k-linked in the proof of Theorem 2.

to C_i. Note that $V(G_{i+1})$ contains a large subset $T_{i+1} = T_i \cap (V_{i+1} \setminus V(C_i))$ of vertices of T, and that x_i is well defined for every $i = 1, \ldots, k + k' + 1$. Indeed, $|T_i| \geq \frac{|T_{i-1}|-k'-1}{k} - k$. Equivalently, this means that $|T_i| + \frac{k'+1+k^2}{k-1} \geq \frac{1}{k}(|T_{i-1}| + \frac{k'+1+k^2}{k-1})$. Thus the sequence $(|T_i| + \frac{k'+1+k^2}{k-1})$ follows a geometric progression of factor $\frac{1}{k}$. So that $|T_{k+k'+1}| \geq 1$ we need that $|T| + \frac{k'+1+k^2}{k-1} = |T_0| + \frac{k'+1+k^2}{k-1} \geq k^{k+k'+1}(\frac{k'+1+k^2}{k-1} + |T_{k+k'+1}|) \geq k^{k+k'+1}(\frac{k'+1+k^2}{k-1} + 1)$, that is $|T| \geq k^{k+k'+1}(\frac{k'+1+k^2}{k-1} + 1) - \frac{k'+1+k^2}{k-1}$, which is the assumption of the Theorem.

If the algorithm did not stop after step $k + k' + 1$, this means that for every $i = 1, \ldots, k + k' + 1$, x_i is not k'-strongly $(z|T \setminus \{x_i\})$-k-linked in G_i.

Let us denote $T_e := \{x_i : i = 0, \ldots, k + k'\}$. We will show that $x_{k+k'+1}$ is k'-strongly $(z|T_e)$-k-linked in G, which implies that $x_{k+k'+1}$ is k'-strongly $(z|T \setminus \{x_{k+k'+1}\})$-$k$-linked in G. Consider a path P between $x_{k+k'+1}$ and z. Note that $C_i \cap C_j = \emptyset$ for $i \neq j$, which implies that each edge of C_i has one endpoint in L_i and one endpoint in L_{i+1}. For every $i = 0, \ldots, k + k'$, P intersects L_i since P intersects C_i. For every $i = 0, \ldots, k+k'$, let P_i be a path between x_i and z in G which is included in P outside L_i. Such a path exists because L_{i+1} is connected. This implies that, for every set $S_e \subseteq T_e$ of size at least $k + k' + 1 - k' = k + 1$, and for every path P between $x_{k+k'+1}$ and z, there exist $k+1$ paths P_j for $j = 1, \ldots, k+1$ between vertices of S_e and z, such that $P_j \cap P_l \subseteq P$ for $j \neq l$. Hence, every path P between $x_{k+k'+1}$ and z must be cut by every cut of size at most k between z and S_e, which thus separates $x_{k+k'+1}$ and z. □

Lemma 2 *Rule 2 is safe and can be applied in FPT time.*

Proof: Let t be a vertex endpoint of at least $g(k) = f(k, k)$ requests, and $\tilde{T}$ be the corresponding endpoints. Since $|\tilde{T}| \geqslant f(k, k)$, Theorem 2 implies that there exists a vertex $\tilde{t} \in \tilde{T}$ which is k-strongly $(t|\tilde{T} \setminus \{\tilde{t}\})$-$k$-linked in G, so $\tilde{t}$ is $(t|\tilde{T} \setminus \{\tilde{t}\})$-k linked in G. Let F be a k-multicut of $(G, R \setminus (t, \tilde{t}), k)$. By definition, F is a $(t, \tilde{T} \setminus \{\tilde{t}\})$-cut in G, and hence a $(t, \tilde{t})$-cut in G. It follows that F is actually a k-multicut for (G, R, k) and thus that the request $(t, \tilde{t})$ is irrelevant. □

Lemma 3 *Rule 3 is safe and can be applied in FPT time.*

Proof: Let T be a gathered set of terminals of size at least $4f(k, k)^3$, and $S = \{s|(s, t) \in R, t \in T\}$. By Rule 2 each vertex $t \in T$ is the endpoint of at most $f(k, k)$ requests. Hence there exist two sets $T' = \{t'_1, \ldots, t'_p\} \subseteq T$ and $S' = \{s', \ldots, s'_p\} \subseteq S$ with $p = 4f(k, k)^2$ such that (t'_i, s'_i) is a request for every $i = 1, \ldots, p$.

We now define an auxiliary bipartite graph $B = (T' \cup S', E')$ where $t'_i s'_j \in E'$ if $i \neq j$ and there exists a cut of size at most k in G such that t'_i and s'_j lies in a same connected component different from C_F in $G \setminus F$. For every $i \neq j$ we use Theorem 1 on the graph obtained from G by contracting $T' \setminus \{t'_i\}$ into a single vertex x to test whether there exists a $(t'_i s'_j|x)$-cut of size at most k. If so, then t'_i and s'_j are in the same connected component in $G \setminus F$, which is not C_F since t'_i is cut from any vertex of $T' \setminus \{t'_i\}$, and t'_i and s'_j are adjacent in B. Hence B can be constructed in FPT time in k.

Claim 2 *Every vertex $s' \in S'$ has degree at most $f(k, k)$ in B.*

Proof. Assume there exists a vertex $s' \in S'$ with degree at least $f(k,k)$, and let x be any of its neighbours. By Theorem 2, x is k-strongly $(s'|N_B(s') \setminus \{s'\})$-$k$-linked. Since x and s' are adjacent in B, there exists a set $F \subseteq E(G)$, $|F| \leq k$ such that s' and x belongs to a same component different from C_F in $G \setminus F$. Since T is a gathered set, such a cut F must cut x from $N_B(s) \setminus \{x\}$ and is thus a cut between x and s', a contradiction. $\diamond$

Claim 3 *There exists $T'' \subseteq T'$ and $S'' \subseteq S'$ of size $f(k,k)$ such that for every set F of at most k edges, if $t'_i \in T''$ and $s'_j \in S''$ both lie in a component $C \neq C_F$ of $G \setminus F$, then $i = j$.*

Proof. By Claim 2, we have $|E'| \leqslant |S'| \cdot f(k,k)$. Hence there exists a set J of at least $\frac{|T'|}{2}$ vertices of T' with degree at most $2f(k,k)$ in B. Let $I = \{s : (t,s) \in R, t \in J\}$. The degree of $B[J \cup I]$ is at most $2f(k,k)$ implying that we can greedily find two sets $T'' = \{t''_1, \ldots, t''_{f(k,k)}\} \subseteq J$ and $S'' = \{s''_1, \ldots, s''_{f(k,k)}\} \subseteq I$ such that (t''_i, s''_i) is a request for every i and $B[T'' \cup S'']$ does not contain any edge. By construction, this means that for every set F of at most k edges if t''_i and s''_j both lie in a component $C \neq C_F$ of $G \setminus F$ then $i = j$. $\diamond$

By Theorem 2, there exists a vertex $s'' \in S''$ which is k-strongly $(z|S'' \setminus \{s''\})$-$k$-linked in $\tilde{G}$, where $\tilde{G}$ is the graph obtained from G by contracting T'' to a single vertex z. We conclude the proof by showing the following:

Claim 4 *The request (t'', s'') is irrelevant.*

Proof. Let F be a k-multicut for the instance $(G, R \setminus (t'', s''), k)$. In particular, F is a (T''_F, S''_F)-cut where $T''_F = T'' \cap C_F$ and S''_F denotes the set of terminals corresponding to $T'' \cap C_F$. Observe that F is a (z, S''_F)-cut in $\tilde{G}$: otherwise this would imply that there exists a vertex of $T'' \setminus T''_F$ lying in a component $C \neq C_F$ of $G \setminus F$ which also contains a vertex of S''_F in $G \setminus F$, which cannot be by Claim 3. Hence F is a (s'', z)-cut in $\tilde{G}$ and thus a (t'', s'')-cut in G, implying that F is actually a k-multicut for (G, R, k). $\diamond$

This concludes the proof of Lemma 3. $\square$

The rest of the paper is essentially devoted to finding a situation where Rule 3 can be applied.

4 Clique Minor

In this section, we assume that G has a large clique minor. Let H be an $f_2(k)$-clique model of G, for a large enough $f_2(k)$. Such a model can be computed in FPT time in k. Let a be a vertex of G lying in a part h_a of H. The vertex a is *special* if there exist $k+1$ vertex-disjoint paths internally in h_a between a and distinct parts of H different from h_a.

4.1 Finding a nice model

We say that a model H of a graph G is *nice* whenever it satisfies the following properties:

(1) $|V(H)| \geq f_2(k)$ and H is $(k+1)$-vertex connected.

(2) There exists at most one special vertex a in G (and we denote its part by h_a).

(3) All parts but possibly h_a have degree at most $(k+1)^2 + 1$ in H.

(4) The special vertex a separates $h_a \setminus \{a\}$ from $V(G) \setminus h_a$ in G.

Theorem 3 *Let (G, R, k) be an instance of* Multicut *reduced under Rule 1 which admits an $f_2(k)$-clique model H. There exists a nice model of G which can be computed in FPT time in k.*

Proof: Let us first show that H respects the second property of the nice model definition. Observe that any $f_2(k)$-clique model for G is in particular $(k+2)$-vertex connected since we may assume $f_2(k) \geqslant k+2$.

Claim 5 *Let (G, R, k) be an instance of* Multicut *reduced under Rule 1 that admits a $(k+1)$-vertex connected model H. Then G contains at most one special vertex.*

Proof. Assume by contradiction that there exist two vertices a and a' having at least $k+1$ vertex-disjoint paths $P_1, \ldots, P_{k+1}$ and $P'_1, \ldots, P'_{k+1}$ in their own parts h_a and $h_{a'}$ towards distinct parts different from h_a (resp. $h_{a'}$) in H. Since G is reduced under Rule 1, there are at most k vertex-disjoint paths between a and a' in G. Hence by Menger's theorem [18] there exists a set S of at most k vertices of G which removal disconnects a and a'. Let $\mathcal{S}$ be the set of parts of H containing vertices of S. By hypothesis, there exists at least one path P_i towards a neighbour h_i of h_a and one path P'_i towards a neighbour h'_i of h'_a such that $P_i \cup h_i$ and $P'_i \cup h'_i$ are not intersected by $\mathcal{S}$. By $(k+1)$-vertex connectivity of H there exists a path between h_i and h'_i in $H \setminus \mathcal{S}$. Since every part of $H \setminus \mathcal{S}$ is connected, such a path can be extended to a path between a and a' in $G \setminus S$, leading to a contradiction. ◇

We now show a technical result:

Lemma 4 *Let $G = (V, E)$ be a graph that admits a p-vertex connected model H with $p > k$ and assume G contains at most one special vertex a. If there is a part $h \neq h_a$ with degree greater than $(p-1)(k+1)$ in H, then there exists a bipartition h_1, h_2 of h such that h_i is connected in G for $i = 1, 2$ and the partition $\mathcal{P} \setminus h \cup \{h_1, h_2\}$ of $V(G)$ is a p-vertex connected model of G, where $\mathcal{P}$ is the partition of $V(G)$ associated to H.*

Proof: We need the following claims:

Claim 6 *Let T be a tree of maximum degree d, with a weight function ω from the nodes of T into $\{0, \ldots, M\}$, such that $\omega(T) > q(d+1)$ with $q \geq M$. There exists an edge of T separating T into two subtrees T_1 and T_2 such that $\omega(T_i) \geqslant q+1$ for $i = 1, 2$.*

Proof. Assume that T contradicts the lemma. For each edge e in T, let T_1^e and T_2^e be the connected components of $T \setminus e$. We have $\omega(T_1^e) \leq q$ or $\omega(T_2^e) \leq q$ (and these cases are mutually exclusive). Let us orient T: e gets the orientation $T_1^e \to T_2^e$ if $\omega(T_1^e) \leq q$. Note that each edge uv gets the orientation $u \to v$ whenever u is a leaf. Since this orientation is acyclic, there exists an internal node x of T such that all edges incident to x are oriented towards x. Let $T_1^x, \ldots, T_l^x$ be the connected components of $T \setminus x$. We have $l \leq d$. Thus $w(T) = \omega(T_1^x) + \ldots + \omega(T_l^x) + \omega(x) \leq l * q + M \leq q(d+1)$, a contradiction. This concludes the proof of Claim 6. ◇

Claim 7 *Let H be a c-vertex-connected graph and $h \in V(H)$ be a vertex of degree at least $2c$. Let N_1, N_2 be a bipartition of $N(h)$, with $|N_i| \geq c$ for $i = 1, 2$. Let H' be the graph obtained from H by deleting h and adding two vertices h_1 and h_2, of respective neighbourhoods $N_1 \cup \{h_2\}$ and $N_2 \cup \{h_1\}$. Then H' is c-connected.*

Proof. Assume that there exists a set S of $c-1$ vertices which removal disconnects H'. If $h_i \notin S$ for $i = 1, 2$, then S disconnects H, a contradiction. If h_1 and h_2 both lie in S, then $(S \setminus \{h_1, h_2\}) \cup \{h\}$ disconnects H, a contradiction. If h_i lies in S, for $i = 1$ or $i = 2$, then $\{h_{3-i}\}$ is not a connected component of $H \setminus S$ since it has at least c neighbours besides h_i in H'. Thus $S \setminus \{h_i\} \cup \{h\}$ is a $(c-1)$-cut in H, a contradiction. This concludes the proof of Claim 7. ◇

We use these two results to prove Lemma 4. Consider a part $h \neq h_a$ of degree greater than $(p-1)(k+1)$ in H. For each part h' adjacent to h in H, we distinguish one vertex $x \in h$ such that there exists an edge xx' in G with $x' \in h'$. Denote this vertex x by $v(h')$. Let T be a tree of $G[h]$ spanning all vertices x such that $v^{-1}(x) \neq \emptyset$, which is minimal by inclusion. The leaves of T are vertices such that $v^{-1}(x) \neq \emptyset$. Let ω be a weight function on h, such that $\omega(x) = |v^{-1}(x)|$.

Since G contains at most one special vertex, vertices distinct from a can have at most k neighbours in distinct other parts, implying that $\omega(x) \leqslant k$ for every vertex $x \neq a$. Moreover, the degree of any vertex in T is at most k, since a vertex with degree at least $k+1$ would have $k+1$ vertex-disjoint paths towards leaves of T and hence towards $k+1$ distinct parts different from its own part, which cannot be by hypothesis. By construction, $\omega(T)$ is equal to the degree of h in H, so $\omega(T) > (p-1)(k+1)$. By Claim 6 applied with $q = p-1$ and $d = k$, there exists a bipartition T_1, T_2 of T such that $\omega(T_i) \geqslant p$. Observe that this bipartition can be extended into a bipartition of h into two connected sets h_1, h_2 such that $|N_H(h_1)|, |N_H(h_2)| \geqslant p$, by definition of the weight function ω. By Claim 7 it follows that the partition $\mathcal{P} \setminus h \cup \{h_1, h_2\}$ defines a model H' which is p-connected. This concludes the proof of Lemma 4. □

Assume now that H does not contain a special vertex, and let h be a part of H with degree at least $(k+1)^2+1$ in H. Since H is $(k+2)$-vertex connected, it follows by Lemma 4 applied with $p = k+2$ that we can find in polynomial time a new model of G which is $(k+2)$-vertex connected, which contains at most one special vertex by Claim 5, and with size strictly greater than $V(H)$. We apply repeatedly Lemma 4 until no part has degree at least $(k+1)^2+1$ in H_i or there exists a special vertex a in a part h_a. In the former case, since the model finally obtained is $(k+2)$-vertex connected and does not contain any special vertex, it is actually a nice model, thus we are done. In the latter case, we modify the finally obtained model to obtain a new model H' such that the special vertex a cuts $h_a \setminus \{a\}$ from $V(G) \setminus h_a$ in G as follows.

For every vertex $v \neq a \in h_a$ having a neighbour in a part $h_v \neq h_a$, we denote by A_v the set of vertices disconnected from a in h_a by the removal of v. We remove the set $A_v \cup \{v\}$ from h_a and add it to h_v, repeating this process until no vertex in h_a but a is adjacent to other parts. Observe that the model H' thus obtained may no longer be $(k+2)$-vertex connected after this process: however, $H' \setminus h_a$ remains $(k+1)$-vertex connected. Since h_a has degree at least $k+1$ in H_i, it follows that H' is a $(k+1)$-vertex connected model. Observe that H' contains exactly one special vertex a by Claim 5.

Since this process may increase the degree of some parts different from h_a in H', we apply repeatedly Lemma 4 to H' to reduce its degree while preserving $(k+1)$-vertex connectivity. Once this process is over, we obtain a $(k+1)$-vertex connected model H' such that $|V(H')| \geqslant f_2(k)$, which special vertex a separates $h_a \setminus \{a\}$ from $V(G) \setminus h_a$ in G, and such that every part but possibly h_a has degree at most $(k+1)^2$. It follows that H' is a nice model, which concludes the proof of Theorem 3. □

4.2 Small and giant components

In the following we consider a nice model H of the instance (G, R, k) of the Multicut problem.

Lemma 5 *Let H be a nice model of G. Two parts not touched by a set $F \subseteq E(G)$ of at most k edges are included in the same connected component of $G \setminus F$.*

Proof: Assume that a set F of at most k edges does not touch parts h_1 and h_2. It follows that part h_1 (resp h_2) is completely included in a connected component of $G \setminus F$. By $(k+1)$-connectivity in H, there are $k+1$ disjoint paths between h_1 and h_2 in H, which cannot all be cut by a set of at most k edges. □

Corollary 2 *Let H be a nice model of G, h_1, h_2 be two parts of H and (t_1, t_2) a request with $t_1 \in h_1$ and $t_2 \in h_2$. Every k-multicut contains an edge that belongs to h_1 or an edge that belongs to h_2.*

Observe that any set of at most k edges cuts the graph G into at most $k+1$ connected components. Given a set F of edges, we call *giant component* of $G \setminus F$ and denote by C_F a connected component of $G \setminus F$ containing entirely at least $|V(H)| - k$ parts.

Lemma 6 *Let F be a set of at most k edges of a graph G admitting a nice model H. Then $G \setminus F$ has a giant component, which contains all parts not touched by F.*

Proof: Let h_1 and h_2 be two parts not touched by a set F of at most k edges. Then h_1 and h_2 belong to the same connected component in $G \setminus F$ by Lemma 5. Since there are at most k parts touched by F, all other parts are entirely included in the same connected component of $G \setminus F$. □

The connected components distinct from the giant component are called *small components*. They intersect vertex-wise altogether at most k parts.

Lemma 7 *Assume that G admits a nice model. For every set F of at most k edges, the special vertex belongs to the giant component of $G \setminus F$.*

Proof: Since a is adjacent to at least $k+1$ parts, a is adjacent to at least one vertex lying in a part not touched by F in $G \setminus F$. By Lemma 6, a cannot belong to a small component. □

Lemma 8 *Let F be an optimal k-multicut of a graph G admitting a nice model. Every small component C of $G \setminus F$ contains a vertex which is a terminal.*

Proof: Let F be an optimal k-multicut of G and e be any edge of F which touches C. If C contains no terminal then $F \setminus \{e\}$ is still a multicut, contradicting the optimality of F. □

Let $H_a = H \setminus h_a$. Adjacent vertices x and y which are far away in H_a from any terminal can be contracted as stated in the following Lemma:

Lemma 9 *Let G be a graph admitting a nice model H. Let h be a part of H_a not containing any terminal, such that every part at distance at most $k+2$ from h in H_a has no terminal. Let e be an edge of G with at least one endpoint in h. Then e does not belong to any optimal k-multicut.*

Proof: Consider an optimal k-multicut F of G, and let h' be a part adjacent to h in H. Every small component contains a terminal by Lemma 8 and intersects at most k parts by definition. Since part h (resp. h') is assumed to be too far from any terminal in H_a, part h (resp. h') can intersect a small component C of $G \setminus F$ only if $a \in C$, which contradicts Lemma 7. Hence neither part h nor part h' can intersect a small component. It follows that if e belongs to F then $F \setminus \{e\}$ is also a k-multicut, which contradicts the optimality of F. □

Hence the following reduction rule is correct:

Rule 4 *Let G be a graph admitting a nice model H and h be a part of H_a without any terminal. If every part at distance at most $k+2$ from h in H_a has no terminal, then contract an arbitrary edge e of G with at least one endpoint in h.*

When this reduction rule does no longer apply, every part is at distance at most $k+2$ in H_a from a part containing a terminal.

4.3 Reducing the instance

Let us review the structure of the graph when none of Rule 1, Rule 2 and Rule 4 applies. Recall that we consider an instance (G, R, k) that contains a nice model H.

We consider the level decomposition of H_a starting from some part L_0 of H_a. Since H is a nice model, it follows that every part of H_a has degree at most $(k+1)^2$ in H_a. Hence every level L_i has size at most $(k+1)^2|L_{i-1}|$. Denote by d the number of non-empty levels in this decomposition. We have $f_2(k) \leq |V(H_a)| + 1 \leq (\sum_{i=0}^{d}((k+1)^2)^i) + 1 \leq (k+1)^{2(d+1)}$, so $d \geq log_{(k+1)^2}(f_2(k)-1) - 1$.

Let us show that we can find a gathered set and thus that the graph can be reduced using Rule 3. Let T be a set of maximum size of terminals in $V_G(H_a)$ belonging to different parts lying pairwise at distance

at least $k+1$ in H_a. Since every part is at distance at most $k+2$ from a terminal in H_a (Rule 4), there exists at least one terminal in every $2(k+2)+1$ consecutive levels, implying that T has sufficiently large size (recall that we assumed $f_2(k)$ to be large enough).

Let us argue that T is a gathered set. Indeed, consider any set F of at most k edges. Since parts containing terminals of T are pairwise at distance at least $k+1$ in H_a, a small component of $G \setminus F$ can contain two terminals of T only if it contains the special vertex a (Lemma 6). Since a belongs to the giant component by Lemma 7, it follows that $G \setminus F$ contains at most one component containing more than one vertex of T (namely the giant component), and hence T is a gathered set. Using Rule 3, an irrelevant request can be found in FPT time. Hence we have shown in this Section that whenever G has a large clique model, we can reduce the instance in FPT time either by safely contracting an edge or by finding an irrelevant request. This concludes the large clique minor part.

5 Grid Minor

The aim of this section is to complete the treewidth reduction by showing that a graph with a large grid minor but no large clique minor can also be reduced in FPT time. Indeed, every graph with treewidth at least 2^{2m^5} contains an $(m \times m)$-grid minor [21], and a model for such a grid minor can be computed in FPT time in m.

Given a grid model $H = (V(H), E(H))$ of a graph G we denote by $\bar{H}$ a graph $(V(H), E(\bar{H}))$ where $E(\bar{H}) \subseteq E(H)$ is such that $\bar{H}$ is a grid. We call the set of vertices of degree two or three in a grid H' the *border* of the grid, and denote it by $B(H')$.

5.1 On grid minors without clique minors

In this section, we concentrate on structural properties of graphs with large grid minors but no large clique minors. Let G_k be the graph obtained from the $(k \times k)$-grid by adding the two diagonals to each internal face of the grid (see Figure 2 in Appendix).

Lemma 10 *The crossed grid G_{2k} has a k-clique minor.*

Proof: We are looking for a k-clique model inside G_{2k}, that is for k vertex-disjoint connected sets of vertices of G_{2k} such that any two of these sets are adjacent in G_{2k}. Roughly speaking, these subsets will be the union of the ith column and ith row of G_{2k} for odd i. These sets are not vertex disjoint, but this can easily be dealt with: for any crossing of two sets, we use two diagonals inside a face of the original grid to preserve connectivity while uncrossing them.

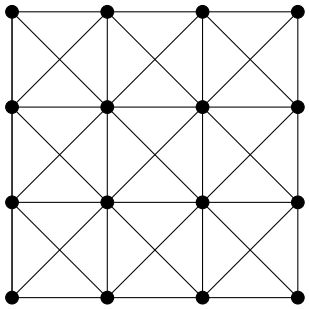


Figure 2: The crossed grid G_4.

More formally, denote by (i,j) the vertex lying on row i and column j of the crossed grid G_{2k}. Let S_i be the set of vertices defined as follows:

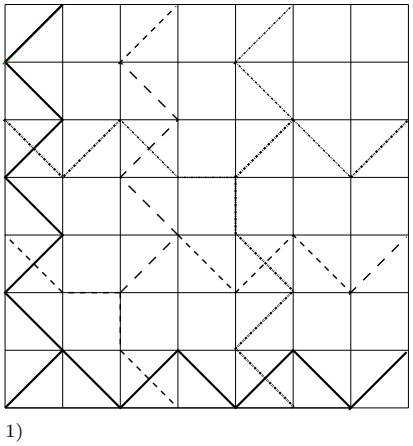


Figure 3: A clique model in the crossed grid G_8

$$S_i := \begin{cases} (i, i+l) \text{ and } (i+l, i) \text{ for even } l,\ l \geqslant 0 \\ (i+1, i+l) \text{ and } (i+l, i+1) \text{ for even } l,\ l < 0 \\ (i+1, i+l) \text{ and } (i+l, i+1) \text{ for odd } l,\ l \geqslant 0 \\ (i, i+l) \text{ and } (i+l, i) \text{ for odd } l,\ l < 0 \end{cases}$$

with $1 \leqslant i + l \leqslant 2k$. The sets S_i for odd i are a model of a K_k of G_{2k}. Indeed they are disjoint, and for every odd i, j, $i < j$ we have $(i, j) \in S_i$ and $(i + 1, j) \in S_j$ (see Figure 5.1). □

Given a grid G, an edge $e \notin E(G)$ is said to be *non-planar* if the graph $G \cup \{e\}$ is not planar. We call a planar supergraph of a grid an *augmented grid*.

We now show that a grid together with many non-planar edges has a large clique minor. In [20], Robertson and Seymour give a result (7.4) similar to what we need. To reformulate it in a form closer to our terminology:

Lemma 11 *[20] For every integer m, there exists an integer $r(m)$ such that a graph obtained from an augmented grid H by adding $\binom{m}{2}$ non-planar edges in $(V(H) \times V(H)) \setminus E(H)$, which endpoints lie at distance at least $r(m)$ from $B(H)$ in H and are pairwise lying at distance at least $r(m)$ in H, has a K_m minor.*

Note that result (7.4) from [20] concerns walls rather than grids and is stated with a different distance function, but Lemma 11 can be deduced from there.

It is important that the extra edges are non-planar in Lemma 11, and that they lie far away from the border of the grid. The fact that the extra edges are long is actually not necessary, as stated in the following stronger result.

Lemma 12 *For every integer m, there exists two integers $r(m)$ and $f_3(m)$ such that a graph obtained from an augmented grid H by adding $f_3(m)$ non-planar endpoint disjoint edges in $(V(H) \times V(H)) \setminus E(H)$, which endpoints lie at distance at least $r(m)$ from $B(H)$ in H, has a K_m minor.*

Proof: In the following we let:

$$f_3(m) = 2 \cdot 8^{2(r(m)+m)} \binom{m-1}{2} + 2 \cdot 8^{r(m)} \binom{m}{2}$$

If $2 \cdot 8^{r(m)} \binom{m}{2}$ non-planar edges have length at least m in the grid, then there exists a subset of size $\binom{m}{2}$ of these non-planar edges which endpoints lie pairwise at distance at least $r(m)$ in the grid. By Lemma 11, it follows

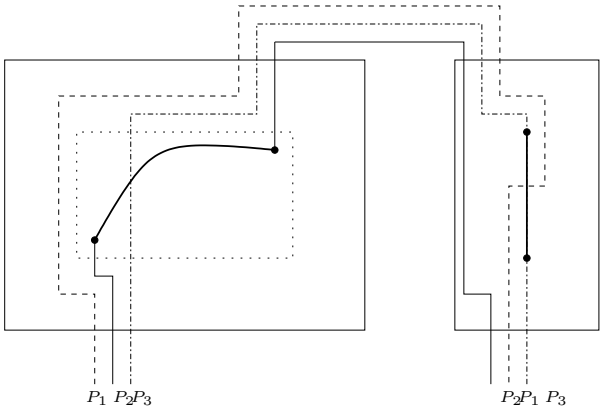


Figure 4: Illustration of the swaps made within two consecutive zones in the Proof of Lemma 12.

that H has a clique minor of size m. Otherwise, there exists a subset X of at least $f_3(m) - 2 \cdot 8^{r(m)}\binom{m}{2} \geq 2 \cdot 8^{2(r(m)+m)}\binom{m-1}{2}$ non-planar edges of length at most $r(m)$. Hence there exists a subset $S \subseteq X$ of non-planar edges which endpoints lie pairwise at distance at least $2(r(m)+m)$ of size at least $\binom{m-1}{2}$. For every edge $e = xy \in S$, let S_e be the minimal subgrid containing both x and y. The *zone* Z_e of e is the subgrid obtained from S_e by adding m layers around S_e. Note that for any two edges $e, e' \in S$ we have $Z_e \cap Z_{e'} = \varnothing$. There exists a set of m vertex disjoint paths $P_1, \ldots, P_m$ which are parallel and adjacent outside the zones and enter all zones in the grid exactly once.

A zone can be used to exchange two adjacent paths, as shown in Figure 4.

- use the $(m-1)$ first encountered zones to place P_m before P_1
- use the $(m-2)$ next zones to place P_{m-1} before P_1
- ...
- use the $(m-i)$ next zones to place P_{m-i} before P_1
- ...
- use the next zone to place P_2 before P_1

We have $\frac{m(m-1)}{2}$ zones, enough to do the $\sum_{i=1}^{m-1} i = \frac{m(m-1)}{2}$ above swaps, and a given zone is large enough to make two adjacent paths cross within it. We have reversed the original planar left-to-right order of paths P'_i, $i \in \{1, \ldots, m\}$ by making consecutive swaps, hence any two paths have crossed and hence are adjacent in the grid. Thus paths P_i, $i \in \{1, \ldots, m\}$ form a K_m minor model. □

Given a grid model H we say that $H' \subseteq H$ is a *proper subgrid* of H if it satisfies the following properties:

- $|V(H')| \geq f_4(k)$.
- H' is an induced augmented grid.
- There are at most $f_5(k)^2$ vertices in $V(G) \setminus V(H')$ with neighbours in $V(H') \setminus B(H')$. Such vertices are called *special vertices* and are denoted by W.
- Each special vertex has at least $k^2/4$ neighbours in $V(H') \setminus B(H')$.

Theorem 4 *Let G be a graph admitting an $(f_6(k) \times f_6(k))$-grid model H, with $f_6(k)$ large enough, but no $K_{f_2(k)}$ minor. There exists a proper subgrid which can be computed in polynomial time.*

Proof: We first show a technical result that will be used in the proof of Theorem 4. Given a grid model H of a graph G and an induced augmented grid H' which lies at distance at least $r(f_2(k))$ from $B(\bar{H})$ in $\bar{H}$, an edge wz where $z \in V_G(H') \setminus V_G(B(H'))$ lies at distance at least $r(f_2(k))$ of the border of H' in H' and $w \in V_G(H \setminus H')$ is called a *H'-matched edge.*

Lemma 13 *Let G be a graph admitting a grid model H with an induced augmented grid H' of size at least $f_4(k)$ which lies at distance at least $r(f_2(k)+1)$ in $\bar{H}$ from the border of $\bar{H}$. If there exist at least $f_5(k)$ endpoint disjoint H'-*matched edges *in H, with $f_5(k)$ large enough, then one of the following holds:*

(1) either G admits a grid model H_1 and an induced augmented grid $H'_1 \subseteq H_1$ such that $|H'_1| \geqslant f_4(k)$ and every set of endpoint disjoint H'_1-matched edges has size at most $f_5(k)$),

(2) or G has an $f_2(k)$ clique minor.

Proof: In the following we assume that all considered functions are large enough to prove the results. The following result follows by definition of a minor:

Observation 1 *Let G be a graph admitting a grid model H with an induced augmented grid H' such that there exist p paths internally in $G \setminus V_G(H')$ which are vertex-disjoint, between pairs of vertices $z_i, z'_i \in V_G(H')$ at distance at least $r(f_2(k))$ from the border of H' in H', $1 \leq i \leq p$. If edges $z_i z'_i$ are non-planar in H' for $i = 1, \ldots, p$, then G has as a minor a grid $\check{H}$ with p non-planar edges which endpoints lie at distance at least $r(f_2(k))$ from the border of $\check{H}$ in $\check{H}$.*

Let $W = \{w_1, \ldots, w_p\}$ be the endpoints of the H'-matched edges that belong to $V_G(H \setminus H')$, and assume $|W| \geqslant f_5(k)$. Since the graph $G \setminus V_G(H')$ is connected, let T be a tree spanning W in $G \setminus V_G(H')$. We can assume that this tree where parts have been contracted is a path, since $H \setminus V_G(H')$ has a hamiltonian path. Moreover, we can assume that any vertex of a given part has at most one neighbour in T in each of its two adjacent parts. Indeed, assume that a vertex v in a part h has more than one neighbour in T in one of its adjacent part h'. We choose any vertex $u \in N_{h'}(v)$ and remove all edges between v and $N_{h'}(v) \setminus u$ from T. We now use the connectivity of h' to find a spanning tree of all vertices in $N_{h'}(v)$, which gives a new spanning tree T' having the claimed property. Finally, we choose such a tree T with minimum maximum degree. We again distinguish two cases.

Case 1: T has degree bounded by $f_8(k)$ We use the following Lemma to exhibit a large clique minor in G:

Lemma 14 *Let T be a tree of degree at most d. Let W be a set of nodes of T. There exist $\frac{|W|-1}{d+1}$ vertex-disjoint paths in T with distinct endpoints in W.*

Proof: We root T arbitrarily. The hypothesis trivially holds for $|W| = 1$, so assume that $|W| \geq 2$. Among all pairs of vertices in W, we choose one pair (x, y) which least common ancestor u is minimum for the ancestor relation. Let T_u be the subtree of T rooted at u. As u is minimum, each subtree rooted in a son of u contains at most one vertex of W. Hence, as u has degree at most d, T_u contains at most $d+1$ vertices of W. By induction on $T \setminus T_u$, there exist $\frac{|W|-(d+1)-1}{d+1} = \frac{|W|-1}{d+1} - 1$ vertex-disjoint paths in $T \setminus T_u$ with distinct endpoints in W. We add to these paths the disjoint path $(x, y) \subseteq T_u$ to obtain the desired bound. □

Applying Lemma 14 to T and W, we find $\frac{f_5(k)-1}{f_8(k)+1}$ (which we assume to be greater than $f_3(f_2(k))$) vertex-disjoint paths between distinct vertices $w_{i,j}$. Using Lemma 1, G has as a minor a grid with $f_3(f_2(k))$ endpoint disjoint non planar edges with endpoints at distance at least $r(f_2(k))$ from the border of H'. Using Lemma 12, we deduce that G has a $K_{f_2(k)}$ minor.

Case 2: T has a node u of degree more than $f_8(k)$ Assume that u has maximum degree in T. As T where each part has been contracted to a vertex is a path, u has at least $f_8(k) - 2$ neighbours in T lying in its own part h_u. Among these vertices, we distinguish at most 4 of them, which are vertices connecting h_u to its four adjacent parts in $\bar{H}$. As T has minimum maximum degree, there must exist a set $W_u \subseteq W \cap h_u$ of size at least $f_8(k) - 6$ such that there exist vertex disjoint paths between u and vertices in W_u. For any $w = w_i \in W_u$, we use z_w to denote the corresponding vertex z_i, Z_w to denote the part of z_w, and let A_w be the set of vertices disconnected from u in T by the deletion of w. Note that $A_w \cup W_u = \emptyset$ since u has vertex disjoint paths to vertices in W_u. Let us denote by T_w the set of neighbours of $A_w \cup \{w\}$ distinct from z_w in H'.

Assume that for some vertex $w \in W_u$, there does not exist any part b_w containing a vertex in T_w such that $b_w Z_w$ would be a non-planar edge in H'. We change the partition H by removing $A_w \cup \{w\}$ from part h_u and adding it to part Z_w.

The resulting partition is still a grid model of G since all parts are still connected sets in G and the four distinguished vertices connecting h_u to its four adjacent parts in $\bar{H}$ are still in h_u. This also implies that H' still lies at distance at least $r(f_2(k))$ from the border of $\bar{H}$ in $\bar{H}$. Moreover, H' is still an induced augmented grid of H since no non-planar edge has been created in H'. We go back to the beginning of the proof of Lemma 13 with this new grid model H and augmented subgrid H' of G. We cannot loop more than n times as the number of vertices of G contained in $V_G(H')$ strictly increases. When this process terminates, if we did not find either a clique minor of G or a grid model H with an induced augmented grid H' containing at most $f_5(k)$ endpoint disjoint matched-edges towards vertices in $V_G(H \setminus H')$, then we are still in Case 2. Thus, for every vertex $w \in W_u$ there exists a vertex in $A_w \cup \{w\}$ with a neighbour z'_w in a part $Z'_w \in V(H')$ such that $Z_w Z'_w$ would be a non-planar edge in H'. Hence there exist paths P_w between z_w and z'_w for $w \in W_u$ which are vertex disjoint in h_u, as $A_w \cap A_{w'} = \emptyset$ for $w \neq w' \in W_u$.

Let $Z = \{Z_w | w \in W_u\}$ and $Z' = \{Z'_w | w \in W_u\}$. Note that by construction $Z_w \neq Z_{w'}$ for $w \neq w' \in W_u$. Consider the bipartite graph $B = (Z' \cup W_u, E)$, where $E := \{Z'_w w | w \in W_u\}$.

Claim 8 *Every part in Z' has degree at most $f_9(k)$.*

Proof. Assume by contradiction that there exists a part h in Z' of degree at least $f_9(k)$. Observe that a vertex $x \in h$ cannot have $k+1$ distinct neighbours in W_u, as there would exist $k+1$ edge-disjoint paths between u and x, which contradicts the fact that the instance is reduced under Rule 1. Let $Z'' = \{z'_w | w \in W_u\} \cap h$. Since h is connected, consider a tree T_h of minimum maximum degree spanning Z'' in h.

Assume first that T_h has minimum degree bounded by $k+1$. We apply Lemma 14 to Z'' and T_h to find $\frac{f_9(k)}{k \cdot (k+1+1)}$ (which we assume to be greater than $f_3(f_2(k)+1)$) vertex disjoint paths in T_h between vertices in Z''. This gives $f_3(f_2(k)+1)$ vertex disjoint paths in $G' := (G \setminus V_G(H') \cup \{h\}$ between vertices in Z (see Figure 5). Consider $H'' := (H' \setminus \{h\}) \cup h'$, where h' is a new part having the same adjacency with H' than h. As H'' defines an induced augmented grid, we can apply Lemma 1 to H'' and the above $f_3(f_2(k)+1)$ vertex-disjoint paths of G' to find a grid with $f_3(f_2(k)+1)$ non-planar endpoint disjoint edges with endpoints at distance at least $r(f_2(k)+1)$ from the border of H'. By Lemma 12, the graph H'' has a $K_{f_2(k)+1}$-minor, implying that G has a $K_{f_2(k)}$ minor.

Assume now that T_h has a vertex u_h of degree at least $k+1$. Assume that u_h has maximum degree in T_h. In particular, this means that there exists $k+1$ vertices of Z' with vertex disjoint paths to u_h in T_h. As we may assume $\frac{f_9(k)}{k} \geq k+1$, there exist $k+1$ edge-disjoint paths between u and u_h, which contradicts the fact that the instance is reduced under Rule 1.

This completes the proof of Claim 8. ◇

By Claim 8 we can greedily find $f_{10}(k) = \frac{|W_u|}{f_9(k)}$ vertex-disjoint paths in $V(G) \setminus H'$ between distinct vertices of H' of the form $z_i w_i z'_i$. By Observation 1, G admits as a minor a grid $\bar{H}$ with at least $\frac{f_8(k)-6}{f_9(k)}$ (assumed to be greater than $f_3(f_2(k))$) endpoint disjoint non-planar edges lying at distance at least $r(f_2(k))$ from the border of $\bar{H}$ in $\bar{H}$. Together with Lemma 12, this implies that G admits a $K_{f_2(k)}$-minor. This completes the proof of Lemma 13.

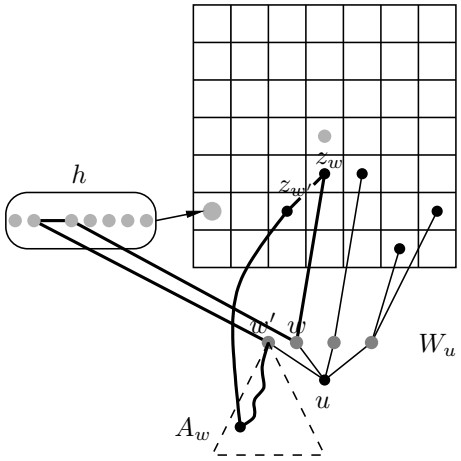


Figure 5: Illustration of the vertex disjoint paths found in the proof of Claim 8.

□

Proof of Theorem 4. Once again we assume that the considered functions are large enough for our purpose. We need the following technical result:

Lemma 15 *Let H be an $(m \times m)$-grid and Z be a set of i^2 vertices of H. Then H contains an $(\lfloor \frac{m}{i+1} \rfloor \times \lfloor \frac{m}{i+1} \rfloor)$-subgrid that does not contain any vertex from Z.*

Proof: We refine H by considering $(i+1)^2$ vertex-disjoint subgrids of size $\lfloor \frac{m}{i+1} \rfloor \cdot \lfloor \frac{m}{i+1} \rfloor$ each. As there are more grids than elements of Z there exists one such subgrid containing no vertex from Z, which is of size $\lfloor \frac{m}{i+1} \rfloor \cdot \lfloor \frac{m}{i+1} \rfloor$. □

Consider the refinement of the subgrid obtained from H by removing its $r(f_2(k))$ first layers in $\bar{H}$, in vertex disjoint subgrids $H_{i,j}$, for $1 \leq i, j \leq 2f_2(k)$ of size $\frac{f_6(k)}{2f_2(k)} \cdot \frac{f_6(k)}{2f_2(k)}$ each.

Claim 9 *If every subgrid $H_{i,j}$ is a non-planar graph, then G has the $(2f_2(k) \times 2f_2(k))$-crossed grid as a minor.*

Together with Lemma 10, Claim 9 implies that there exist $1 \leq i, j \leq 2f_2(k)$ such that $H_{i,j}$ is an induced augmented grid. In the following, we use H' to denote $H_{i,j}$.

Claim 10 *If every set of H'-matched edges has size at most $f_5(k)$ then G admits a proper subgrid model.*

Proof. Recall that an H'-matched edge is an edge wz where $z \in V_G(H') \setminus V_G(B(H'))$ lies at distance at least $r(f_2(k))$ of the border of H' in H' and $w \in V_G(H \setminus H')$. Let $Out = V(G) \setminus V_G(H')$ denote the set of vertices of G outside the grid H', and In the set of parts of $V(H')$ lying at distance at least $r(f_2(k))$ from the border of H' in H'. Let B be the bipartite graph with vertex bipartition (Out, In), and where a part $h \in In$ is adjacent to a vertex $x \in Out$ if there exists a vertex $y \in h$ such that xy is an edge of G. Denote by A the set of parts in In with degree at least $f_5(k)$ in B. Since every set of H'-matched edges has size at most $f_5(k)$, we have that $|A| < f_5(k)$. Hence there exists a subgrid $H'' \subseteq In$ of size at least $\frac{f_6(k)}{\sqrt{|f_5(k)|}}$ which contains no part of A by Lemma 15. Let C be the set of parts in H'' with degree at least 1 in B. If $|C| \leq f_5(k)$, Lemma 15 gives a subgrid H''' of H'' of size at least $\frac{f_6(k)}{f_5(k)}$ (which we assume to be greater than $f_4(k)$) which contains no part in C. Parts in H''' have no neighbours in $V(G) \setminus V_G(H')$, hence H''' is a

proper subgrid. Otherwise if $|C| > f_5(k)$, let D be the set of vertices in Out with at least one neighbour in H''. We have by hypothesis that $|D| \leqslant f_5(k)^2$ (otherwise we would find a large set of H'-matched edges).

Hence there exists a grid model with a subgrid H'' that respects all properties of the definition of a proper subgrid but the last one. We now show how to deal with the special vertices. Let $H_0 = H''$. For $i \geq 0$, if there exists a vertex in Out adjacent in B to a set P_i of parts in H_i of size less than $k^2/4$, then we let H_{i+1} be the subgrid of H_i obtained by Lemma 15 which contains no vertex in P_i. Otherwise H_i is a proper subgrid and we stop. The above process can be repeated at most $D < f_5(k)^2$ times. Hence the proper subgrid found by this process has size at least $\frac{f_6(k)}{f_5(k)^{f_5(k)}}$ (assumed to be greater $f_4(k)$). $\diamond$

To complete the proof of Theorem 4, it remains to show that we can find a subgrid H_m of sufficiently large size with at most $f_5(k)$ H_m-matched edges. Assume there are at least $f_5(k)$ H'-matched edges. Using Lemma 13, this implies either that G has a large clique minor or that G admits a grid model H with an induced augmented grid H_m having at most $f_5(k)$ H_m-matched edges. Since the former case is impossible by assumption, it follows by Claim 10 that G admits a proper subgrid model. This completes the proof of Theorem 4.

$\square$

5.2 Reducing the instance

Assume now that G has an $(f_6(k) \times f_6(k))$ grid model H but no $f_2(k)$-clique model. By Theorem 4 we can moreover assume that G has a proper subgrid H'. Given a set F of edges, a *giant component* denotes in this case a connected component of $G \setminus F$ containing entirely at least $|V(H)| - k^2/4$ parts. We do not give detailed proofs of our claim when they are similar to claims in the previous section.

Lemma 16 *Let F be a set of at most k edges of a graph G admitting a grid model. Then $G \setminus F$ has a* giant component*, denoted by C_F.*

This is analogous to Lemma 6. The way to cut the most vertices in a grid with k edges is to cut off a corner of size $(k/2 \times k/2)$. We call the other connected components the *small components* (and they intersect altogether at most $k^2/4$ parts).

Lemma 17 *For every set F of at most k edges the special vertices belong to the giant component of $G \setminus F$.*

This is analogous to Lemma 7.

Lemma 18 *Let F be an optimal k-multicut of a graph G admitting a grid model H. Every small component C of $G \setminus F$ contains a vertex which is a terminal.*

This is analogous to Lemma 8.

Let us now consider the refinement of the subgrid H' into vertex-disjoint subgrids $H'_{i,j}$ of size $f_7(k) \cdot f_7(k)$ each, where $f_7(k)$ is large enough.

5.2.1 There exists a subgrid without terminal

Let $1 \leqslant i, j \leqslant f_7(k)$ be such that $H'_{i,j}$ does not contain any terminal, and $h_{i,j}$ be a part of $H'_{i,j}$ at maximum distance in $H'_{i,j}$ of the border of the grid $H'_{i,j}$. We define $x_{i,j}$ to be a vertex of $h_{i,j}$ and $y_{i,j}$ to be any neighbour of $x_{i,j}$ in G.

Claim 11 *For every k-multicut F, $x_{i,j}$ and $y_{i,j}$ both belong to the giant component of $G \setminus F$.*

Proof. Recall that every small component contains a terminal and intersects at most $k^2/4$ parts, by Lemma 16 and Lemma 18. If part $h_{i,j}$ lies at distance more than $k^2/4+1$ in H of a part containing a terminal, then $x_{i,j}$ has no path in G to a terminal intersecting at most $k^2/4+1$ parts, and so $x_{i,j}$ and $y_{i,j}$ must both belong to the giant component of any k-multicut. Assume that there exists a path in G between $x_{i,j}$ and a terminal intersecting at most $k^2/4+1$ parts. This path uses an edge $z_{i,j}w_{i,j}$ with $z_{i,j} \in H'_{i,j}$ for some special vertex $w_{i,j}$. Indeed $H'_{i,j}$ has size greater than $2(k^2/4+2)$, so $h_{i,j}$ is at distance more than $k^2/4+1$ from the border of $H'_{i,j}$, and $H'_{i,j}$ contains no terminal. Since the special vertices belong to the giant component of $G \setminus F$ for every set F of at most k edges by Lemma 17, it follows that $x_{i,j}$ and $y_{i,j}$ belong to the giant component of $G \setminus F$. ◇

Since $x_{i,j}$ and $y_{i,j}$ belong to the giant component of $G \setminus F$ for every k-multicut F, we can safely contract them.

5.2.2 All subgrids contain a terminal

We find a gathered set using arguments similar to the ones used in the previous section. Consider the level decomposition of $G \setminus W$ starting from $L_0 := (H \setminus (H' \cup W)) \cup B(H')$. Since H' is a proper subgrid and hence contains no non-planar edges, if follows that there are at least $\frac{f_4(k)}{2}$ levels in the level decomposition of $G \setminus W$ starting from L_0. Let T be a set of maximum size of terminals in $V_G(L_1 \cup \ldots \cup L_d)$ belonging to different parts lying at distance at least $k^2/4$ in $H_{>1} := V_H(L_1 \cup \ldots \cup L_d)$. Since every subgrid $H'_{i,j}$ in H' contains a terminal and has size $f_7(k) \cdot f_7(k)$ there is a terminal in every $3f_7(k)$ consecutive levels, implying that T has sufficiently large size to apply Rule 3.

We now show that T is a gathered set. Consider any set F of at most k edges. Since parts containing terminals of T are pairwise at distance at least $k^2/4$ in $H_{>1}$ and since small components intersect at most $k^2/4$ parts by Lemma 16, a small component of $G \setminus F$ can contain two terminals of T only if it contains a special vertex $w \in W$. Since w belongs to the giant component by Lemma 17, it follows that $G \setminus F$ contains at most one component (the giant component) containing more than one vertex of T and hence T is a gathered set. Using Rule 3, there exists an irrelevant request adjacent to a terminal in T which can be found in FPT time.

This completes the reduction of this section, showing that we can reduce in FPT time in k a graph with a large grid minor but with no large clique minor.

6 FPT algorithm for Integer Weighted Multicut In Trees

Multicut is known to be FPT on trees [13], and even to have a polynomial kernel [1]. One of the simplest subcases of Multicut left open is that of trees with multi-edges [1, 13]. In this section we give a fixed-parameter algorithm for the integer weighted case of Multicut In Trees, defined as follows. Given a tree $T = (V, E)$, a set of requests P, a weight function $\omega : E \to \mathbb{N}^+$ and an integer k, decide if there is a multicut S such that $\omega(S) \leqslant k$, where the weight of a set of edges is defined to be the sum of the weights of its edges.

In [14], Guo *et. al.* gave an algorithm solving this problem in time $O(3^d.mn^2)$, where the parameter d is the maximum number of requests going through a node or an edge of the given tree. To our knowledge, no fixed-parameter algorithm for such a problem when parameterized by the size of the multicut is known. We provide such an algorithm, using branching and reduction rules, running in time $O^\star((2k+1)^k)$.

Rule 5 *If there exists a request (x, y) of length at most $2k+1$, then branch on every edge e in the path $P_{x,y}$ between x and y in T, adding e to the multicut and letting $k = k - \omega(e)$.*

We exhaustively apply branching Rule 5, answering FALSE if $k < 0$. Note that this branching rule can be applied at most k times since every application strictly decreases the parameter. If the resulting instance has no request and $k \geq 0$ we answer TRUE. If there exists an edge of weight at least $k+1$ we contract it.

Then we use the following reduction rule:

Rule 6 *Let $P = \{u_1, \ldots, u_p\}$ be a path of maximum length in T and let $e_i = u_i u_{i+1}$ for all $1 \leqslant i \leqslant k+1$. If there exists i such that $1 \leqslant i \leqslant k+1$ and $\omega(e_i) \geqslant \omega(e_{i+1})$, then we can contract e_i.*

Proof: Observe that P has length at least $2k+2$ since T is reduced under Rule 5. First of all, we show that if there is a request (x, y) with $x = u_i$ for $1 \leqslant i \leqslant k+1$, then $P_{x,y}$ contains $\{u_i, \ldots, u_{k+2}\}$. Assume by contradiction that there exists a request (u_i, y) with $1 \leqslant i \leqslant k+1$ that does not go through u_{k+2}. As we assumed that all requests have length at least $2k+1$, we know that the distance between u_i and y in T must be at least $k+1$, that is $|P_{y,u_i}| > k$. But then the path $P_{y,u_i} \cup (P \setminus \{u_1, \ldots, u_{i-1}\})$ is a path of length larger than P, which is a contradiction. Thus any request than contains e_i also contains e_{i+1}. As $w(e_i) \geq w(e_{i+1})$, this implies that there always exists an optimal k-multicut not containing e_i, and thus it is safe to contract e_i. □

When none of Rule 5 and Rule 6 apply, if there still exists a request, then the longest path of T has length more than $2k+1$. Denote by $(e_i)_{1 \leq i \leq i+2}$ the first $i+2$ edges of this path. We have $w(e_{i+1}) > w(e_i)$ for $1 \leq i \leq i+1$, hence $w(e_{k+2}) \geq k+1$, and it is safe to contract e_{k+2}. Hence when this branching/reduction process stops after at most $O^\star((2k+1)^k)$ steps.

Conclusion

Deciding whether the Multicut problem parameterized by the solution size k is fixed parameter tractable is one of the most important open question in parameterized complexity [5]. In this paper we have shown that this problem can be reduced in FPT time to graphs of treewidth bounded by a function of k. Hence the main open question that remains is whether it is fixed parameter tractable to decide the Multicut problem in graphs of treewidth bounded by a function of k. As a first step, we proposed an algorithm for the Integer Weighted Multicut In Trees problem, which is a particular subcase of Multicut for graphs of bounded treewidth. We believe that Multicut with this parameterization should be fixed parameter tractable on graphs of treewidth bounded in k. We thus conclude our paper with the following:

Conjecture 1 *The* Multicut *problem parameterized by the solution size k is fixed-parameter tractable on graphs of treewidth bounded in k.*

Aknowledgements. We thank Gwenaël Joret for pointing out the proof of Lemma 10.

References

[1] N. Bousquet, J. Daligault, S. Thomassé, and A. Yeo. A polynomial kernel for multicut in trees. In *STACS*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 183–194, 2009.

[2] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Comput. Complex.*, 15(2):94–114, 2006.

[3] J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[4] M.-C. Costa, L. Létocart, and F. Roupin. Minimal multicut and maximal integer multiflow: A survey. *European Journal of Operational Research*, 162(1):55–69, 2005.

[5] E. D. Demaine, M. Hajiaghayi, and D. Marx. 09511 open problems – parameterized complexity and approximation algorithms. In *Parameterized complexity and approximation algorithms*, number 09511 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010.

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[7] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer-Verlag New York Inc, 2006.

[8] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *STOC*, pages 698–707, 1993.

[9] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[10] G. Gottlob and S. Tien Lee. A logical approach to multicut problems. *Inf. Process. Lett*, 103(4):136–141, 2007.

[11] S. Guillemot. FPT algorithms for path-transversals and cycle-transversals problems in graphs. In *IWPEC*, pages 129–140, 2008.

[12] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, and J. Uhlmann. Complexity and exact algorithms for multicut. In *SOFSEM*, pages 303–312, 2006.

[13] J. Guo and R. Niedermeier. Fixed-parameter tractability and data reduction for multicut in trees. *Networks*, 46(3):124–135, 2005.

[14] J. Guo and R. Niedermeier. Exact algorithms and applications for tree-like weighted set cover. *J. Discrete Algorithms*, 4(4):608–622, 2006.

[15] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[16] D. Marx, B. O'Sullivan, and I. Razgon. Treewidth reduction for constrained separation and bipartization problems. In *STACS*, 2010.

[17] D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 647–658. Springer, 2009.

[18] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[19] R. Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*. Oxford University Press, USA, March 2006.

[20] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B*, 63(1):65–110, 1995.

[21] N. Robertson, P.D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Combin. Theory Ser. B*, 62(2):323–348, 1994.

Abstract

In this thesis, we study the parameterized complexity of several NP-complete problems. More precisely, we study the existence of polynomial kernels for graph and relations modification problems. In particular, we introduce the concept of branches, which provides polynomial kernels for some graph modification problems when the target graph class admits a so-called adjacency decomposition. This technique allows us to obtain the first known polynomial kernels for the CLOSEST 3-LEAF POWER, COGRAPH EDITION and PROPER INTERVAL COMPLETION problems. Regarding relations modification problems, we develop and push further the concept of Conflict Packing, a technique that has already been used in a few parameterized problems and that provides polynomial kernels for several problems. We thus present a linear vertex-kernel for the FEEDBACK ARC SET IN TOURNAMENTS problem, and adapt these techniques to obtain a linear vertex-kernel for the DENSE ROOTED TRIPLET INCONSISTENCY problem as well. In both cases, our results improve the best known bound of $O(k^2)$ vertices. Finally, we apply the Conflict Packing technique on the DENSE BETWEENNESS and DENSE CIRCULAR ORDERING problems, obtaining once again linear vertex-kernels. To the best of our knowledge, these results constitute the first known polynomial kernels for these problems.

Keywords: *parameterized complexity, kernelization algorithms, graph modification problems, branches, conflict packing*

Résumé

Dans le cadre de cette thèse, nous considérons la complexité paramétrée de problèmes NP-complets. Plus précisément, nous nous intéressons à l'existence d'algorithmes de noyau polynomiaux pour des problèmes d'édition de graphes et de relations. Nous introduisons en particulier la notion de branches, qui permet d'obtenir des algorithmes polynomiaux pour des problèmes d'édition de graphes lorsque la classe de graphes cible respecte une décomposition d'adjacence. Cette technique nous permet ainsi d'élaborer les premiers algorithmes de noyaux polynomiaux pour les problèmes CLOSEST 3-LEAF POWER, COGRAPH EDITION et PROPER INTERVAL COMPLETION. Concernant les problèmes d'édition de relations, nous étendons la notion de Conflict Packing, qui a déjà été utilisée dans quelques problèmes paramétrés et permet d'élaborer des algorithmes de noyau linéaires pour différents problèmes. Nous présentons un noyau linéaire pour le problème FEEDBACK ARC SET IN TOURNAMENTS, et adaptons les techniques utilisées pour obtenir un noyau linéaire pour le problème DENSE ROOTED TRIPLET INCONSISTENCY. Dans les deux cas, nos résultats améliorent la meilleure borne connue, à savoir un noyau quadratique. Finalement, nous appliquons cette technique sur les problèmes DENSE BETWEENNESS et DENSE CIRCULAR ORDERING, obtenant à nouveau des noyaux linéaires, qui constituent les premiers algorithmes de noyau polynomiaux connus pour ces problèmes.

Mots clefs : *complexité paramétrée, noyau, édition de graphes, édition de relations, branches, conflict packing*

LIRMM — 161, rue Ada — 34095 Montpellier cedex 5 — France