Thèse de Doctorat en Informatique

-

Conservatoire National des Arts et Métiers

Laboratoire Cedric

# Multicoupes et sous-graphes induits : complexité et algorithmes

**Nicolas Derhy**

Membres du jury:

| | |
|---|---|
| **Marie-Christine Costa**<br>Professeur au CNAM Paris | Directrice de thèse |
| **Christophe Picouleau**<br>Professeur au CNAM Paris | Co-encadrant |
| **Frédéric Roupin**<br>Maître de conférences au CNAM Paris | Co-encadrant |
| **Cristina Bazgan**<br>Professeur à l'Université Paris Dauphine | Rapporteur |
| **Dominique de Werra**<br>Professeur à l'EPFL, Lausanne, Suisse | Rapporteur |
| **Cédric Bentz**<br>Maître de conférences à l'Université Paris XI | Examinateur |
| **Marc Demange**<br>Professeur à l'ESSEC | Examinateur |
| **Nicolas Trotignon**<br>Chargé de recherche CNRS à l'Université Paris Diderot | Examinateur |

soutenue publiquement le 4 décembre 2008

# Remerciements

Et voici venu le moment des remerciements. Si ce chapitre apparaît pour le lecteur en début de thèse, il a été rédigé très peu de temps avant la soutenance. J'y remercie les personnes qui m'ont accompagné ces trois dernières années et qui ont contribué, chacune à leur manière, à la bonne marche de cette thèse. Ne vous étonnez pas si certains noms reviennent plusieurs fois : ce n'est pas une étourderie de ma part. De plus, je tiens à indiquer que j'ai opté, la plupart du temps, pour un classement des noms par ordre alphabétique : si X est cité avant Y, cela ne veut absolument rien dire !

Tout d'abord, je voudrais remercier Marie-Christine Costa, Christophe Picouleau et Frédéric Roupin qui m'ont encadré durant ces trois années. Merci à eux pour ce temps passé ensemble que cela soit à réfléchir à l'étude de nouveaux problèmes, à travailler sur des parties de rédaction ou tout simplement pour les nombreuses discussions que nous avons pu avoir ensemble. La thèse est une expérience très enrichissante qui dépasse de loin le cadre universitaire et pendant laquelle vous m'avez beaucoup apporté.

Je souhaiterais ensuite remercier Cristina Bazgan et Dominique de Werra qui ont eu la gentillesse d'accepter d'être rapporteurs de cette thèse. Ils ne me connaissent pas ou peu mais ils ont accepté de relire en détail le manuscrit que vous tenez actuellement dans vos mains.

Enfin, j'exprime ma profonde reconnaissance à Cédric Bentz, Marc Demange et Nicolas Trotignon qui ont accepté de faire partie des examinateurs.

Dans un second temps, je voudrais également remercier les personnes qui m'ont fait confiance pendant ces trois années en acceptant de me confier certains travaux dirigés ou travaux pratiques. Je pense notamment à Maria-Virginia Aponte, Sourour Elloumi, Bernard Lemaire et Christophe Picouleau. L'enseignement est, pour moi, une composante importante du travail d'enseignant-chercheur et c'est en partie grâce à la confiance dont vous m'avez témoignée que j'ai pu autant l'apprécier.

Si ces trois années ont été aussi plaisantes, c'est aussi grâce à la bonne ambiance régnant au laboratoire CEDRIC et plus particulièrement au sein de l'équipe Optimisation Combinatoire. Je pense notamment à Alain Billionnet, Cédric Bentz, Marie-Christine Costa, Sourour Elloumi, Alain Faye, Amélie Lambert, Aurélie Le Maître, Christophe Picouleau, Agnès Plateau, Marie-Christine Plateau, Frédéric Roupin, Eric Soutif, Hélène Topart, Mathieu Trampont.

Je voudrais également saluer les personnes qui ont eu la "chance" de partager leur bureau avec moi et qui ont, tant bien que mal, réussi à me supporter : tout d'abord, Marie-Christine (Costa) et Eric (Gressier) pendant le stage de Master et les quelques mois que j'ai passé au second étage du 55. Puis Cédric Bentz, Aurélie Le Maître et Marie-Christine Plateau qui ont partagé mon quotidien pendant les deux premières années de thèse. Même si leur niveau à Tetris était plutôt décevant, ce fut un vrai plaisir de passer ces deux années en leur compagnie. Entre Cédric sans qui cette thèse n'aurait pas été ce qu'elle est, Marie qui est "la fille la plus sympa que je connaisse" et Aurélie dont nos parcours jumeaux nous ont permis de nous soutenir l'un l'autre dans toutes nos petites galères du quotidien ; je ne peux que reconnaître avoir été particulièrement gâté.

Hélène avait donc une tâche peu aisée à assumer dès son arrivée : elle devait prendre la succession de Marie et Cédric, partis pour de nouveaux horizons une fois leur diplôme de doctorat en poche... et elle a su se montrer à la hauteur des attentes placées en elle, n'hésitant pas à remettre en cause ma supériorité Tétriesque sur le bureau 12.B2.45 ! Enfin, une petite pensée pour certains voisins de l'accès 12 comme Sévan, Jean-Remy et Rodrigo...

Cette bonne ambiance au bureau s'est également prolongée lors de nombreuses sorties : restaurants, cinémas... ou pistes de ski ! Quelques personnes se sont jointes à nous lors de ces occasions, à commencer par les conjoint(e)s des uns et des autres ; Lucas, Sylvie, Thomas et Vincent ; mais également plusieurs autres personnes telles que Laurence, Dominique, Sévan et Sabine avec qui j'ai partagé pleins de bons moments.

Je termine cette partie des remerciements avec un grand merci à Nicolas (Trotignon) et Christophe qui se sont beaucoup investis dans mon travail et sans qui la seconde partie de ce mémoire n'aurait certainement jamais vu le jour.

Mais ces trois dernières années ne se résument pas aux personnes qui ont partagé mon quotidien au CNAM. En plus d'Aurélie, on retrouve quelques "rescapés" des années IIE : Olivier et Amandine que j'ai toujours plaisir à revoir, Vincent et Leslie qui sont devenus petit à petit des amis très proches. Je tiens à adresser un grand merci en particulier à Leslie grâce à laquelle j'ai pu devenir un "step-danseur" confirmé. J'ai découvert un nouveau groupe composé de personnes extras, un nouvel environnement, une nouvelle activité agrémentée de nombreuses sorties pendant lesquelles j'ai passé pleins de moments inoubliables avec eux et notamment Anne-Claire, Bertrand, Elsa, Isa, Julie, Hinanui, Marc, Gwen, Vincent, Leslie, Marie, Hélène, Pierre-Alain, Boris, Céline, Aurore, Cécile, Laetitia ainsi que nos "mentors" Caroline, David, Alice et Nadège. Que cela soit pour la soirée du 31, au ski, pour un anniversaire, au diabolo, au bowling, au restaurant, vous avez toujours réussi, chacun à votre manière, à me redonner le sourire.

Une petite pensée également pour Audrey - la néo-québecoise - et pour Marie - ma judoka préférée - que je n'ai pas eu l'occasion de beaucoup voir ces dernières années mais que je n'oublie pas pour autant ! Enfin, je voudrais

également adresser un merci tout particulier à Céline pour sa présence à mes côtés ces derniers mois.

Je termine ce premier chapitre par ma famille qui a su m'entourer pendant ces trois dernières années. Tout d'abord mes parents et mon frère (sans oublier Gémini) qui ne savent toujours pas de quoi parlent ce mémoire mais qui seront là pour moi ce jeudi 4 décembre. Je voudrais aussi adresser un énorme merci à mes grands-parents qui m'accueillent quasiment tous les samedis midis depuis de nombreuses années malgré les nombreux capots et mes larges victoires au rumi. Vous êtes toujours là pour moi quand j'en ai besoin, ce mémoire est donc aussi un peu le vôtre.

Enfin, un grand merci général à toutes les personnes qui d'une manière ou d'une autre ont contribué à ce que ce mémoire soit ce qu'il est aujourd'hui.

# Table des matières

# Table des figures

# Chapitre 1

# Introduction générale

Les travaux de cette thèse s'articulent autour de deux grandes classes de problèmes combinatoires issus de la théorie des graphes : les problèmes de multicoupes et les problèmes de recherche de sous-graphes induits. Ainsi, si ces deux problématiques ne sont pas directement voisines, leur étude s'inscrit dans un cadre commun, qui constitue le fil conducteur de nos travaux : on s'intéresse à la complexité algorithmique associée à la résolution de ces problèmes. En d'autres termes, on cherche à montrer leur $\mathcal{NP}$-complétude (ou $\mathcal{NP}$-difficulté) ou à exhiber des algorithmes polynomiaux de moindre complexité permettant de les résoudre.

En raison du grand nombre d'applications existantes, les problèmes de coupes et de multicoupes jouissent d'une certaine popularité dans le milieu scientifique notamment depuis la parution du papier fondateur de Ford et Fulkerson. En effet, cet article fournit un algorithme polynomial à la fois simple et efficace pour résoudre les problèmes de la coupe minimum et du flot maximum ([25]). Si de nombreuses recherches ont permis de perfectionner les différentes méthodes de résolution et d'apporter des améliorations notables concernant la compréhension et l'utilisation de ces deux problèmes ([1]), d'autres travaux se sont également focalisés sur l'étude de problèmes connexes ou de généralisations comme les problèmes de multicoupes. Ces derniers ont la particularité de ne pas nécessairement vérifier un certain nombre de propriétés importantes, utilisées dans la conception d'algorithmes polynomiaux pour le problème de la coupe minimum. La plupart de ces problèmes sont donc $\mathcal{NP}$-difficiles ([20]).

La recherche de sous-graphes induits a connu un cheminement bien différent au cours de ces cinquante dernières années. Certains problèmes tels que la recherche d'un ensemble stable de taille maximum ou la détermination d'une clique de taille maximum ont toujours fait partie des problèmes les plus étudiés en raison du nombre important d'applications existant au sein d'un panel relativement large de domaines ([51]). D'autres problèmes - dont notamment ceux étudiés dans cette thèse - sont issus de travaux plus récents liés en grande partie à la démonstration du théorème fort des graphes parfaits réalisée par Chudnovsky et al. en 2002 ([17]). On peut ainsi citer les problèmes de recherche

d'un sous-graphe induit qui soit un prisme ([46]), une pyramide ([15]) ou un thêta ([16]). Notons cependant que ces derniers sont des problèmes de décision concernant l'existence d'une certaine structure alors que pour les problèmes de cliques ou de stables, il est trivial de déterminer si un graphe en contient mais très difficile de déterminer le plus grand.

Après avoir donné l'ensemble des notations utilisées dans ce mémoire et rappelé certaines définitions essentielles de la théorie des graphes, nous décrirons précisément les différents problèmes étudiés. Puis, nous ferons un rapide survol de l'état de l'art ayant trait aux problèmes de multicoupes et aux problèmes de recherche de sous-graphes induits. Enfin, nous présenterons les principaux résultats que nous avons obtenus tout en détaillant l'organisation du mémoire.

## 1.1 Notations et notions de base en théorie des graphes

Nous commençons par rappeler quelques notations de la théorie des ensembles. Soit $A$ un ensemble fini. Nous notons $|A|$ sa cardinalité. Si $w : A \to \mathbb{N}^*$ est une fonction de poids des éléments de $A$ alors pour tout sous-ensemble $A'$ de $A$, nous posons $w(A') = \sum_{a \in A'} w(a)$. Enfin l'inclusion (au sens large) sera représentée par le symbole "$\subset$".

La quasi-totalité des problèmes évoqués dans ce mémoire sont issus de la théorie des graphes pour laquelle nous adoptons une terminologie classique (sauf mention explicite contraire) : tout graphe considéré est noté $G$. L'ensemble de ses sommets est représenté par l'ensemble $V$ et $n$ désigne le nombre d'éléments de $V$. L'ensemble des arêtes (ou des arcs si $G$ est orienté) est noté $E$ et son cardinal est égal à $m$. Par défaut, les graphes considérés sont non orientés et dans le cas contraire, l'orientation du graphe est explicitement mentionnée dans le texte. De plus, un graphe est dit *pondéré* s'il existe une fonction de poids $w : E \to \mathbb{N}^*$.

La plupart du temps, nous utilisons la lettre $e$ pour désigner une arête. Si une arête $e$ relie les sommets $u$ et $v$ alors nous la notons indifféremment $e_{uv}$ ou $u - v$ et on dira que $u$ et $v$ sont les *extrémités* de $e$. Dans le cas des graphes orientés, les arcs sont notés $a$. Si $a$ relie le sommet $u$ au sommet $v$, nous notons $u \to v$ cet arc et $u$ et $v$ sont également appelés *extrémités* de $a$. Nous dirons qu'une arête $e$ est *incidente* à un sommet $u$ si $u$ est une extrémité de $e$. De plus, deux arêtes $e$ et $e'$ sont *adjacentes* si elles ont une extrémité commune. Le *degré* d'un sommet $u$, noté $d^\circ(u)$ est alors égal au nombre d'arêtes adjacentes à $u$. Un *voisin* d'un sommet $u$ est un sommet $v$ tel que l'arête $u - v$ existe. Dans le cas des graphes orientés, $u$ a pour *successeur* (resp. *prédécesseur*) $v$ si l'arc $u \to v$ (resp. $v \to u$) existe.

Une *chaîne* est une suite d'arêtes adjacentes et un *chemin* est une suite d'arcs où aucun sommet n'a deux prédécesseurs ou deux successeurs et où l'on obtient une chaîne si l'on remplace chaque arc par une arête. Les chemins et les chaînes considérés seront toujours *élémentaires*, c'est-à-dire qu'ils ne contiennent

pas deux fois un même sommet. Si une chaîne (resp. un chemin) est composée de $k$ sommets $v_1, \ldots, v_k$ alors nous la notons $v_1 - \cdots - v_k$ (resp. $v_1 \to \cdots \to v_k$). Un *cycle* (resp. un *circuit*) est l'union d'une chaîne $v_1 - \cdots - v_k$ (resp. d'un chemin $v_1 \to \cdots \to v_k$) et de l'arête $v_k - v_1$ (resp. de l'arc $v_k \to v_1$). La *longueur* d'une chaîne ou d'un cycle (resp. d'un chemin ou d'un circuit) est égale à son nombre d'arêtes (resp. d'arcs). Un triangle est un cycle de longueur 3 et la *maille* d'un graphe est égale à la longueur de son plus petit cycle. Enfin, nous dirons que deux chaînes sont disjointes par les arêtes (resp. sommets) si elles n'ont aucune arête (resp. aucun sommet) en commun. Deux chaînes disjointes par les sommets sont donc également disjointes par les arêtes.

Tous les graphes considérés dans ce mémoire sont *finis*, *simples* et *sans boucle*, c'est-à-dire que l'ensemble $V$ est fini, qu'il n'y a pas d'arête (ou d'arc) parallèle et qu'il n'existe pas d'arête $u - u$ (ou d'arc $u \to u$).

Un graphe $G' = (V', E')$ est un *sous-graphe induit* de $G$ si $V' \subset V$ et $E' = \{e_{uv} \in E | u \in V' \text{ et } v \in V'\}$. De plus, si $A$ est un ensemble de sommets de $G$, le *graphe induit* par $A$ est le sous-graphe induit de $G$ dont l'ensemble des sommets est $A$. Enfin, notons que le terme *sous-graphe* désignera systématiquement, et de manière implicite, un *sous-graphe induit*.

Un graphe $G$ est *connexe* s'il existe au moins une chaîne entre toute paire de sommets de $V$. Les *composantes connexes* de $G$ sont les sous-graphes connexes de $G$ maximaux au sens de l'inclusion. Par convention, nous considérons que le graphe vide (c'est-à-dire $V = \emptyset$) est connexe. Sauf indication contraire, les graphes définis dans les chapitres suivants seront toujours connexes.

Un *arbre* est un graphe non orienté connexe sans cycle. La *hauteur* d'un arbre est égale à la longueur d'une plus longue chaîne diminuée de 1. Un *arbre orienté* est un graphe orienté qui devient un arbre si l'on remplace chaque arc par une arête. Une *feuille* est un sommet de degré 1 d'un arbre. Enfin, une *arborescence* est un arbre orienté qui possède un unique sommet $r$ tel qu'il existe un unique chemin de $r$ à tout autre sommet.

Nous allons maintenant donner la définition de plusieurs classes particulières de graphes que nous rencontrons fréquemment dans la suite de ce mémoire.

**Définition 1.1.** *Un graphe $G = (V, E)$ est biparti si $V$ peut être partitionné en deux ensembles $V_1$ et $V_2$ tels que chaque arête de $E$ ait une extrémité dans $V_1$ et l'autre extrémité dans $V_2$.*

Si $G$ est biparti, nous préférerons, le plus souvent, la notation $G = (V_1, V_2, E)$ à $G = (V, E)$ afin d'indiquer clairement au lecteur le caractère biparti du graphe considéré. Notons également qu'un résultat bien connu sur les graphes bipartis est qu'ils ne possèdent aucun cycle de longueur impaire (donc aucun triangle).

**Définition 1.2.** *Nous notons $K_{i,j}$ le graphe biparti $G = (V_1, V_2, E)$ tel que $|V_1| = i$, $|V_2| = j$ et $E$ contient toutes les arêtes possibles reliant un sommet de $V_1$ à un sommet de $V_2$.*

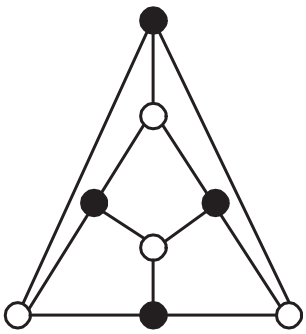

FIGURE 1.1 – Un exemple de graphe biparti cubique planaire (les couleurs noires et blanches indiquent la bipartition $(V_1, V_2)$)

**Définition 1.3.** *Un graphe $G = (V, E)$ est cubique si tous ses sommets sont de degré 3.*

**Définition 1.4.** *Un graphe $G = (V, E)$ est planaire s'il peut être dessiné dans le plan sans que deux ou plusieurs de ses arêtes ne se croisent ailleurs qu'en des sommets.*

Notons qu'un graphe peut être à la fois biparti, cubique et planaire comme le démontre la figure 1.1 (qui représente en fait un cube).

**Définition 1.5.** *Une étoile est un arbre de hauteur 1.*

De manière similaire aux arbres, une étoile orientée est une étoile où chaque arête a été remplacée par un arc.

Pour des notions et des définitions de théorie des graphes non présentes dans cette section, le lecteur est invité à consulter [6].

Nous pouvons désormais donner la définition de l'ensemble des problèmes considérés dans les chapitres suivants.

## 1.2 Définition des problèmes étudiés

### 1.2.1 Définition des problèmes de multicoupes

Dans cette section, nous définissons de manière formelle les problèmes évoqués dans nos travaux sur les multicoupes. Soit un graphe $G = (V, E)$ orienté ou non, soit une fonction $w : E \to \mathbb{N}^*$ et soit un ensemble $\mathcal{T} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ de $k$ paires de sommets distincts de $G$. Ces sommets sont appelés *terminaux* du graphe $G$. De plus, si $k = 1$ nous préférerons noter les terminaux $s$ et $t$ plutôt que $s_1$ et $t_1$. Enfin, pour tout $i \in \{1, \ldots, k\}$, soit $\mathcal{P}_i$, l'ensemble des chaînes (ou des chemins si le graphe est orienté) reliant $s_i$ à $t_i$ et soit $p$ un entier naturel donné.

Nous définissons tout d'abord la notion de multicoupe :

**Définition 1.6.** *Une* ***multicoupe*** *est un ensemble d'arêtes (ou d'arcs) de $G$ dont la suppression ne laisse aucune chaîne (ou chemin) entre $s_i$ et $t_i$ pour $i \in \{1, \ldots, k\}$.*

Le poids d'une multicoupe $C$ correspond à la somme des poids de ses arêtes et est donc égal à $w(C)$. La cardinalité d'une multicoupe $C$ est égale au nombre d'arêtes sélectionnées, c'est-à-dire à $|C|$.

Pour faciliter la lecture, nous adoptons les conventions suivantes pour la nomenclature des problèmes. La lettre "C" est présente dans la quasi-totalité des noms de problèmes pour indiquer qu'il s'agit d'un problème de coupe. De plus, nous utilisons les préfixes "Min", "$R$-Cri" et "Multi" pour indiquer respectivement un problème de minimisation, un problème de décision à $R$ critères et un problème pour lequel $k$ peut être strictement plus grand que 1. Nous utilisons le suffixe "Card" pour signaler que le problème contient une contrainte de cardinalité. Les préfixes et le suffixe ne sont pas forcément exclusifs : le problème MinMultiCCard consiste en la recherche d'une multicoupe minimum satisfaisant une contrainte de cardinalité.

**Définition 1.7.** ***Problème de la coupe minimum*** *(noté* MinC *par la suite) : étant donné un graphe $G = (V, E)$ et deux sommets $s$ et $t$, l'objectif est de déterminer une partition de $V$ en deux ensembles $V_1$ et $V_2$ telle que $s \in V_1$ et $t \in V_2$, tout en minimisant la somme des poids des arêtes (ou arcs) ayant une extrémité dans $V_1$ et une extrémité dans $V_2$.*

La coupe est constituée ici de l'ensemble des arêtes incidentes à la fois à un sommet de $V_1$ et à un sommet de $V_2$. En généralisant ce problème au cas où nous disposons de plusieurs paires source-puits, nous obtenons le problème suivant :

**Définition 1.8.** ***Problème de la multicoupe minimum*** *(noté* MinMultiC *par la suite) : l'objectif est de déterminer une multicoupe de poids minimum.*

On peut aisément montrer que MinC correspond en fait au cas particulier de MinMultiC obtenu en posant $k = 1$. Nous pouvons à présent introduire les deux problèmes principaux intégrant une contrainte de cardinalité :

**Définition 1.9.** ***Problème de la coupe minimum de cardinalité inférieure à une valeur donnée*** *(noté* MinCCard *par la suite) : l'objectif est de déterminer une coupe de poids minimum séparant $s$ de $t$ et dont la cardinalité est inférieure ou égale à $p$.*

**Définition 1.10.** ***Problème de la multicoupe minimum de cardinalité inférieure à une valeur donnée*** *(noté* MinMultiCCard *par la suite) : l'objectif est de déterminer une multicoupe de poids minimum dont la cardinalité est inférieure ou égale à $p$.*

Nous définissons également quelques variantes secondaires de ces deux derniers problèmes :

**Définition 1.11.** ***Problème de la coupe minimum de cardinalité égale à une valeur donnée*** *(noté* MinC=Card *par la suite) : l'objectif est de déterminer une coupe de poids minimum séparant $s$ de $t$ et dont la cardinalité est égale à $p$.*

**Définition 1.12.** ***Problème de la coupe minimum de cardinalité supérieure à une valeur donnée*** *(noté* MinC≥Card *par la suite) : l'objectif est de déterminer une coupe de poids minimum séparant $s$ de $t$ et dont la cardinalité est supérieure ou égale à $p$.*

**Définition 1.13.** ***Problème de la multicoupe minimum de cardinalité égale à une valeur donnée*** *(noté* MinMultiC=Card *par la suite) : l'objectif est de déterminer une multicoupe de poids minimum dont la cardinalité est égale à $p$.*

Nous verrons dans la section 1.3.1 que MinCCard et MinMultiCCard peuvent être vus comme les problèmes d'optimisation associés à deux problèmes particuliers de coupe bicritère et de multicoupe bicritère.

Soit $w_1, \ldots, w_R$ $R$ fonctions à valeurs entières strictement positives pondérant les arêtes de $G$ et soit $B_1, \ldots, B_R$ $R$ entiers positifs donnés. On définit alors les deux problèmes de décision suivants :

**Définition 1.14.** ***Problème de la coupe multicritère*** *(noté $R-$CriC par la suite) : l'objectif est de déterminer s'il existe une coupe $C$ séparant $s$ de $t$ telle que $w_i(C) \leq B_i$ pour $i \in \{1, \ldots, R\}$.*

**Définition 1.15.** ***Problème de la multicoupe multicritère*** *(noté $R-$CriMultiC par la suite) : l'objectif est de déterminer s'il existe une multicoupe $C$ telle que $w_i(C) \leq B_i$ pour $i \in \{1, \ldots, R\}$.*

Enfin, concluons cette section en définissant un problème de flot :

**Définition 1.16.** ***Problème du multiflot entier maximum*** *(noté* MaxMultiFlot *par la suite) : l'objectif est de router le nombre maximum d'unités de flot ; chaque unité étant routée de $s_i$ vers $t_i$ ($i \in \{1, \ldots, k\}$), de façon à ce que, pour chaque arête $e$ (resp. chaque arc $a$), le nombre total d'unités de flot qui la (resp. le) traversent soit inférieur ou égal à $w(e)$ (resp. $w(a)$).*

Nous verrons dans la section 1.3.1 les liens étroits qui relient ce dernier problème à MinMultiC.

### 1.2.2 Définition des problèmes de sous-graphes induits

Nous définissons maintenant les problèmes rencontrés dans la seconde partie du mémoire. Ces problèmes peuvent être divisés en trois catégories : les problèmes de recherche d'arbres induits, de chaînes induites et de cycles induits. Soit $G = (V, E)$ un graphe non orienté et soit $T = \{x_1, \ldots, x_k\}$ un ensemble de $k$ sommets distincts de $G$ appelés *sommets terminaux* de $G$.

A l'instar de la section précédente, nous adoptons les conventions suivantes pour la nomenclature des problèmes. Les préfixes "$k-$", "MIN" et "POND" indiquent respectivement que le problème considéré possède $k$ terminaux, que c'est un problème de minimisation et que c'est un problème de décision dans un graphe pondéré. Les mots-clés "ARBRE", "CHAINE" et "CYCLE" indiquent si l'on cherche un sous-graphe de $G$ qui soit un arbre, une chaîne ou un cycle. Enfin la lettre I, présente dans tous les noms de problème, est l'abréviation du mot "Induit".

Les trois premiers problèmes définis ci-dessous correspondent aux problèmes d'existence d'un sous-graphe de $G$ qui soit un arbre, une chaîne ou un cycle :

**Définition 1.17.** ***Problème d'existence d'un arbre induit couvrant les*** $k$ ***terminaux*** *(noté* $k-$ ARBREI *par la suite) : l'objectif est de déterminer s'il existe un ensemble de sommets* $A \subset V$ *tel que* $T \subset A$ *et le graphe induit par* $A$ *est un arbre.*

**Définition 1.18.** ***Problème d'existence d'une chaîne induite couvrant les*** $k$ ***terminaux*** *(noté* $k-$ CHAINEI *par la suite) : l'objectif est de déterminer s'il existe un ensemble de sommets* $A \subset V$ *tel que* $T \subset A$ *et le graphe induit par* $A$ *est une chaîne.*

**Définition 1.19.** ***Problème d'existence d'un cycle induit couvrant les*** $k$ ***terminaux*** *(noté* $k-$CYCLEI *par la suite) : l'objectif est de déterminer s'il existe un ensemble de sommets* $A \subset V$ *tel que* $T \subset A$ *et le graphe induit par* $A$ *est un cycle.*

Nous appellerons arbre induit, chaîne induite et cycle induit une solution admissible pour respectivement $k-$ ARBREI, $k-$ CHAINEI et $k-$ CYCLEI.

Nous définissons également les problèmes d'optimisation associés aux trois précédents problèmes consistant à minimiser le nombre d'arêtes (et donc de sommets) des solutions admissibles. En effet, si $n_A$ et $m_A$ sont respectivement le nombre de sommets et d'arêtes d'un sous-graphe $A$ de $G$, alors on a $n_A = m_A + 1$ pour un arbre et une chaîne, et $m_A = n_A$ pour un cycle.

**Définition 1.20.** ***Problème de minimisation d'un arbre induit couvrant les*** $k$ ***terminaux*** *(noté* $k-$MINARBREI *par la suite) : l'objectif est de déterminer un arbre induit de taille minimum.*

**Définition 1.21.** ***Problème de minimisation d'une chaîne induite couvrant les*** $k$ ***terminaux*** *(noté* $k-$ MINCHAINEI *par la suite) : l'objectif est de déterminer une chaîne induite de taille minimum.*

**Définition 1.22.** ***Problème de minimisation d'un cycle induit couvrant les*** $k$ ***terminaux*** *(noté* $k-$MINCYCLEI *par la suite) : l'objectif est de déterminer un cycle induit de taille minimum.*

$k-$MINARBREI, $k-$MINCHAINEI et $k-$MINCYCLEI peuvent être également vus comme la recherche d'un sous-graphe de poids minimum dans un graphe où chaque arête est pondérée par l'entier 1. Soit $w : E \to \mathbb{N}$ une fonction de poids sur les arêtes de $G$ et soit $B$ un entier naturel donné. Nous généralisons alors les trois problèmes précédents en définissant les problèmes de décision suivants :

**Définition 1.23.** ***Problème de recherche d'un arbre induit couvrant les $k$ terminaux et de poids inférieur à une valeur donnée*** *(noté $k-$*PondArbreI *par la suite) : l'objectif est de déterminer s'il existe un arbre induit dont la somme des poids de ses arêtes est inférieure ou égale à $B$.*

**Définition 1.24.** ***Problème de recherche d'une chaîne induite couvrant les $k$ terminaux et de poids inférieur à une valeur donnée*** *(noté $k-$*PondChaineI *par la suite) : l'objectif est de déterminer s'il existe une chaîne induite dont la somme des poids de ses arêtes est inférieure ou égale à $B$.*

**Définition 1.25.** ***Problème de recherche d'un cycle induit couvrant les $k$ terminaux de poids inférieur à une valeur donnée*** *(noté $k-$*PondCycleI *par la suite) : l'objectif est de déterminer s'il existe un cycle induit dont la somme des poids de ses arêtes est inférieure ou égale à $B$.*

## 1.3 Etat de l'art

Nous présentons dans cette section un aperçu des différents résultats existants pour les problèmes de multicoupes et de recherche de sous-graphes induits.

### 1.3.1 Problèmes de multicoupes

Les problèmes de coupes et de multicoupes ont de nombreuses applications concrètes comme par exemple, la fiabilité des réseaux de télécommunications. Ainsi, dans [38, Chapitre *Cut problems and their application to divide-and-conquer*, pages 192–235], Shmoys donne un cadre général d'applications pour MinMultiC et certaines de ses variantes : de nombreux problèmes combinatoires de théorie des graphes peuvent être résolus, de façon exacte ou heuristique, par des approches de type *diviser-pour-régner* (*divide-and-conquer* en anglais). Or, ces approches reposent, le plus souvent, sur un partitionnement judicieux des sommets ou des arêtes d'un graphe, qui se ramène souvent à des problèmes de multicoupes ([52]). On peut aussi citer des problèmes de reconnaissance des communautés virtuelles d'internet, de partitionnement dans des réseaux sociaux (ou assimilés, voir [48] par exemple) ou dans des graphes de dépendances de type processeurs-tâches pour des problèmes de placement de tâches sur un système multiprocesseur.

En ce qui concerne la complexité de ces problèmes, on sait depuis longtemps que le problème de la coupe minimum est polynomial ([25]). Cependant, ce résultat ne se généralise pas au problème de la multicoupe minimum qui, bien que polynomial dans les graphes non orientés pour $k = 2$ ([56]), est $\mathcal{NP}$-difficile au sens fort pour $k = 2$ dans les graphes orientés ([31]) et pour $k = 3$ dans les graphes non orientés ([21]).

Pour une valeur de $k$ non fixée, Dahlhaus et al. ont montré en 1994 dans [21] que MinMultiC est $\mathcal{NP}$-difficile au sens fort dans les graphes planaires (en fait, ils ont montré la $\mathcal{NP}$-difficulté du problème de la coupe multiterminale minimum qui est un cas particulier de multicoupes). Puis en 1996,

Garg et al. ont donné un résultat plus fort en démontrant que MinMultiC est également $\mathcal{NP}$-difficile au sens fort dans les étoiles même si tous les poids sont unitaires ([30]). Néanmoins, MinMultiC admet plusieurs cas polynomiaux comme les arbres orientés, les chaînes, les cycles et les circuits ([19, 20]).

Une des raisons expliquant les différences de complexité entre les problèmes de coupes et les problèmes de multicoupes vient du fait que certaines propriétés très intéressantes vérifiées pour $k = 1$ (c'est-à-dire pour une seule paire source-puits) disparaissent lorsque $k > 1$. Par exemple, les problèmes de coupe minimum et de flot maximum sont modélisables sous la forme de programmes linéaires en nombres entiers (PLNE en abrégé) dont la matrice des contraintes est *totalement unimodulaire* ([1]). Considérons alors les PLNE suivants, modélisant MinMultiC et MaxMultiFlot :

$$\text{(PLNEcoupe)} \begin{cases} Min & \sum_{e \in E} w(e) z_e & \\ \text{s. c.} & \sum_{e \in P} z_e \geq 1 & \forall P \in \mathcal{P}_i \ \forall i \in \{1, \ldots, k\} \\ & z_e \in \{0, 1\} & \forall e \in E \end{cases}$$

$z_e$ est la variable de décision valant 1 si l'arête $e$ est placée dans la coupe et 0 sinon. Rappelons que $w(e)$ est le poids de l'arête $e$ et que $\mathcal{P}_i$ est l'ensemble des chaînes (ou chemins) reliant $s_i$ à $t_i$ ($i \in \{1, \ldots, k\}$). La seule contrainte du problème indique qu'au moins une arête de chaque chaîne (ou chemin) reliant une source à son puits doit être sélectionnée dans la coupe.

$$\text{(PLNEflot)} \begin{cases} Max & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_P & \\ \text{s. c.} & \sum_{P/e \in P} f_P \leq w(e) & \forall e \in E \\ & f_P \in \mathbb{N} & \forall P \in \mathcal{P}_i \ \forall i \in \{1, \ldots, k\} \end{cases}$$

$f_P$ est égal au nombre d'unités de flot routés sur la chaîne (ou le chemin) $P$ et la contrainte indique que la somme des flots passant par une arête ne peut excéder sa capacité.

Tout d'abord nous remarquons que le nombre de contraintes de ($PLNEcoupe$) et le nombre de variables de ($PLNEflot$) peuvent être exponentiels en la taille du graphe puisque proportionnels au nombre total de chaînes (ou chemin) reliant les sources à leurs puits respectifs (cependant, il existe d'autres formulations sous forme de PLNE pour MinMultiC et MaxMultiFlot, pour lesquelles le nombre de variables et de contraintes est polynomial en la taille du graphe ([29])). De plus, nous remarquons que les relaxations continues de ces deux programmes linéaires sont *duales l'une de l'autre*.

Même si les matrices des contraintes de ($PLNEcoupe$) et de ($PLNEflot$) ne sont généralement pas totalement unimodulaires, la propriété de dualité est un

outil très intéressant. Dans [30], Garg et al. montrent que MinMultiC est $\mathcal{NP}$-difficile au sens fort dans les étoiles et que MaxMultiFlot est $\mathcal{NP}$-difficile au sens fort dans les arbres. Puis, ils utilisent la relation de dualité entre ces deux problèmes pour élaborer un algorithme fournissant à la fois une solution 2-approchée pour MinMultiC (c'est-à-dire une solution dont la valeur est inférieure à deux fois la valeur d'une multicoupe optimale) et une solution $\frac{1}{2}$-approchée pour MaxMultiFlot (c'est-à-dire une solution dont la valeur est supérieure à la moitié de la valeur d'un multiflot entier maximum) lorsque le graphe considéré est un arbre.

De plus, dans certains cas, la matrice des contraintes de ($PLNEcoupe$) et de ($PLNEflot$) est totalement unimodulaire, et le nombre de variables et de contraintes est polynomial en la taille du graphe. Dans [19], Costa et al. montrent que c'est le cas lorsque le graphe considéré est un arbre orienté : MinMultiC et MaxMultiFlot sont donc polynomiaux dans ce cas. Ils fournissent dans ce même papier un algorithme polynomial en $O(Min(nk, n^2))$ pour résoudre à la fois MinMultiC et MaxMultiFlot dans les arborescences.

Concernant l'approximation de MinMultiC, nous avons évoqué précédemment un algorithme 2-approché dans les arbres ([30]). Ce n'est pas le seul puisque d'autres algorithmes 2-approchés reposant sur une démarche d'arrondi d'une solution fractionnaire ont été proposés dans [33, 45]. De plus, Tardos et Vazirani ont également élaboré un algorithme $O(1)$-approché pour les graphes planaires non orientés ([55]) et Garg et al. ont proposé un algorithme $O(\log(k))$-approché pour les graphes non orientés ([29]). Enfin, Chawla et al. ont montré plus récemment qu'il n'existe pas d'algorithme $O(1)$-approché si la *Unique Games Conjecture* est vraie ([12]).

Dans les graphes orientés, un algorithme $O(\sqrt{n})$-approché pour MinMultiC a été proposé en 2003 par Gupta [35] (voir aussi [14, 43]). De plus, pour ce même problème un algorithme $O(k)$-approché trivial existe : il consiste à calculer, pour chaque $i$, une coupe minimum entre $s_i$ et $t_i$, puis à prendre l'union de toutes ces coupes.

La spécificité des problèmes de coupes et de multicoupes étudiés dans cette première partie du mémoire réside, en partie, dans l'ajout d'une *contrainte de cardinalité*. Ainsi, pour certains de ces problèmes comme MinCCard et MinMultiCCard, nous cherchons à déterminer une (multi)coupe de poids minimum parmi les (multi)coupes dont la cardinalité est inférieure ou égale à la valeur $p$ donnée. On peut retrouver ce type de contrainte dans de nombreux problèmes d'optimisation combinatoire comme par exemple le problème consistant à déterminer une partition des sommets d'un graphe en deux ensembles $V_1$ et $V_2$ telle que $|V_1| \leq B$, $|V_2| \leq B$ et de manière à minimiser le nombre arêtes ayant une extrémité dans $V_1$ et l'autre dans $V_2$ ([27]). On pense également au problème du $k$-Cluster qui consiste à déterminer un sous-graphe d'au plus $k$ sommets le plus dense possible ([9]). Enfin, citons le problème du $p$-Centre qui consiste à localiser au maximum $p$ entrepôts parmi $M$ sites possibles et à allouer chacun des $N$ clients à l'entrepôt le plus proche, de façon à ce que la

distance maximale entre un client et l'entrepôt qui le dessert soit minimale ([26]). Bruglieri et al. donnent également dans [11] une bibliographie de différents problèmes d'optimisation combinatoire mettant en jeu des contraintes de cardinalité.

Concernant plus spécifiquement des problèmes de coupes, Bruglieri et al. ont étudié dans [10] la conséquence de l'ajout d'une contrainte de cardinalité au problème de la coupe minimum. Ils donnent un ensemble de résultats de complexité pour MinC=Card et MinC≥Card mais laissent ouverte la complexité de MinCCard dans les graphes généraux.

Les autres résultats existants pour MinCCard sont en fait issus de l'étude de problèmes de *coupes multicritères*. En effet, le problème de décision associé à MinCCard (resp. MinMultiCCard) peut être vu comme un cas particulier de $2-$CriC (resp. $2-$CriMultiC) où l'on a $w_2(e) = 1$ pour toute arête $e$ du graphe. Ainsi, dans [50], Papadimitriou et Yannakakis ont montré que $2-$CriC est $\mathcal{NP}$-complet au sens fort. Dans [4], Armon et Zwick ont étudié des versions multicritères du problème de la coupe minimum dont une semblable à $R-$CriC à la différence qu'il n'y a pas de source et de puits donnés. Ils ont montré que ce problème est polynomial lorsque le nombre de critères est fixé mais devient $\mathcal{NP}$-complet lorsque ce n'est plus le cas. Enfin, Aissi et al. se sont focalisés dans [2] sur plusieurs versions multicritères du problème de la coupe minimum en considérant le cas min-max, le cas de la minimisation du regret maximum ainsi que celui où les poids des arêtes sont compris dans un intervalle donné.

Il est également intéressant de noter que la donnée ou non du couple de sommets source-puits peut avoir une grande influence sur la complexité du problème. Ainsi, les résultats obtenus dans [2], [4] et [50], démontrent l'existence de plusieurs problèmes de coupes multicritères $\mathcal{NP}$-complets lorsque $s$ et $t$ sont donnés et polynomiaux dans le cas contraire.

Pour ce qui est des problèmes de multicoupes multicritères, il n'existe à notre connaissance aucun résultat spécifique (les résultats existants étant des conséquences immédiates de résultats pour la coupe multicritère).

### 1.3.2 Problèmes de sous-graphes induits

Le problème de l'arbre de Steiner qui consiste à déterminer un arbre de taille minimum couvrant un ensemble donné de sommets terminaux est un problème bien connu et étudié depuis de nombreuses années ([39]). Il est $\mathcal{NP}$-difficile dans de nombreux cas - comme par exemple dans les graphes bipartis ([28]) - et il devient polynomial lorsque la cardinalité de l'ensemble des terminaux est fixée ([24]).

Nos travaux ont débuté lorsque nous nous somme interrogés sur la complexité du problème de l'arbre de Steiner si nous y ajoutons une *contrainte d'induction*. Contrairement au problème de l'arbre de Steiner qui admet toujours une solution si le graphe considéré est connexe, il n'y a aucune raison pour que cela soit également le cas pour le problème $k-$MinArbreI. Ainsi, sans même s'intéresser au problème d'optimisation associé, décider s'il existe ou non un arbre induit

couvrant un ensemble de sommets terminaux donnés semble être un problème peu évident à résoudre.

Les problèmes de recherche de sous-graphes induits particuliers constituent un domaine relativement vaste. Certains d'entre eux sont des problèmes très classiques en optimisation combinatoire. On pense par exemple à la recherche d'un ensemble stable de taille maximum ([37]) ou d'une clique de taille maximum ([51]) qui consistent à déterminer respectivement un sous-graphe maximum sans arête et un sous-graphe maximum complet, et qui sont des problèmes $\mathcal{NP}$-difficiles dans le cas général mais polynomiaux dans certaines classes de graphes comme les graphes bipartis ([54]). D'autres problèmes, comme la recherche d'un sous-graphe induit qui soit un cycle ([8]), une pyramide ([15]), un thêta ([16]) ou un prisme ([46]), ont des liens très forts avec les graphes parfaits.

Rappelons qu'un graphe est dit parfait si pour chacun de ses sous-graphes induits, la taille maximum d'une clique est égale à son nombre chromatique (c'est-à-dire au nombre minimum de couleurs nécessaires à sa coloration). C'est une classe de graphes relativement vaste, qui contient par exemple les graphes bipartis, les graphes des arêtes (*line graph* en anglais) et les graphes triangulés, et qui possède de nombreuses propriétés très intéressantes au niveau de l'optimisation combinatoire puisque de nombreux problèmes classiques de théorie des graphes y sont polynomiaux ([53]). En 2002, Chudnovsky et al. ont démontré le théorème fort des graphes parfaits : un graphe est parfait si et seulement si ni lui ni son complémentaire ne possèdent un cycle induit impair de longueur supérieure ou égale à 5 ([17]).

De nombreux travaux de recherche se sont donc focalisés sur la détection de cycles induits. Bienstock a ainsi montré dans [8] que déterminer s'il existe un cycle induit passant par deux sommets donnés est un problème $\mathcal{NP}$-complet et McDiarmid et al. ont prouvé que ce problème est polynomial dans les graphes planaires ([47]). D'autres études se sont focalisées sur la détection de prismes, de pyramides et de thêtas. En effet, tout graphe admettant une pyramide comme sous-graphe induit, possède un cycle de longueur impaire. A l'inverse, s'il existe un sous-graphe induit qui soit un thêta ou un prisme alors il existe nécessairement un cycle induit de longueur paire.

Chudnovsky et al. ont élaboré dans [15] un algorithme de complexité $O(n^9)$ permettant de tester s'il existe un sous-graphe qui soit une pyramide. De plus, Chudnovsky et Kapadia ont proposé dans [18] un algorithme polynomial permettant de tester s'il existe un sous-graphe qui soit un thêta **ou** un prisme. Cependant, déterminer l'existence ou non d'un sous-graphe qui soit un prisme est un problème $\mathcal{NP}$-difficile ([46]). Enfin, dans [16], Chudnovsky et Seymour proposent un algorithme en $O(n^{11})$ pour tester s'il existe un sous-graphe qui soit un thêta. Cet algorithme repose en grande partie sur un second algorithme élaboré dans le même papier permettant de résoudre $3-$ ArbreI et de complexité $O(n^4)$.

Notons également que Lévêque et al. donnent dans [44] un ensemble conséquent de résultats pour les problèmes de détection de sous-graphes particuliers.

## 1.4 Organisation du mémoire

Ce mémoire est divisé en deux parties : la première traite de problèmes de coupes et de multicoupes tandis que la seconde s'intéresse aux problèmes de recherche de sous-graphes induits.

Le chapitre 2 traite de la complexité de MinCCard dont nous montrons la $\mathcal{NP}$-difficulté. Puis, dans le chapitre 3, nous étudions une généralisation de MinCCard au problème de la multicoupe minimum. Nous donnons plusieurs résultats de $\mathcal{NP}$-complétude ainsi que des algorithmes polynomiaux reposant notamment sur la programmation dynamique et la relaxation lagrangienne. Enfin, le chapitre 4 se focalise davantage sur les problèmes de coupes et de multicoupes multicritères pour lesquels nous montrons principalement des résultats de $\mathcal{NP}$-complétude.

Le chapitre 6 donne un ensemble de résultats pour les problèmes de recherche de cycles induits, de chaînes induites et d'arbres induits. Après avoir montré la $\mathcal{NP}$-complétude des cas généraux de ces problèmes, nous étudions plus en détail leur complexité lorsque le nombre de sommets terminaux est fixé. Le chapitre 7 expose un algorithme de résolution pour $4 - $ ArbreI dans les graphes sans triangle. Nous décrivons la structure des graphes pour lesquels ce problème n'admet aucune solution puis nous montrons comment déterminer si un graphe admet ou non une telle structure.

# Première partie

# Etude de problèmes de coupe et de multicoupe

# Chapitre 2

# Problème de la coupe minimum sous contrainte de cardinalité

## 2.1 Préliminaires

Dans ce chapitre nous abordons les problèmes de coupe simple avec contrainte de cardinalité et nous nous intéressons plus particulièrement à la complexité de MinCCard.

Rappelons tout d'abord que le problème de la coupe minimum est un problème polynomial, que l'on peut résoudre en utilisant par exemple l'algorithme de Ford et Fulkerson ([1, 25]). Cependant, l'ajout d'une contrainte forçant la cardinalité de la coupe à être égale (ou même supérieure ou égale) à une certaine valeur donnée rend le problème difficile. Ainsi, Bruglieri et al. ont montré dans [10] que MinC=Card et MinC≥Card sont $\mathcal{NP}$-difficiles dans les graphes généraux même si tous les poids des arêtes sont égaux à 1, et également dans certains graphes particuliers comme les graphes complets et les graphes bipartis complets. A l'inverse, MinC=Card et MinC≥Card restent polynomiaux dans certaines classes de graphes comme par exemple les arbres où la cardinalité d'une coupe minimum est toujours égale à 1.

Même dans les graphes généraux, la complexité de MinCCard est un problème laissé ouvert par Bruglieri et al. Nous allons répondre à cette question dans ce chapitre en montrant que MinCCard est $\mathcal{NP}$-difficile même dans certains graphes particuliers orientés et non orientés.

D'autres résultats intéressants pour des problèmes très proches de ceux étudiés par Bruglieri et al. dans [10] ont été fournis par Armon et Zwick dans [4]. Ils ont notamment montré que GlobaleCCard était polynomial :

GlobaleCCard
**Données** : Un graphe non orienté $G = (V, E)$ où chaque arête $e$ est pondérée par un entier positif $w_e$, deux entiers $B$ et $p$.
**Question** : Existe-t-il une partition de $V$ en deux ensembles $V_1$ et $V_2$ telle que le nombre d'arêtes ayant une extrémité dans $V_1$ et l'autre dans $V_2$ soit inférieur

ou égal à $p$ et telle que la somme des poids de ces mêmes arêtes soit inférieure ou égale à $B$ ?

Pour obtenir ce résultat, ils ont prouvé qu'il suffit d'énumérer un certain nombre de coupes dans le graphe où le poids de chaque arête $e$ est égal à $w(e) + 1$ puis de sélectionner parmi celles-ci la coupe $C$ minimisant $w(C)$ et vérifiant la contrainte de cardinalité . En fait ils ont même prouvé un résultat plus fort : la version multicritère de GlobaleCCard est polynomial lorsque le nombre de critères est fixé. Notons que la seule différence entre MinCCard et GlobaleCCard consiste en l'absence de la donnée d'un sommet source et d'un sommet puits.

Enfin, terminons cette brève introduction par un théorème et par une remarque sur les poids utilisés dans les graphes.

Lorsque nous considérons MinCCard et également MinMultiCCard pour une valeur de $p$ fixée, ces problèmes sont polynomiaux :

**Théorème 2.1.** *Si l'on fixe la valeur de $p$ alors* MinCCard *et* MinMultiCCard *sont polynomiaux.*

*Démonstration.* Si la valeur de $p$ est fixée, l'algorithme - de complexité en $O(m^p)$ - consistant à énumérer tous les ensembles d'au plus $p$ arêtes et d'en sélectionner l'ensemble de poids minimum correspondant à une (multi)coupe admissible, permet de résoudre les problèmes MinCCard et MinMultiCCard. □

Enfin, l'infini peut être codé par un seul caractère ("$\infty$" par exemple) ayant la propriété d'être toujours plus grand que n'importe quelle valeur finie. Cela nous permettra ainsi d'envisager des instances de taille finie où certaines valuations sont considérées comme infinies.

## 2.2 MinCCard dans les graphes généraux

Dans [10], Bruglieri et al. utilisent une réduction à partir du problème de la coupe maximale pour montrer la $\mathcal{NP}$-difficulté de MinC=Card et de MinC≥Card. Cependant, cette réduction n'est pas réutilisable pour MinCCard car elle consiste à augmenter autant que possible la cardinalité de la coupe afin d'obtenir celle de cardinalité maximum. Nous considérons alors le problème $\mathcal{NP}$-complet suivant ([28]) :

BiSection
**Données** : Un graphe $G = (V, E)$ tel que $|V| = 2n$ et $|E| = m$, un entier $B$ donné tel que $B < m$.
**Question** : Existe-t-il une partition de $V$ en deux ensembles $V_1$ et $V_2$ telle que $|V_1| = |V_2| = n$ et telle que le nombre d'arêtes ayant une extrémité dans $V_1$ et l'autre dans $V_2$ soit inférieur ou égal à $B$ ?

**Théorème 2.2.** MinCCard *est $\mathcal{NP}$-difficile au sens fort.*

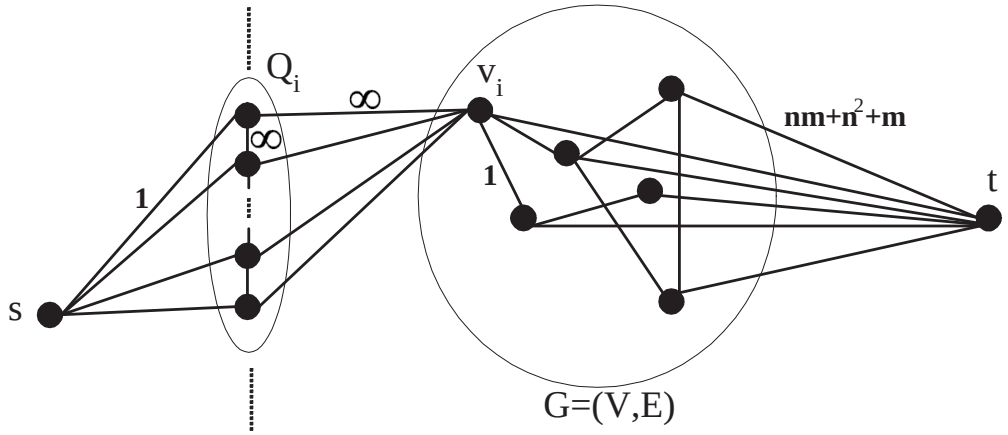


Figure 2.1 – Le graphe obtenu pour MinCCard à partir de $G$

*Démonstration.* Nous réalisons une réduction à partir de BiSection.

Soit $I_{bi}$ une instance de BiSection composée d'un graphe $G = (V, E)$ possédant $2n$ sommets et $m$ arêtes, et d'un entier $B$.

Soit $I_{cut}$ une instance du problème de décision associé à MinCCard construite à partir de $I_{bi}$ de la manière suivante (voir la figure 2.1) : assignons un poids de 1 à toutes les arêtes de $G$. Ajoutons à $G$ un nouveau sommet $t$ ainsi que $2n$ arêtes de poids $nm + n^2 + m$ le reliant à chaque sommet de $G$. Pour chaque sommet $v_i$ de $G$ ($i \in \{1, \ldots, 2n\}$), ajoutons une chaîne $Q_i$ de $m + n$ sommets composée d'arêtes de poids infini, ainsi que $m + n$ arêtes de poids infini reliant chaque sommet de $Q_i$ à $v_i$. Enfin, ajoutons un nouveau sommet $s$ et $2n(m + n)$ arêtes de poids 1 reliant $s$ à chacun des sommets des $Q_i$ ($i \in \{1, \ldots, 2n\}$).

Montrons qu'il existe une solution de $I_{bi}$ si et seulement s'il existe une coupe $C$ séparant $s$ et $t$ telle que :

$$w(C) \leq (nm + n^2 + m)n + (m + n)n + B \tag{2.1}$$

$$|C| \leq n + (m + n)n + B \tag{2.2}$$

Supposons que nous possédons une solution de $I_{bi}$ et construisons une solution de $I_{cut}$ de la manière suivante. Pour chaque sommet de $V_1$, nous coupons l'arête le reliant à $t$ et pour chaque sommet $v_i$ de $V_2$, nous coupons les arêtes reliant $s$ aux sommets de $Q_i$. Enfin, nous coupons les arêtes ayant une extrémité dans $V_1$ et l'autre dans $V_2$.

$$C = \{e_{v_i t} | v_i \in V_1\} \cup \{e_{sq} | v_i \in V_2, q \in Q_i\} \cup \{e_{v_i v_j} | v_i \in V_1, v_j \in V_2\}$$

De cette manière, $C$ sépare $\{s\} \cup V_1 \cup \{q \in Q_i | v_i \in V_1\}$ de $\{t\} \cup V_2 \cup \{v \in Q_i | v_i \in V_2\}$. De plus,
$|C| \leq |V_1| + (m + n)|V_2| + B = n + (m + n)n + B$
$w(C) \leq (nm + n^2 + m)|V_1| + (m + n)|V_2| + B = (nm + n^2 + m)n + (m + n)n + B$

Réciproquement, soit $C$ une solution de $I_{cut}$, c'est-à-dire une coupe séparant $s$ de $t$ et telle que $w(C) \leq (nm+n^2+m)n+(m+n)n+B$ et $|C| \leq n+(m+n)n+B$. Construisons alors une solution de $I_{bi}$ : $V_1$ contient les sommets de $G$ reliés à $s$ et $V_2$ les sommets de $G$ reliés à $t$.

Tout d'abord, notons que $C$ ne peut évidemment contenir aucune arête de poids infini. De plus, si l'arête $e_{sq}$ ($q \in Q_i$) est contenu dans $C$ alors toutes les arêtes $e_{sq_i}$ ($q_i \in Q_i$) sont dans $C$, l'arête $e_{v_it}$ n'est pas dans $C$ et $v_i$ appartient donc à $V_2$.

Montrons que les ensembles $V_1$ et $V_2$ que nous avons construits ont la même cardinalité.
$|V_2| \geq n$ car sinon $C$ contiendrait au moins $n+1$ arêtes reliant $t$ à des sommets de $G$ et l'on aurait :
$w(C) \geq (n+1)(nm+n^2+m) = (nm+n^2+m)n+(m+n)n+m > (nm+n^2+m)n+(m+n)n+B$ ce qui violerait (2.1).
$|V_1| \geq n$ car sinon $C$ contiendrait au moins $(n+1)(m+n)$ arêtes reliant $s$ à des sommets de $Q_i$ ($i|v_i \in V_2$) et l'on aurait :
$|C| \geq (n+1)(m+n) = (m+n)n+m+n > (m+n)n+B+n$ ce qui violerait (2.2).
Puisque $|V_1|+|V_2| = 2n$, nous avons nécessairement $|V_1| = |V_2| = n$.

Il reste désormais à montrer que le nombre d'arêtes ayant une extrémité dans $V_1$ et l'autre dans $V_2$ est inférieur ou égal à $B$.
Puisque $C$ contient les $n$ arêtes reliant $t$ aux sommets de $V_1$ ainsi que les $(m+n)n$ arêtes reliant $s$ aux sommets de $Q_i$ ($i|V_i \in V_2$), nous obtenons d'après (2.2) que le nombre d'arêtes de $G$ contenues dans $C$ est inférieur ou égal à $B$. □

## 2.3 Extension aux graphes bipartis de degré maximum 3

Nous avons montré dans la section précédente que MinCCard est $\mathcal{NP}$-difficile dans le cas général. Nous allons voir qu'il est en fait possible de se ramener systématiquement à des graphes bipartis de degré maximum 3 en utilisant deux transformations simples du graphe.

Considérons la transformation suivante (voir la figure 2.2) : soit $v$ un sommet tel que $d^\circ(v) > 3$ et soit $e_1, \ldots, e_{d^\circ(v)}$ les arêtes incidentes à $v$. Remplaçons alors $v$ par la chaîne $v_1 - v_2 - \cdots - v_{d^\circ(v)-2}$ de longueur $d^\circ(v)-3$ dont les arêtes sont pondérées par des poids infinis et telle que :

- $e_1$ est connecté à $v_1$
- $e_{d^\circ(v)}$ est connecté à $v_{d^\circ(v)-2}$
- pour $i \in \{2, \ldots, d^\circ(v)-1\}$ $e_i = e_{vx}$ est remplacée par $e_{v_{i-1}x}$.

Si une source ou un puits était placé en $v$, nous le plaçons en $v_1$. Notons également que cette transformation conserve la planarité en ordonnant les arêtes incidentes à $v$ de manière adéquate (comme c'est le cas sur la figure 2.2).

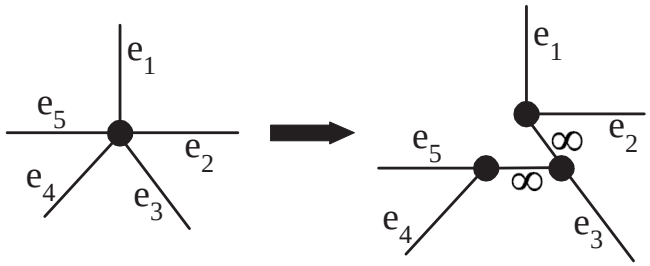


FIGURE 2.2 – Obtention d'un graphe de degré maximum 3


FIGURE 2.3 – Obtention d'un graphe biparti

Considérons maintenant la transformation suivante (voir la figure 2.3) : soit $u - v$ une arête de $G$. Remplaçons cette arête par deux nouvelles arêtes - *l'une de même poids que* $u - v$*, l'autre de poids infini* - et un nouveau sommet de telle sorte que $u$ et $v$ soient désormais reliés par une chaîne de longueur 2.
Ainsi, en appliquant cette seconde transformation à toutes les arêtes d'un graphe, on obtient un graphe biparti avec d'un côté les sommets originaux du graphe et de l'autre les nouveaux sommets ajoutés.
Notons que cette transformation, tout comme la précédente, conserve la planarité.

En utilisant ces deux transformations, nous obtenons alors le théorème suivant :

**Théorème 2.3.** *Toute instance d'un problème de coupe ou de multicoupe, monocritère ou multicritère peut être transformée en une nouvelle instance équivalente dans laquelle le graphe est biparti et de degré maximum* 3.

*Démonstration.* Tout d'abord, précisons que pour le cas multicritère, une arête de poids infini est une arête pondérée par un vecteur dont toutes les composantes sont infinies.

Puisque la seconde transformation ne modifie pas le degré maximum du graphe, nous allons appliquer la première transformation à tous les sommets de degré supérieur ou égal à 4 puis appliquer la seconde transformation à toutes les arêtes.

Ainsi, il apparaît qu'un ensemble d'arêtes $C$ est une (multi)coupe pour le graphe d'origine si et seulement si $C$ est une (multi)coupe pour le graphe biparti de degré maximum 3 obtenu. En effet, les arêtes ajoutées lors des deux transformations ne peuvent appartenir à aucune (multi)coupe du nouveau graphe car elles sont de poids infinis. De plus, les chaînes reliant la(les) source(s) au(x) puits

ne sont pas véritablement modifiées puisque qu'on leur ajoute uniquement des arêtes de poids infinis. □

Nous avons alors :

**Théorème 2.4.** MinCCard *est $\mathcal{NP}$-difficile au sens fort dans les graphes bipartis de degré maximum* 3.

*Démonstration.* Ce théorème est une conséquence directe du théorème précédent et du théorème 2.2 (page 34). En effet, la coupe simple avec contrainte de cardinalité peut être vue comme une coupe bicritère particulière où la seconde valuation de chaque arête est égale à 1. □

Il est à noter qu'en utilisant la seconde transformation, la source et le puits se retrouvent du même côté de la bipartition. Or, ce n'est pas le cas dans certains problèmes classiques comme par exemple les problèmes de transports ([1]). Une idée intuitive pour éviter cette situation consisterait à appliquer les deux transformations puis à ajouter un sommet fictif relié au puits et à déplacer le puits sur ce nouveau sommet. Mais en agissant de la sorte, le degré maximum n'est plus nécessairement 3. Nous utilisons donc la manière suivante. Avant d'utiliser les deux transformations, nous ajoutons un sommet $v'$ ainsi qu'une arête $e'$ de poids infini reliant $v'$ au puits et nous déplaçons le puits en $v'$. Puis, nous appliquons la première transformation à tous les sommets de degré supérieur ou égal à 4 et la seconde transformation à toutes les arêtes (donc y compris $e'$). Enfin, dans le graphe biparti de degré maximum 3 obtenu, nous déplaçons le puits sur l'unique voisin de $v'$ et nous supprimons $v'$.

## 2.4 MinCCard dans les graphes orientés

Pour conclure l'étude de MinCCard, nous nous intéressons dans cette section au cas des graphes orientés. De manière similaire aux graphes non orientés, MinCCard admet de manière évidente une solution dans certaines classes de graphes comme les circuits et les arbres orientés, où la cardinalité de toute coupe minimum est égale à 1.

De plus, nous montrons que cette similarité se vérifie pour ce qui concerne le cas général :

**Théorème 2.5.** MinCCard *est $\mathcal{NP}$-difficile au sens fort dans les graphes orientés bipartis.*

*Démonstration.* Nous réalisons une réduction à partir du problème de décision associé à MinCCard dans les graphes non orientés, bipartis, de degré maximum 3 (voir le théorème 2.4).

Soit $I$ une instance de ce problème composée d'un graphe biparti connexe $G = (V_1, V_2, E)$ possédant $n$ sommets et $m$ arêtes, de deux sommets $s$ et $t$ de $G$, et de deux entiers strictement positifs $p$ et $B$.

Soit $I'$ une instance du problème décision associé à MinCCard dans les graphes orientés construite à partir de $I$ de la manière suivante (voir la figure


FIGURE 2.4 – Transformation d'un graphe non orienté en un graphe orienté

2.4) : chaque arête $u-v$ de $E$ est remplacée par deux arcs de même poids reliant $u$ à $v$ et $v$ à $u$. Montrons que $I$ admet une solution si et seulement si $I'$ admet une solution $C'$ séparant $s$ de $t$ telle que $|C'| \leq p$ et $w(C') \leq B$.

Soit $C$ une solution de $I$. Supposons que $C$ est minimale au sens de l'inclusion (si ce n'est pas le cas, on supprime une à une toutes les arêtes inutiles de $C$). $G$ étant connexe, après suppression des arêtes de $C$, nous obtenons un graphe possédant deux composantes connexes $U_1$ et $U_2$, avec $s \in U_1$ et $t \in U_2$ puisqu'il n'existe plus aucune chaîne reliant $s$ à $t$ et que $C$ est minimale. Une solution de $I'$ est alors obtenue en sélectionnant les arcs $u \rightarrow v$ tels que $u - v \in C$, $u \in U_1$ et $v \in U_2$.

Réciproquement, soit $C'$ une solution de $I'$ minimale au sens de l'inclusion. Une solution de $I$ consiste alors à sélectionner les arêtes $u - v$ telles que $u \rightarrow v \in C'$ ou $v \rightarrow u \in C'$. En effet, si la solution construite n'était pas une coupe alors il existerait une chaîne reliant la source au puits et on pourrait alors faire correspondre à cette chaîne un chemin reliant la source au puits dans $I'$ ; ce qui voudrait dire que $C'$ n'est pas solution admissible de $I'$. □

# Chapitre 3

# Multicoupes et contrainte de cardinalité

## 3.1 Introduction

Le problème de la multicoupe minimum (MinMultiC) est une généralisation du classique problème de la coupe simple minimum. Cependant, si ce dernier est polynomial, MinMultiC est $\mathcal{NP}$-difficile au sens fort dans les graphes orientés et non orientés, même lorsque le nombre de paires de terminaux $k$ est fixé pour $k > 2$ ([21]). De plus, Garg et al. ont montré dans [30] que pour des valeurs arbitraires de $k$, MinMultiC reste $\mathcal{NP}$-difficile au sens fort dans les étoiles non orientées même lorsque tous les poids des arêtes sont égaux à 1. A l'inverse, ce problème est polynomial dans les arbres orientés ainsi que dans les chaînes et les anneaux ([19],[20]).

MinMultiCCard étant une généralisation de MinCCard d'une part et MinMultiC pouvant s'y ramener d'autre part (en choisissant une valeur très grande pour $p$), tous les cas difficiles de MinCCard et MinMultiC sont également difficiles pour MinMultiCCard. D'où le théorème suivant :

**Théorème 3.1.** MinMultiCCard *est $\mathcal{NP}$-difficile au sens fort dans les étoiles non orientées ainsi que dans les graphes (non orientés) bipartis de degré maximum* 3.

*Démonstration.* C'est une conséquence immédiate de [30] et du théorème 2.4 (page 38). □

Dans ce chapitre, nous allons donc nous intéresser aux cas particuliers polynomiaux de MinMultiC et de MinCCard afin de déterminer s'ils le restent ou non pour MinMultiCCard.

## 3.2 Matrice des contraintes et totale unimodularité

Tout d'abord, nous rappelons la définition d'une matrice totalement unimodulaire :

**Définition 3.1.** *Une matrice est dite totalement unimodulaire si et seulement si toute sous-matrice carrée a pour déterminant* $0$, $1$ *ou* $-1$.

A titre d'exemple, c'est le cas de la matrice d'incidence de tout graphe orienté. Une classe particulièrement connue de matrices totalement unimodulaires est celle des matrices d'intervalle ([54, Chapitre 19]) :

**Définition 3.2.** *Une matrice d'intervalle est une matrice binaire pour laquelle il existe une permutation de ses colonnes permettant d'obtenir une nouvelle matrice où chacune de ses lignes a ses '1' consécutifs.*

Le caractère totalement unimodulaires d'une matrice est très intéressant notamment du point de vue de la programmation linéaire en raison de la propriété suivante :

**Propriété 3.1.** *Soit $A$ une matrice totalement unimodulaire de taille $m \cdot n$ et soit $b \in \mathbb{Z}^m$.*
*Les points extrêmes du polyèdre $P = \{x \mid Ax \leq b\}$ sont entiers.*

Ainsi, Grotschel, Lovasz et Schriver ont montré dans [34] qu'un programme linéaire en nombres entiers dont la matrice des contraintes est totalement unimodulaire peut être résolu en temps polynomial en utilisant la méthode de l'ellipsoïde.

Considérons à présent la formulation "arc-sommet" de MinMultiC dans les arbres orientés. Pour chaque couple source-puits $(s_i, t_i)$ $(i \in \{1, \ldots, k\})$, $P_i$ représente l'unique chemin entre $s_i$ et $t_i$. Rappelons également que $w(a)$ est égal au poids de l'arc $a$. Enfin, définissons $z_a$ comme la variable de décision valant 1 si $a$ appartient à la multicoupe et 0 sinon.

La seule contrainte de MinMultiC consiste à couper au moins un arc sur chaque chemin reliant une source à son puits. On obtient donc le programme linéaire en variables $\{0, 1\}$ suivant :

$$(PLarbre) \begin{cases} Min & \sum_{a \in E} w(a) z_a & \\ \text{s. t.} & \sum_{a \in P_i} z_a \geq 1 & \forall i \in \{1, \ldots, k\} \\ & z_a \in \{0, 1\} & \forall a \in E \end{cases}$$

La matrice des contraintes de $(PLarbre)$ étant totalement unimodulaire ([20]), MinMultiC est polynomial dans ce cas. Cependant, cela n'est plus nécessairement le cas si nous y ajoutons la contrainte de cardinalité $\sum_{a \in E} z_a \leq p$ comme nous pouvons le voir sur l'instance de la figure 3.1.

Dans cette instance, la solution optimale de coût 5 est obtenue en coupant l'arc de poids 5 alors que l'optimum continu de coût 4 est obtenu en posant pour tout arc $a \in E$, $z_a = 0.5$. La matrice des contraintes de cette instance ne peut donc pas être totalement unimodulaire alors que l'instance considérée est à la fois une étoile orientée et une arborescence.
Nous montrerons dans la section suivante que MinMultiCCard est $\mathcal{NP}$-difficile

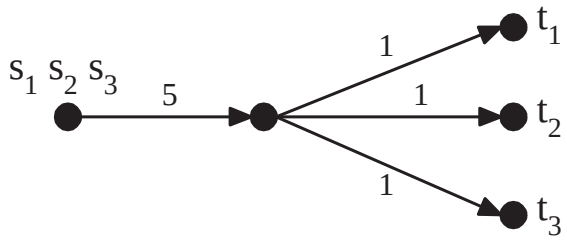


Figure 3.1 – Une instance de MinMultiCCard dans un arbre orienté avec $p = 2$ où la matrice des contraintes n'est plus totalement unimodulaire

au sens fort dans les étoiles orientées.

A l'inverse, on peut aisément remarquer que lorsque le graphe considéré est un chemin, l'ajout de la contrainte de cardinalité ne modifie pas le caractère totalement unimodulaire de la matrice des contraintes. Nous montrons ainsi le théorème suivant :

**Théorème 3.2.** MinMultiCCard *est polynomial dans les chemins.*

*Démonstration.* Dans les chemins, la matrice des contraintes de ($PLarbre$) est une matrice d'intervalle. L'ajout de la contrainte $\sum_{a \in E} z_a \leq p$ nous permet de conserver une matrice d'intervalle. En effet, cela revient à ajouter une ligne de 1 à la matrice des contraintes et la propriété des '1' consécutifs est donc toujours vérifiée. Puisqu'une matrice d'intervalle est totalement unimodulaire, MinMultiCCard est donc polynomial dans les chemins. □

Hassin et Tamir ont proposé dans [36] un algorithme combinatoire pour obtenir la *valeur optimale* (mais sans fournir la solution associée) d'une instance de MinMultiCCard dans les chemins en $O(n^2 \log^2(n))$. Nous proposerons dans la section 3.4.3 une approche primal-dual permettant également de calculer cette valeur. De plus, dans la section 3.4.4, nous présenterons un algorithme polynomial reposant sur la programmation dynamique permettant de calculer une *solution optimale*.

## 3.3 MinMultiCCard dans les étoiles orientées

### 3.3.1 Préliminaires

Commençons par définir plusieurs nouveaux problèmes utiles lors des réductions menant à la $\mathcal{NP}$-difficulté de MinMultiCCard dans les étoiles orientées.

**Définition 3.3.** *Soit $H$ un graphe non orienté. Une* ***couverture des arêtes*** *de $H$ est un ensemble $S$ de sommets de $H$ tel que chaque arête de $H$ a au moins une de ses extrémités dans $S$.*

PondVCCard
**Données** : Un graphe non orienté biparti $H = (V_1, V_2, E)$, une fonction de poids $w : (V_1 \cup V_2) \rightarrow \mathbb{N}^*$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il une couverture $S$ des arêtes de $H$ telle que $w(S) \leq B$ et $|S| \leq p$ ?

VCCardInf
**Données** : Un graphe non orienté biparti $H = (V_1, V_2, E)$ où $VC_{min}$ est la taille minimum d'une couverture des arêtes de $H$, deux entiers positifs $p$ et $q$ tels que $p \leq VC_{min}$, $q \leq VC_{min}$ et $p + q \geq VC_{min}$.
**Question** : Existe-t-il une couverture $S$ des arêtes de $H$ de taille minimum contenant *au plus* $p$ sommets de $V_1$ et *au plus* $q$ sommets de $V_2$ ?

VCCardEg
**Données** : Un graphe non orienté biparti $H = (V_1, V_2, E)$ où $VC_{min}$ est la taille minimum d'une couverture des arêtes de $H$, deux entiers positifs $p^=$ et $q^=$ tels que $p^= + q^= = VC_{min}$.
**Question** : Existe-t-il une couverture $S$ des arêtes de $H$ contenant *exactement* $p^=$ sommets de $V_1$ et $q^=$ sommets de $V_2$ ?

Notons que le terme "VC" utilisé dans le nom des problèmes est l'abréviation de "Vertex Cover".

Rappelons que König a montré que le problème consistant à déterminer une couverture des arêtes de taille minimum dans un graphe biparti est polynomial et peut être résolu en $O(m)$ en utilisant les liens unissant ce problème à celui du couplage de taille maximum ([42],[54]). Ainsi, la valeur $VC_{min}$ utilisée dans la définition des problèmes VCCardInf et VCCardEg peut être calculée en temps polynomial. De plus, Chen et Kanj ont montré dans [13] que VCCardInf est $\mathcal{NP}$-complet en utilisant une réduction à partir d'un problème de clique.

Dans toute la section suivante, le graphe $G = (V_1, O, V_2, E)$ représentera une étoile orientée dont les arcs sont pondérés par un entier strictement positif. $O$ sera le sommet central de l'étoile, c'est-à-dire le seul sommet de degré supérieur ou égal à 2. $V_1$ (respectivement $V_2$) correspondra aux sommets ayant pour successeur (resp. prédécesseur) le sommet $O$. Nous supposerons qu'il existe un chemin entre toute source et son puits (dans le cas contraire, on peut supprimer la paire de terminaux) et que ce chemin est de longueur 2 (s'il est de longueur 1, on supprime le seul arc séparant la source du puits). Par conséquent, il n'y aura aucun terminal placé en $O$. Enfin nous supposerons également qu'il y a au moins un terminal placé sur chaque sommet de $V_1 \cup V_2$ car sinon nous pouvons supprimer ces sommets sans terminaux (voir la partie gauche de la figure 3.2 pour un exemple d'étoiles orientées vérifiant les propriétés énoncées).

### 3.3.2 $\mathcal{NP}$-difficulté de MinMultiCCard dans les étoiles orientées

Nous allons montrer successivement que :

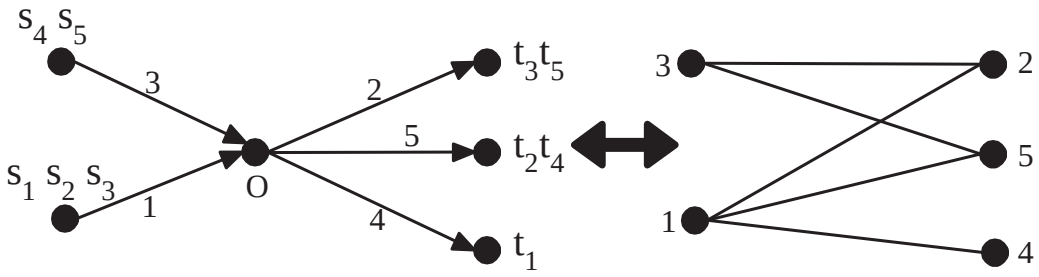


Figure 3.2 – Equivalence entre une instance de MinMultiCCard dans une étoile orientée et une instance de PondVCCard

1. PondVCCard est équivalent au problème de décision associé à MinMultiCCard dans les étoiles orientées
2. VCCardEg est $\mathcal{NP}$-complet, grâce à une réduction à partir de VCCardInf
3. PondVCCard est $\mathcal{NP}$-complet, grâce à une réduction à partir de VCCardEg

Nous aurons alors prouvé la $\mathcal{NP}$-difficulté de MinMultiCCard dans les étoiles orientées.

**Lemme 3.1.** PondVCCard *est équivalent au problème de décision associé à* MinMultiCCard *dans les étoiles orientées.*

*Démonstration.* Nous utilisons une preuve similaire à celle employée par Garg et al. dans [30] pour montrer la $\mathcal{NP}$-difficulté au sens fort de MinMultiC dans les étoiles non orientées. Ils y effectuent une réduction à partir du problème de couverture des arêtes de taille minimum en passant par l'intermédiaire du graphe de demande.

Soit $I$ une instance du problème de décision associé à MinMultiCCard dans une étoile orientée et construisons l'instance de PondVCCard associée. Considérons le graphe (non orienté) de demande $H = (V_1, V_2, E^H)$ associé à $I$ (voir la figure 3.2) : pour chaque paire source-puits $(s_i, t_i)$ avec $s_i$ placé en $v$ et $t_i$ placé en $v'$, $E^H$ contient l'arête $v - v'$. De plus, le poids de chaque sommet $v$ dans $H$ est égal au poids de l'unique arc dont les extrémités sont $v$ et $O$ dans $G$. Puisqu'il n'y a aucun terminal placé en $O$, $H$ est donc nécessairement un graphe biparti.

A toute couverture des arêtes de $H$, dont le poids est inférieur ou égal à $B$ et la cardinalité inférieure ou égale à $p$, correspond une solution de $I$ de poids inférieur ou égal à $B$ et de cardinalité inférieure ou égale à $p$ (et réciproquement). En effet, il suffit de remarquer qu'ajouter à la multicoupe un arc $a$ de $G$ ayant pour extrémités les sommets $v$ à $O$, revient à sélectionner $v$ dans la couverture des arêtes de $H$. □

**Lemme 3.2.** VCCardEg *est* $\mathcal{NP}$*-complet.*

*Démonstration.* Nous réalisons une réduction à partir de VCCardInf qui est $\mathcal{NP}$-complet ([13]).

Soit $I$ une instance de VCCardInf composée d'un graphe $H = (V_1, V_2, E)$ non orienté biparti dans lequel $VC_{min}$ est la taille minimum d'une couverture des arêtes, et de deux entiers $p$ et $q$. Remarquons que, puisque $p + q \geq VC_{min}$, $p \geq VC_{min} - q$. Nous construisons $O(|V_1|+|V_2|)$ instances $I_i$ de VCCardEg de la manière suivante : la $i$-ème instance est composée de $H$, $p^{=} = i$ et $q^{=} = VC_{min} - i$ ($i \in \{VC_{min} - q, \ldots, p\}$).

Montrons que $I$ a une solution si et seulement s'il existe $i \in \{VC_{min} - q, \ldots, p\}$ tel que $I_i$ a une solution.

Soit $VC_I$ une solution de $I$ et soit $p'$ et $q'$ respectivement le nombre de sommets de $V_1$ et de $V_2$ appartenant à $VC_I$. Par construction, nous avons $p' + q' = VC_{min}$, $p' \leq p$ et $q' \leq q$ d'où $p' \geq VC_{min} - q$. Donc $p' \in \{VC_{min} - q, \ldots, p\}$ et $VC_I$ est ainsi solution de l'instance $I_{p'}$. Réciproquement, toute solution d'une instance $I_i$ est également une solution pour $I$ puisque l'on a toujours $p^{=} \leq p$ et $q^{=} \leq q$. □

**Lemme 3.3.** PondVCCard *est $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* Nous réalisons une réduction à partir de VCCardEg dont nous avons montré la $\mathcal{NP}$-complétude dans le lemme précédent.

Soit $I$ une instance de VCCardEg composée d'un graphe $H = (V_1, V_2, E)$ non orienté biparti, dont $VC_{min}$ est la taille minimum d'une couverture des arêtes, et de deux entiers positifs $p^{=}$ et $q^{=}$ tels que $p^{=} + q^{=} = VC_{min}$. Soit $n$ le nombre de sommets de $H$ d'où $n = |V_1| + |V_2|$.

Nous construisons une instance $I'$ pour PondVCCard de la manière suivante (voir la figure 3.3). Assignons un poids de 1 aux sommets de $V_1$ et un poids égal à $2n^2 + 2n + 1$ aux sommets de $V_2$. Pour chaque sommet $v$ de $V_2$, ajoutons $2n + 1$ sommets de poids 1 tous reliés à $v$ et notons $V'$ l'ensemble de ces $|V_2|(2n + 1)$ nouveaux sommets et $E'$ l'ensemble des nouvelles arêtes. Le graphe $H' = (V_1 \cup V', V_2, E \cup E')$ obtenu est donc un graphe biparti dont chaque sommet est pondéré par un entier strictement positif.

Montrons qu'il existe une couverture des arêtes de $H$ comprenant $p^{=}$ sommets de $V_1$ et $q^{=}$ sommets de $V_2$ si et seulement s'il existe une couverture $VC_{H'}$ des arêtes de $H'$ telle que :

$$w(VC_{H'}) \leq p^{=} + (2n^2 + 2n + 1)q^{=} + (2n + 1)(|V_2| - q^{=}) \tag{3.1}$$

$$|VC_{H'}| \leq p^{=} + q^{=} + (2n + 1)(|V_2| - q^{=}) \tag{3.2}$$

Soit $VC_H$ une solution de $I$, construisons une solution $VC_{H'}$ de $I'$ de la manière suivante. $VC_{H'}$ contient tous les sommets de $VC_H$ ainsi que les sommets de $V'$ reliés aux sommets de $V_2 \setminus VC_H$ :

$$VC_{H'} = VC_H \cup \{v' \in V' | \exists v \in V_2 \setminus VC_H \text{ tel que } v' - v \in E'\}$$

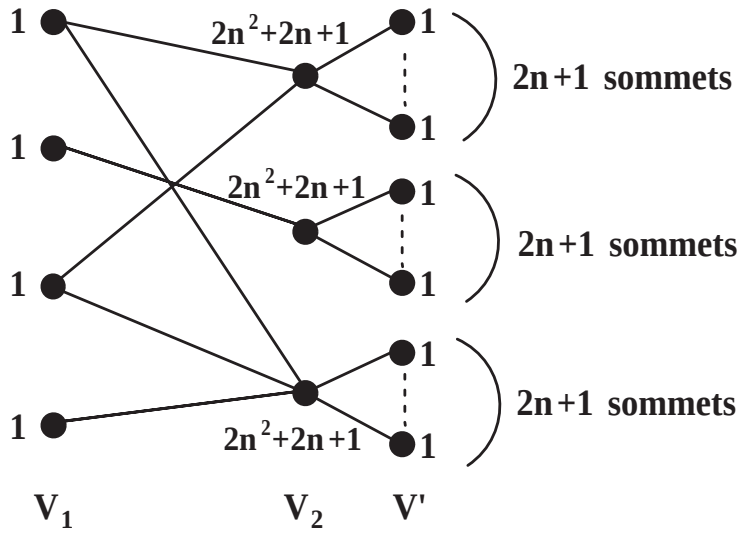


FIGURE 3.3 – Le graphe obtenu pour PondVCCard à partir de $H = (V_1, V_2, E)$

De cette manière, $VC_{H'}$ est une couverture des arêtes de $H'$ et possède exactement $p^=$ sommets de $V_1$, $q^=$ sommets de $V_2$ et $(2n+1)(|V_2| - q^=)$ de $V'$. D'où $|VC_{H'}| = p^= + q^= + (2n+1)(|V_2| - q^=)$ et $w(VC_{H'}) = p^= + (2n^2+2n+1)q^= + (2n+1)(|V_2| - q^=)$.

Réciproquement, soit $VC_{H'}$ une solution de $I'$ et soit $VC_H$ une solution de $I$ construite de la manière suivante :

$$VC_H = VC_{H'} \cap (V_1 \cup V_2)$$

$VC_H$ étant de manière évidente une couverture des arêtes de $H$, il reste à montrer qu'elle contient exactement $p^=$ sommets de $V_1$ et $q^=$ sommets de $V_2$.

$VC_{H'}$ possède au moins $q^=$ sommets de $V_2$ car sinon le nombre de sommets de $V'$ sélectionnés dans $VC_{H'}$ serait supérieur ou égal à $(2n+1)(|V_2| - q^= + 1)$ d'où :
$|VC_{H'}| \geq 2n + 1 + (2n+1)(|V_2| - q^=) > p^= + q^= + (2n+1)(|V_2| - q^=)$, ce qui violerait (3.2).

$VC_{H'}$ possède au plus $q^=$ sommets de $V_2$ car sinon :
$w(VC_{H'}) \geq (2n^2+2n+1)(q^= + 1) = n + (2n^2+2n+1)q^= + (2n+1)n + 1 > p^= + (2n^2+2n+1)q^= + (2n+1)(|V_2| - q^=)$, ce qui violerait (3.1).

Donc, $VC_{H'}$ possède exactement $q^=$ sommets de $V_2$. De plus, puisque pour chaque sommet $v \in V_2 \setminus VC_{H'}$, $VC_{H'}$ contient les $2n+1$ sommets de $V'$ connectés à $v$, $VC_{H'}$ contient également au moins $(2n+1)(|V_2| - q^=)$ sommets de $V'$.

$VC_{H'}$ contient au plus $p^=$ sommets de $V_1$ car sinon :

$w(VC_{H'}) \geq (p^{=}+1) + (2n^2+2n+1)q^{=} + (2n+1)(|V_2| - q^{=})$, ce qui violerait (3.1). De plus, $VC_{H'}$ ne peut contenir moins de $p^{=}$ sommets de $V_1$ car sinon $VC_H$ deviendrait une couverture des arêtes de $H$ de taille strictement inférieure à $VC_{min}$. Donc, $VC_{H'}$ possède exactement $p^{=}$ sommets de $V_1$. □

Nous obtenons finalement le théorème désiré :

**Théorème 3.3.** MinMultiCCard *est $\mathcal{NP}$-difficile au sens fort dans les étoiles orientées.*

*Démonstration.* Nous avons montré dans le lemme 3.1 que PondVCCard est équivalent au problème de décision associé à MinMultiCCard dans les étoiles orientées, puis dans le lemme 3.3 que PondVCCard est $\mathcal{NP}$-complet au sens fort. □

## 3.4 Etude de MinMultiCCard dans les chaînes

Si MinMultiCCard est $\mathcal{NP}$-difficile au sens fort dans les étoiles orientées, il reste polynomial dans les chemins à l'instar de MinMultiC (voir le théorème 3.2 page 43).

Dans cette section, nous élaborons deux algorithmes polynomiaux pour MinMultiCCard dans les chaînes : tout d'abord un algorithme reposant sur une approche primal-dual permettant d'obtenir la *valeur optimale* puis un second algorithme utilisant la programmation dynamique pour calculer une *solution optimale*.

### 3.4.1 Préliminaires

Déterminons tout d'abord la complexité de MinMultiCCard dans les chaînes :

**Théorème 3.4.** MinMultiCCard *est polynomial dans les chaînes.*

*Démonstration.* MinMultiCCard dans les chaînes et MinMultiCCard dans les chemins sont deux problèmes équivalents. Pour passer du premier au second, il suffit d'orienter toutes les arêtes vers la même extrémité de la chaîne et d'inverser, si nécessaire, une source et son puits pour qu'il existe toujours un chemin les reliant (voir la figure 3.4). Réciproquement, pour passer du chemin à la chaîne, il suffit de remplacer chaque arc par une arête de même poids.

MinMultiCCard étant polynomial dans les chemins (voir le théorème 3.2 page 43), nous en déduisons le théorème désiré. □

De ce fait, tout algorithme résolvant MinMultiCCard dans les chaînes permet d'obtenir un algorithme de même complexité pour MinMultiCCard dans les chemins.

Dans toute la suite de la section 3.4, $G = (V, E)$ est une chaîne composée de $n$ sommets et de $n-1$ arêtes. Les deux sommets extrémités de $G$ sont notés $b$ et $b'$. Chaque arête $e$ est pondérée par un entier strictement positif $w(e)$. Pour chaque

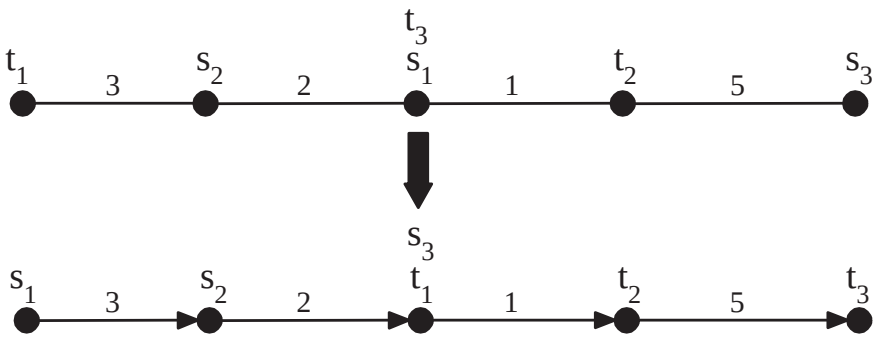


FIGURE 3.4 – Transformation d'une chaîne en un chemin pour MinMultiCCard

paire source-puits $(s_i, t_i)$ $(i \in \{1, \ldots, k\})$, $P_i$ représente l'unique chaîne reliant $s_i$ à $t_i$. Nous disons qu'*une arête $e$ coupe une chaîne $P_i$* si $e$ appartient à $P_i$. Nous recherchons alors dans ce graphe une multicoupe de poids minimum séparant les $k$ paires source-puits et dont la cardinalité est inférieure ou égale à l'entier $p$ donné.

Nous constatons les deux propriétés suivantes concernant la valeur de $p$ :

**Propriété 3.2.** *Soit $C$ une multicoupe de poids minimum séparant les $k$ paires source-puits. Si $p \geq |C|$ alors $C$ est également une solution optimale de l'instance de* MinMultiCCard *considérée.*

Nous montrons dans la section 3.4.2 comment obtenir en $O(nk)$ une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum et ainsi vérifier s'il est nécessaire de conserver la contrainte de cardinalité.

**Propriété 3.3.** *Soit $p_{min}$ le nombre maximum de chaînes disjointes par les arêtes que l'on peut former à partir des $k$ chaînes $P_1, \ldots, P_k$. Si $p < p_{min}$ alors l'instance considérée n'admet aucune solution.*

*Démonstration.* En effet, il est nécessaire pour toute multicoupe séparant les $k$ paires source-puits de posséder au moins une arête de chacune de ces $p_{min}$ chaînes disjointes. □

Nous supposons par la suite que $p \geq p_{min}$ et donc que l'instance considérée admet toujours une solution.

Nous supposons également que les propriétés 3.4, 3.5, 3.6 sont vérifiées (voir la figure 3.5 pour un exemple d'application des différents pré-traitements).

**Propriété 3.4.** *Soit $i$ et $j$ deux entiers appartenant à l'ensemble $\{1, \ldots, k\}$ tels que $i \neq j$. Alors $P_i$ n'est pas inclus dans $P_j$.*

Cette propriété signifie qu'il n'existe pas de paires source-puits *inutiles*. En effet, si $P_i$ est inclus dans $P_j$ alors toute arête coupant $P_i$ coupe également $P_j$.

Nous pouvons alors supprimer la paire $(s_j, t_j)$.

Nous allons voir que nous pouvons réaliser un pré-traitement en $O(n+k)$ utilisant une file afin de supprimer de toute instance les paires source-puits inutiles. Soit $F$ une file vide et $T$ un tableau initialement vide. Nous parcourons la chaîne de $b$ à $b'$ :

- à chaque source $s_i$ rencontrée, nous enfilons la paire $(s_i, t_i)$ dans $F$
- à chaque puits $t_i$ rencontré, si $s_i - t_i$ est dans la file, nous défilons $F$ jusqu'à l'élément $(s_i, t_i)$ (inclus) puis nous ajoutons la paire $(s_i, t_i)$ à $T$

Lorsque le parcours de la chaîne se termine, $T$ correspond à notre nouvelle liste de paires source-puits. Cet algorithme ne parcoure qu'une seule fois la chaîne et l'ensemble des paires source-puits. De plus, vérifier si une paire $s_i - t_i$ est dans la file est réalisé en temps constant en utilisant un tableau de booléen indiquant si une paire source-puits est dans la file ou non. La complexité obtenue est donc $O(n + k)$.

**Propriété 3.5.** *Il n'existe pas deux arêtes consécutives coupant exactement les mêmes $P_i$ ($i \in \{1, \ldots, k\}$).*

Cette seconde propriété permet de supposer l'absence de certaines arêtes *inutiles*. En effet, si deux arêtes consécutives interceptent les mêmes $P_i$, nous pouvons contracter l'arête de plus grand poids puisque celle-ci ne pourra jamais appartenir à aucune solution optimale.

**Propriété 3.6.** *Toute arête coupe au moins l'un des $P_i$ ($i \in \{1, \ldots, k\}$).*

Si $e$ ne coupe aucune liaison source-puits, elle n'appartient à aucune solution optimale. Nous pouvons donc la contracter.

Pour réaliser un pré-traitement permettant de contracter *toutes* les arêtes inutiles, il suffit de parcourir la chaîne de $b$ à $b'$ en :

- comparant chaque arête à sa précédente et en la contractant si son poids est plus important et si elle coupe les mêmes $P_i$
- contractant toute arête ne coupant aucun $P_i$

Un tel pré-traitement se réalise donc en $O(n)$ puisque pour vérifier si deux arêtes consécutives interceptent exactement les mêmes $P_i$, il suffit de vérifier qu'il n'y a ni source ni puits sur le sommet qui est une extrémité pour les deux arêtes.

Il est important de remarquer que pour obtenir une instance vérifiant les propriétés 3.4, 3.5 et 3.6, il est nécessaire de supprimer d'abord les paires source-puits inutiles puis les arêtes inutiles. Sinon, il est possible d'obtenir une instance ne vérifiant pas 3.5 ou 3.6 comme cela serait par exemple le cas sur l'instance de la figure 3.5 où aucune arête ne serait supprimée si on inversait l'ordre d'exécution des deux pré-traitements.

Enfin, nous supposons que les paires source-puits sont désormais ordonnées de telle sorte que : si $i < j$ alors $s_i$ est strictement plus proche de $b$ que $s_j$.

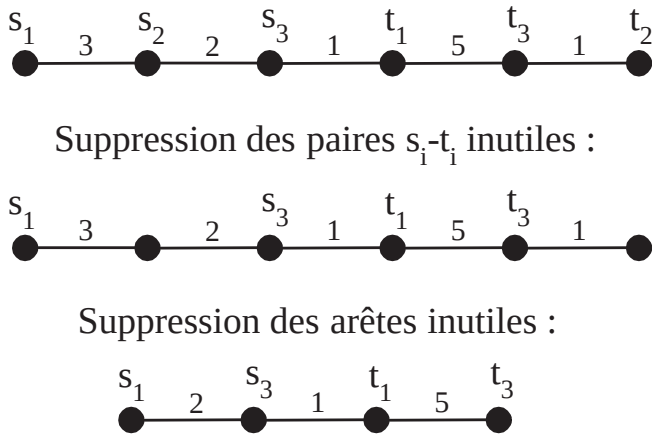


Figure 3.5 – Application des deux pré-traitements à une instance de MinMultiCCard dans les chaînes

### 3.4.2 Recherche d'une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum

Avant d'étudier plus en détail la résolution de MinMultiCCard dans les chaînes, nous allons montrer qu'une solution optimale d'une instance de MinMultiC dans les chaînes n'est plus nécessairement admissible si nous ajoutons la contrainte de cardinalité. De plus, nous décrivons comment obtenir une solution de cardinalité minimum parmi les solutions de poids minimum.

MinMultiC dans les chaînes est polynomial et Costa et al. ont proposé dans [19] l'algorithme polynomial (noté $Algo_{coupe}$) suivant :

- Parcourir la chaîne de $b'$ à $b$ et pour chaque puits $t_i$ rencontré, router un maximum d'unités de flot $f_i$ entre $s_i$ et $t_i$. Notons $C_0$ l'ensemble des arêtes saturées par le multiflot.
- Soit $C$ un ensemble d'arêtes initialement vide. Parcourir la chaîne de $b$ à $b'$ et pour chaque source $s_i$ rencontrée telle que $f_i > 0$ et telle que $C$ ne sépare pas $s_i$ de $t_i$, ajouter à $C$ l'arête de $C_0 \cap P_i$ la plus proche de $s_i$ puis supprimer de $C_0$ toute arête de $P_i$.

Costa et al. ont montré que la première étape de leur algorithme - que nous noterons par la suite $Algo_{flot}$ - détermine un multiflot entier maximum et qu'à la fin de la seconde étape, $C$ correspond à une multicoupe de poids minimum. De plus, la complexité d'$Algo_{coupe}$ est en $O(nk)$.

Cependant, parmi les multicoupes de poids minimum, cet algorithme ne détermine pas nécessairement celle de cardinalité minimum comme nous pouvons le voir sur l'instance de la figure 3.6 : $Algo_{coupe}$ donne une multicoupe de poids 2 composée des deux arêtes de poids 1 alors qu'en coupant l'arête de poids 2, nous obtenons une multicoupe de même poids mais de cardinalité moindre.

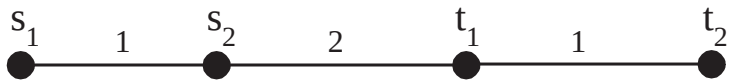


FIGURE 3.6 – Une multicoupe minimum de cardinalité non minimale

Pour déterminer une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum, nous allons voir qu'il suffit de modifier la fonction de poids des arêtes de la chaîne puis d'appliquer $Algo_{coupe}$. Nous avons alors le théorème suivant :

**Théorème 3.5.** *Déterminer une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum peut être réalisé en $O(nk)$.*

*Démonstration.* Soit $I$ et $I'$ deux instances de MINMULTIC dans les chaînes. $I$ est composée d'une chaîne $G = (V, E)$ telle que $|V| = n$, d'une fonction de poids $w : E \to \mathbb{N}^*$ et d'un ensemble $\mathcal{T}$ de $k$ paires source-puits de sommets de $G$. $I'$ est composée de $G$, de $\mathcal{T}$ et d'une fonction de poids $w'$ définie par :

$$\forall e \in E \quad w'(e) = n \cdot w(e) + 1$$

Montrons que toute multicoupe de poids minimum de $I'$ est une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum de $I$.

Puisque $I$ et $I'$ partagent le même graphe support $G$, $C$ est solution admissible de $I'$ si et seulement si $C$ est solution admissible de $I$. De plus, par définition de $w'$, on a :

$$\forall C \quad w'(C) = n \cdot w(C) + |C|$$

Soit $C'$ une solution optimale de $I'$ et $C$ une solution optimale de $I$. Nécessairement, $w'(C') \leq w'(C)$ d'où $n \cdot w(C') + |C'| \leq n \cdot w(C) + |C|$ soit :

$$w(C') \leq w(C) + \frac{|C| - |C'|}{n} < w(C) + 1$$

Ainsi, $w(C') = w(C)$ puisque $C$ est une solution optimale de $I$ et que tous les poids sont entiers. De plus, par définition de $C'$, $n \cdot w(C') + |C'| \leq n \cdot w(C) + |C|$. Ce qui donne $|C'| \leq |C|$.

$C'$ est donc bien une multicoupe de cardinalité minimum parmi les multicoupes de poids minimum de $I$. Déterminer $C'$ peut être réalisé en $O(nk)$ en résolvant l'instance $I'$ avec $Algo_{coupe}$ □

### 3.4.3 Résolution optimale du problème dual

Dans cette section, nous présentons un algorithme pour calculer la valeur optimale d'une instance de MINMULTICCARD dans les chaînes. Cet algorithme

consiste à résoudre le programme dual associé au programme linéaire modélisant MinMultiCCard dans les chaînes.

De manière semblable au programme mathématique obtenu dans la section 3.2 pour modéliser MinCCard dans les arbres orientés, considérons le programme linéaire en variables booléennes modélisant MinMultiCCard dans les chaînes décrit ci-dessous :

$$\text{(PLcoupe)} \begin{cases} Min & \sum_{e \in E} w(e) z_e & & \\ \text{s. c.} & \sum_{e \in P_i} z_e \geq 1 & \forall i \in \{1, \ldots, k\} & (C1) \\ & \sum_{e \in E} z_e \leq p & & (C2) \\ & z_e \in \{0,1\} & \forall e \in E & (C3) \end{cases}$$

En relâchant les contraintes d'intégrité (il est alors inutile d'ajouter les contraintes $z_e \leq 1$ car elles sont toujours vérifiées par à la fois l'optimum entier et l'optimum continu), nous obtenons le programme linéaire dual suivant :

$$\text{(PLdual)} \begin{cases} Max & \sum_{i=1}^{k} f_i - p\ \lambda & & \\ \text{s. c.} & \sum_{i/e \in P_i} f_i \leq w(e) + \lambda & \forall e \in E & (C4) \\ & \lambda, f_i \in \mathbb{R}^+ & \forall i \in \{1, \ldots, k\} & (C5) \end{cases}$$

où $\lambda$ est la variable duale associée à $(C2)$ et les $f_i$ sont les variables duales associées à la contrainte $(C1)$ (préalablement multipliée par $-1$).

Soit la fonction $D : \mathbb{R}^+ \to \mathbb{R}$ définie par :

$$D(\lambda) = \begin{cases} Max & \sum_{i=1}^{k} f_i - p\ \lambda & & \\ \text{s. c.} & \sum_{i/e \in P_i} f_i \leq w(e) + \lambda & \forall e \in E & (C4) \\ & f_i \in \mathbb{N} & \forall i \in \{1, \ldots, k\} & (C6) \end{cases}$$

**Propriété 3.7.** *$\max_{\lambda \in \mathbb{R}^+} D(\lambda)$ est égal à la valeur optimale de $(PLcoupe)$ et cette valeur est atteinte pour une valeur entière de $\lambda$.*

*Démonstration.* La matrice des contraintes de $(PLcoupe)$ est totalement unimodulaire (voir la section 3.2). Par dualité, c'est également le cas pour la matrice des contraintes de $(PLdual)$. Dans ce dernier, on peut donc remplacer $(C5)$ par

$$\lambda, f_i \in \mathbb{N} \quad \forall i \in \{1, \ldots, k\}$$

□

Nous allons maintenant élaborer un algorithme pour calculer $\max_{\lambda \in \mathbb{R}^+} D(\lambda)$. Pour cela, nous allons montrer que toute valeur de $D$ est calculable en $O(nk)$, que la fonction $D$ est concave et qu'il existe une valeur $B$ pour laquelle $D(B) \geq D(B+1)$. Cela permettra alors une approche par dichotomie pour obtenir le maximum atteint par la fonction $D$.

Reformulons $D$ de la manière suivante :

$$D(\lambda) = -p \cdot \lambda + \left| \begin{array}{lll} Max & \sum_{i=1}^{k} f_i & \\ \text{s. c.} & \sum_{i/e \in P_i} f_i \leq w(e) + \lambda & \forall e \in E \quad (C4) \\ & f_i \in \mathbb{N} & \forall i \in \{1, \ldots, k\} \quad (C6) \end{array} \right.$$

**Théorème 3.6.** *Pour une valeur $\lambda$ donnée, $D(\lambda)$ peut être calculé en $O(nk)$.*

*Démonstration.* Pour une valeur de $\lambda$ fixée, considérons le programme linéaire suivant :

$$\left\{ \begin{array}{lll} Max & \sum_{i=1}^{k} f_i & \\ \text{s. c.} & \sum_{i/e \in P_i} f_i \leq w(e) + \lambda & \forall e \in E \quad (C4) \\ & f_i \in \mathbb{N} & \forall i \in \{1, \ldots, k\} \quad (C6) \end{array} \right.$$

Remarquons que ce programme mathématique correspond à une modélisation de MaxMultiFlot dans la chaîne où toutes les capacités ont été augmentées de la valeur $\lambda$. Pour $i \in \{1, \ldots, k\}$, $f_i$ est alors égal au nombre d'unités de flot routées entre $s_i$ et $t_i$.

Pour en calculer la valeur optimale, il suffit donc d'appliquer $Algo_{flot}$ (voir la section 3.4.2) après avoir augmenté toutes les capacités de la valeur $\lambda$. La complexité de cet algorithme est en $O(nk)$. □

**Théorème 3.7.** *$D$ est une fonction concave.*

*Démonstration.* Dans la dernière formulation de $D$, le second terme du membre de droite est un programme linéaire dont la matrice des contraintes est totalement unimodulaire. Nous pouvons donc remplacer ce programme par son dual et ajouter une contrainte d'intégrité sur les variables duales :

$$D(\lambda) = -p \cdot \lambda + \left| \begin{array}{lll} Min & \sum_{e \in E} (w(e) + \lambda) z_e & \\ \text{s. c.} & \sum_{e \in P_i} z_e \geq 1 & \forall i \in \{1, \ldots, k\} \quad (C1) \\ & z_e \in \{0, 1\} & \forall e \in E \quad (C3) \end{array} \right.$$

Notons $l : \mathbb{R}^+ \rightarrow \mathbb{R}$ la fonction lagrangienne associée à $(PLcoupe)$ où seule la contrainte de cardinalité a été dualisée et $u$ le multiplicateur positif de Lagrange

associé à $(C2)$ :

$$l(u) = \begin{cases} Min & \sum_{e\in E} w(e)z_e + u\left(-p + \sum_{e\in E} z_e\right) & & \\ \text{s. c.} & \sum_{e\in P_i} z_e \geq 1 & \forall i \in \{1,\ldots,k\} & (C1) \\ & z_e \in \{0,1\} & \forall e \in E & (C3) \end{cases}$$

Soit :

$$l(u) = -p\cdot u + \begin{cases} Min & \sum_{e\in E}(w(e)+u)z_e & & \\ \text{s. c.} & \sum_{e\in P_i} z_e \geq 1 & \forall i \in \{1,\ldots,k\} & (C1) \\ & z_e \in \{0,1\} & \forall e \in E & (C3) \end{cases}$$

D'où :

$$\forall x \geq 0 \quad D(x) = l(x)$$

Or, $l$ est une fonction lagrangienne et est, de ce fait, concave ([49]). $D$ est donc également concave. □

Posons $B = Max(0, w_{max} - 2w_{min})$ où $w_{max}$ et $w_{min}$ sont respectivement les poids maximum et minimum des arêtes de la chaîne $G$.

Nous allons établir plusieurs lemmes pour montrer que $D(B+1) \leq D(B)$.

Soit $G'$ le graphe obtenu en ajoutant la valeur $B$ au poids de chaque arête de G. Pour toute arête $e$ de G, notons $w'(e)$ son poids dans G'.

**Lemme 3.4.** *Soit $e_1$, $e_2$ et $e_3$ trois arêtes de $G'$. Nous avons $w'(e_1) \leq w'(e_2) + w'(e_3)$.*

*Démonstration.* Montrons que $w'(e_1) - w'(e_2) - w'(e_3) \leq 0$.
Cas 1 : $B > 0$
$w'(e_1) = w(e_1)+w_{max}-2w_{min}$ et $w'(e_2)+w'(e_3) = w(e_2)+w(e_3)+2w_{max}-4w_{min}$ d'où $w'(e_1)-w'(e_2)-w'(e_3) = (w(e_1)-w_{max})+(w_{min}-w(e_2))+(w_{min}-w(e_3))$. Chaque terme du membre de droite est négatif ou nul d'où $w'(e_1) - w'(e_2) - w'(e_3) \leq 0$.
Cas 2 : $B = 0$
$w_{max} - 2w_{min} \leq 0$ et $w'(e_1) - w'(e_2) - w'(e_3) = w(e_1) - w(e_2) - w(e_3)$.
Puisque $w(e_1)$ est inférieur à $w_{max}$ et, $w(e_2)$ et $w(e_3)$ sont supérieurs à $w_{min}$, on obtient :
$w(e_1)-w(e_2)-w(e_3) \leq w_{max}-2w_{min} \leq 0$ d'où $w'(e_1)-w'(e_2)-w'(e_3) \leq 0$. □

Rappelons qu'$Algo_{flot}$ (voir la section 3.4.2) permet d'obtenir un multiflot entier maximum dans les chaînes et consiste à parcourir la chaîne de $b'$ à $b$ en routant un maximum d'unités de flot entre $s_i$ et $t_i$ à chaque puits $t_i$ rencontré. Le premier flot routé est donc $f_k$ et le dernier est $f_1$.

**Lemme 3.5.** *Soit $i < k$ tel que $P_i \cap P_k \neq \emptyset$. Alors, après application d'$Algo_{flot}$ à $G'$, au moins une arête de $P_i \cap P_k$ est saturée.*

*Démonstration.* Lorsque nous appliquons $Algo_{flot}$, le premier flot routé est $f_k$. Si une arête de $P_k \cap P_i$ est saturée alors le lemme est montré. Sinon, une arête $e_1$ de $P_k \setminus P_i$ est nécessairement saturée et $f_k = w'(e_1)$. Nous routons ensuite les flots $f_{k-1}, \ldots, f_i$. Au moins une arête de $P_i$ est alors nécessairement saturée et ainsi :

- Si une arête de $P_k \cap P_i$ est saturée alors le lemme est montré.
- Sinon, aucune arête de $P_k \cap P_i$ n'est saturée et soit $e_2 \in P_i \setminus P_k$, une arête saturée. Puisque le flot $f_k$ ne passe pas par $e_2$, nous obtenons :

$$\exists h \in \{i, \ldots, k-1\} \quad w'(e_2) = \sum_{j=i}^{h} f_j \leq \sum_{j=i}^{k-1} f_j$$

Soit une arête $e_3$ de $P_k \cap P_i$. Puisque $e_3$ n'est pas saturée, nous en déduisons que sa capacité doit vérifier :

$$w'(e_3) > \sum_{j=i}^{k} f_j \geq f_k + \sum_{j=i}^{k-1} f_j \geq w'(e_1) + w'(e_2)$$

Ce qui contredirait alors le lemme 3.4. Il existe donc au moins une arête de $P_k \cap P_i$ qui est saturée.

□

Nous venons d'établir un résultat concernant le routage des flots dans $G'$. Nous allons maintenant nous intéresser au cas où nous augmentons de 1 tous les poids de $G'$.

**Lemme 3.6.** *Si nous augmentons de 1 tous les poids de $G'$, alors dans le multiflot entier obtenu avec $Algo_{flot}$, la valeur de $f_k$ augmente de 1 unité et la valeur de tous les flots $f_i$, tels que $P_i \cap P_k \neq \emptyset$ et $i < k$, reste inchangée.*

*Démonstration.* Soit $G''$ le graphe obtenu en ajoutant 1 au poids des arêtes de $G'$ et soit $F = \{P_i | i \neq k \text{ et } P_i \cap P_k \neq \emptyset\}$. Si $F$ est vide alors la valeur de $f_k$ augmente de 1 unité et le lemme est montré. Supposons que $F$ n'est pas vide et montrons ce lemme par induction.

D'après le lemme précédent, après avoir appliqué $Algo_{flot}$ à $G'$, il existe au moins une arête $e$ de $P_{k-1} \cap P_k$ qui est saturée. D'où :

$$f_k + f_{k-1} = w'(e) \tag{3.3}$$

Lorsque nous appliquons $Algo_{flot}$ à $G''$, la valeur de $f_k$ augmente de 1 unité par rapport à sa valeur dans $G'$. Si la valeur de $f_{k-1}$ augmente de $\alpha$ unités, en faisant le bilan des flots routés sur $e$, nous obtenons alors :

$$(f_k + 1) + (f_{k-1} + \alpha) \leq w'(e) + 1 \Rightarrow f_k + f_{k-1} + \alpha \leq w'(e)$$

Donc, d'après l'équation (3.3), $\alpha = 0$ et la valeur de $f_{k-1}$ reste inchangée.

Supposons que le lemme est vérifié pour $f_{k-1}, \ldots, f_{i+1}$. D'après le lemme précédent, après avoir appliqué $Algo_{flot}$ à $G'$, il existe au moins une arête de $P_i \cap P_k$ qui est saturée. Soit $e$ une telle arête, d'où :

$$\sum_{j=i}^{k} f_j = w'(e) \tag{3.4}$$

Lorsque nous appliquons $Algo_{flot}$ à $G''$, la valeur du flot $f_k$ augmente de 1 unité par rapport à sa valeur dans $G'$ et par hypothèse d'induction, la valeur des flots $f_{k-1},\ldots,f_{i+1}$ reste inchangée. Si la valeur de $f_i$ augmente de $\alpha$ unités, en faisant le bilan des flots routés sur $e$, nous obtenons alors :

$$(f_k + 1) + \sum_{j=i+1}^{k-1} f_j + (f_i + \alpha) \leq w'(e) + 1 \Rightarrow \sum_{j=i}^{k} f_j + \alpha \leq w'(e)$$

Donc d'après l'équation (3.4), $\alpha = 0$ et la valeur de $f_i$ reste inchangée. □

Nous obtenons finalement le théorème désiré :

**Théorème 3.8.** *Soit $p_{min}$ le nombre maximum de chaînes disjointes par les arêtes obtenu à partir des $k$ chaînes $P_1, \ldots, P_k$ en partant de $b'$. Alors :*

$$D(B+1) = D(B) + p_{min} - p$$

*Démonstration.* Soit $G''$ le graphe obtenu en augmentant de 1 le poids des arêtes de $G'$ et soit $F = \{P_i | i \neq k \text{ et } P_i \cap P_k \neq \emptyset\}$. D'après le lemme précédent, après avoir appliqué $Algo_{flot}$ à $G'$ puis à $G''$, la valeur de $f_k$ a augmenté de 1 unité et pour tout $P_i \in F$, la valeur de $f_i$ reste inchangée. Nous supprimons alors la paire $s_k - t_k$ ainsi que toutes les paires $s_i - t_i$ correspondant à un élément de $F$. Nous supprimons également les arêtes devenues inutiles qui ne coupent plus aucun $P_i$. Puis, nous appliquons à nouveau le même raisonnement sur la chaîne résultante.

En passant de $G'$ à $G''$, le multiflot obtenu par $Algo_{flot}$ a augmenté de $p_{min}$ unités où $p_{min}$ est le nombre de chaînes disjointes par les arêtes que l'on peut former à partir de $P_1, \ldots, P_k$ en partant du sommet $b'$.

D'où $D(B+1) = D(B) + p_{min} - p$ (le terme "$-p$" venant du terme "$-p \cdot \lambda$" de la formule de $D$). □

D'où :

**Corollaire 3.1.**

$$D(B+1) \leq D(B)$$

*Démonstration.* En raison de la propriété 3.3 (page 49), nous avons supposé que $p \geq p_{min}$ car sinon il ne peut exister de multicoupe respectant la contrainte de cardinalité. □

Puisque $D(B+1) \leq D(B)$, que $D$ est concave et que chaque valeur de $D$ peut être calculée en $O(nk)$, une approche dichotomique permet de calculer $\max_{\lambda \in \mathbb{R}^+} D(\lambda)$ (ainsi que la valeur de $\lambda$ et des $f_i$ correspondants) en $O\left(n \cdot k \cdot Max(1, \log(B))\right)$.

En effet, nous savons que $D$ atteint son maximum pour une valeur entière de $\lambda$ comprise entre 0 et $B$. Soit $\lambda^* = \left\lfloor \frac{B-0}{2} \right\rfloor$. Nous calculons alors $D(\lambda^*)$ et $D(\lambda^*+1)$. Si $D(\lambda^*) > D(\lambda^*+1)$ alors l'optimum se trouve dans l'ensemble $\{0, \ldots, \lambda^*\}$, sinon il se trouve dans l'ensemble $\{\lambda^*+1, \ldots, B\}$. Nous réitérons ce procédé jusqu'à atteindre le maximum.

L'inconvénient de cette approche est qu'elle ne fournit que la valeur optimale et que l'on ne dispose pas d'une méthode efficace pour déduire une solution du primal à partir de la solution du dual. On pourrait par exemple utiliser les conditions des écarts complémentaires entre le primal et le dual et résoudre le système obtenu mais la complexité d'un tel algorithme serait très coûteuse, de l'ordre de $O\left(n^3 + n \cdot k \cdot Max(1, \log(B))\right)$.

De plus, nous aurions pu espérer déduire une solution du primal à partir de la chaîne où tous les poids ont été augmentés de la valeur $\lambda^*$. En effet, nous pouvons montrer assez aisément qu'une solution du primal est une multicoupe de poids minimum dans ce graphe et nous avons montré dans la section 3.4.2 que nous savons déterminer une multicoupe de cardinalité minimum parmi les multicoupes minimum d'une chaîne. Cependant, une telle multicoupe ne correspond pas forcément à une solution du problème primal car rien n'assure qu'elle est de poids minimum dans le graphe d'origine.

Enfin, nous pouvons également rapprocher cette méthode de celle proposée par Hassin et Tamir dans [36] qui, à l'instar de notre approche, permet uniquement de calculer la valeur optimale de l'instance considérée. Leur algorithme a pour complexité $O(n^2 \log^2(n))$ et peut donc s'avérer plus efficace que le nôtre si la valeur de $w_{max} - 2w_{min}$ est très importante.

### 3.4.4 Un algorithme de programmation dynamique pour MinMultiCCard dans les chaînes

Dans cette section, nous présentons un algorithme de programmation dynamique pour résoudre MinMultiCCard dans les chaînes. Contrairement à l'approche précédente qui ne permet d'obtenir que la *valeur optimale*, nous sommes ici capable de déterminer une *solution optimale*.

#### Elaboration de l'algorithme

Afin de faciliter l'élaboration de la fonction d'optimisation, nous commençons par définir la fonction $c$ suivante :

**Définition 3.4.** *Soit $\alpha$ et $\alpha'$ deux entiers tels que $0 \leq \alpha' < \alpha \leq k$.*
*Si $(P_{\alpha'+1} \cap P_\alpha) \setminus P_{\alpha+1} \neq \emptyset$ alors*

$$c(\alpha', \alpha) = \min_{e \in (P_{\alpha'+1} \cap P_\alpha) \setminus P_{\alpha+1}} w(e)$$

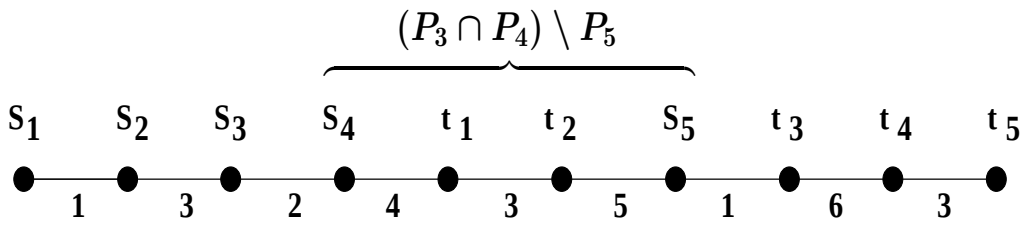


FIGURE 3.7 – Une instance où $c(2,4) = 3$

*sinon*

$$c(\alpha', \alpha) = \infty$$

Ainsi $c(\alpha', \alpha)$ peut être vue comme le poids de la "meilleure" arête (c'est-à-dire celle de poids minimum) à ajouter à une multicoupe séparant exactement les $\alpha'$ premières paires source-puits afin d'obtenir une multicoupe séparant exactement les $\alpha$ premières paires source-puits (voir la figure 3.7). $c(\alpha', \alpha) = \infty$ lorsqu'une telle arête n'existe pas.

L'algorithme de programmation dynamique va reposer sur la propriété suivante : si $C$ et $C'$ sont deux multicoupes séparant exactement les $\alpha$ premières paires source-puits alors $C$ sera préférée à $C'$ si et seulement si $w(C) < w(C')$ et $|C| = |C'|$. L'algorithme va comparer les multicoupes de même cardinalité séparant exactement le même ensemble de paires source-puits. Si deux multicoupes ont même poids et même cardinalité, seule l'une des deux - sélectionnée de manière arbitraire - sera conservée pour la suite de l'algorithme.

Nous définissons la fonction d'optimisation $g$ :

**Définition 3.5.**
$g : \{0, \ldots, k\} \times \{0, \ldots, p\} \to \mathbb{N}$
$g(0,0) = 0$, $g(\alpha, 0) = \infty \ \forall \alpha \in \{1, \ldots, k\}$ *et* $g(\alpha, \beta) = \infty \ \forall \beta > \alpha$
$g(\alpha, \beta) = \min_{\alpha' \in \{0,\ldots,\alpha-1\}} \{g(\alpha', \beta - 1) + c(\alpha', \alpha)\}$ *pour* $\alpha \geq \beta > 0$.

Montrons alors le théorème suivant sur $g$ :

**Théorème 3.9.** *Soit* $\alpha \in \{1, \ldots, k\}$ *et* $\beta \in \{1, \ldots, p\}$.
*La valeur de* $g(\alpha, \beta)$ *est égal au poids minimum d'une multicoupe de cardinalité* $\beta$ *séparant* $s_1$ *de* $t_1$, ..., $s_\alpha$ *de* $t_\alpha$ *mais pas* $s_{\alpha+1}$ *de* $t_{\alpha+1}$, ..., $s_k$ *de* $t_k$. *Si une telle multicoupe n'existe pas,* $g(\alpha, \beta) = \infty$.

*Démonstration.* La preuve se fait par récurrence sur $\beta$.

Pour $\beta = 1$, puisque pour tout $\alpha' > 0$ $g(\alpha', 0) = \infty$, nous avons nécessairement $g(\alpha, 1) = g(0,0) + c(0, \alpha)$. $c(0, \alpha)$ est fini si l'ensemble $(P_1 \cap P_\alpha) \setminus P_{\alpha+1}$ n'est pas vide, c'est-à-dire s'il existe une arête (nécessairement unique d'après la propriété 3.5 page 50) séparant exactement les $\alpha$ premières paires source-puits. Dans ce cas, $g(\alpha, 1)$ est égal soit au poids d'une multicoupe minimum de cardinalité 1 séparant les $\alpha$ premières paires source-puits, soit à l'infini.

Supposons désormais que le théorème est vrai pour une valeur $\beta \geq 1$ donnée.
Cas 1 : $g(\alpha, \beta+1) = \infty$
Par l'absurde, supposons qu'il existe une multicoupe minimum $C$ de cardinalité $\beta+1$ séparant exactement les $\alpha$ premières paires source-puits. Soit $e \in C \cap P_\alpha$. $e$ est unique puisque $C$ est minimum et que la propriété 3.5 est vérifiée. Il existe donc un entier $\alpha' < \alpha$ tel que $C \setminus \{e\}$ soit une multicoupe de cardinalité $\beta$ séparant exactement les $\alpha'$ premières paires source-puits. Donc par hypothèse de récurrence $g(\alpha', \beta) \neq \infty$ et par définition de $g$, $g(\alpha, \beta+1) \leq g(\alpha', \beta) + w(e)$. Ceci est contradictoire avec $g(\alpha, \beta+1) = \infty$ donc $C$ n'existe pas.

Cas 2 : $g(\alpha, \beta+1) \neq \infty$
Montrons tout d'abord qu'il existe au moins une multicoupe de cardinalité $\beta+1$ séparant exactement les $\alpha$ premières paires source-puits et que l'une d'elles a pour poids $g(\alpha, \beta+1)$ : puisque $g(\alpha, \beta+1)$ est finie, il existe $\alpha_1$ et $e_1 \in (P_{\alpha_1+1} \cap P_\alpha) \setminus P_{\alpha+1}$ tels que $g(\alpha, \beta+1) = g(\alpha_1, \beta) + w(e_1)$. Par hypothèse de récurrence, $g(\alpha_1, \beta)$ est égal au poids d'une multicoupe minimale $C_1$ de cardinalité $\beta$ séparant exactement les $\alpha_1$ premières paires source-puits. $C_1 \cup \{e\}$ est donc une multicoupe de cardinalité $\beta+1$ séparant exactement les $\alpha$ premières paires source-puits et $w(C_1 \cup \{e\}) = g(\alpha, \beta+1)$.

Soit $C$ une multicoupe minimum de cardinalité $\beta+1$ séparant les $\alpha$ premières paires source-puits. Puisque $C$ est minimum, il existe une unique arête $e$ appartenant à $C \cap P_\alpha$. En effet, s'il en existait d'autres, on pourrait conserver celle située la plus près de $s_\alpha$ et supprimer toutes les autres de $C$ ; ce qui serait contradictoire avec la minimalité de $C$. Soit $\alpha'$ le nombre de paires source-puits séparées par $C \setminus \{e\}$. Par hypothèse de récurrence $g(\alpha', \beta) \leq w(C \setminus \{e\})$ et par définition de $g$, $g(\alpha, \beta+1) \leq g(\alpha', \beta) + w(e)$, d'où :

$$g(\alpha, \beta+1) \leq g(\alpha', \beta) + w(e) \leq w(C \setminus \{e\}) + w(e) = w(C)$$

Ainsi, $g(\alpha, \beta+1) = w(C)$. □

Nous en déduisons de manière immédiate le corollaire suivant :

**Corollaire 3.2.** *Le poids d'une multicoupe minimum de cardinalité inférieure ou égale à $p$ est égal à* $\min_{\beta \in \{1,\ldots,p\}} \{g(k, \beta)\}$.

Nous venons ainsi de montrer que pour obtenir la valeur minimale d'une multicoupe de cardinalité au plus $p$ séparant les $k$ paires source-puits, il est suffisant de calculer les valeurs prises par la fonction $g$.

### Evaluation de la complexité de l'algorithme

Remarquons tout d'abord qu'en raison de la définition de la fonction $g$, il est relativement facile de construire de manière récursive une multicoupe correspondant à une certaine valeur de $g$ :

– $g(0, 0)$ correspond à la multicoupe vide

- si $g(\alpha,\beta) = g(\alpha',\beta-1) + w(e)$ et si $C'$ est la multicoupe correspondant à $g(\alpha',\beta-1)$ alors $C \cup \{e\}$ est la multicoupe correspondant à $g(\alpha,\beta)$

Pour obtenir la solution optimale de toute instance de MinMultiCCard dans les chaînes, nous allons procéder de la manière suivante :

- $1^{ère}$ étape : calcul des valeurs $c(\alpha',\alpha)$ pour $\alpha \in \{1,\ldots,k\}$ et $\alpha' \in \{0,\ldots,\alpha-1\}$ en recherchant l'arête de poids minimum dans $(P_{\alpha'+1} \cap P_\alpha) \setminus P_{\alpha+1}$
- $2^{ème}$ étape : calcul des valeurs $g(\alpha,\beta)$ pour $\alpha \in \{1,\ldots,k\}$ et $\beta \in \{1,\ldots,p\}$ en utilisant la définition récursive de $g$

La première étape de l'algorithme peut être réalisée "grossièrement" en $O(nk^2)$. En effet, nous devons évaluer $1+2+\cdots+k = \frac{k(k+1)}{2} = O(k^2)$ fois la fonction $c$ et chaque évaluation se fait en $O(n)$ puisque nous sélectionnons l'arête de poids minimum parmi un certain ensemble d'arêtes de la chaîne. Cependant, en calculant les valeurs de $c$ de manière plus élaborée, il est possible de ramener la complexité à $O(nk)$ en remarquant la propriété suivante. Pour $\alpha \in \{1,\ldots,k\}$ et $\alpha' \in \{0,\ldots,\alpha-2\}$ :

$$(P_{\alpha'} \cap P_\alpha) \subset (P_{\alpha'+1} \cap P_\alpha)$$

En effet, toute arête permettant de passer d'une multicoupe séparant les $\alpha'$ premières paires source-puits à une multicoupe séparant les $\alpha$ premières paires source-puits permet également de passer d'une multicoupe séparant les $\alpha'+1$ premières paires source-puits à une multicoupe séparant les $\alpha$ premières paires source-puits. D'où la formule de récurrence suivante pour $c$ :

$$c(\alpha',\alpha) = min\left\{ c(\alpha'-1,\alpha), \min_{e \in (P_{\alpha'+1} \cap P_\alpha) \setminus (P_{\alpha+1} \cup P_\alpha)} \{w(e)\} \right\}$$

Ainsi, pour une valeur $\alpha$ donnée, nous pouvons désormais calculer $c(0,\alpha)$,$c(1,\alpha)$,...,$c(\alpha-1,\alpha)$ en $O(n)$ puisque pour toutes ces valeurs, un seul parcours des arêtes de la chaîne suffit. $\alpha$ variant entre 1 et $k$, la nouvelle complexité de cette première étape de l'algorithme est donc $O(nk)$.

La deuxième partie de l'algorithme est effectuée en calculant $O(pk)$ valeurs pour $g$, chaque valeur de $g$ étant calculée en $O(k)$ à l'aide de sa définition récursive. La complexité de cette seconde partie est donc en $O(pk^2)$.

**Nous obtenons finalement une complexité globale en $O(nk+pk^2)$.**

Notons tout d'abord que la valeur de $k$ est toujours inférieure ou égale à $n-1$ en raison des pré-traitements réalisés dans la section 3.4.1 (pour un graphe quelconque, elle peut être de l'ordre de $n^2$).

Rappelons également qu'Hassin et Tamir ont proposé dans [36] un algorithme pour ce même problème calculant uniquement la valeur optimale et dont la complexité est $O(n^2 \log^2(n))$. Lorsque les valeurs de $p$ et de $k$ sont importantes,

c'est-à-dire en $O(n)$, l'utilisation de leur algorithme peut s'avérer plus intéressante si seule la valeur optimale est recherchée (et non une solution optimale). A l'inverse, pour des chaînes peu denses où le nombre de paires source-puits est faible en comparaison de la longueur totale de la chaîne, il semble plus judicieux d'utiliser notre algorithme.

Enfin, notre algorithme peut être utilisé pour résoudre MinMultiC dans les chaînes. Il suffit de choisir une valeur suffisamment grande pour $p$ (par exemple $p = k$) puis d'appliquer l'algorithme. Néanmoins, la complexité obtenue est alors moins intéressante que celle obtenue par Costa et al. avec $Algo_{coupe}$ (voir la section 3.4.2).

## 3.5 Extension de l'algorithme aux cycles et aux circuits

Dans cette section, nous montrons comment adapter l'algorithme reposant sur la programmation dynamique élaboré pour MinMultiCCard dans les chaînes, aux cas de MinMultiCCard dans les cycles et les circuits.

Soit $G = (V, E)$ un cycle de $n$ sommets, soit $k$ paires source-puits de sommets de $G$ et soit $p$ un entier donné. Rappelons que dans les cycles le nombre de chaînes entre une source et son puits est exactement égal à deux.

**Théorème 3.10.** MinMultiCCard *est polynomial dans les cycles et l'algorithme reposant sur la programmation dynamique élaboré pour les chaînes peut s'y adapter afin d'obtenir un algorithme dont la complexité est en* $O(n^2k+npk^2)$.

*Démonstration.* Soit $P_{min}$, la plus courte chaîne du cycle reliant une source à son puits. $P_{min}$ est nécessairement de longueur inférieure ou égale à $\lfloor \frac{n}{2} \rfloor$. Or, toute multicoupe doit nécessairement contenir au moins une arête de $P_{min}$.

L'algorithme pour résoudre MinMultiCCard dans les cycles consiste à supprimer tour à tour chaque arête de $P_{min}$, puis à déterminer une multicoupe minimum de cardinalité inférieure ou égale à $p-1$ sur la chaîne résultante. Une solution optimale de l'instance initiale est alors obtenue en sélectionnant la solution de plus petit poids parmi les $|P_{min}|$ solutions calculées.

La complexité s'établit en constatant que nous avons appliqué au plus $\lfloor \frac{n}{2} \rfloor$ fois l'algorithme de résolution de MinMultiCCard dans des chaînes contenant $n-1$ arêtes, puis que nous avons sélectionné la solution de poids minimum parmi celles obtenues.

La complexité de cet algorithme est donc en $O(n^2k + npk^2)$. □

L'algorithme élaboré pour MinMultiCCard dans les cycles peut être, de manière évidente, utilisé pour résoudre toute instance de MinMultiCCard dans les circuits (la seule différence est que nous résolvons dans ce cas $|P_{min}|$ instances de MinMultiCCard dans des chemins et non des chaînes). D'où le théorème suivant :

**Théorème 3.11.** MinMultiCCard *est polynomial dans les circuits et l'algorithme reposant sur la programmation dynamique élaboré pour les chaînes peut s'y adapter afin d'obtenir un algorithme dont la complexité est en* $O(n^2k+npk^2)$.

## 3.6 Résultats de complexité pour MinMultiC=Card

MinMultiC=Card diffère de MinMultiCCard en imposant à la cardinalité de la multicoupe d'être égale à la valeur $p$ donnée. Nous avons alors le théorème suivant :

**Théorème 3.12.** *Si* MinMultiCCard *est* $\mathcal{NP}$*-difficile au sens fort dans une classe* $\mathcal{C}$ *de graphes particuliers alors* MinMultiC=Card *est également* $\mathcal{NP}$*-difficile au sens fort dans* $\mathcal{C}$.

*Démonstration.* Supposons que MinMultiC=Card puisse être résolu par un algorithme pseudo-polynomial dans $\mathcal{C}$. Soit $I$ une instance de MinMultiCCard dans $\mathcal{C}$ consistant en un graphe $G$, un ensemble $T$ de $k$ paires source-puits de sommets de $G$ et un entier $p$ donné. Pour obtenir la solution optimale de $I$, il suffirait alors de résoudre $p$ instances de MinMultiC=Card dans $\mathcal{C}$ telles que la $i$-ème instance serait composée de $G$, $\mathcal{T}$ et d'un entier $p' = i$, puis de sélectionner la solution de poids minimum parmi les $p$ solutions obtenues. □

D'où les corollaires suivants :

**Corollaire 3.3.** MinMultiC=Card *est* $\mathcal{NP}$*-difficile au sens fort dans les étoiles orientées.*

*Démonstration.* C'est une conséquence immédiate du théorème précédent et du théorème 3.3 (page 48). □

**Corollaire 3.4.** MinMultiC=Card *est* $\mathcal{NP}$*-difficile au sens fort dans les étoiles non orientées et dans les graphes bipartis de degré maximum* 3.

*Démonstration.* C'est une conséquence immédiate des théorèmes 3.1 (page 41) et 3.12 (page 63). □

De plus, l'algorithme élaboré dans la section 3.4.4 permet de résoudre MinMultiC=Card dans les chaînes. En effet, nous avons montré que pour $\alpha \in \{1, \ldots, k\}$ et $\beta \in \{1, \ldots, p\}$, $g(\alpha, \beta)$ est égal au poids d'une multicoupe minimum de cardinalité $\beta$ séparant les $\alpha$ premières paires sources-puits. $g(k, p)$ est donc égal au poids minimum d'une multicoupe de cardinalité $p$ séparant toutes les paires de terminaux. Ainsi :

**Théorème 3.13.** MinMultiC=Card *est polynomial dans les chaînes et peut être résolu en utilisant l'algorithme élaboré pour* MinMultiCCard *dans les chaînes.*

De ce fait, nous obtenons alors les mêmes résultats de complexité pour MinMultiC=Card dans les chemins, cycles et circuits, que ceux obtenus pour MinMultiCCard :

**Théorème 3.14.** MinMultiC=Card *est polynomial dans les chemins, les cycles et les circuits ; il peut y être résolu en utilisant l'algorithme élaboré pour* MinMultiCCard *respectivement dans les chemins, les cycles et les circuits.*

Au niveau de la complexité des algorithmes de résolution, ceux-ci ont donc la même complexité que pour MinMultiCCard, c'est-à-dire en $O(nk+pk^2)$ pour les chaînes et les chemins et en $O(n^2k+npk^2)$ pour les cycles et les circuits.

## 3.7 Quelques variations autour de l'étoile

### 3.7.1 $\mathcal{NP}$-complétude de plusieurs problèmes connexes

Dans la section 3.3, nous avons prouvé la $\mathcal{NP}$-complétude de plusieurs problèmes liés à MinMultiCCard dans les étoiles orientées. Dans cette section, nous complétons ces résultats en établissant la complexité de plusieurs problèmes proches de ceux étudiés dans la section 3.3.

Nous avons montré dans la section 3.3.2 que le problème PondVCCard, consistant à rechercher une couverture des arêtes de poids inférieur ou égal à $B$ et de cardinalité inférieure ou égale à $p$, est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis. En raison des liens entre ensemble stable et couverture des arêtes dans un graphe biparti, nous pouvons établir un résultat similaire pour le problème PondSCard défini ci-dessous.

**Définition 3.6.** *Soit $H$ un graphe non orienté. Un* ***ensemble stable*** *de $H$ est un ensemble $S$ de sommets de $H$ tel qu'aucune arête de $H$ n'a ses deux extrémités dans $S$.*

PondSCard
**Données** : Un graphe non orienté biparti $H=(V_1,V_2,E)$, une fonction de poids $w:(V_1\cup V_2)\to\mathbb{N}^*$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il un ensemble stable $S$ de $H$ tel que $w(S)\geq B$ et $|S|\geq p$ ?

**Théorème 3.15.** PondSCard *est $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* Nous réalisons une réduction à partir de PondVCCard qui est $\mathcal{NP}$-complet au sens fort (voir le lemme 3.3 page 46).

Soit $H=(V_1,V_2,E)$ un graphe non orienté biparti et $w:(V_1\cup V_2)\to\mathbb{N}^*$ une fonction de poids. Pour montrer ce théorème, il suffit de rappeler que $S$ est une couverture des arêtes de $H$ si et seulement si $(V_1\cup V_2)\setminus S$ (c'est-à-dire le complémentaire de $S$) est un ensemble stable de $H$. Ainsi, à toute couverture des arêtes de poids inférieur ou égal à $B$ et de cardinalité inférieure ou égale à $p$ correspond un ensemble stable de poids supérieur ou égal à $w(V_1\cup V_2)-B$ et de cardinalité supérieure ou égale à $|V_1\cup V_2|-p$ (et réciproquement). □

Nous avons montré dans les sections précédentes que MinMultiCCard et MinMultiC=Card sont $\mathcal{NP}$-difficiles au sens fort dans les étoiles orientées. Regardons ce qu'il advient si la contrainte de cardinalité ne s'applique que pour

une certaine partie des arcs de l'étoile. Pour cela, considérons les problèmes de décision suivants :

CEtoileInf
**Données** : Une étoile orientée $G = (V_1, O, V_2, E)$, une fonction de poids $w : (V_1 \cup V_2) \to \mathbb{N}^*$, $k$ paires $(s_i, t_i)$ $(i \in \{1, \ldots, k\})$ de sommets de $G$ telles que $s_i \in V_1$ et $t_i \in V_2$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il une multicoupe $C$ séparant $s_i$ de $t_i$ pour $i \in \{1, \ldots, k\}$ telle que $w(C) \leq B$ et le nombre d'arcs de $C$ *reliant des sommets de* $V_1$ *à* $O$ est inférieur ou égal à $p$ ?

CEtoileEg
**Données** : Une étoile orientée $G = (V_1, O, V_2, E)$, une fonction de poids $w : (V_1 \cup V_2) \to \mathbb{N}^*$, $k$ paires $(s_i, t_i)$ $(i \in \{1, \ldots, k\})$ de sommets de $G$ telles que $s_i \in V_1$ et $t_i \in V_2$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il une multicoupe $C$ séparant $s_i$ de $t_i$ pour $i \in \{1, \ldots, k\}$ telle que $w(C) \leq B$ et le nombre d'arcs de $C$ *reliant des sommets de* $V_1$ *à* $O$ est égal à $p$ ?

La seule différence entre ces deux derniers problèmes et les problèmes de décision associé à MinMultiCCard et MinMultiC=Card dans les étoiles orientées provient de la contrainte de cardinalité. En effet, nous imposons ici uniquement le nombre d'arcs de la multicoupe reliant des sommets de $V_1$ à $O$ et l'on peut donc sélectionner autant d'arcs que l'on souhaite dans ceux reliant $O$ à des sommets de $V_2$.

De manière similaire à la preuve du théorème 3.3 (page 48) et afin de prouver la $\mathcal{NP}$-complétude de CEtoileInf et de CEtoileEg, introduisons les deux problèmes suivants.

PondVCInf
**Données** : Un graphe non orienté biparti $H = (V_1, V_2, E)$, une fonction de poids $w : (V_1 \cup V_2) \to \mathbb{N}^*$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il une couverture $S$ des arêtes de $H$ telle que $w(S) \leq B$ et $|S \cap V_1| \leq p$ ?

PondVCEg
**Données** : Un graphe non orienté biparti $H = (V_1, V_2, E)$, une fonction de poids $w : (V_1 \cup V_2) \to \mathbb{N}^*$, deux entiers positifs $B$ et $p$.
**Question** : Existe-t-il une couverture $S$ des arêtes de $H$ telle que $w(S) \leq B$ et $|S \cap V_1| = p$ ?

Nous avons les deux lemmes suivants :

**Lemme 3.7.** CEtoileInf *est équivalent à* PondVCInf.

**Lemme 3.8.** CEtoileEg *est équivalent à* PondVCEg.

La démonstration de ces deux lemmes est identique à celle du lemme 3.1 (page 45) où l'on considère le graphe de demande associé à l'instance de

multicoupe.

Nous pouvons désormais établir la complexité de CEtoileInf et CEtoileEg.

**Lemme 3.9.** PondVCEg *est $\mathcal{NP}$-complet au sens fort même si tous les poids des sommets sont égaux à* 1.

*Démonstration.* Nous utilisons une réduction à partir de VCCardEg qui est $\mathcal{NP}$-complet (voir le lemme 3.2 page 45).

Soit $I$ une instance de VCCardEg composée d'un graphe biparti $H = (V_1, V_2, E)$, dans lequel $VC_{min}$ est la taille minimum d'une couverture des arêtes, et de deux entiers positifs $p^=$, $q^=$ tels que $p^= + q^= = VC_{min}$. Soit $I'$ l'instance de PondVCEg construite de la manière suivante : nous conservons le même graphe $H$ dont nous pondérons tous les sommets à 1.

De manière évidente, $I$ admet une solution si et seulement s'il existe une couverture $S$ des arêtes de $H$ telle que $w(S) \leq VC_{min}$ et $|S \cap V_1| = p^=$. En effet, toute solution $S$ vérifie $|S| = w(S) = VC_{min}$ d'où $|S \cap V_2| = VC_{min} - p^= = q^=$. □

**Théorème 3.16.** CEtoileEg *est $\mathcal{NP}$-complet au sens fort même si tous les poids des arcs sont égaux à* 1.

*Démonstration.* C'est une conséquence immédiate du lemme précédent et du lemme 3.8. □

Etablissons maintenant la complexité de CEtoileInf

**Lemme 3.10.** PondVCInf *est $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* Le raisonnement étant identique à celui de la preuve du lemme 3.3 (page 46), nous ne donnons que la construction de la réduction. Nous effectuons une réduction à partir de VCCardEg qui est $\mathcal{NP}$-complet (voir le lemme 3.2 page 45).

Soit $I$ une instance de VCCardEg composée d'un graphe $H = (V_1, V_2, E)$ non orienté biparti, dont $VC_{min}$ est la taille minimum d'une couverture des arêtes, et de deux entiers positifs $p^=$ et $q^=$ tels que $p^= + q^= = VC_{min}$. Posons $n = |V_1| + |V_2|$.

Nous construisons une instance $I'$ pour PondVCInf de la manière suivante. Assignons un poids de 1 aux sommets de $V_1$ et un poids égal à $2n^2 + 2n + 1$ aux sommets de $V_2$. Pour chaque sommet $v$ de $V_2$, ajoutons $2n+1$ sommets de poids 1 tous reliés à $v$ et notons $V'$ l'ensemble de ces $|V_2|(2n+1)$ nouveaux sommets et $E'$ l'ensemble des nouvelles arêtes. Le graphe $H' = (V_1 \cup V', V_2, E \cup E')$ obtenu est donc un graphe biparti dont chaque sommet est pondéré par un entier strictement positif.

Il existe une couverture des arêtes de $H$ comprenant $p^=$ sommets de $V_1$ et $q^=$ sommets de $V_2$ si et seulement s'il existe une couverture $VC_{H'}$ des arêtes $H'$

telle que :

$$w(VC_{H'}) \leq p^{=} + (2n^2 + 2n + 1)q^{=} + (2n + 1)(|V_2| - q^{=})$$

$$|VC_{H'} \cap (V_1 \cup V')| \leq p^{=} + (2n + 1)(|V_2| - q^{=})$$

□

**Théorème 3.17.** CEtoileInf *est $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* C'est une conséquence immédiate du lemme précédent et du lemme 3.7. □

### 3.7.2 Un cas particulier polynomial de MinMultiCCard dans les étoiles orientées

Nous avons montré dans la section 3.3.2 que MinMultiCCard est $\mathcal{NP}$-difficile au sens fort dans les étoiles orientées. Dans cette section, nous prouvons que dans le cas où il y a au plus deux terminaux sur chaque sommet de l'étoile, MinMultiCCard devient polynomial.

Nous réutilisons des notations similaires à celles données dans la section 3.3.1 : $G = (V_1, O, V_2, E)$ est une étoile orientée dont chaque arc $a$ est pondéré par un entier strictement positif $w(a)$. $O$ est le seul sommet de degré supérieur ou égal à 2. Il n'y a aucun terminal en $O$ et il existe un chemin de longueur 2 entre toute source et son puits.

De plus, dans ce cas précis, il y a un ou deux terminaux placés en chaque sommet de $G$ (excepté $O$).

L'algorithme que nous avons élaboré consiste à transformer l'étoile en plusieurs chaînes et cycles puis à appliquer sur ces derniers l'algorithme de programmation dynamique présenté dans la section 3.4.4.

**Définition 3.7.** *Soit $I$ une instance de* MinMultiCCard *dans les étoiles orientées. $I$ est dite* élémentaire *si chaque sommet - excepté $O$ - possède un ou deux terminaux et si le graphe de demande associé est connexe.*

Notons que si $I$ est élémentaire, alors le graphe de demande est nécessairement une chaîne ou un cycle. Dans le cas contraire, le graphe de demande est la réunion de plusieurs chaînes et cycles. De manière plus intuitive, $I$ est élémentaire si elle ne peut être vue comme la réunion de plusieurs étoiles orientées (voir la figure 3.8).

**Propriété 3.8.** *Si $I$ est une instance élémentaire de* MinMultiCCard *dans les étoiles orientées, alors $I$ est équivalente à une instance particulière de* MinMultiCCard *dans les chaînes ou les cycles.*

*Démonstration.* Supposons qu'un des sommets de $G$ ne possède qu'un seul terminal. Si c'est un puits, nous inversons tous les arcs de $G$, toute source $s_i$ devenant le puits $t_i$ (et vice-versa), nous obtenons une instance équivalente où il existe au

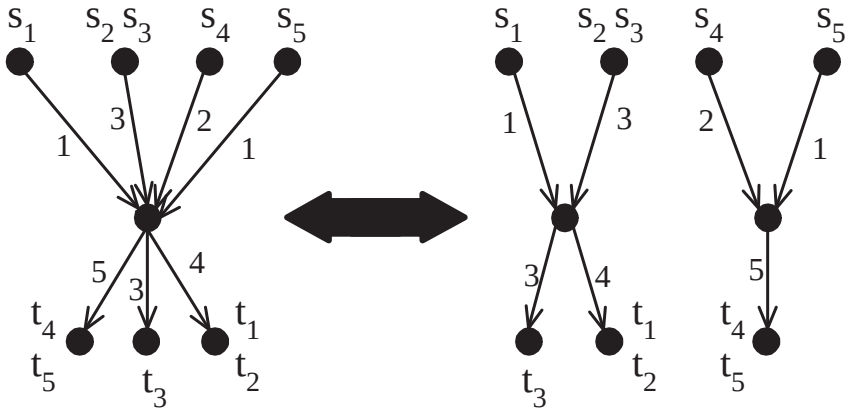


FIGURE 3.8 – Un exemple d'instance non élémentaire

moins un sommet $v_1$ possédant un unique terminal qui est une source et que nous notons $s_1$.

Soit $v_2$ le sommet où se trouve $t_1$. Pour obtenir une multicoupe il est nécessaire de couper l'arc $v_1 \to O$ ou l'arc $O \to v_2$. Si $v_2$ possède un second puits, que nous notons $t_2$ et dont la source associée est située en $v_3$, alors il est nécessaire de couper l'arc $O \to v_2$ ou l'arc $v_3 \to O$. Si l'on considère la chaîne dont les trois arêtes correspondent respectivement à $v_1 \to O$, $O \to v_2$ et $v_3 \to O$, il faut couper au moins l'une des deux premières arêtes et au moins l'une des deux dernières. Nous réitérons ce raisonnement en parcourant un à un les arcs jusqu'au sommet $v'$ de l'étoile qui ne contient qu'un seul terminal. $v'$ existe nécessairement puisque le nombre de terminaux est pair, qu'il y a au plus deux terminaux par sommet et que $v_1$ ne possède qu'un seul terminal. Nous construisons ainsi une instance de MINMULTICCARD dans les chaînes, équivalente à $I$ (voir la figure 3.9 pour un exemple).

Si tous les sommets de $G$ possèdent exactement deux terminaux alors nous choisissons un sommet au hasard, appliquons le même raisonnement et obtenons une instance de MINMULTICCARD dans les cycles. □

Nous supposons désormais que $G$ a été décomposé en $d$ instances élémentaires et nous notons $H_1, \ldots, H_{d_1}$ les chaînes obtenues à partir des instances élémentaires et $H'_1, \ldots, H'_{d'_1}$ les cycles obtenus à partir des instances élémentaires ($d_1 + d'_1 = d$). Notons $n_i$ le nombre de sommets de $H_i$, $n'_i$ le nombre de sommets de $H'_i$, $k_i$ le nombre de paires source-puits dans $H_i$ (on a nécessairement $k_i = n_i - 2$) et $k'_i$ le nombre de paires source-puits dans $H'_i$ (on a nécessairement $k'_i = n'_i - 1$). Nous travaillons désormais uniquement sur les $d_1$ chaînes et les $d'_1$ cycles.

**Propriété 3.9.** *Calculer toutes les valeurs finies de la fonction d'optimisation $g$ (voir la section 3.4.4) peut être réalisé en $O(k^2)$ pour les chaînes de type*

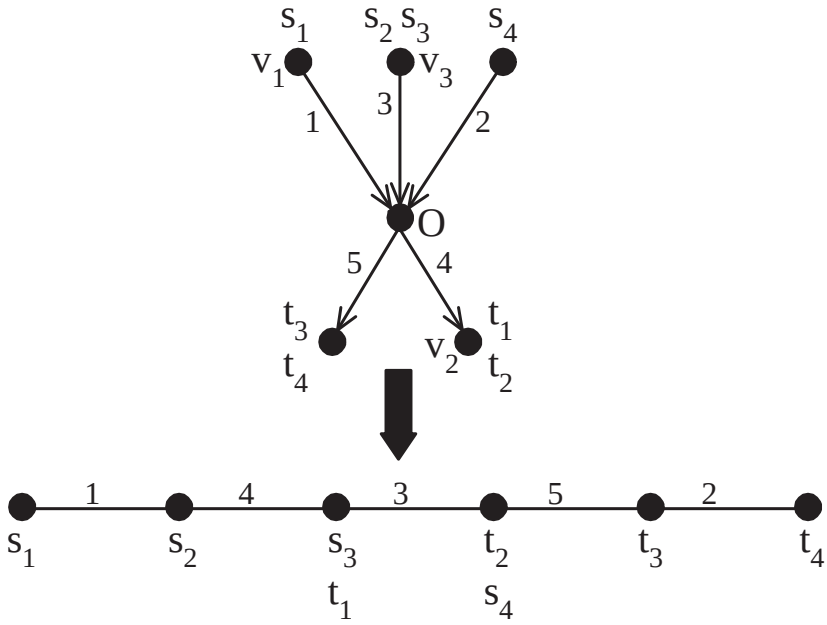


FIGURE 3.9 – Transformation d'une instance élémentaire

$H_1, \ldots, H_{d_1}$.

*Démonstration.* L'algorithme de programmation dynamique présenté dans la section 3.4.4 consiste à calculer toutes les valeurs finies de la fonction $g$ et a pour complexité $O(nk + pk^2)$. Nous allons maintenant montrer qu'il est possible d'adapter cet algorithme au cas présent dans lequel les chaînes considérées sont très particulières.

Tous les graphes $H_1, \ldots, H_{d_1}$ vérifient $k = |E| - 1$ et pour tout $i \in \{2, \ldots, k-1\}$, $P_i \cap P_j \neq \emptyset$ uniquement si $j = i-1$ ou $j = i+1$. Plus précisément, $|P_i \cap P_{i-1}| = |P_i \cap P_{i+1}| = 1$. Nous pouvons alors simplifier la formule de calcul de la fonction d'optimisation $g$ (voir la section 3.4.4 pour plus de détails sur $g$ et la fonction $c$) :
$g : \{0, \ldots, k\} \times \{0, \ldots, k\} \to \mathbb{N}$
$g(0,0) = 0$ et $g(1,1) = c(0,1)$
$g(\alpha, 0) = \infty \ \forall \alpha \in \{1, \ldots, k\}$ et $g(\alpha, \beta) = \infty \ \forall \beta > \alpha$
$g(\alpha, \beta) = \min\{g(\alpha - 2, \beta - 1) + c(\alpha - 2, \alpha), g(\alpha - 1, \beta - 1) + c(\alpha - 1, \alpha)\}$ pour $\alpha \geq 2, \beta \geq 1$ et $\alpha \geq \beta$.

Rappelons que $c(\alpha-1, \alpha)$ et $c(\alpha-2, \alpha)$ sont respectivement égaux au poids de l'arête de poids minimum de $P_\alpha \cap P_\alpha \setminus P_{\alpha+1} = P_\alpha \setminus P_{\alpha+1}$ et de $P_{\alpha-1} \cap P_\alpha \setminus P_{\alpha+1} = P_{\alpha-1} \cap P_\alpha$. Or, la cardinalité de ces deux ensembles est égale à 1 donc $c(\alpha-1, \alpha)$ et $c(\alpha-2, \alpha)$ se calculent en temps constant. C'est également le cas pour le calcul des valeurs de la fonction $g$ où nous réalisons le minimum entre deux éléments.

Puisqu'il est nécessaire de calculer $O(k^2)$ valeurs de $g$, la complexité globale

de notre algorithme est en $O(k^2)$. □

Nous pouvons désormais décrire le déroulement de notre algorithme. Nous allons calculer les valeurs de la fonction $g$ pour chacun des graphes $H_1, \ldots, H_{d_1}$ puis pour les graphes $H'_1, \ldots, H'_{d'_1}$. Cependant, nous ne traitons pas de manière indépendante chacun de ces graphes. Nous nous servons des résultats obtenus pour le graphe précédent afin de modifier l'initialisation de la fonction $g$ pour le nouveau graphe considéré. Cela nous permet d'obtenir un algorithme global de programmation dynamique calculant une solution optimale pour l'instance initiale.

Nous débutons en calculant toutes les valeurs finies de $g$ pour $H_1$ et nous posons $g^*(\beta) = g(k_1, \beta)$ pour toute valeur $\beta \in \{1, \ldots, k_1\}$. Nous calculons ensuite les valeurs de la fonction $g$ pour $H_2$ de la manière suivante :
$g : \{0, \ldots, k_2\} \times \{1, \ldots, k_1 + k_2\} \to \mathbb{N}$
$g(0, \beta) = g^*(\beta)\ \forall \beta \in \{1, \ldots, k_1\}$ et $g(0, \beta) = \infty\ \forall \beta \in \{k_1 + 1, \ldots, k_1 + k_2\}$
$g(1, \beta + 1) = g^*(\beta) + c(1, 1)\ \forall \beta \in \{1, \ldots, k_1\}$ et $g(1, \beta + 1) = \infty$ $\forall \beta \in \{k_1 + 1, \ldots, k_1 + k_2 - 1\}$
$g(\alpha, 0) = \infty\ \forall \alpha \in \{0, \ldots, k_2\}$ et $g(\alpha, 1) = \infty\ \forall \alpha \in \{1, \ldots, k_2\}$
$g(\alpha, \beta) = \min\{g(\alpha - 2, \beta - 1) + c(\alpha - 2, \alpha), g(\alpha - 1, \beta - 1) + c(\alpha - 1, \alpha)\}$ pour $\alpha \geq 2$ et $\beta \geq 2$.
Puis, nous posons $g^*(\beta) = g(k_2, \beta)$ pour toute valeur de $\beta \in \{1, \ldots, k_1 + k_2\}$ et nous réitérons de manière similaire le calcul de la fonction $g$ pour $H_3, \ldots, H_{d_1}$. Il est à noter que nous recalculons $g^*$ après avoir traité chaque chaîne $H_i$. $g^*(\beta)$ est alors égal au poids minimum d'une multicoupe de cardinalité $\beta$ pour les $i$ premières chaînes. Ainsi, après le calcul de $g$ pour le graphe $H_{d_1}$, nous posons $g^*(\beta) = g(k_{d_1}, \beta)$ pour toute valeur de $\beta \in \{1, \ldots, k_1 + \cdots + k_{d_1}\}$.

Il reste désormais à traiter le cas des cycles $H'_1, \ldots, H'_{d'_1}$. Nous commençons tout d'abord par $H'_1$. Soit $e$ et $e'$ deux arêtes consécutives de $H'_1$. Toute multicoupe de $H'_1$ contient nécessairement $e$ ou $e'$ puisqu'il existe $i$ tel que $P_i = \{e, e'\}$. Nous supprimons $e$ ainsi que les deux paires source-puits coupées par $e$. Nous obtenons ainsi une chaîne pour laquelle nous calculons les valeurs de la fonction $g$ de manière similaire aux chaînes $H_1, \ldots, H_{d_1}$ et nous posons $g_1^*(\beta+1) = g(k'_1-2, \beta) + w(e)$ pour toute valeur de $\beta \in \{0, \ldots, k_1 + \cdots + k_{d_1} + k'_1 - 1\}$. De la même manière, nous supprimons de $H'_1$ l'arête $e'$ ainsi que les deux paires source-puits coupées par $e'$, calculons les valeurs de $g$ sur la chaîne résultante et posons $g_2^*(\beta + 1) = g(k'_1 - 2, \beta) + w(e')$ pour toute valeur de $\beta \in \{0, \ldots, k_1 + \cdots + k_{d_1} + k'_1 - 1\}$. Nous fusionnons alors $g_1^*$ et $g_2^*$ en $g^*$ de la manière suivante :

$$\forall \beta \in \{1, \ldots, k_1 + \cdots + k_{d_1} + k'_1\} \quad g^*(\beta) = min\{g_1^*(\beta), g_2^*(\beta)\}$$

Nous réitérons ce raisonnement sur chacun des cycles restants $H'_2, \ldots, H'_{d'_1}$. Après le calcul des valeurs de $g$ et de $g^*$ pour le dernier cycle $H'_{d'_1}$, nous obtenons :

**La valeur optimale de l'instance initiale est égale à** $Min_{\beta \leq p}\{g^*(\beta)\}$.

La complexité de l'algorithme s'établit de manière immédiate. Nous avons montré dans la démonstration de la propriété 3.9 que toute valeur de la fonction $g$ se calcule en temps constant. De plus nous calculons $O\left((k_1 + \cdots + k_{d_1} + k'_1 + \cdots + k'_{d'_1})^2\right)$ valeurs de $g$. La complexité globale de l'algorithme est donc en $O(k^2)$ (où $k = k_1 + \cdots + k_{d_1} + k'_1 + \cdots + k'_{d'_1}$ correspond au nombre de paires source-puits de l'étoile).

# Chapitre 4

# Généralisations aux coupes et multicoupes multicritères

## 4.1 Introduction

Dans ce chapitre, nous étudions les problèmes $R-$CriC et $R-$CriMultiC, généralisations multicritères du problème de la coupe minimum et de la multicoupe minimum. $R-$CriC et $R-$CriMultiC ont été relativement peu étudiés à ce jour, le principal résultat les concernant ayant été apporté par Papadimitriou et Yannakakis qui ont montré que $2-$CriC est $\mathcal{NP}$-complet au sens fort dans les graphes non orientés ([50]).

De manière évidente, tout résultat de $\mathcal{NP}$-difficulté obtenu pour MinMultiC se transforme en un résultat de $\mathcal{NP}$-complétude pour $R-$CriMultiC puisque $1-$CriMultiC correspond au problème de décision associé à MinMultiC. Ainsi, $R-$CriMultiC est $\mathcal{NP}$-complet au sens fort dans les étoiles non orientées même si tous les poids des arêtes sont égaux à 1 ([30]).

De plus, les problèmes de décision associés à MinCCard et MinMultiCCard sont des cas particuliers de $2-$CriC et de $2-$CriMultiC où la seconde valuation de chaque arête est égale à 1. Les cas "difficiles" de MinCCard et MinMultiCCard l'étant donc également pour $R-$CriC et $R-$CriMultiC, nous en déduisons les deux théorèmes suivants :

**Théorème 4.1.** $2-$CriC *est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis de degré maximum* 3.

*Démonstration.* C'est une conséquence directe du théorème 2.4 (page 38). □

**Théorème 4.2.** $2-$CriMultiC *est $\mathcal{NP}$-complet au sens fort dans les étoiles orientées.*

*Démonstration.* C'est une conséquence directe du théorème 3.3 (page 48). □

Dans les sections suivantes, nous donnons d'autres résultats de $\mathcal{NP}$-complétude pour $R-$CriC dans des graphes particuliers et nous déterminons si

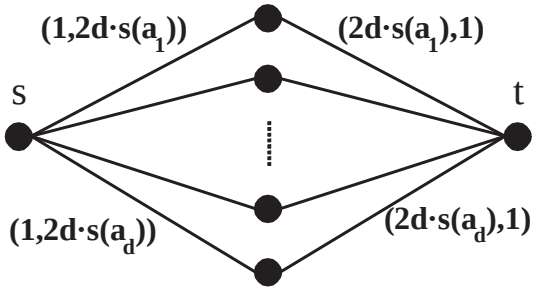


FIGURE 4.1 – Le graphe $K_{2,d}$ construit pour $2-$CRIC

les cas polynomiaux de MINMULTICCARD - comme les chaînes ou les cycles - le restent pour $R-$CRIMULTIC.

## 4.2 Etude du problème de la coupe simple multicritère

Nous donnons dans cette section de nouveaux résultats de complexité pour $R-$CRIC dans des classes de graphes particuliers.

Rappelons que nous notons par $K_{i,j}$, le graphe biparti complet $G=(V_1,V_2,E)$ où $|V_1|=i$ et $|V_2|=j$. De plus, le problème $\mathcal{NP}$-complet PARTITION ([27]) est défini de la manière suivante :

PARTITION
**Données** : Un ensemble $A$ composé de $d$ éléments, une fonction de poids $s$ : $A \rightarrow \mathbb{N}^*$.
**Question** : Existe-t-il un sous ensemble $A'$ inclus dans $A$ tel que $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a)$ ?

**Théorème 4.3.** *$2-$CRIC est $\mathcal{NP}$-complet dans $K_{2,d}$ même si la source et le puits sont les deux seuls sommets de degré supérieur ou égal à 3.*

*Démonstration.* Nous réalisons une réduction à partir de PARTITION.

Soit $I$ une instance de PARTITION composée d'un ensemble $A$ de $d$ éléments, d'une fonction de poids $s : A \rightarrow \mathbb{N}^*$ et d'un entier positif $S$ tel que $\sum_{a\in A} s(a) = S$. Nous construisons une instance $I'$ de $2-$CRIC de la manière suivante (voir la figure 4.1) : soit $G=(V_1,V_2,E)$ un graphe biparti complet avec $V_1=\{s,t\}$ (d'où $|V_1|=2$) et $|V_2|=d$. $G$ est donc composé de $d$ chaînes disjointes par les arêtes - que nous notons $P_a$ ($a\in A$) - de longueur 2 reliant $s$ à $t$. Pour $a\in A$, soit $e_a$ et $e'_a$ les deux arêtes de $P_a$ et posons alors $w_1(e_a)=w_2(e'_a)=1$ et $w_2(e_a)=w_1(e'_a)=2d\cdot s(a)$.

Montrons qu'il existe une solution de $I$ si et seulement s'il existe une coupe $C$ séparant $s$ et $t$ telle que $w_1(C)\leq d+dS$ et $w_2(C)\leq d+dS$.

Soit $A'$ une solution de $I$, construisons alors une solution $C$ de $I'$ de la manière suivante :

$$C = \{e_a | a \in A'\} \cup \{e'_a | a \notin A'\}$$

De cette manière, il y a exactement une arête de chaque $P_a$ dans $C$ et nous avons :
$w_1(C) = \sum_{a\in A'} w_1(e_a) + \sum_{a\notin A'} w_1(e'_a) = \sum_{a\in A'} 1 + 2d \sum_{a\notin A'} s(a) = |A'| + dS \leq d + dS$
$w_2(C) = \sum_{a\in A'} w_2(e_a) + \sum_{a\notin A'} w_2(e'_a) = dS + (d - |A'|) \leq dS + d$

Réciproquement, soit $C$ une solution de $I'$. Nous construisons alors une solution $A'$ de $I$ de la manière suivante :

$$A' = \{a | e_a \in C\}$$

Montrons que l'ensemble $A'$ ainsi construit vérifie $\sum_{a\in A'} s(a) = \sum_{a\notin A'} s(a) = \frac{S}{2}$.

Puisque $C$ est solution de $I'$, $w_1(C) \leq d + dS$. De plus, par construction, $w_1(C) \geq \sum_{a\in A'} w_1(e_a) + \sum_{a\notin A'} w_1(e'_a) = |A'| + 2d \sum_{a\notin A'} s(a)$.
D'où $d + dS \geq |A'| + 2d \sum_{a\notin A'} s(a)$ soit :

$$\sum_{a\notin A'} s(a) \leq \frac{S}{2} + \frac{d - |A'|}{2d} < \frac{S}{2} + 1 \tag{4.1}$$

Puisque $C$ est solution de $I'$, $w_2(C) \leq d + dS$. De plus, par construction, $w_2(C) \geq \sum_{a\in A'} w_2(e_a) + \sum_{a\notin A'} w_2(e'_a) = 2d \sum_{a\in A'} s(a) + d - |A'|$.
D'où $d + dS \geq 2d \sum_{a\in A'} s(a) + d - |A'|$ soit :

$$\sum_{a\in A'} s(a) \leq \frac{S}{2} + \frac{|A'|}{2d} < \frac{S}{2} + 1 \tag{4.2}$$

Puisque $\sum_{a\in A} s(a) = S$ et d'après les équations (4.1) et (4.2), nous obtenons $\sum_{a\in A'} s(a) = \sum_{a\notin A'} s(a) = \frac{S}{2}$ □

Pour obtenir un résultat de $\mathcal{NP}$-complétude *au sens fort* pour $R - $ CRIC, nous définissons les graphes suivants :

**Définition 4.1.** *Soit $i$ et $j$ deux entiers strictement positifs. $H_{i,j}$ est le graphe composé de deux sommets connectés par $i$ chaînes disjointes (par les sommets) de longueur $j \geq 2$.*

Remarquons que $H_{i,j}$ est un graphe biparti planaire quelles que soient les valeurs de $i$ et de $j$. De plus, rappelons également la définition du problème 3-PARTITION qui est $\mathcal{NP}$-complet au sens fort ([27]) :

3-PARTITION
**Données** : Un ensemble $A$ composé de $3d$ éléments, une fonction de poids $s : A \to \mathbb{N}^*$ telle qu'il existe un entier positif $S$ vérifiant $s(A) = dS$.
**Question** : Existe-t-il une partition de $A$ en $d$ ensembles $A_1, \ldots, A_d$ telle que pour $i \in \{1, \ldots, d\}$, $|A_i| = 3$ et $s(A_i) = S$ ?

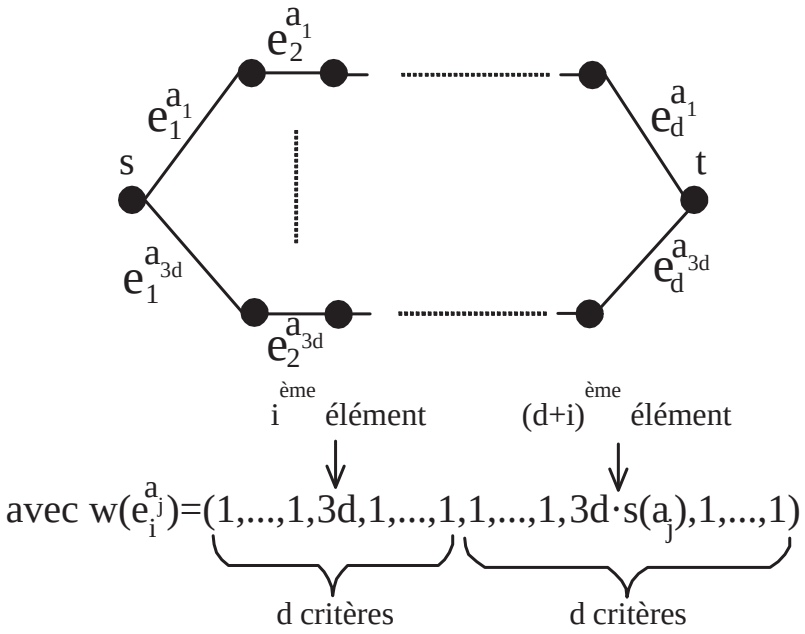


FIGURE 4.2 – Le graphe $H_{3d,d}$ obtenu pour $2d-$CRIC

**Théorème 4.4.** *$2d-$CRIC est $\mathcal{NP}$-complet au sens fort dans $H_{3d,d}$ même quand la source et le puits sont les deux seuls sommets de degré supérieur ou égal à 3.*

*Démonstration.* Nous réalisons une réduction à partir de 3-PARTITION.

Soit $I$ une instance de 3-PARTITION composée d'un ensemble $A$ de $3d$ éléments, d'une fonction de poids $s : A \to \mathbb{N}^*$ et d'un entier positif $S$ tel que $\sum_{a\in A} s(a) = dS$. Nous construisons une instance $I'$ de $2d-$CRIC de la manière suivante (voir la figure 4.2) : considérons le graphe $H_{3d,d}$ où $s$ et $t$ sont les deux sommets connectés par les $3d$ chaînes disjointes - notées $P_a$ $(a \in A)$ - de longueur $d$. Pour chaque $a \in A$, notons $e_1^a, \ldots, e_d^a$ les $d$ arêtes de $P_a$ et posons pour tout $i \in \{1, \ldots, d\}$, $w_i(e_i^a) = 3d$, $w_{d+i}(e_i^a) = 3ds(a)$ et pour $j \neq i$, $w_i(e_j^a) = w_{d+i}(e_j^a) = 1$. Les $d$ premiers critères vont permettre de s'assurer que chaque $A_i$ possède exactement 3 éléments et les $d$ autres critères que chaque $A_i$ vérifie $s(A_i) = S$.

Montrons qu'il existe une solution de $I$ si et seulement s'il existe une coupe $C$ séparant $s$ de $t$ telle que :

$$w_i(C) \leq 12d - 3 \quad \forall i \in \{1, \ldots, d\} \tag{4.3}$$

$$w_i(C) \leq 3dS + 3d - 3 \quad \forall i \in \{d+1, \ldots, 2d\} \tag{4.4}$$

Soit $A_1, \ldots, A_d$ une solution de $I$. Construisons une solution $C$ de $I'$ de la manière suivante :

$$C = \{e_i^a | a \in A_i\}$$

De cette manière, il y a exactement une arête de chaque $P_a$ dans $C$ et nous avons :
$\forall i \in \{1,\ldots,d\} \quad w_i(C) = \sum_{a\in A_i} w_i(e_i^a) + \sum_{a\notin A_i} 1 = |A_i| \cdot 3d + (|A| - |A_i|) = 9d + 3d - 3 = 12d - 3$
$\forall i \in \{1,\ldots,d\} \quad w_{d+i}(C) = \sum_{a\in A_i} w_{d+i}(e_i^a) + \sum_{a\notin A_i} 1 = 3d \sum_{a\in A_i} s(a) + |A| - |A_i| = 3dS + 3d - 3$.

Réciproquement, soit $C$ une solution de $I'$. Construisons une solution $A_1,\ldots,A_d$ de $I$ de la manière suivante :

$$\forall i \in \{1,\ldots,d\} \quad A_i = \{a | e_a^i \in C \text{ et } \forall j < i \; e_a^j \notin C\}$$

Montrons tout d'abord que pour tout $i$ appartenant à $\{1,\ldots,d\}$, $|A_i| = 3$. Soit $i \in \{1,\ldots,d\}$. Par construction, $w_i(C) \geq \sum_{a\in A_i} 3d + \sum_{a\notin A_i} 1 = 3d|A_i| + (3d - |A_i|)$. D'où, d'après l'équation (4.3), $3d|A_i| + (3d - |A_i|) \leq 12d - 3$ soit :

$$|A_i| \leq 3$$

Puisque $\sum_{i=1}^{d} |A_i| = 3d$, nous avons nécessairement pour tout $i$ de $\{1,\ldots,d\}$, $|A_i| = 3$.

Il nous reste désormais à prouver que $s(A_i) = S$ pour tout $i$ de $\{1,\ldots,d\}$. Soit $i$ appartenant à $\{1,\ldots,d\}$. Par construction, $w_{d+i}(C) \geq \sum_{a\in A_i} 3d \cdot s(a) + \sum_{a\notin A_i} 1 = 3d \cdot s(A_i) + (3d - 3)$. D'où, d'après l'équation (4.4), $3d \cdot s(A_i) + (3d - 3) \leq 3dS + 3d - 3$ soit :

$$s(A_i) \leq S$$

Puisque $\sum_{i=1}^{d} s(A_i) = s(A) = dS$, nous avons nécessairement pour tout $i$ de $\{1,\ldots,d\}$, $s(A_i) = S$. □

**Théorème 4.5.** *$R-$CRIC est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis planaires de degré maximum 3.*

*Démonstration.* Nous venons de montrer que $2d-$CRIC est $\mathcal{NP}$-complet au sens fort dans $H_{3d,d}$ qui est toujours un graphe biparti planaire. Pour obtenir un degré maximum de 3, il suffit alors de réutiliser les deux transformations proposées dans la section 2.3 permettant de ramener le degré maximum de tout graphe à 3 puis de le rendre biparti. Ces deux transformation conservant la planarité, nous obtenons le théorème désiré. □

Cependant, pour les arbres non orientés (donc également pour les chaînes) et les cycles, $R-$CRIC reste polynomial :

**Théorème 4.6.** *$R-$CRIC est polynomial dans les arbres non orientés.*

*Démonstration.* Il n'y a qu'une seule chaîne $P$ reliant la source au puits. Il suffit donc de parcourir $P$ et de déterminer s'il existe une arête $e$ de $P$ vérifiant $w_i(e) \leq B_i$ pour tout $i \in \{1,\ldots,R\}$. □

**Théorème 4.7.** *$R-$CriC est polynomial dans les cycles.*

*Démonstration.* La preuve est similaire à celle du théorème précédent puisqu'ici, il y a exactement deux chaînes reliant la source au puits. □

Enfin, il est très facile d'adapter les théorèmes de cette section au cas des graphes orientés : par exemple, pour le théorème 4.3, il suffit d'orienter les arêtes de telle sorte que le graphe obtenu consiste en $d$ *chemins* disjoints (par les sommets) de longueur 2 reliant $s$ à $t$. D'où le théorème suivant :

**Théorème 4.8.** *$R-$CriC est $\mathcal{NP}$-complet au sens fort dans les graphes orientés bipartis planaires mais polynomial dans les arbres orientés et les circuits.*

## 4.3 Etude du problème de la multicoupe multicritère

Dans cette section, nous nous intéressons à la recherche d'une multicoupe multicritère. $R-$CriMultiC peut ainsi être vu comme le problème le plus général traité dans cette première partie du mémoire. En effet, MinCCard, $R-$CriC, MinMultiC ou encore MinMultiCCard sont en réalité des cas particuliers de $R-$CriMultiC. De ce fait, les cas "difficiles" pour les différents problèmes cités, comme les graphes bipartis planaires de degré maximum 3, les étoiles orientées et les étoiles non orientées, sont également "difficiles" pour $R-$CriMultiC.

Nous allons maintenant nous intéresser aux chaînes et aux cycles qui sont des cas polynomiaux pour les problèmes MinCCard, $R-$CriC, MinMultiC et MinMultiCCard.

### 4.3.1 $R-$CriMultiC dans les chaînes

Nous montrons le théorème suivant :

**Théorème 4.9.** *$2-$CriMultiC est $\mathcal{NP}$-complet dans les chaînes.*

*Démonstration.* Nous réalisons une réduction à partir de $2-$CriC dans $K_{2,d}$ où la source et le puits sont les deux seuls sommets de degré supérieur ou égal à 3. Nous avons prouvé la $\mathcal{NP}$-complétude de ce problème dans le théorème 4.3 (page 74).

Soit $I$ une instance de $2-$CriC composée d'un graphe biparti complet $G = (V_1, V_2, E)$ tel que $V_1 = \{s, t\}$ et $|V_2| = d$, de deux fonctions $w_1$ et $w_2$ définies de $E$ dans $\mathbb{N}^*$, et de deux entiers $B_1$ et $B_2$. Il y a exactement $d$ chaînes disjointes $P_1, \ldots, P_d$ de longueur 2 reliant $s$ à $t$. Pour $i \in \{1, \ldots, d\}$, notons $e_1^i$ et $e_2^i$ les deux arêtes de $P_i$.

Nous construisons une instance $I'$ de $2-$CriMultiC de la manière suivante (voir la figure 4.3). Considérons une chaîne $G' = (V', E')$ de $2d$ arêtes $e_1'^1, e_2'^1, \ldots, e_1'^d, e_2'^d$ et deux fonctions de poids $w_1'$ et $w_2'$ pour les arêtes de $G'$ telles que pour tout $i \in \{1, \ldots, d\}$, $w_1'(e_1'^i) = w_1(e_1^i)$, $w_2'(e_1'^i) = w_2(e_1^i)$, $w_1'(e_2'^i) = w_1(e_2^i)$

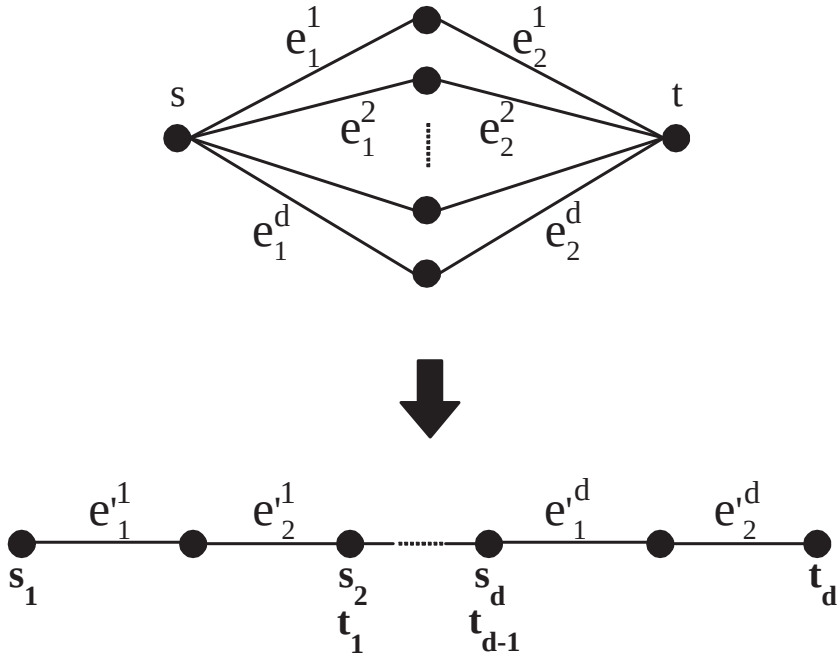


FIGURE 4.3 – Transformation d'une instance de $2-$CRIC en une instance de $2-$CRIMULTIC

et $w'_2(e'^i_2) = w_2(e^i_2)$. De plus, pour $i \in \{1,\ldots,d\}$ nous plaçons la source $s_i$ et le puits $t_i$ de telle sorte que l'unique chaîne les reliant soit composée de $e'^i_1, e'^i_2$.

Montrons qu'il existe une solution $C$ de $I$ si et seulement s'il existe une multicoupe $C'$ séparant $s_1$ de $t_1$,..., $s_d$ de $t_d$ telle que $w'_1(C') \leq B_1$ et $w'_2(C') \leq B_2$.

Un ensemble d'arêtes $C$ (resp. $C'$) est une solution de $I$ (resp. de $I'$) si et seulement si pour tout $i \in \{1,\ldots,d\}$ $e^i_1 \in C$ ou $e^i_2 \in C$ (resp. $e'^i_1 \in C'$ ou $e'^i_2 \in C'$). Ainsi, nous pouvons construire $C$ à partir de $C'$ - et réciproquement - de la manière suivante : pour tout $i \in \{1,\ldots,d\}$ et pour tout $j \in \{1,2\}$, $e^i_j \in C$ si et seulement si $e'^i_j \in C'$. □

Une conséquence immédiate de ce théorème est le corollaire suivant :

**Corollaire 4.1.** *Pour toute valeur fixée de $R \geq 2$, $R-$*CRIMULTIC *est $\mathcal{NP}$-complet dans les chaînes.*

Dans le cas où le nombre de critères n'est pas fixé, nous obtenons :

**Théorème 4.10.** *$R-$*CRIMULTIC *est $\mathcal{NP}$-difficile au sens fort dans les chaînes.*

*Démonstration.* Nous réalisons une réduction à partir de $2d-$CRIC dans $H_{3d,d}$ où la source et le puits sont les deux seuls sommets de degré supérieur ou égal à 3. Nous avons prouvé la $\mathcal{NP}$-complétude de ce problème dans le théorème 4.4

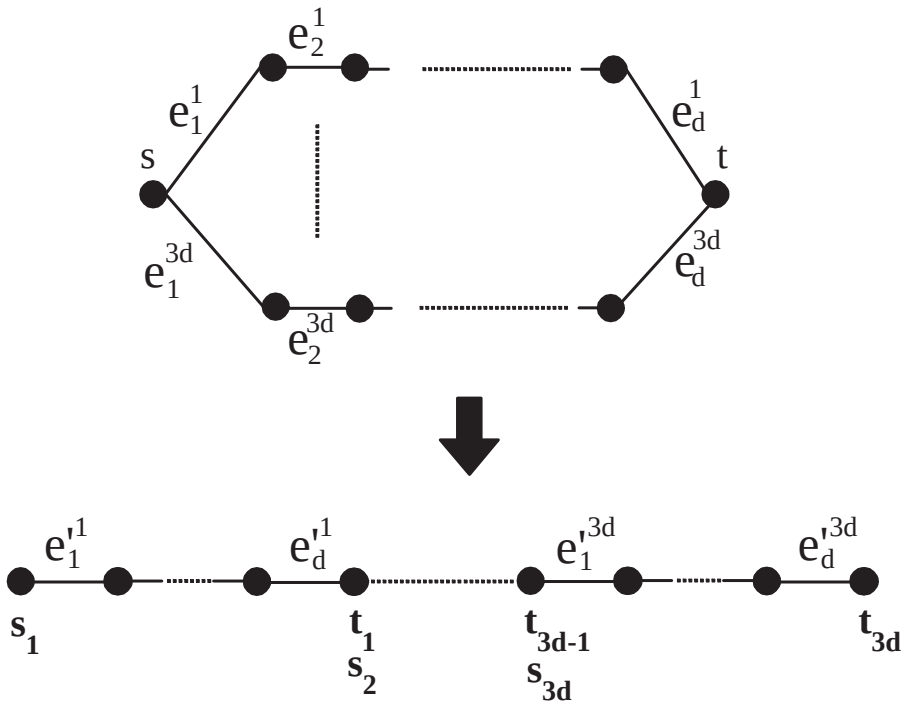


FIGURE 4.4 – Transformation d'une instance de $R-$CRIC en une instance de $R-$CRIMULTIC

(page 76).

Soit $I$ une instance de $2d-$CRIC composée du graphe $H_{3d,d}$, de $2d$ fonctions de poids $w_1,\ldots,w_{2d}$ pondérant les arêtes de $H_{3d,d}$ et de $2d$ entiers $B_1,\ldots,B_{2d}$. Il y a exactement $3d$ chaînes disjointes $P_1,\ldots,P_{3d}$ de longueur $d$ reliant $s$ à $t$ et pour $i\in\{1,\ldots,3d\}$, notons $e_1^i,\ldots,e_d^i$ les $d$ arêtes de $P_i$.

Nous construisons une instance $I$ pour $2d-$CRIMULTIC de la manière suivante (voir la figure 4.4). Considérons une chaîne $G'=(V',E')$ de $3d^2$ arêtes $e_1^{'1},\ldots,e_d^{'1},\ldots,e_1^{'3d},\ldots,e_d^{'3d}$ et $2d$ fonctions de poids $w'_1,\ldots,w'_{2d}$ pondérant les arêtes de $G'$ telles que :

$$\forall i\in\{1,\ldots,3d\}\ \forall j\in\{1,\ldots,d\}\ \forall k\in\{1,\ldots,2d\}\quad w'_k(e_j^{'i})=w_k(e_j^i)$$

De plus, pour $i\in\{1,\ldots,3d\}$ nous plaçons la source $s_i$ et le puits $t_i$ de telle sorte que l'unique chaîne les reliant soit composée de $e_1^{'i},\ldots,e_d^{'i}$.

Montrons qu'il existe une solution $C$ de $I$ si et seulement s'il existe une multicoupe $C'$ séparant $s_1$ de $t_1$,..., $s_{3d}$ de $t_{3d}$ telle que $w'_k(C')\leq B_k$ pour tout $k\in\{1,\ldots,2d\}$.

Un ensemble d'arêtes $C$ (resp. $C'$) est une solution de $I$ (resp. de $I'$) si et seulement si pour tout $i\in\{1,\ldots,3d\}$ il existe $j\in\{1,\ldots,d\}$ tel que $e_j^i\in C$ (resp.

$e'^i_j \in C'$). Ainsi, nous pouvons construire $C$ à partir de $C'$ - et réciproquement - de la manière suivante : pour tout $i \in \{1, \ldots, 3d\}$ et pour tout $j \in \{1, d\}$, $e^i_j \in C$ si et seulement si $e'^i_j \in C'$. □

Il nous reste désormais à déterminer la complexité exacte de $R-$CriMultiC pour $R$ fixé, c'est-à-dire déterminer si ce problème est $\mathcal{NP}$-complet au *sens fort* ou non.

Afin d'apporter une réponse à cette question, nous allons élaborer un algorithme pseudo-polynomial résolvant $R-$CriMultiC pour $R$ fixé. Pour cela, introduisons la fonction d'optimisation $h$ suivante :

**Définition 4.2.** $h(0, 0, \ldots, 0) = 0$. *Soit* $\alpha \in \{1, \ldots, k\}$ *et* $\beta_i \leq B_i$ *pour* $2 \leq i \leq R$.
*Si* ($\alpha = 0$ *et* $\exists \beta_i \neq 0$) *ou* ($\exists \beta_i < 0$) *alors* $h(\alpha, \beta_2, \ldots, \beta_R) = \infty$.
*Dans le cas général :*
*S'il existe* $\alpha' \in \{0, \ldots, \alpha - 1\}$ *tel que* $(P_{\alpha'+1} \cap P_\alpha) \setminus P_{\alpha+1} \neq \emptyset$ *alors* $h(\alpha, \beta_2, \ldots, \beta_R) =$

$$\min_{0 \leq \alpha' \leq \alpha-1,\ e \in (P_{\alpha'+1} \cap P_\alpha) \setminus P_{\alpha+1}} \{h(\alpha', \beta_2 - w_2(e), \ldots, \beta_R - w_R(e)) + w_1(e)\} \quad (4.5)$$

*sinon*

$$h(\alpha, \beta_2, \ldots, \beta_R) = \infty$$

Notons que $h$ est similaire à la fonction $g$ que nous avons définie dans la section 3.4.4 du chapitre précédent et qui nous permettait de résoudre MinMultiCCard dans les chaînes. Ainsi, par une preuve similaire (et plus simple, car n'utilisant pas la fonction auxiliaire $c$) à celle du théorème 3.9 (page 59), nous obtenons le théorème suivant :

**Théorème 4.11.** *Soit* $\alpha \in \{1, \ldots, k\}$, $\beta_i \in \{0, \ldots, B_i\}$ *pour* $2 \leq i \leq R$.
$h(\alpha, \beta_2, \ldots, \beta_R)$ *est égal à la valeur minimum, relativement au premier critère, d'une multicoupe* $C$ *séparant exactement les* $\alpha$ *premières paires source-puits telle que* $w_i(C) = \beta_i$ *pour tout* $i \in \{2, \ldots, R\}$. *Si une telle multicoupe n'existe pas alors* $h(\alpha, \beta_2, \ldots, \beta_R) = \infty$.

D'où le corollaire suivant :

**Corollaire 4.2.** *L'instance admet une solution si et seulement s'il existe* $\beta_2, \ldots, \beta_R$ *tels que* $h(k, \beta_2, \ldots, \beta_R) \leq B_1$.

Dans la pratique, nous calculons toutes les valeurs de $h$ nécessaires, soit $O(kB_2 \ldots B_R)$ valeurs. Pour $\alpha, \beta_2, \ldots, \beta_R$ donnés, nous calculons $h(\alpha, \beta_2, \ldots, \beta_R)$ en $O(kn)$ puisque nous recherchons le minimum parmi $O(kn)$ valeurs (voir l'équation (4.5)).

**Nous obtenons finalement une complexité globale en $O(mk^2 B_2 \ldots B_R)$.**

Nous pouvons remarquer que pour $R = 2$ et $B_2 = p$, la complexité de cet algorithme est moins intéressante que celle obtenue dans la section 3.4.4 pour résoudre MinMultiCCard dans les chaînes. En effet, l'utilisation de la fonction intermédiaire $c$ nous avait permis d'améliorer notablement la complexité globale de l'algorithme.

D'autre part, nous remarquons que $B_1$ n'intervient pas dans la complexité. Il est donc plus efficace de choisir $B_1$ de manière à ce que $B_1 \geq B_i$ pour tout $i$ de $\{2, \ldots, R\}$. Enfin, si l'on considère le cas des instances où $R$ est fixé et où $B_i = O(m^\gamma)$ pour une valeur fixée de $\gamma$ ($i \in \{2, \ldots, R\}$), le problème est alors polynomial.

D'où finalement :

**Théorème 4.12.** *$R -$ CriMultiC peut être résolu par un algorithme pseudo-polynomial dans les chaînes lorsque $R$ est fixé.*

De ce fait, à moins que $\mathcal{P} = \mathcal{NP}$, $R -$ CriMultiC n'est pas $\mathcal{NP}$-complet *au sens fort* dans les chaînes pour une valeur de $R$ donnée.

### 4.3.2 Cas des cycles, des chemins et des circuits

Concluons ce chapitre avec plusieurs résultats de $\mathcal{NP}$-complétude dans des cas où MinMultiCCard et $R -$ CriC sont pourtant polynomiaux.

**Théorème 4.13.** *Dans les cycles, $2 -$ CriMultiC et $R -$ CriMultiC sont respectivement $\mathcal{NP}$-complet et $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* Il suffit de remarquer que pour ces deux problèmes, nous pouvons obtenir une instance dans un cycle à partir d'une instance dans une chaîne en ajoutant une paire source-puits, la source étant placée à l'une des extrémités de la chaîne et le puits à l'autre extrémité, puis en ajoutant une arête de poids quelconque reliant les deux extrémités. □

Les résultats sur les chaînes et les cycles se généralisent aisément aux cas des chemins et des cycles d'où :

**Théorème 4.14.** *Dans les chemins et les circuits, $2 -$ CriMultiC et $R -$ CriMultiC sont respectivement $\mathcal{NP}$-complet et $\mathcal{NP}$-complet au sens fort.*

*Démonstration.* C'est une conséquence immédiate des théorèmes 4.9, 4.10 et 4.13. □

# Chapitre 5

# Conclusion

Cette première partie du mémoire traite de problèmes de coupes et de multicoupes à plusieurs critères ou soumises à une contrainte de cardinalité.

Si le problème bien connu de la coupe minimum a été largement étudié ces cinquante dernières années, notamment depuis la publication de l'algorithme de Ford et Fulkerson permettant de le résoudre ([25]), plusieurs travaux récents se sont davantage focalisés sur des versions multicritères de ce problème ([2, 4, 10, 50]). Ainsi, dans [10], Bruglieri et al. ont étudié l'influence sur la complexité du problème de l'ajout d'une contrainte de cardinalité et ont laissé ouverte la question suivante : "Quelle est la complexité du problème consistant à déterminer une coupe minimum de cardinalité inférieure à une valeur donnée ?".

Cette question a été le point de départ de nos travaux et nous y avons répondu dans le chapitre 2 en démontrant que ce problème est $\mathcal{NP}$-difficile au sens fort dans les graphes bipartis de degré maximum 3.

Puisque ce problème peut en fait être vu comme un cas particulier de coupe bicritère, il nous a semblé naturel de poursuive ces travaux par l'étude du problème de la coupe multicritère. Plusieurs résultats existaient au préalable, dont notamment la $\mathcal{NP}$-complétude au sens fort de la version bicritère de ce problème dans les graphes généraux démontrée par Papadimitriou et Yannakakis dans [50].

Nous avons apporté de nouveaux résultats en démontrant que cette $\mathcal{NP}$-complétude se conserve pour des graphes particuliers. Ainsi, dans le chapitre 4, nous avons prouvé que $2 - \textsc{CriC}$ est $\mathcal{NP}$-complet dans les graphes bipartis de type $K_{2,d}$ et que, pour un nombre de critères non fixé, le problème est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis planaires de degré maximum 3.

Puis, nous avons généralisé les problèmes précédemment étudiés aux cas des multicoupes : au lieu de séparer uniquement une source donnée d'un puits donné, il faut désormais séparer un ensemble de sommets sources de leurs puits respectifs. Le chapitre 3 a ainsi été consacré à l'étude de MinMultiCCard qui correspond au problème de la multicoupe minimum soumise à une contrainte de cardinalité. Puisque MinMultiCCard est une généralisation de MinCCard et puisque MinMultiC peut s'y ramener, les cas $\mathcal{NP}$-difficiles pour ces deux

derniers problèmes le sont également pour MinMultiCCard. Notre étude s'est alors articulée autour de la question suivante : "Les cas polynomiaux de MinMultiC et de MinCCard - notamment les arbres orientés - le restent-ils pour MinMultiCCard ?".

Nous avons alors démontré la $\mathcal{NP}$-complétude au sens fort de MinMultiCCard dans les étoiles orientées. La démonstration de ce résultat - qui constitue la contribution principale du chapitre 3 - a nécessité d'aborder d'autres problèmes d'optimisation combinatoire fréquemment utilisés mais sans lien apparent avec les problèmes de coupes et de multicoupes. Nous avons par exemple établi que rechercher une couverture des arêtes de poids minimum et de cardinalité inférieure à une valeur donnée dans un graphe biparti pondéré est $\mathcal{NP}$-difficile au sens fort. A titre d'indication, un tel problème devient polynomial lorsque nous supprimons la contrainte de cardinalité ([42]).

En raison de sa $\mathcal{NP}$-difficulté dans les étoiles, orientées ou non, nous avons voulu étudier la complexité de MinMultiCCard dans des graphes plus "simples" : les chaînes. En utilisant certains arguments polyédraux, nous avons tout d'abord montré que ce problème est polynomial. Puis nous avons proposé deux approches différentes pour résoudre ce problème. La première repose sur l'utilisation d'une relaxation lagrangienne permettant d'obtenir la valeur optimale (mais pas une solution optimale). La seconde utilise des techniques de programmation dynamique et permet de déterminer une solution optimale en $O(nk + pk^2)$. Par la suite, nous avons généralisé cette seconde approche aux cas des cycles, des chemins et des circuits pour lesquels MinMultiCCard est également polynomial.

Ce chapitre 3 a ainsi permis de donner une vue d'ensemble de la complexité de MinMultiCCard, problème qui n'avait été que très peu étudié jusqu'à présent. Le tableau 5 - qui indique la complexité de l'ensemble des problèmes de coupes évoqués dans ce mémoire pour différentes topologies - permet d'apprécier les différences notables de complexité entre les divers problèmes étudiés.

A l'instar de $R-$CriC et de MinCCard, nous avons finalement étudié $R-$CriMultiC, généralisation du problème bicritère particulier MinMultiCCard. Nous avons ainsi montré que dans le cas des chaînes, $2-$CriMultiC est $\mathcal{NP}$-complet et $R-$CriMultiC est $\mathcal{NP}$-complet au sens fort. Enfin, nous avons élaboré un algorithme pseudo-polynomial pour résoudre $R-$CriMultiC lorsque le nombre de critères est fixé et que le graphe considéré est une chaîne.

Cette première partie du mémoire remplit donc en grande partie les objectifs que nous nous étions fixés : elle apporte un certain nombre de résultats de complexité significatifs concernant des problèmes de coupes et de multicoupes à plusieurs critères ou soumises à une contrainte de cardinalité. Une partie de ses résultats a ainsi fait l'objet d'une publication en revue ([5]).

Cependant, certains points mériteraient d'être davantage approfondis. On pense notamment à la complexité de MinCCard dans le cas des graphes complets, des graphes sans circuits et des graphes planaires. On pense également à

| **Complexité des problèmes** | Graphes bipartis de degré max 3 | Etoiles non orientées | Etoiles orientées | Arborescences | Chaîne-chemins cycle-circuits |
|---|---|---|---|---|---|
| MinC | Polynomial [1] | Polynomial [1] | Polynomial [1] | Polynomial [1] | Polynomial [1] |
| MinCCard | $\mathcal{NP}$-difficile au sens fort | Polynomial | Polynomial | Polynomial | Polynomial |
| $2-$CriC | $\mathcal{NP}$-complet au sens fort | Polynomial | Polynomial | Polynomial | Polynomial |
| $R-$CriC | $\mathcal{NP}$-complet au sens fort | Polynomial | Polynomial | Polynomial | Polynomial |
| MinMultiC | $\mathcal{NP}$-difficile au sens fort | $\mathcal{NP}$-difficile au sens fort [30] | Polynomial [19] | Polynomial [19] | Polynomial [19] |
| MinMultiCCard | $\mathcal{NP}$-difficile au sens fort | $\mathcal{NP}$-difficile au sens fort [30] | $\mathcal{NP}$-difficile au sens fort | Ouvert | Polynomial |
| $2-$CriMultiC | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort [30] | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet |
| $R-$CriMultiC | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort [30] | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort |

Table 5.1 – Tableau récapitulatif de la complexité des différents problèmes de coupes et de multicoupes étudiés. Une référence est fournie pour tous les résultats antérieurs à nos travaux.

des versions min-max, de minimisation du regret maximum ou à des versions où le poids de chaque arête (ou arc) est dans un intervalle donné. Une autre perspective de recherche, très prometteuse à ce jour, consisterait à se diversifier davantage en étudiant l'ajout d'une contrainte de cardinalité à d'autres problèmes classiques de théorie des graphes. On a notamment pu constater certains liens étroits unissant des problèmes de coupes, de couvertures des arêtes et de stables.

Enfin, déterminer la complexité de MinMultiCCard pour le cas des arborescences reste également un défi très intéressant (MinMultiCCard étant polynomial dans les chemins et $\mathcal{NP}$-difficile dans les étoiles orientées).

Deuxième partie

# Recherche de sous-graphes induits

# Chapitre 6

# Quelques résultats généraux pour la recherche de chaînes, d'arbres et de cycles induits

Le problème de l'arbre de Steiner consiste à déterminer un arbre de taille minimum couvrant un ensemble $T = \{x_1, \ldots, x_k\}$ de $k$ sommets terminaux de $G = (V, E)$. Ce problème bien connu est l'un des 21 problèmes dont Karp a montré la $\mathcal{NP}$-complétude en 1972 ([40]). Il reste $\mathcal{NP}$-difficile dans certaines classes de graphes comme par exemple les graphes bipartis ([7]) mais devient polynomial lorsque le nombre de terminaux est fixé ([24]).

Dans ce chapitre, on s'intéresse à l'influence sur la complexité de ce problème de l'ajout d'une contrainte d'induction. Cela revient à rechercher un ensemble $A$ de sommets du graphe tel que :

- $A$ est de cardinalité minimum
- $T \subset A$
- le graphe induit par $A$ est un arbre

Contrairement au problème de l'arbre de Steiner qui admet toujours une solution lorsque le graphe considéré est connexe, la figure 6.1 montre que ce n'est pas toujours le cas pour $k-$MinArbreI et $k-$ArbreI : tout arbre couvrant $x_1, x_2, x_3$ utilise au moins deux arêtes du triangle central et ne peut donc être induit.

Ce chapitre est donc consacré aux problèmes de recherche de sous-graphes induits particuliers - tels que les chaînes, les arbres et les cycles - couvrant un ensemble donné de sommets terminaux de $G$.

Rappelons deux des principaux résultats pour ce type de problèmes. Le premier est un résultat de polynomialité pour $3 -$ ArbreI obtenu par Chudnovsky et Seymour dans [16], où ils ont élaboré un algorithme en $O(n^4)$ pour résoudre ce problème. Le second résultat, plus ancien, est l'oeuvre de Bienstock qui a montré dans un article de Discrete Mathematics ([8]) que $2 -$ CycleI est $\mathcal{NP}$-complet. Il est à noter que cet article, initialement faux, a été complété par

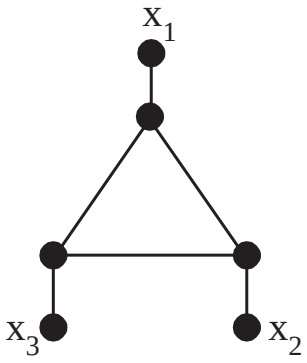


FIGURE 6.1 – Une instance de $3-$ ARBREI n'admettant pas de solution

un Corrigendum de Reed dans la même revue.

Dans la suite de ce chapitre, nous allons tout d'abord montrer la $\mathcal{NP}$-complétude des cas généraux de $k-$ ARBREI, $k-$ CHAINEI et $k-$ CYCLEI ainsi que de leur version pondérée. Puis, nous allons établir plusieurs résultats de polynomialité et de $\mathcal{NP}$-complétude pour certaines classes de graphes lorsque le nombre de sommets terminaux est fixé.

## 6.1 Préliminaires

Remarquons que si l'ensemble $T$ induit un graphe contenant au moins un cycle, alors $k-$ CHAINEI et $k-$ ARBREI (ainsi que leurs versions optimisations et pondérées) n'admettent aucune solution. Nous supposerons dans tout ce chapitre que le graphe induit par $T$ est acyclique.

Intéressons-nous maintenant au cas où $G$ possède une arête $x_i - x_j$ avec $x_i$ et $x_j$ dans $T$ et considérons le graphe $G' = (V', E')$ construit à partir de $G$ de la manière suivante : nous supprimons $x_i$ et $x_j$ ainsi que tous leurs voisins communs puis nous ajoutons un nouveau sommet $x$ et une arête $x - v$ pour tout sommet $v$ voisin de $x_i$ ou (exclusif) de $x_j$ et différent de $x_i$ et $x_j$. Enfin, nous mettons à jour l'ensemble des sommets terminaux en posant $T' = T \cup \{x\} \setminus \{x_i, x_j\}$. Nous avons alors :

**Propriété 6.1.** *$k-$ ARBREI (resp. $k-$ CHAINEI et $k-$ CYCLEI) admet une solution dans $G$ si et seulement si $(k-1)-$ ARBREI (resp. $(k-1)-$ CHAINEI et $(k-1)-$ CYCLEI) admet une solution dans $G'$.*

*Démonstration.* Si $A$ est un arbre induit couvrant $T$ dans $G$ alors $x_i - x_j$ est nécessairement une arête de $A$. De plus, $A$ ne peut contenir un sommet voisin à la fois de $x_i$ et de $x_j$ car cela induirait un triangle. D'où, en remplaçant $x_i$ et $x_j$ par le sommet $x$, nous obtenons un arbre induit $A'$ couvrant $T'$ dans $G'$.

Réciproquement, si $A'$ est un arbre induit qui couvre $T'$ dans $G'$, en remplaçant $x$ par $x_i - x_j$, nous obtenons un arbre induit $A$ couvrant $T$ dans $G$. □

Cette transformation nous permet ainsi de réduire la taille de certaines instances en diminuant le nombre de sommets terminaux et la taille du graphe.

Enfin, nous donnons un premier résultat que nous utiliserons fréquemment au cours de ce chapitre :

**Propriété 6.2.** *Une plus courte chaîne entre deux sommets d'un graphe est nécessairement induite.*

*Démonstration.* Rappelons tout d'abord que les graphes considérés sont simples. Ainsi, il existe au plus une arête entre toute paire de sommets. Si une telle chaîne n'est pas induite, il existe une arête reliant deux sommets non voisins de la chaîne et cette arête permettrait d'obtenir une chaîne de longueur strictement inférieure. □

## 6.2 Complexité des problèmes généraux

### 6.2.1 Etude des problèmes pondérés

Dans cette section, nous nous intéressons à $k$ − PondChaineI, $k$ − PondArbreI et $k$ − PondCycleI, versions pondérées de $k$ − ChaineI, $k$ − ArbreI et $k$ − CycleI. Afin de montrer la $\mathcal{NP}$-complétude de ces trois problèmes, introduisons le problème de décision $\mathcal{NP}$-complet suivant ([32]) :

Monotone 3-SAT
**Données** : Un ensemble $U$ de variables booléennes, une collection $C$ de clauses sur $U$ telle que pour tout $c \in C$, $|c| = 3$ et $c$ contient uniquement des variables complémentées ou uniquement des variables non complémentées.
**Question** : Existe-t-il une affectation de valeurs aux variables de $U$ permettant de satisfaire toutes les clauses de $C$ ?

Nous montrons alors le théorème suivant :

**Théorème 6.1.** $2-$PondChaineI *est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis même si les poids des arêtes sont* 0 *ou* 1.

*Démonstration.* Nous réalisons une réduction à partir de Monotone 3-SAT. Nous dirons qu'une clause est positive (resp. négative) si elle ne contient que des variables non complémentées (resp. complémentées).

Soit $I$ une instance de Monotone 3-SAT composée de $U = \{u_1, \ldots, u_n\}$ et de $C = \{c_1, \ldots, c_m\}$ où $c_1, \ldots, c_{m'}$ sont les clauses positives et $c_{m'+1}, \ldots, c_m$ les clauses négatives. Nous construisons une instance $I'$ pour $2 -$ ChaineI de la manière suivante. Chaque clause $c_j$ est représentée par le gadget $\mathcal{C}_j$ décrit sur la figure 6.2 et composé d'arêtes de poids nul. Les sommets $v_{1,j}, v_{2,j}$ et $v_{3,j}$ représentent chacun une variable de la clause $c_j$ et les sommets $w_j$ et $w'_j$

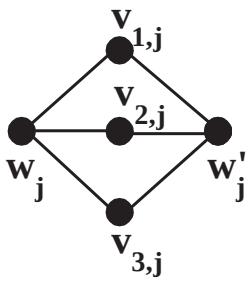


FIGURE 6.2 – Le gadget $\mathcal{C}_j$ représentant la clause $c_j$

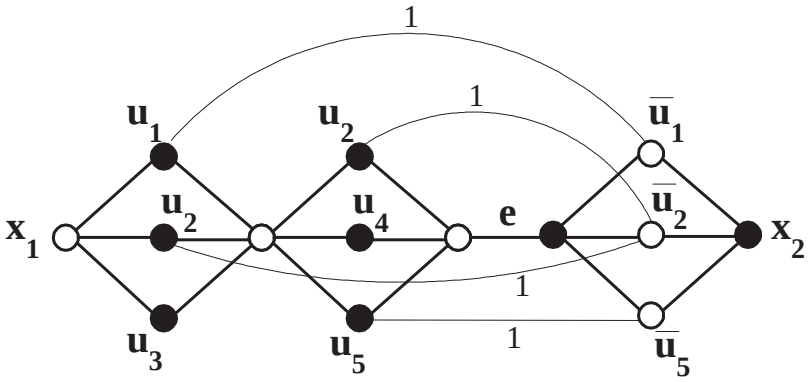


FIGURE 6.3 – Le graphe obtenu pour $(u_1 \vee u_2 \vee u_3) \wedge (u_2 \vee u_4 \vee u_5) \wedge (\bar{u}_1 \vee \bar{u}_2 \vee \bar{u}_5)$ (les sommets noirs et blancs indiquent la bipartition)

sont utilisés pour relier les gadgets entre eux. Pour $j \in \{1, \ldots, m' - 1\}$ et pour $j \in \{m' + 1, \ldots, m - 1\}$, nous fusionnons les sommets $w'_j$ et $w_{j+1}$. Puis, nous connectons $w'_{m'}$ et $w_{m'+1}$ par une arête $e$ de poids nul. Enfin, nous ajoutons une arête de poids 1 entre toute paire de sommets lorsque l'un correspond à $u_i$ et l'autre à $\bar{u}_i$. Le graphe construit est donc un graphe biparti dont les arêtes sont pondérées par 0 ou 1 (voir la figure 6.3). En effet, nous avons d'un côté les sommets $v_{1,j}, v_{2,j}$ et $v_{3,j}$ des clauses positives et les sommets $w_j$ et $w'_j$ des clauses négatives et de l'autre côté de la bipartition, les sommets $w_j$ et $w'_j$ des clauses positives et les sommets $v_{1,j}, v_{2,j}$ et $v_{3,j}$ des clauses négatives. Posons $T = \{x_1, x_2\}$ où $x_1$ et $x_2$ sont respectivement $w_1$ et $w'_m$.

Montrons que $I$ admet une solution si et seulement s'il existe une chaîne induite couvrant $x_1$ et $x_2$ de poids inférieur ou égal à 0.

Soit $S$ une solution de $I$. Nous construisons une chaîne induite de $x_1$ à $x_2$ de la manière suivante : nous traversons chacun des gadgets $\mathcal{C}_j$ en passant par un de ses 3 sommets correspondant à une variable égale à VRAI dans la solution $S$.

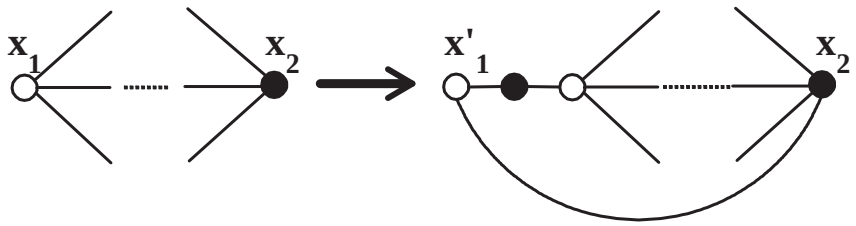


FIGURE 6.4 – La transformation permettant de prouver la $\mathcal{NP}$-complétude de $1-$PONDCYCLEI

Ainsi, la chaîne construite est de poids nul et est nécessairement induite puisque $u_i$ et $\bar{u}_i$ ne peuvent être égaux à VRAI simultanément.

Réciproquement, soit $P$ une chaîne induite de poids nul couvrant $x_1$ et $x_2$. $P$ ne peut utiliser des arêtes de poids 1 et puisqu'elle est induite, elle ne peut passer à la fois par un sommet représentant $u_i$ et par un sommet représentant $\bar{u}_i$. Nous obtenons alors une solution pour $I$ en affectant la valeur VRAI aux variables correspondant aux sommets par lesquels passe $P$. □

En remarquant que dans cette dernière preuve, tout arbre induit de poids nul couvrant $x_1$ et $x_2$ est nécessairement une chaîne induite, nous obtenons :

**Corollaire 6.1.** $2-$PONDARBREI *est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis même si les poids des arêtes sont* 0 *ou* 1.

De plus, si nous remplaçons $x_1$ par une chaîne de longueur 2 dont nous notons $x'_1$ le sommet de degré 1 puis que nous ajoutons une arête connectant $x'_1$ à $x_2$, toute chaîne induite couvrant $x_1$ et $x_2$ correspond à un cycle induit couvrant $x'_1$ (voir la figure 6.4). Réciproquement, tout cycle induit couvrant $x'_1$ correspond à une chaîne induite couvrant $x_1$ et $x_2$ si nous supprimons les arêtes et les sommets ajoutés, d'où :

**Corollaire 6.2.** $1-$PONDCYCLEI *est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis même si les poids des arêtes sont* 0 *ou* 1.

Nous allons maintenant montrer que nous pouvons restreindre la preuve du théorème 6.1 au cas des graphes bipartis cubiques (voir la définition 1.3 page 20). Pour cela, nous allons tout d'abord effectuer deux transformations pour supprimer les sommets de degré supérieur ou égal à 4.

La première est une substitution concernant les sommets $w_j$ et $w'_j$ de degré 6 et est décrite sur la figure 6.5. De plus, cette transformation s'adapte de manière immédiate au cas des $w_j$ et $w'_j$ de degré 4 en ne conservant qu'une "moitié" de la transformation. Enfin, il est à noter que toutes les nouvelles arêtes sont de poids nul.

La seconde transformation est également une substitution mais concerne les sommets $v_{1,j}, v_{2,j}$ et $v_{3,j}$. Soit $u_i$ la variable représentée par $v_{k,j}$ ($k = 1, 2$ ou 3).

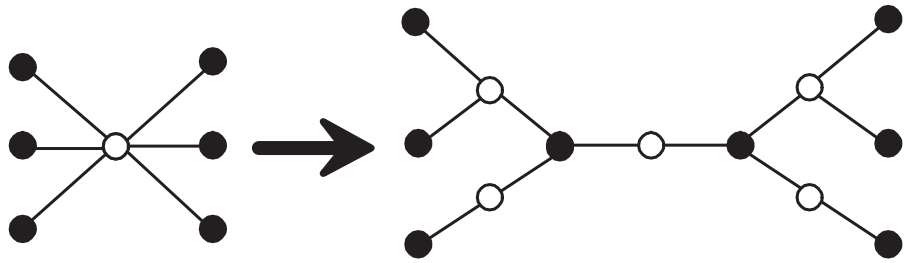

FIGURE 6.5 – Transformation des sommets de degré 6

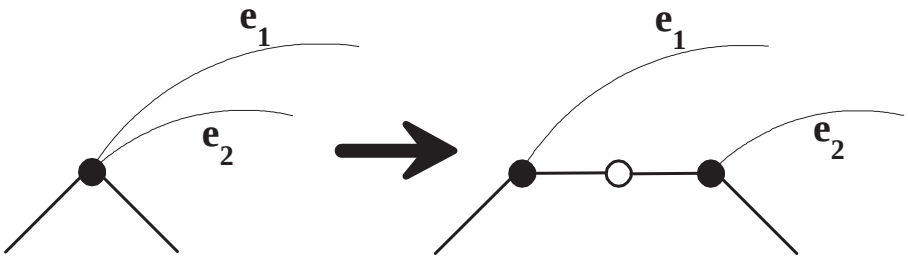


FIGURE 6.6 – Remplacement du "sommet variable" $u_i$ par une chaîne avec $n(\bar{u}_i) = 2$

Par construction, $v_{k,j}$ est relié à tous les sommets correspondant à $\bar{u}_i$. Notons alors $n(\bar{u}_i)$ le nombre de fois où $\bar{u}_i$ apparaît dans l'ensemble des clauses et $e_1, \ldots, e_{n(\bar{u}_i)}$ les arêtes reliant $v_{k,j}$ aux sommets représentant $\bar{u}_i$. Nous remplaçons $v_{k,j}$ par une chaîne $\mathcal{C}$ comportant $2n(\bar{u}_i) - 1$ sommets et composée d'arêtes de poids nul. De plus, pour tout $l \in \{1, \ldots, n(\bar{u}_i)\}$, l'une des extrémités de $e_l$ n'est plus $v_{k,j}$ mais le $(2i-1)$-ème sommet de $\mathcal{C}$ (voir la figure 6.6). Remarquons que si $n(\bar{u}_i) = 1$, cette transformation ne modifie pas le graphe.

Après avoir appliqué ces deux transformations, nous obtenons une instance équivalente composée d'un graphe $G' = (V_1', V_2', E')$ biparti et de degré maximum égal à 3. En effet, les deux transformations ne modifient pas la structure des solutions existantes et ont pour seule conséquence de diminuer les degrés de certains sommets du graphe. Cependant, $G'$ n'est pas encore cubique puisqu'il existe des sommets de degré 2. Soit $p_1$ et $p_2$ le nombre de sommets de degré 2 de $V_1'$ et $V_2'$ et soit $q_1$ et $q_2$ le nombre de sommets de degré 3 de $V_1'$ et $V_2'$. Nous supposons par la suite que $p_1 \geq p_2$.
Nécessairement, $\sum_{v \in V_1'} d^\circ(v) = \sum_{v \in V_2'} d^\circ(v)$ d'où $2p_1 + 3q_1 = 2p_2 + 3q_2$ soit encore $2(p_1 - p_2) = 3(q_2 - q_1)$. Puisque 2 et 3 sont premiers entre eux alors, d'après le théorème de Gauss, nous avons :

$$p_1 - p_2 \equiv 0 \ [3]$$

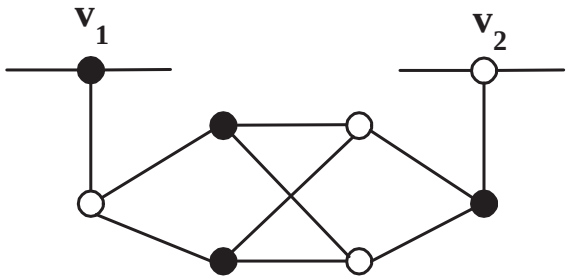


FIGURE 6.7 – Augmentation du degré d'un sommet de $V_1'$ et d'un sommet de $V_2'$ (le poids des arêtes du gadget est égal à 1)

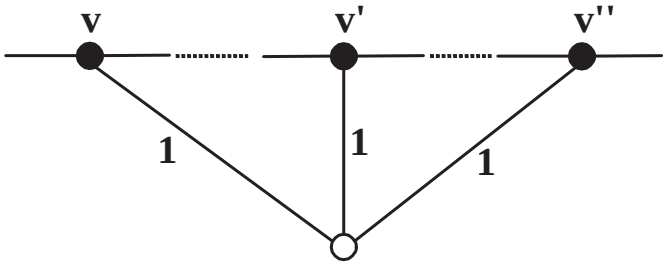


FIGURE 6.8 – Augmentation du degré de trois sommets $v$, $v'$ et $v''$ de $V_1'$

Soit $v_1$ un sommet de degré 2 de $V_1'$ et $v_2$ un sommet de degré 2 de $V_2'$. Pour augmenter de 1 le degré de ces deux sommets, nous utilisons le gadget de la figure 6.7. Le poids de toutes les arêtes du gadget est égal à 1. Nous réitérons cette transformation autant de fois que possible, c'est-à-dire jusqu'à ce qu'il n'y ait plus aucun sommet de degré 2 dans $V_2'$. Il reste alors $p_1 - p_2$ sommets de degré 2 dans $V_1'$. Puisque $p_1 - p_2$ est multiple de 3, nous pouvons utiliser le gadget de la figure 6.8 pour qu'il n'y ait plus aucun sommet de degré 2. Ce gadget consiste à relier trois par trois les sommets restants à un nouveau sommet en utilisant des arêtes de poids 1.

L'instance construite est équivalente à l'instance de départ puisque les deux gadgets sont composés d'arêtes de poids 1 et aucun cycle ou chaîne induite de poids nul ne peut donc les emprunter.

Nous obtenons alors le théorème désiré :

**Théorème 6.2.** $2-$PONDCHAINEI, $2-$PONDARBREI *et* $1-$PONDCYCLEI *sont* $\mathcal{NP}$*-complets au sens fort dans les graphes bipartis cubiques même si les poids des arêtes sont* 0 *ou* 1.

Précisons que pour $1-$PONDCYCLEI, il faut d'abord appliquer la transformation de la figure 6.4 puis utiliser les deux transformations et les deux gadgets

rendant le graphe cubique.

Si nous considérons les problèmes d'optimisation associés à $2-$PondChaineI, $2-$PondArbreI et $1-$PondCycleI, nous avons :

**Corollaire 6.3.** *Si $\mathcal{P} \neq \mathcal{NP}$, il ne peut exister d'algorithme polynomial approché à garantie de performance relative pour les problèmes d'optimisation associés à* $2-$PondChaineI, $2-$PondArbreI *et* $1-$PondCycleI *même dans les graphes bipartis cubiques où tous les poids sont égaux à 0 ou 1.*

*Démonstration.* Ce corollaire est une conséquence directe de la réduction du théorème 6.1 (page 91) car si un tel algorithme existait, nous aurions alors un algorithme polynomial pour Monotone 3-SAT. En effet, dans cette réduction nous cherchons une solution de poids inférieur ou égal à 0 donc tout algorithme à garantie de performance relative fournirait une solution pour l'instance de Monotone 3-SAT. □

Enfin, remarquons que tous les résultats de $\mathcal{NP}$-difficulté obtenus ici se généralisent pour $k-$PondChaineI, $k-$PondArbreI et $k-$PondCycleI pour toute valeur $k$ fixée ou non ($k \geq 2$) : pour $k-$PondChaineI et $k-$PondArbreI il suffit d'ajouter de nouveaux sommets terminaux connectés uniquement à $x_1$ tandis que pour $k-$PondCycleI on peut remplacer l'arête $x_1 - x_2$ par une chaîne contenant autant de terminaux que l'on souhaite.

### 6.2.2 Etude des problèmes non pondérés

Puisque les versions pondérées des problèmes considérés sont $\mathcal{NP}$-complets au sens fort pour deux terminaux et ce, même dans des graphes très particuliers, nous allons désormais nous intéresser au cas des graphes non pondérés.

Tout d'abord, afin d'établir la complexité de $k-$ChaineI, $k-$ArbreI et $k-$CycleI, nous introduisons le problème $\mathcal{NP}$-complet suivant ([28]) :

Chaîne Hamiltonienne
**Données** : un graphe $H = (S, A)$ non orienté et connexe
**Question** : Existe-t-il une chaîne passant une et une seule fois par chaque sommet de $H$ ?

Akiyama et al. ont montré dans [3] que Chaîne Hamiltonienne est $\mathcal{NP}$-complet dans les graphes bipartis cubiques planaires. Nous proposons le théorème suivant :

**Théorème 6.3.** *$k-$ChaineI est $\mathcal{NP}$-complet dans les graphes bipartis cubiques planaires.*

*Démonstration.* Nous réalisons une réduction à partir de Chaîne Hamiltonienne dans les graphes bipartis cubiques planaires. Soit $I$ une instance de ce problème composée d'un graphe $H = (S, A)$ biparti cubique planaire. Notons $S = \{s_1, \ldots, s_n\}$ et $m = |A|$.

Soit $I'$ une instance de $n-$ChaineI construite de la manière suivante. Chaque sommet $s_i$ de $H$ est remplacé par le gadget $\mathcal{S}_i$ représenté sur la figure 6.9. $x_i$

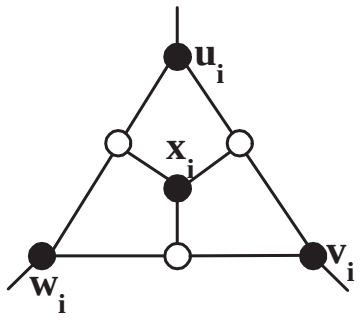


FIGURE 6.9 – Le gadget $\mathcal{S}_i$ remplaçant chaque sommet $s_i$ de $H$

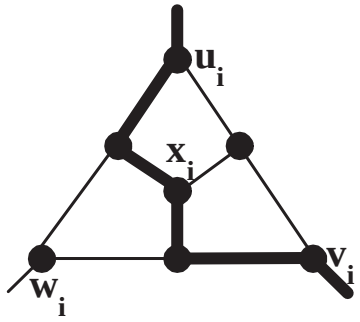


FIGURE 6.10 – La chaîne induite traversant $\mathcal{S}_i$ et passant par $x_i$

est qualifié de sommet *central* et $u_i, v_i, w_i$ de sommets *extérieurs*. Si $s_i - s_j$ est une arête de $H$ alors nous relions l'un des sommets extérieurs de $\mathcal{S}_i$ à l'un des sommets extérieurs de $\mathcal{S}_j$ de manière à obtenir un graphe cubique planaire. Le graphe $G$ ainsi construit est biparti, cubique et planaire. Finalement, l'ensemble $T$ des sommets terminaux est composé de tous les sommets centraux.

Montrons qu'il existe dans $G$ une chaîne induite couvrant tous les sommets centraux si et seulement si $H$ admet une chaîne hamiltonienne.

Soit $P$ une chaîne hamiltonienne de $H$. Soit $s_j$ et $s_k$ respectivement le prédécesseur et le successeur de $s_i$ dans $P$. Nous supposons que $\mathcal{S}_j$ et $\mathcal{S}_i$ sont connectés par une arête incidente à $u_i$ et que $\mathcal{S}_i$ et $\mathcal{S}_k$ sont connectés par une arête incidente à $v_i$. Nous construisons alors une chaîne induite traversant $s_i$ comme indiqué sur la figure 6.10. Le sous-graphe induit par les sommets de cette chaîne est ainsi acyclique et couvre l'ensemble des sommets terminaux.

Réciproquement, soit $C$ une chaîne induite couvrant $T$. Nous supposons que les deux extrémités de $C$ sont des terminaux car dans le cas contraire, nous supprimons de tels sommets de $C$. La seule manière pour $C$ de traverser un gadget $\mathcal{S}_i$ de $u_i$ à $v_i$ en passant par $x_i$ est représenté sur la figure 6.10. Ainsi,

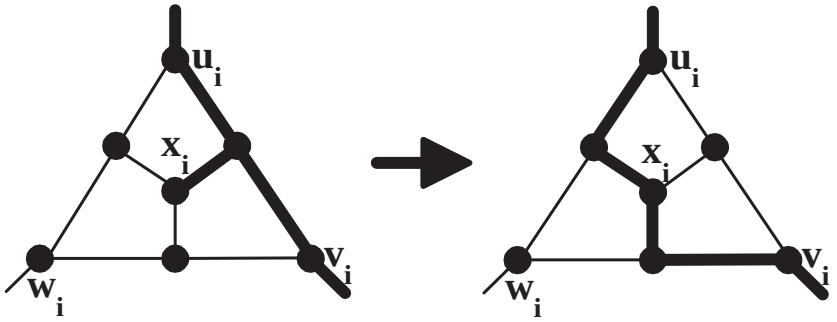


FIGURE 6.11 – Transformation d'un arbre induit en une chaîne induite

$C$ traverse successivement chaque gadget de $H$. Nous obtenons alors une chaîne hamiltonienne $P$ pour $H$ de la manière suivante : si $\mathcal{S}_i$ est le $k$-ème gadget traversé par $C$ alors $s_i$ est le $k$-ème sommet de $P$. □

Nous obtenons également :

**Théorème 6.4.** *$k-$ARBREI est $\mathcal{NP}$-complet dans les graphes bipartis cubiques planaires.*

*Démonstration.* La preuve est similaire à celle du théorème précédent. La seule différence est qu'un arbre induit couvrant le sommet central de chaque gadget n'est pas nécessairement une chaîne. Dans ce cas, nous modifions l'arbre induit obtenu afin d'obtenir une chaîne équivalente comme le montre la figure 6.11. □

Bienstock a montré dans [8] que $2-$CYCLEI (et donc $k-$CYCLEI) est $\mathcal{NP}$-complet dans les graphes généraux. Nous montrons que $k-$CYCLEI est également $\mathcal{NP}$-complet dans certains graphes très particuliers :

**Théorème 6.5.** *$k-$CYCLEI est $\mathcal{NP}$-complet dans les graphes bipartis cubiques planaires.*

*Démonstration.* La preuve est identique à celle du théorème 6.3 en utilisant une réduction à partir du problème du cycle hamiltonien qui est $\mathcal{NP}$-complet dans les graphes bipartis cubiques planaires ([3]). □

## 6.3 Algorithme et complexité des problèmes lorsque le nombre de sommets terminaux est fixé

Puisque $k-$CYCLEI, $k-$CHAINEI et $k-$ARBREI sont $\mathcal{NP}$-complets dans les graphes bipartis cubiques planaires, nous allons désormais étudier le cas où le nombre de sommets terminaux est fixé.

### 6.3.1 k=1 : un cas trivial pour tous les problèmes ?

De façon immédiate, $1-$ ChaineI, $1-$ ArbreI, $1-$ MinChaineI, $1-$ MinArbreI, $1-$ PondChaineI et $1-$ PondArbreI sont polynomiaux puisqu'il suffit de sélectionner l'unique sommet terminal.

Concernant la recherche d'un cycle induit, nous obtenons :

**Théorème 6.6.** $1-$ CycleI *et* $1-$ MinCycleI *sont polynomiaux.*

*Démonstration.* Soit $v_1, \ldots, v_{d^\circ(x_1)}$ les voisins du sommet terminal $x_1$. Pour $i \in \{1, \ldots, d^\circ(x_1)\}$, nous supprimons l'arête $x_1 - v_i$ et calculons ensuite une plus courte chaîne $P_i$ entre $x_1$ et $v_i$. Une plus courte chaîne entre deux sommets d'un graphe étant toujours induite (voir la propriété 6.2 page 91), pour chaque chaîne $P_i$ obtenue nous obtenons un cycle induit passant par $x_1$ en ajoutant l'arête $x_1 - v_i$. Si aucune chaîne $P_i$ n'existe (dans ce cas, $x_1$ est un sommet d'articulation), alors $1-$ CycleI n'admet pas de solution et, dans le cas contraire, le cycle induit de taille minimum obtenu est alors solution optimale de $1-$ MinCycleI. □

Enfin, rappelons que nous avons montré dans la section 6.2.1 que $1-$ PondCycleI est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis cubiques même si les poids sont 0 ou 1 (voir le théorème 6.2 page 95).

### 6.3.2 Complexité des problèmes pour 2 terminaux

Pour $2-$ChaineI et $2-$ArbreI, nous montrons les deux propriétés suivantes :

**Propriété 6.3.** $2-$ ChaineI *et* $2-$ ArbreI *admettent toujours une solution.*

*Démonstration.* C'est une conséquence immédiate de la propriété 6.2 (page 91). □

**Propriété 6.4.** $2-$ MinChaineI *et* $2-$ MinArbreI *admettent les mêmes solutions optimales et l'une d'elles peut être déterminée en* $O(m)$.

*Démonstration.* Un parcours en profondeur à partir de l'un des deux sommets terminaux permet de déterminer une chaîne de longueur minimum entre les deux terminaux. La complexité de cet algorithme est donc $O(m)$. □

Cependant, ce résultat de polynomialité ne se généralise pas pour la version pondérée de ces deux problèmes puisqu'une chaîne de poids minimum n'est pas nécessairement induite. Nous avons ainsi montré dans la section 6.2.1 que ces problèmes sont alors $\mathcal{NP}$-complets au sens fort.

A l'inverse, $2-$ CycleI est $\mathcal{NP}$-complet ([8]) et Lévêque et al. ont montré récemment dans [44] que ce problème reste $\mathcal{NP}$-complet même si tous les sommets sont de degrés 3 exceptés les deux terminaux qui sont de degré 2.

Nous allons montrer que $2-$ CycleI est également $\mathcal{NP}$-complet dans les graphes bipartis avec une preuve assez similaire à celle de Bienstock dans [8].

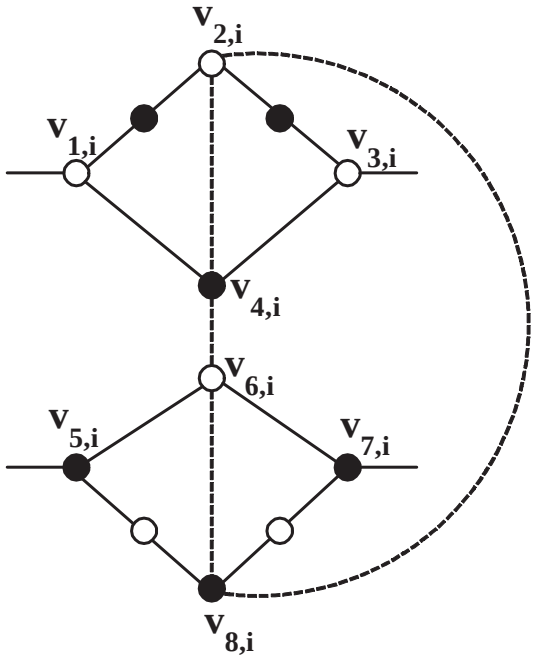


FIGURE 6.12 – Le gadget $\mathcal{U}_i$ obtenu pour la variable $u_i$ (les sommets noirs et blancs indiquent la bipartition)

**Théorème 6.7.** $2-$CYCLEI *est* $\mathcal{NP}$*-complet dans les graphes bipartis.*

*Démonstration.* Nous utilisons une réduction à partir de MONOTONE 3-SAT (voir la page 91).

Soit $I$ une instance de MONOTONE 3-SAT composée des variables $U = \{u_1, \ldots, u_n\}$ et des clauses $C = \{c_1, \ldots, c_m\}$ où $c_1, \ldots, c_{m'}$ sont les clauses positives et $c_{m'+1}, \ldots, c_m$ les clauses négatives. Nous construisons une instance $I'$ pour $2-$CYCLEI de la manière suivante. Chaque variable $u_i$ est représentée par le gadget $\mathcal{U}_i$ décrit sur la figure 6.12. Les arêtes en pointillé sont qualifiées d'arêtes *interdites*. Chaque clause $c_j$ est représentée par le gadget $\mathcal{C}_j$ décrit sur la figure 6.13. Les sommets $w_{3,j}, w_{4,j}, w_{5,j}$ représentent les trois littéraux de la clause $c_j$. Si $w_{k,j}$ ($k \in \{3,4,5\}$) correspond à $u_i$ (resp. $\bar{u}_i$) alors nous ajoutons deux arêtes - dites également *interdites* - reliant $w_{k,j}$ à $v_{4,i}$ et $v_{8,i}$ (resp. à $v_{2,i}$ et $v_{6,i}$). Enfin, nous connectons l'ensemble des gadgets créés en ajoutant plusieurs sommets et arêtes dont les deux terminaux $x_1$ et $x_2$, comme l'illustre la figure 6.14. Le graphe construit est donc biparti comme on peut le voir sur cette dernière figure.

Montrons que $I$ admet une solution si et seulement s'il existe un cycle induit passant par $x_1$ et $x_2$.

Soit $S$ une solution de $I$, nous construisons un cycle induit couvrant $x_1$ et $x_2$ de la manière suivante. Soit $S' = \{x_1, x_2\}$. Pour chaque gadget $\mathcal{U}_i$, si $u_i = VRAI$

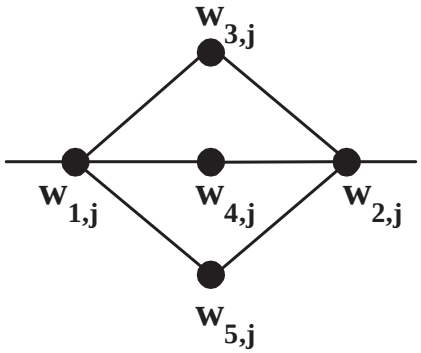


FIGURE 6.13 – Le gadget $\mathcal{C}_j$ obtenu pour la clause $c_j$

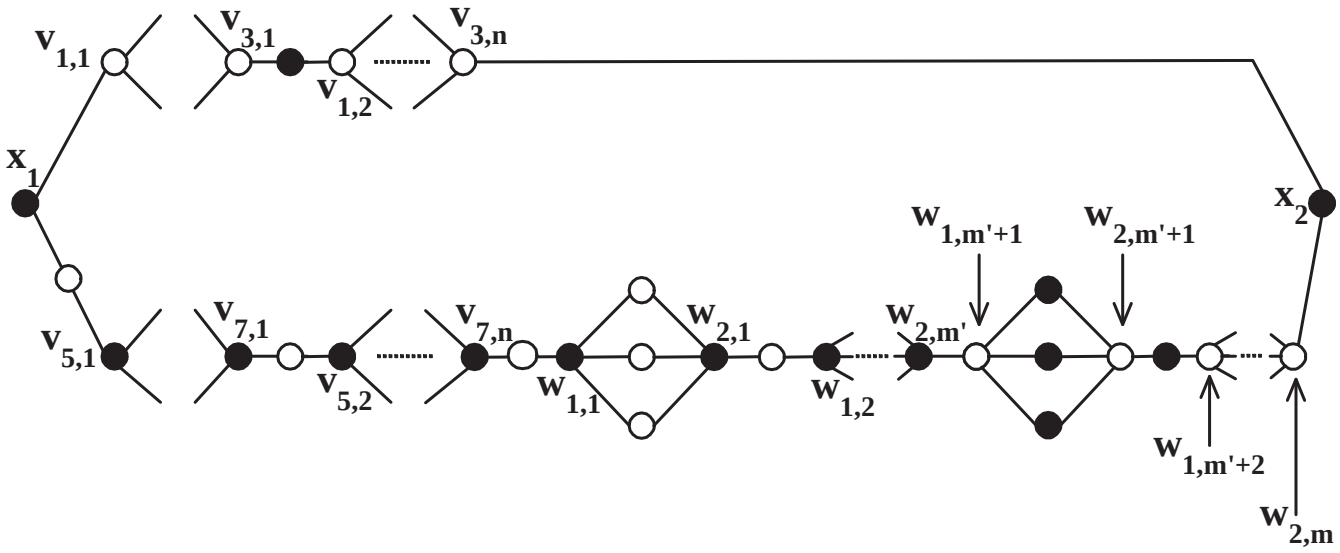


FIGURE 6.14 – L'allure générale du graphe obtenu

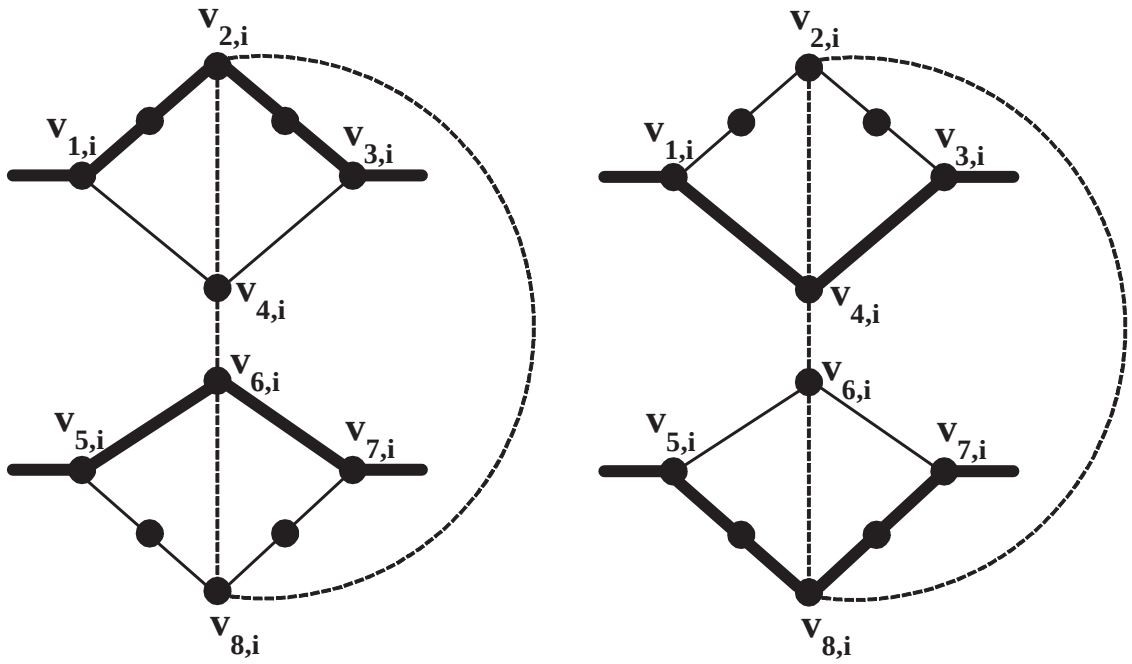


FIGURE 6.15 – Les deux manières possibles de traverser $\mathcal{U}_i$ (la partie gauche correspond à $u_i = VRAI$ et la partie droite à $u_i = FAUX$)

alors nous ajoutons à $S'$ les sommets de la chaîne reliant $v_{1,i}$ à $v_{3,i}$ et passant par $v_{2,i}$, ainsi que les sommets $v_{5,i}, v_{6,i}, v_{7,i}$. Si $u_i = FAUX$ alors nous ajoutons à $S'$ les sommets $v_{1,i}, v_{4,i}, v_{3,i}$ ainsi que les sommets de la chaîne reliant $v_{5,i}$ à $v_{7,i}$ et passant par $v_{8,i}$ (voir la figure 6.15).

Pour $j \in \{1, \ldots, m\}$, au moins un littéral de la clause $c_j$ est vrai et soit $w_{k,j}$ ($k = 3, 4$ ou 5) le sommet associé à l'un de ces littéraux. Nous ajoutons à $S'$ les sommets $w_{1,j}$, $w_{k,j}$ et $w_{2,j}$.

Enfin nous ajoutons à $S'$ tous les sommets n'appartenant à aucun gadget (comme par exemple ceux reliant les gadgets entre eux). Le graphe induit par $S'$ est alors nécessairement un cycle et couvre $x_1$ et $x_2$.

Réciproquement, soit $S'$ un cycle induit couvrant $x_1$ et $x_2$. Montrons qu'aucune arête *interdite* ne peut appartenir à $S'$. $v_{1,1}$ et $v_{5,1}$ sont nécessairement dans $S'$. En énumérant toutes les manières possibles de "traverser" le gadget $\mathcal{U}_1$, on s'aperçoit qu'il n'y a que deux alternatives (voir la figure 6.15) :

- soit $S'$ contient les sommets de la chaîne reliant $v_{1,1}$ à $v_{3,1}$ passant par $v_{2,1}$, ainsi que les sommets $v_{5,1}, v_{6,1}, v_{7,1}$
- soit $S'$ contient les sommets $v_{1,1}, v_{4,1}, v_{3,1}$ ainsi que les sommets de la chaîne reliant $v_{5,1}$ à $v_{7,1}$ passant par $v_{8,1}$

$v_{3,1}, v_{7,1}, v_{1,2}, v_{5,2}$ sont donc dans $S'$ et on réitère le même raisonnement pour les gadgets $\mathcal{U}_2, \ldots, \mathcal{U}_n$. Nous construisons alors une solution pour $I$ de la manière suivante : si $\mathcal{U}_i$ est traversé comme sur la partie gauche de la figure 6.15 alors

$u_i = VRAI$ et dans le cas contraire $u_i = FAUX$.

Il reste à vérifier que la solution construite satisfait bien chacune des clauses. Pour $j \in \{1, \ldots, m\}$, $S$ traverse $\mathcal{C}_j$ par l'un des trois sommets $w_{3,j}$,$w_{4,j}$ ou $w_{5,j}$. Ce sommet, supposons $w_{3,j}$, correspond à une variable $u_i$ (resp. $\bar{u}_i$). Or $\mathcal{U}_i$ est alors nécessairement traversé comme décrit sur la partie gauche (resp. droite) de la figure 6.15 à cause des arêtes $v_{4,i} - w_{3,j}$ et $v_{8,i} - w_{3,j}$ (resp. $v_{2,i} - w_{3,j}$ et $v_{6,i} - w_{3,j}$). Donc $u_i = VRAI$ (resp. $\bar{u}_i = VRAI$) et la clause $c_j$ est bien satisfaite. □

Comme nous l'avons fait en fin de section 6.2.1, nous allons modifier la preuve précédente pour obtenir un résultat de $\mathcal{NP}$-complétude dans les graphes bipartis de degré maximum 4. En effet, dans la réduction que nous venons d'utiliser, les seuls sommets pouvant être de degré strictement supérieur à 4 sont les sommets $v_{2,i}, v_{4,i}, v_{6,i}, v_{8,i}$. Pour $i \in \{1, \ldots, n\}$, soit $n(u_i)$ et $n(\bar{u}_i)$ le nombre d'occurrences de $u_i$ et de $\bar{u}_i$ dans les clauses de $C$. Pour $i \in \{1, \ldots, n\}$, nous remplaçons $v_{2,i}$ et $v_{6,i}$ par deux chaînes, $P_2$ et $P_6$, composées de $2n(\bar{u}_i) + 1$ sommets, et $v_{4,i}$ et $v_{8,i}$ par deux chaînes, $P_4$ et $P_8$, composées de $2n(u_i) + 1$ sommets (voir la figure 6.16). De plus, pour $k \in \{1, \ldots, n(\bar{u}_i)\}$ (resp. $k \in \{1, \ldots, n(u_i)\}$), on ajoute une arête entre les $(2k+1)$-ème sommets de $P_2$ et de $P_6$ (resp. $P_4$ et $P_8$) et le sommet du gadget $\mathcal{C}_j$ représentant la $k$-ème occurrence de $\bar{u}_i$ (resp. $u_i$) dans $C$. Enfin, le cycle de longueur 4 composé de $v_{2,i}, v_{4,i}, v_{6,i}$ et $v_{8,i}$ est désormais composé du premier sommet des chaînes $P_2$, $P_4$, $P_6$ et $P_8$.

La preuve du théorème 6.7 s'applique toujours sur l'instance transformée car la "forme" des solutions (voir la figure 6.15) reste inchangée empruntant soit $P_2$ et $P_6$ soit $P_4$ et $P_8$. Nous obtenons alors le théorème suivant :

**Théorème 6.8.** $2 - \textsc{Cycle}$I *est* $\mathcal{NP}$*-complet dans les graphes bipartis de degré maximum* 4.

Enfin, ce dernier résultat se généralise pour un nombre de terminaux fixé quelconque :

**Théorème 6.9.** $k - \textsc{Cycle}$I *est* $\mathcal{NP}$*-complet dans les graphes bipartis de degré maximum* 4 *pour toute valeur fixée de* $k \geq 2$.

*Démonstration.* Il suffit de reprendre la preuve du théorème précédent en remplaçant l'arête $x_1 - v_{1,1}$ par une chaîne de longueur impaire contenant autant de terminaux que l'on souhaite. □

### 6.3.3 Complexité des problèmes pour 3 terminaux

D'après le théorème 6.9, $3-\textsc{Cycle}$I est $\mathcal{NP}$-complet dans les graphes bipartis de degré maximum 4 et nous allons désormais nous focaliser sur $3 - \textsc{Chaine}$I et $3 - \textsc{Arbre}$I.

**Théorème 6.10.** $3-\textsc{Chaine}$I *est* $\mathcal{NP}$*-complet dans les graphes bipartis de degré maximum* 4.

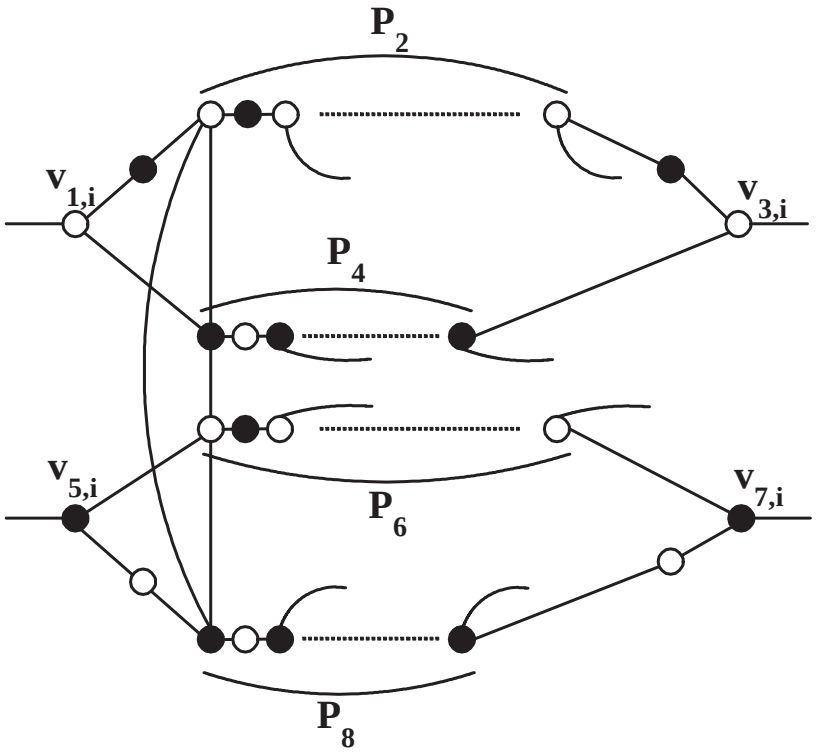


FIGURE 6.16 – Le nouveau gadget $\mathcal{U}_i$ obtenu pour la variable $u_i$

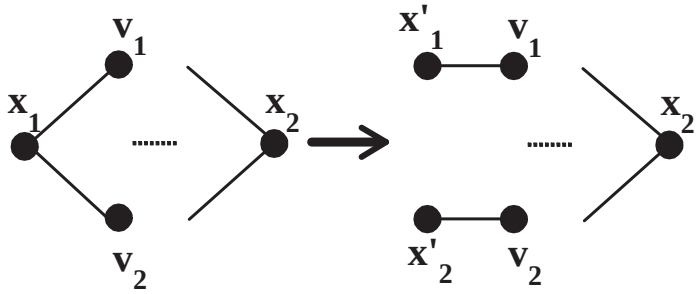


FIGURE 6.17 – Transformation d'une instance de $2-$CYCLEI en une instance de $3-$CHAINEI

*Démonstration.* Nous effectuons une réduction à partir de $2-$CYCLEI dans les graphes bipartis de degré maximum 4 dont la $\mathcal{NP}$-complétude a été montrée dans le théorème 6.8. Or, dans la preuve de ce théorème les deux terminaux $x_1$ et $x_2$ sont de degré 2. Notons alors $v_1$ et $v_2$ les deux voisins de $x_1$. Soit $G'$ le graphe obtenu en supprimant le sommet $x_1$ et en ajoutant deux nouveaux sommets $x'_1$ et $x'_2$ connectés respectivement à $v_1$ et $v_2$ (voir la figure 6.17).

De manière immédiate, il existe un cycle induit couvrant $x_1$ et $x_2$ si et seulement s'il existe dans $G'$ une chaîne induite couvrant $x'_1$, $x'_2$ et $x_2$. □

Contrairement à $3-$CHAINEI, $3-$ARBREI est polynomial ([16]) et nous montrons :

**Théorème 6.11.** $3-$MINARBREI *peut être résolu en* $O(m)$ *dans les graphes sans triangle.*

*Démonstration.* Soit $n_1(v)$, $n_2(v)$ et $n_3(v)$ la longueur d'une plus courte chaîne entre le sommet $v$ et respectivement $x_1$, $x_2$ et $x_3$. Soit $s$ un sommet tel que $n_1(s)+n_2(s)+n_3(s) = \min_{v\in V}\{n_1(v)+n_2(v)+n_3(v)\}$.

Soit $P_{x_1s}$ (resp. $P_{x_2s}$, $P_{x_3s}$) une plus courte chaîne entre $x_1$ (resp. $x_2$ et $x_3$) et $s$. Montrons que le graphe $H$ induit par $P_{x_1s} \cup P_{x_2s} \cup P_{x_3s}$ est un arbre.

Puisque $P_{x_1s}$, $P_{x_2s}$ et $P_{x_3s}$ sont trois plus courtes chaînes, elles sont nécessairement induites. Supposons que $H$ n'est pas un arbre et qu'il existe par exemple une arête $v-v'$ avec $v \in P_{x_1s}$ et $v' \in P_{x_2s}$ (voir la figure 6.18). Puisque le graphe est sans triangle, $v$ et $v'$ ne peuvent être simultanément voisins de $s$ et nous supposons alors que $s(v)$, la longueur de la chaîne $P_{vs} \subset P_{x_1s}$, est au moins 2. Notons $s(v')$ la longueur de la chaîne $P_{v's} \subset P_{x_2s}$. Ainsi, nous avons trois chaînes issues de $v'$, l'une de longueur $n_1(s)-s(v)+1$ rejoignant $x_1$, l'une de longueur $n_2(s)-s(v')$ rejoignant $x_2$ et l'une de longueur $n_3(s)+s(v')$ atteignant $x_3$.

D'où, $n_1(v')+n_2(v')+n_3(v') \leq n_1(s)-s(v)+1+n_2(s)-s(v')+n_3(s)+s(v') = n_1(s)+n_2(s)+n_3(s)-s(v)+1 < n_1(s)+n_2(s)+n_3(s)$ ce qui contredit la définition du sommet $s$. Donc $H$ est un arbre et est, par construction, solution optimale de l'instance de $3-$MINARBREI considérée.

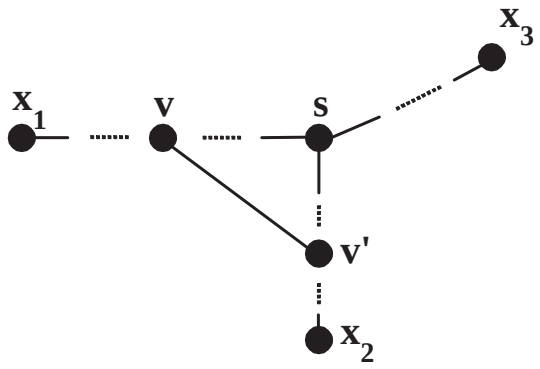


FIGURE 6.18 – Détermination de l'arbre induit de taille minimum pour un graphe sans triangle

Trois parcours en largeur à partir de $x_1$, $x_2$ et $x_3$ permettent d'obtenir $n_1(v)$, $n_2(v)$ et $n_3(v)$ pour tout sommet $v$ du graphe (ainsi que les chaînes correspondantes). Un arbre induit couvrant $x_1$, $x_2$ et $x_3$ s'obtient donc en $O(m)$. □

Nous avons alors :

**Corollaire 6.4.** $3-$ARBREI *admet toujours une solution dans les graphes sans triangle.*

*Démonstration.* C'est une conséquence directe de la preuve du théorème précédent où l'on peut toujours construire l'arbre induit $H$. □

Il est à noter que ces deux derniers résultats ne se généralisent pas directement pour les graphes contenant des triangles. A notre connaissance, $3-$MINARBREI reste donc un problème ouvert.

### 6.3.4 D'autres résultats pour un nombre de terminaux fixé

Nous n'avons pas réussi à déterminer la complexité de $k-$ARBREI et de $k-$MINARBREI pour une valeur de $k$ fixée supérieure ou égale à 4. Cependant, dans le cas particulier où la maille du graphe (c'est-à-dire la taille du plus petit cycle) est strictement supérieure au nombre de terminaux, nous montrons que ces deux problèmes sont polynomiaux.

**Lemme 6.1.** *Si la maille du graphe est supérieure ou égale à $k+1$ alors un arbre de taille minimum couvrant les $k$ terminaux (c'est-à-dire un arbre de Steiner de taille minimum) est un arbre induit.*

*Démonstration.* Soit $A = (V_A, E_A)$ un arbre de taille minimum couvrant $T = \{x_1, \ldots, x_k\}$. $A$ étant de taille minimum, toutes ses feuilles sont des terminaux.

Supposons qu'il existe une arête $e$ n'appartenant pas à $A$ et connectant deux sommets de $A$. Alors, le graphe $A' = (V_A, E_A \cup \{e\})$ possède exactement un cycle

$C$. Par hypothèse, la longueur de $C$ est strictement plus grande que $k$ et donc, au moins un de ses sommets n'est pas un terminal.

Supposons qu'aucun sommet de $C \setminus T$ n'est de degré 2 dans $A'$. Puisque la longueur de $C$ est strictement plus grande que $k$ et que chaque feuille de $A$ est un terminal, il devrait y avoir au moins $k+1$ sommets terminaux. Donc il existe un sommet $v_C \in C \setminus T$ de degré 2 dans $A'$ et soit $e_1$ et $e_2$ les deux arêtes de $C$ incidentes à $v_C$. Le graphe $A'' = (V_A \setminus \{v_C\}, (E_A \setminus \{e_1, e_2\}) \cup \{e\})$ est alors un arbre couvrant $T$ et sa taille est strictement inférieure à celle de $A$. Ceci est contradictoire avec la minimalité de $A$ et $e$ ne peut donc exister. □

Nous démontrons également le théorème suivant pour le problème de l'arbre de Steiner :

**Théorème 6.12.** *Déterminer un arbre de taille minimum couvrant $k$ sommets terminaux est un problème $\mathcal{NP}$-difficile dans les graphes bipartis de maille supérieure ou égale à $k+1$.*

*Démonstration.* Il est important de préciser que dans ce théorème la valeur de $k$ n'est pas fixée.

Nous effectuons une réduction à partir du problème $\mathcal{NP}$-difficile ([27]) de l'arbre de Steiner de taille minimum dans les graphes bipartis : nous remplaçons chaque arête du graphe par une chaîne de longueur $2k+1$. Le graphe obtenu est biparti et sa maille est supérieure ou égale à $k+1$. Puisque la taille de tout arbre couvrant les $k$ sommets terminaux a été multipliée par $2k+1$, toute arbre de taille minimum couvrant les $k$ terminaux dans le nouveau graphe nous permet d'obtenir une solution pour l'instance initiale (et réciproquement). □

Nous obtenons alors les théorèmes désirés :

**Théorème 6.13.** *$k-$* ArbreI *admet toujours une solution dans les graphes de maille supérieure ou égale à $k+1$.*

*Démonstration.* C'est une conséquence immédiate du lemme 6.1 puisqu'un arbre de Steiner existe toujours dans un graphe connexe. □

**Théorème 6.14.** *$k-$* MinArbreI *est $\mathcal{NP}$-difficile dans les graphes bipartis de maille supérieure ou égale à $k+1$.*

*Démonstration.* C'est une conséquence directe du lemme 6.1 et du théorème 6.12. □

**Théorème 6.15.** *Pour une valeur de $k$ fixée, $k-$* MinArbreI *est polynomial dans les graphes de maille supérieure ou égale à $k+1$.*

*Démonstration.* Le problème de l'arbre de Steiner de taille minimum est polynomial pour une valeur de $k$ fixée ([24]). Un algorithme polynomial dont la complexité est en $O(3^k n + 2^k m \cdot \log(n))$ est donné dans [41]. □

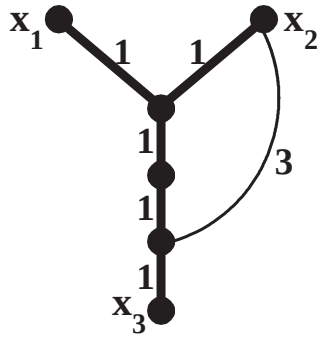


FIGURE 6.19 – Une instance du problème de l'arbre de Steiner où la solution optimale n'est pas un arbre induit

Cependant, le lemme 6.1 et le théorème 6.15 ne se généralisent pas au cas des graphes pondérés car un arbre de Steiner de poids minimum n'est pas nécessairement un arbre induit même si la maille du graphe est supérieure ou égale à $k+1$. Sur la figure 6.19, le poids minimum d'un arbre induit couvrant les trois terminaux vaut 6 tandis que le poids minimum d'un arbre de Steiner (représenté en gras) vaut 5.

De plus, rappelons que nous avons montré dans la section 6.2.1 que $2-$PONDARBREI est $\mathcal{NP}$-complet au sens fort dans les graphes bipartis cubiques même si tous les poids sont 0 ou 1 (voir le théorème 6.2 page 95).

# Chapitre 7

# A la recherche d'un arbre induit couvrant 4 terminaux

De nombreux cas de $k-$ ArbreI et de ses variantes ont été étudiés dans le chapitre précédent. Dans la très grande majorité de ces cas, l'étude du problème de l'existence d'un arbre induit couvrant l'ensemble des terminaux a débouché sur un résultat de $\mathcal{NP}$-complétude ou sur une existence certaine. Il n'y a eu aucun résultat permettant de déterminer l'allure des graphes pour lesquels $k-$ ArbreI admet ou non une solution.

Dans ce chapitre, nous étudions plus en détail $4-$ ArbreI pour lequel il existe toujours une solution dans les graphes sans cycle de longueur 3 ou 4 (voir le théorème 6.13 page 107). Nous allons nous focaliser sur le cas des graphes sans triangle pour répondre à la question suivante : "*A quoi peut ressembler un graphe sans triangle qui n'admet pas d'arbre induit couvrant les quatre terminaux ?*".

## 7.1 Préliminaires

Nous supposerons dans toute la suite de ce chapitre que les quatre terminaux, $x_1, x_2, x_3, x_4$, sont de degré 1. En effet, nous avons :

**Propriété 7.1.** *Toute instance de* $4-$ ArbreI *peut être transformée en une instance équivalente où tous les terminaux sont de degré* 1.

*Démonstration.* Soit $I$ une instance de $4-$ArbreI. Pour chaque sommet terminal $x_i$, nous ajoutons un nouveau sommet $x'_i$ et une arête $x_i - x'_i$. Notons alors $T' = \{x'_1, x'_2, x'_3, x'_4\}$.

Il existe un arbre induit couvrant $T$ si et seulement s'il existe un arbre induit couvrant $T'$.

En effet, si $A$ est un ensemble de sommets induisant un arbre et contenant $x_1, x_2, x_3, x_4$, nous obtenons un arbre induit couvrant $T'$ en ajoutant à $A$ les quatre nouveaux sommets. Réciproquement, si $A'$ induit un arbre et contient $x'_1, x'_2, x'_3, x'_4$, il suffit de supprimer de $A'$ les sommets $x'_i$. □

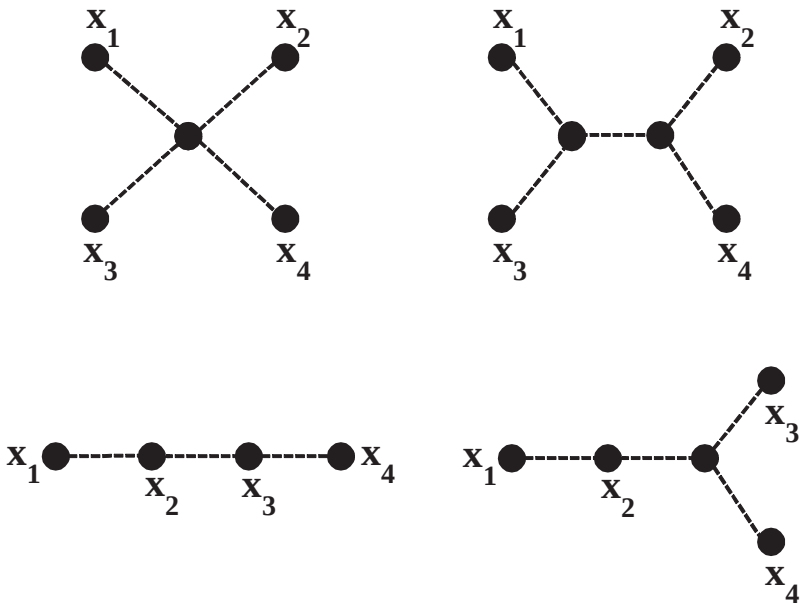


FIGURE 7.1 – Les quatre topologies possibles pour un arbre induit couvrant 4 terminaux (les pointillés représentent des chaînes et non une seule arête)

Il est à noter que ce résultat se généralise pour $k-$ ARBREI ($k$ fixé ou non) mais pas pour $k-$ CHAINEI ($k \geq 3$) et $k-$ CYCLEI ($k \geq 1$) car ces deux problèmes n'admettent aucune solution si les sommets terminaux sont de degré 1.

Nous avons représenté sur la figure 7.1 les quatre topologies possibles pour un arbre induit couvrant quatre sommets terminaux. Or, puisque tous les terminaux sont désormais de degré 1, seules les deux topologies du haut de la figure 7.1 subsistent.

De plus, nous montrons que la topologie en étoile située en haut à gauche de la figure 7.1 correspond à un cas $\mathcal{NP}$-complet :

**Théorème 7.1.** *Le problème d'existence d'une étoile généralisée induite, telle que ses quatre feuilles sont les* 4 *terminaux, est $\mathcal{NP}$-complet dans les graphes bipartis de degré maximum* 4.

*Démonstration.* Une étoile généralisée est un arbre où il existe un unique sommet de degré supérieur ou égal à 3.

Nous réalisons une réduction à partir de $2-$CYCLEI qui est $\mathcal{NP}$-complet dans les graphes bipartis de degré maximum 4 même si les deux sommets terminaux sont de degré 2 (voir le théorème 6.8 page 103).

Soit $I$ une instance de $2-$ CYCLEI composée d'un graphe $G$ biparti de degré maximum 4 et d'un ensemble $T = \{x_1, x_2\}$. Notons $x'_1, x''_1$ et $x'_2, x''_2$ les voisins respectifs de $x_1$ et $x_2$. Nous considérons alors la transformation suivante (voir la figure 7.2) : nous ajoutons 5 nouveaux sommets $c, y_1, y_2, y_3, y_4$ ainsi que les arêtes

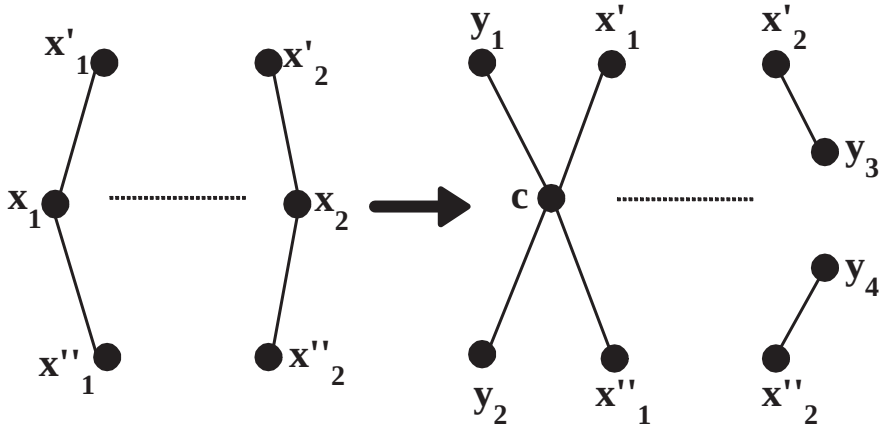


FIGURE 7.2 – Passage de $2-$CYCLEI à $4-$ARBREI

$c-x'_1$, $c-x''_1$, $c-y_1$, $c-y_2$, $x'_2-y_3$ et $x''_2-y_4$ puis nous supprimons les sommets $x_1$ et $x_2$. Le graphe $H$ ainsi obtenu est donc biparti et de degré maximum 4.

Montrons qu'il existe un cycle induit $C$ couvrant $x_1, x_2$ si et seulement s'il existe une étoile généralisée induite $S$ couvrant $y_1, y_2, y_3, y_4$.

Soit $C$ un ensemble de sommets contenant $x_1$ et $x_2$ et induisant un cycle. Le graphe induit par $C \cup \{c, y_1, y_2, y_3, y_4\} \setminus \{x_1, x_2\}$ est nécessairement une étoile généralisée de sommet central $c$ et couvre $y_1, y_2, y_3, y_4$.

Réciproquement, soit $S$ un ensemble de sommets induisant une étoile généralisée et contenant $y_1, y_2, y_3, y_4$. $c$ est nécessairement le sommet central car il est dans $S$ et de degré supérieur ou égal à 3. Le graphe induit par $S \cup \{x_1, x_2\} \setminus \{c, y_1, y_2, y_3, y_4\}$ est donc un cycle passant par $x_1$ et $x_2$. □

Nous n'avons cependant pas réussi à déterminer la complexité de $4-$ARBREI dans le cas général qui reste, à ce jour, un problème ouvert. Nous nous sommes alors intéressés au cas où le graphe ne contient pas de triangle et avons étudié la structure des instances n'admettant pas de solution.

A titre d'exemple, nous représentons sur la figure 7.3 deux instances n'admettant aucune solution. Ces deux instances correspondent à un carré et à un cube et vont être à la base des structures qui nous serviront par la suite. Pour le carré, tout ensemble de sommets couvrant les quatre terminaux induit nécessairement le carré central. Concernant le cube, toute solution de $4-$ARBREI contient au moins 6 sommets du cube - dont les quatre sommets voisins des terminaux - et induit donc nécessairement au moins l'une des faces du cube.

Dans les sections 7.2, 7.3 et 7.4, tous les graphes considérés sont sans triangle. Nous démontrons que dans ce cas, $4-$ARBREI est polynomial et nous élaborons un algorithme permettant de le résoudre dont la complexité est $O(nm)$.

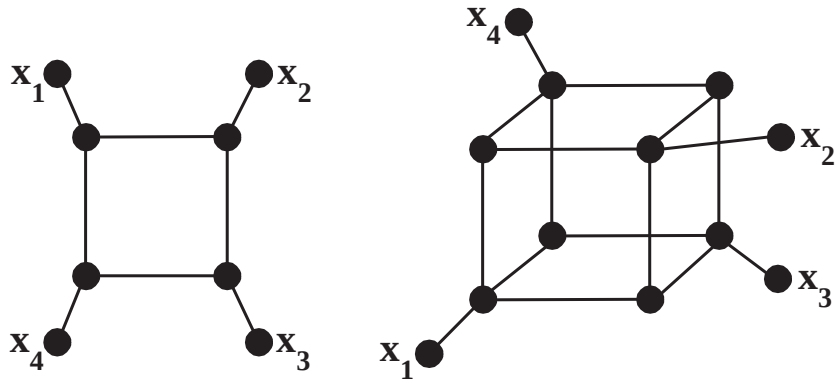


FIGURE 7.3 – Deux instances pour lesquelles 4 – ARBREI n'admet pas de solution

## 7.2 Définition des structures et des algorithmes associés

Nous avons vu que la figure 7.3 représente deux instances de 4 – ARBREI n'admettant pas de solution. De manière plus générale, nous allons définir deux structures particulières de graphes sans triangle pour lesquelles il n'existe jamais de solution à 4 – ARBREI : les *carrés* et les *cubes*.

Tout d'abord, rappelons qu'un ensemble stable $S$ est un ensemble de sommets tel qu'il n'existe pas d'arête entre toute paire de sommets de $S$ (voir la définition 3.6 page 64). De plus, nous introduisons les définitions suivantes :

**Définition 7.1.** *Un ensemble $S$ de sommets est* ***complet*** *à un ensemble $S'$ de sommets s'il existe une arête $s - s'$ pour tout $s \in S$ et pour tout $s' \in S'$.*

**Définition 7.2.** *Un ensemble $S$ de sommets est* ***anticomplet*** *à un ensemble $S'$ de sommets s'il n'existe aucune arête entre un sommet de $S$ et un sommet de $S'$.*

Si $S$ est un ensemble de sommets de $G$, nous notons $G[S]$ le graphe induit par $S$. Pour tout sommet $v$, nous notons $N(v)$ l'ensemble des sommets voisins de $v$. De plus, pour un ensemble de sommets $A$, $N(A)$ contient tous les sommets de $G$ n'étant pas dans $A$ mais ayant au moins un voisin dans $A$. Posons également $N_Z(A) = N(A) \cap Z$ pour toute paire d'ensembles $Z$ et $A$ de sommets. Si $P$ est une chaîne induite et $v$ et $v'$ deux sommets de $P$ alors nous notons $v - P - v'$ le sous-graphe de $P$ qui est une chaîne allant de $v$ à $v'$. Enfin, si nous définissons $k$ ensembles $A_1, \ldots, A_k$, nous notons leur union par $A$. De manière analogue, $B = \bigcup_{i=1}^{k} B_i$ et $S = \bigcup_{i=1}^{k} S_i$.

Définissons une première structure pour les graphes sans triangle :

**Définition 7.3.** Un graphe $G = (V, E)$ est *un carré* s'il existe une partition de $V$ en 9 ensembles $A_1, A_2, A_3, A_4, S_1, S_2, S_3, S_4, R$ telle que :

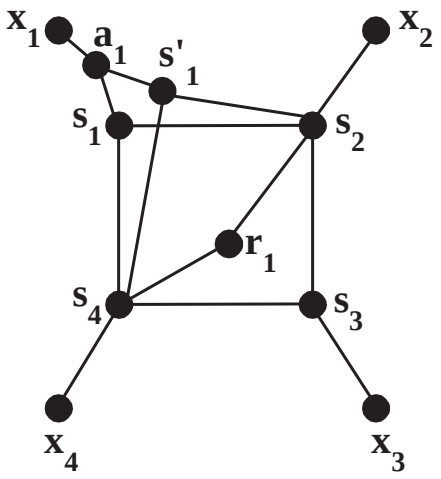


FIGURE 7.4 – Un exemple de carré : $A_1 = \{x_1, a_1\}$, $A_2 = \{x_2\}$, $A_3 = \{x_3\}$, $A_4 = \{x_4\}$, $S_1 = \{s_1, s'_1\}$, $S_2 = \{s_2\}$, $S_3 = \{s_3\}$, $S_4 = \{s_4\}$ et $R = \{r_1\}$.

1. $x_i \in A_i$ pour $i \in \{1, \ldots, 4\}$
2. $S_i$ est un ensemble stable pour $i \in \{1, \ldots, 4\}$
3. $S_i \neq \emptyset$ pour $i \in \{1, \ldots, 4\}$
4. $S_i$ est complet à $S_{i+1}$ pour $i \in \{1, \ldots, 3\}$ et $S_4$ est complet à $S_1$
5. $S_i$ est anticomplet à $S_{i+2}$ pour $i \in \{1, 2\}$
6. $N(A_i) = S_i$ pour $i \in \{1, \ldots, 4\}$
7. $N(R) \subset S$
8. $G[A_i]$ est connexe pour $i \in \{1, \ldots, 4\}$
9. $G[A \cup S \cup R]$ est connexe

Un exemple de *carré* est représenté sur la figure 7.4. De plus, nous dirons qu'une *division* d'un carré consiste en la donnée de 9 ensembles $A_1, A_2, A_3, A_4, S_1, S_2, S_3, S_4, R$ de sommets vérifiant la définition précédente tandis qu'*un carré de* $G$ est un ensemble $Z \subset V$ tel quel $G[Z]$ est un carré.

Le lemme suivant, que nous prouvons dans la section 7.3, nous permet d'affirmer que si nous trouvons dans un graphe sans triangle un carré, nous pouvons ajouter un à un les sommets restants à ce carré à moins d'obtenir à un moment donné un cube ou un arbre induit couvrant les quatre terminaux.

**Lemme 7.1.** *Il existe un algorithme aux spécificités suivantes :*

ENTRÉE : *un graphe sans triangle $G$, quatre terminaux $x_1, x_2, x_3, x_4$, une division d'un carré $Z$ de $G$ et un sommet $v \notin Z$ ayant au moins un voisin dans $Z$*

SORTIE : *un arbre induit couvrant* $x_1, x_2, x_3, x_4$ ou *une division d'un cube de* $G$ ou *une division du carré* $G[Z \cup \{v\}]$

COMPLEXITÉ : $O(m)$

Focalisons-nous maintenant sur le second type de structure :

**Définition 7.4.** Un graphe $G = (V, E)$ est *un cube* s'il existe une partition de $V$ en 17 ensembles $A_1, \ldots, A_4, B_1, \ldots, B_4, S_1, \ldots, S_8, R$ telle que :

1. $x_i \in A_i$ pour $i \in \{1, \ldots, 4\}$
2. $S_i$ est un ensemble stable pour $i \in \{1, \ldots, 8\}$
3. $S_i$ est non vide pour $i \in \{1, \ldots, 4\}$
4. Au plus l'un des ensembles $S_5, S_6, S_7, S_8$ est vide
5. $S_i$ est complet à $(S_5 \cup S_6 \cup S_7 \cup S_8) \setminus S_{i+4}$ pour $i \in \{1, \ldots, 4\}$
6. $S_i$ est anticomplet à $S_{i+4}$ pour $i \in \{1, \ldots, 4\}$
7. $S_i$ est anticomplet à $S_j$ pour $1 \leq i < j \leq 4$
8. $S_i$ est anticomplet $S_j$ pour $5 \leq i < j \leq 8$
9. $N(A_i) = S_i$ pour $i \in \{1, \ldots, 4\}$
10. $N(B_i) \subset S_i \cup N_S(S_i)$ pour $i \in \{1, \ldots, 4\}$
11. $N(R) \subset S_5 \cup S_6 \cup S_7 \cup S_8$
12. $G[A \cup S \cup B \cup R]$ est connexe
13. $G[A_i]$ est connexe pour $i \in \{1, \ldots, 4\}$

Nous donnons deux exemples de *cube* sur la figure 7.5 qui permettent d'apprécier la proximité entre la structure cubique définie ci-dessus et un véritable cube.

Du point de vue de la terminologie, nous dirons qu'une *division* d'un cube consiste en la donnée de 17 ensembles $A_1, \ldots, A_4, B_1, \ldots, B_4, S_1, \ldots, S_8, R$ de sommets vérifiant la définition précédente tandis qu'*un cube de* $G$ est un ensemble $Z \subset V$ tel quel $G[Z]$ est un cube.

De manière similaire à la structure carrée, le lemme suivant - que nous prouvons dans la section 7.4 - atteste que si nous connaissons un cube de $G$, nous pouvons ajouter un à un les sommets restants à cette structure à moins d'obtenir à un moment donné un arbre induit couvrant les quatre terminaux.

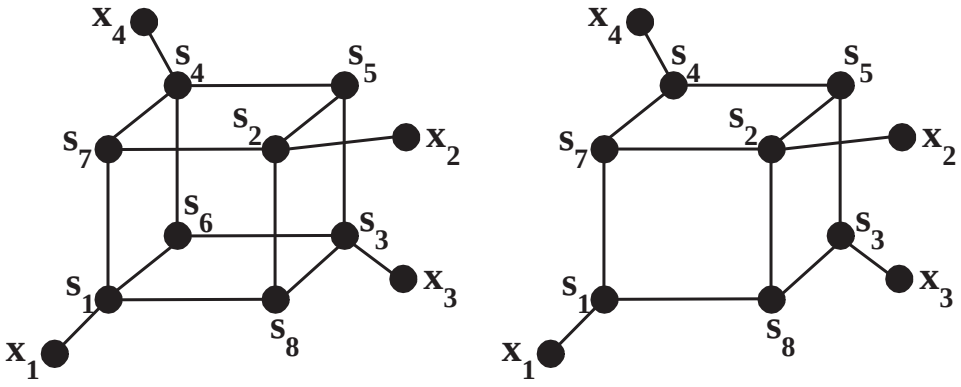


FIGURE 7.5 – Deux exemples de structure cubique

**Lemme 7.2.** *Il existe un algorithme aux spécificités suivantes :*

ENTRÉE : *un graphe sans triangle $G$, quatre terminaux $x_1, x_2, x_3, x_4$, une division d'un cube $Z$ de $G$ et un sommet $v \notin Z$ ayant au moins un voisin dans $Z$*

SORTIE : *un arbre induit couvrant $x_1, x_2, x_3, x_4$* ou *une division du cube $G[Z \cup \{v\}]$*

COMPLEXITÉ : $O(m)$

Notre algorithme pour résoudre $4 - \text{ARBREI}$ dans les graphes sans triangle consiste à déterminer tout d'abord un arbre induit contenant les terminaux ou un premier carré de $G$. Dans le second cas, nous ajoutons les autres sommets à cette structure jusqu'à obtenir un arbre induit couvrant les quatre terminaux ou une partition des sommets de $G$ qui soit la division d'un carré ou d'un cube. Nous montrons ainsi le théorème suivant :

**Théorème 7.2.** *Soit $G$ un graphe sans triangle et $x_1, x_2, x_3, x_4$ quatre sommets terminaux. On a alors l'un des deux cas suivants :*

- *$G$ est un carré ou un cube*
- *$G$ contient un arbre induit couvrant les quatre terminaux*

*De plus, ces deux cas ne peuvent advenir simultanément.*

*Démonstration.* Montrons tout d'abord que les deux cas sont exclusifs.

Supposons que $G$ est un carré et soit $A_1, \ldots, A_4, S_1, \ldots, S_4, R$ une division de $G$. Tout arbre couvrant les quatre terminaux contient nécessairement un sommet de chaque $S_i$ car $x_i \in A_i$ et $N(A_i) = S_i$ pour $i \in \{1, \ldots, 4\}$. Les sommets de l'arbre induisent donc au moins un cycle de longueur 4.

Supposons maintenant que $G$ est un cube et soit $A_1, \ldots, A_4, B_1, \ldots, B_4, S_1, \ldots, S_8, R$ une division de $G$. Soit $A$ un arbre couvrant les quatre terminaux. $A$ possède nécessairement un sommet $s_i$ dans chaque $S_i$ pour $i \in \{1, \ldots, 4\}$ ainsi qu'au moins un sommet dans deux ensembles parmi $S_5, \ldots, S_8$ puisque pour $i \in \{5, \ldots, 8\}$ $S_1 \cup S_2 \cup S_3 \cup S_4 \not\subseteq N_S(S_i)$.

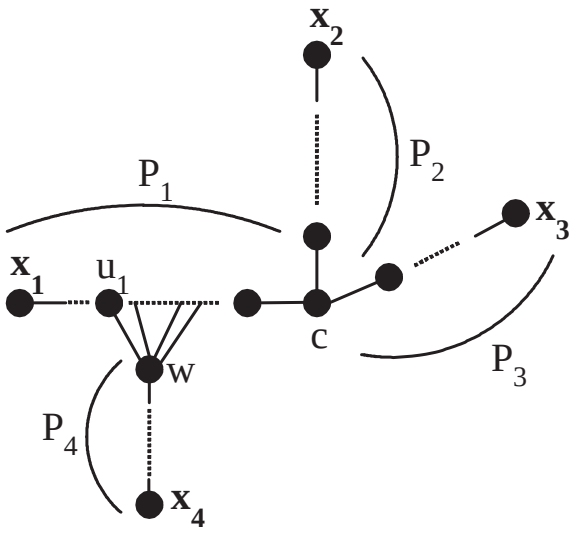


FIGURE 7.6 – Un exemple où $w$ n'a des voisins que dans $P_1$

Supposons par exemple que $A$ contient $s_5 \in S_5$ et $s_6 \in S_6$ (les autres cas étant symétriques). Les sommets de $A$ induisent alors un cycle de longueur 4 composé des sommets $s_3$, $s_6$, $s_5$, $s_4$.

Il reste à montrer que l'on est toujours dans l'un des deux cas du théorème. D'après le corollaire 6.4 (page 106), puisque le graphe est sans triangle il existe un arbre induit $B$ centré en un sommet $c$, couvrant $x_1, x_2$ et $x_3$ et consistant en l'union de trois chaînes induites $P_1 = c - \cdots - x_1$, $P_2 = c - \cdots - x_2$ et $P_3 = c - \cdots - x_3$. En déterminant un parcours en largeur à partir de $x_4$, nous obtenons une plus courte chaîne $P_4 = x_4 - \cdots - w$ telle que $w$ a au moins un voisin dans $B$. Pour $i \in \{1, 2, 3\}$, si $w$ a un voisin dans $P_i$, nous notons $u_i$ le voisin de $w$ le plus près de $x_i$ sur $P_i$ (voir la figure 7.6). Considérons les trois cas suivants :

- $w$ a des voisins dans $P_1$, $P_2$ et $P_3$. Le graphe induit par $P_4 \cup (u_1 - P_1 - x_1) \cup (u_2 - P_2 - x_2) \cup (u_3 - P_3 - x_3)$ est un arbre et contient les quatre terminaux.
- $w$ a des voisins dans exactement un seul des trois ensembles $P_1$, $P_2$ et $P_3$ (par exemple $P_1$). D'après le théorème 6.11 (page 105), puisque le graphe est sans triangle, nous savons déterminer un arbre induit $B'$ dans $G[P_1 \cup \{w\}]$ couvrant $c, x_1$ et $w$. Le graphe induit par $B' \cup P_4 \cup P_2 \cup P_3$ est un arbre et contient les quatre terminaux.
- $w$ a des voisins dans exactement deux des trois ensembles $P_1$, $P_2$ et $P_3$ (par exemple $P_1$ et $P_3$). Nous avons alors deux cas :

  1. Au moins l'une des arêtes $u_1 - c$ et $u_3 - c$ n'existe pas (supposons que $u_1 - c$ n'existe pas). D'après le théorème 6.11 (page 105) et puisque le graphe est sans triangle, nous déterminons un arbre induit $B''$ de

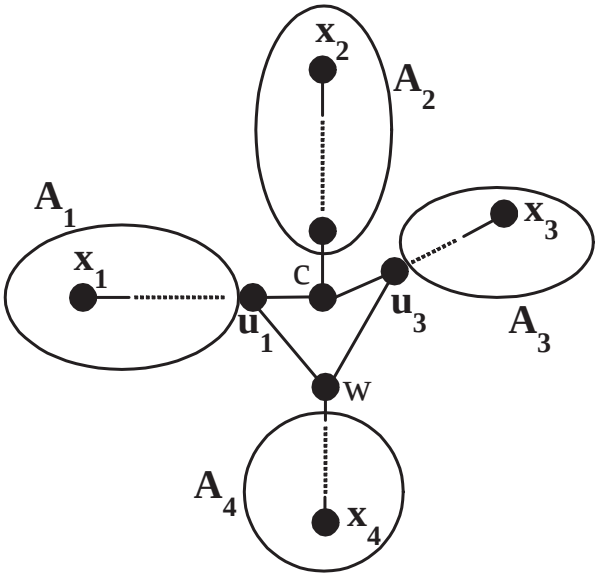


FIGURE 7.7 – Le premier carré de $G$ obtenu

$G[P_3 \cup \{w\}]$ couvrant $w, c$ et $x_3$. Le graphe induit par $B'' \cup P_4 \cup (x_1 - P_1 - u_1) \cup P_2$ est un arbre couvrant tous les terminaux.

2. Les arêtes $u_1 - c$ et $u_3 - c$ existent. Le graphe induit par $Z = P_1 \cup P_2 \cup P_3 \cup P_4$ est un carré et une division de celui-ci est obtenue en posant (voir la figure 7.7) : $A_1 = (x_1 - P_1 - u_1) \setminus \{u_1\}$, $A_2 = P_2 \setminus \{c\}$, $A_3 = (x_3 - P_3 - u_3) \setminus \{u_3\}$, $A_4 = P_4 \setminus \{w\}$, $S_1 = \{u_1\}$, $S_2 = \{c\}$, $S_3 = \{u_3\}$, $S_4 = \{w\}$ et $R = \emptyset$.

Si nous avons obtenu un arbre induit couvrant les quatre terminaux alors le théorème est montré, sinon nous avons déterminé un carré $Z$ de $G$. Nous utilisons alors l'algorithme du lemme 7.1 de la manière suivante : tant qu'il existe un sommet $v$ n'appartenant pas à $Z$ et possédant au moins un voisin dans $Z$, nous essayons d'ajouter $v$ à $Z$. D'après ce lemme, soit nous obtenons à un moment donné un cube $Z'$ de $G$ ou un arbre induit couvrant les quatre terminaux, soit nous réussissons à ajouter tous les sommets de $G$ au carré. Dans deux de ces trois cas, le théorème est montré. Il reste à traiter le cas où nous avons obtenu un cube $Z'$ de $G$.

Nous utilisons alors l'algorithme du lemme 7.2 de la manière suivante : tant qu'il existe un sommet $v$ n'appartenant pas à $Z'$ et possédant au moins un voisin dans $Z'$, nous essayons d'ajouter $v$ à $Z'$. D'après ce lemme, soit nous obtenons à un moment donné un arbre induit couvrant les quatre terminaux, soit nous ajoutons tous les sommets de $G$ au cube et le théorème est donc démontré. □

**Corollaire 7.1.** *Il existe un algorithme aux spécificités suivantes :*

ENTRÉE : *un graphe sans triangle $G$, quatre terminaux $x_1, x_2, x_3, x_4$*

SORTIE : *un arbre induit couvrant* $x_1, x_2, x_3, x_4$ ou *une division du carré* $G$ ou *une division du cube* $G$

COMPLEXITÉ : $O(mn)$

*Démonstration.* C'est une conséquence directe de la preuve précédente. En effet, pour montrer le théorème, nous avons donné une preuve algorithmique puisqu'à partir de $G$ et des quatre terminaux, nous aboutissons soit à un arbre induit couvrant les quatre terminaux, soit au fait que $G$ est un carré ou un cube.

La première partie de l'algorithme nous permet d'obtenir un arbre induit couvrant les quatre terminaux ou un premier carré $Z$ de $G$. Cette première partie utilise $O(1)$ fois un parcours en largeur du graphe et l'algorithme du théorème 6.11 (page 105). Ce dernier s'exécutant en temps $O(m)$, nous obtenons une complexité en $O(m)$.

Puis, nous appliquons $O(n)$ fois les algorithmes des lemmes 7.1 et 7.2 en ajoutant un à un les sommets du graphe au carré ou au cube jusqu'à trouver un arbre induit couvrant les quatre terminaux ou une partition des sommets de $G$ correspondant à la division d'un carré ou d'un cube.

La complexité globale de l'algorithme est donc en $O(nm)$. □

## 7.3 Ajout d'un sommet au carré

Dans cette section, nous apportons la preuve du lemme 7.1. Soit $Z$ un carré de $G$, $A_1, \ldots, A_4, S_1, \ldots, S_4, R$ une division de $G[Z]$ et $v$ un sommet de $G$ n'appartenant pas à $Z$ mais possédant au moins un voisin dans $Z$.

Nous montrons ci-dessous qu'après application de l'algorithme du lemme 7.1, nous obtenons soit un arbre induit couvrant les quatre terminaux, soit un cube de $G$, soit une division du carré $G[Z \cup \{v\}]$. De plus, cette preuve peut être également vue comme la description d'un algorithme de complexité $O(m)$. En effet, il suffit de constater que nous faisons uniquement appel un nombre constant de fois (c'est-à-dire en $O(1)$) à des parcours en largeur du graphe, à des vérification du voisinage de certains ensembles de sommets et à l'algorithme du théorème 6.11 (page 105) qui permet de déterminer un arbre induit couvrant trois terminaux dans un graphe sans triangle et dont la complexité est $O(m)$.

Pour $i \in \{1, \ldots, 4\}$, si $s_i$ (respectivement $a_i$) est un sommet de $S_i \cup A_i$ alors nous notons $P_{s_i}$ (resp. $P_{a_i}$) une plus courte chaîne allant de $s_i$ (resp. $a_i$) à $x_i$ telle que $P_{s_i} \setminus \{s_i\} \subset A_i$ (resp. $P_{a_i} \setminus \{a_i\} \subset A_i$). Une telle chaîne existe toujours puisque d'après le point 8 de la définition 7.3 (page 112), $G[A_i]$ est connexe, et d'après le point 6 tout sommet de $S_i$ à un voisin dans $A_i$. De plus, cette chaîne est nécessairement induite puisque c'est une plus courte chaîne (voir la propriété 6.2 page 91).

Nous entamons la preuve du lemme par la remarque suivante : si $v$ n'a aucun voisin dans $A$ alors $v$ peut être placé dans $R$ et nous obtenons ainsi une division du carré $G[Z \cup \{v\}]$. Nous supposons désormais que $v$ a au moins un voisin dans $A$ par exemple $a_1$ dans $A_1$. Sans perte de généralité, nous supposons également

que $a_1$ a été choisi dans $A_1$ de telle sorte que $P_{a_1}$ soit de longueur minium. Nous montrons tout d'abord la propriété suivante :

**Assertion 7.1.** *Soit une chaîne induite $Q = v - \cdots - w$, minimale au sens de l'inclusion, vérifiant $Q \subset R \cup \{v\}$ et telle que $w$ possède au moins un voisin dans $(A \setminus A_1) \cup (S \setminus S_1)$. On a alors l'un des trois cas suivants :*

1. *il existe un arbre induit dans $G[Z \cup \{v\}]$ couvrant les quatre terminaux*
2. *$N_S(w) = S_2 \cup S_4$ et $N_A(w) \subset A_1$*
3. *$G[Z \cup \{v\}]$ contient un cube*

*Démonstration.* Nous entamons cette preuve par deux remarques que nous utilisons tout au long de la démonstration :

- $Q$ peut être de longueur nulle : par exemple, si $w$ a des voisins dans $A \setminus A_1$ alors $Q = v = w$ puisque d'après le point 7 de la définition 7.3 (page 112), un sommet de $R$ ne peut avoir de voisin dans $A$.
- Un sommet $v$ ne peut être voisin à la fois d'un sommet $v_i \in S_i$ et d'un sommet $v_j \in S_j$ pour $(i, j) \in \{(1, 2), (2, 3), (3, 4), (4, 1)\}$ car d'après le point 4 de la définition 7.3, les sommets $v, v_i, v_j$ induiraient alors un triangle.

Afin de démontrer l'assertion, nous allons prouver successivement 6 propriétés intermédiaires numérotées de (1) à (6).

*(1) Si $w$ a au moins un voisin dans $A_2 \cup A_4$ alors $w = v$ et il existe un arbre induit couvrant $x_1$, $x_2$, $x_3$ et $x_4$.*

D'après l'une des remarques précédentes, on a nécessairement $v = w$. Soit $a_2$ un voisin de $v$ dans $A_2$ tel que la longueur de $P_{a_2}$ soit minimum (le cas où $v$ n'a des voisins que dans $A_4$ est symétrique).

Si $v$ a au moins un voisin dans $A_3 \cup S_3$ et au moins un voisin dans $A_4 \cup S_4$ alors nous choisissons $a_3 \in A_3 \cup S_3$ et $a_4 \in A_4 \cup S_4$ voisins de $v$ tels que $P_{a_3}$ et $P_{a_4}$ soient de longueur minimum (voir la figure 7.8). En fait, on ne peut avoir simultanément $a_3 \in S_3$ et $a_4 \in S_4$ car dans ce cas, $v$, $a_3$ et $a_4$ formeraient un triangle. Ainsi, le graphe induit par $P_{a_1} \cup \{v\} \cup P_{a_2} \cup P_{a_3} \cup P_{a_4}$ est un arbre et contient $x_1$, $x_2$, $x_3$ et $x_4$. Nous supposons désormais que $v$ n'a pas de voisin dans $A_4 \cup S_4$.

Si $v$ a un voisin $a_3$ dans $A_3 \cup S_3$, considérons alors un sommet $s_3$ de $S_3$ (si $a_3 \in S_3$ nous choisissons $s_3 = a_3$) et un sommet $s_4$ de $S_4$. Soit $B_3$ un arbre induit de $G[A_3 \cup \{v, s_3\}]$ qui contient $v$, $s_3$ et $x_3$. $B_3$ existe nécessairement puisque le graphe est sans triangle (voir le corollaire 6.4 page 106). Le graphe induit par $P_{a_1} \cup \{v\} \cup P_{a_2} \cup B_3 \cup P_{s_4}$ est alors un arbre contenant les quatre terminaux car il existe au moins une arête entre $S_3$ et $S_4$ qui sont complets l'un à l'autre. Nous supposons désormais que $v$ n'a pas de voisin dans $A_3 \cup S_3$.

Soit $s_1 \in S_1$, $s_3 \in S_3$ et $s_4 \in S_4$. Considérons un arbre induit $B_1$ sous-graphe de $G[A_1 \cup \{v, s_1\}]$ et qui contient $v$, $s_1$ et $x_1$. $B_1$ existe nécessairement puisque le graphe est sans triangle (voir le corollaire 6.4). Le graphe induit par $B_1 \cup P_{a_2} \cup P_{s_3} \cup P_{s_4}$ est alors un arbre couvrant $x_1$, $x_2$, $x_3$ et $x_4$.

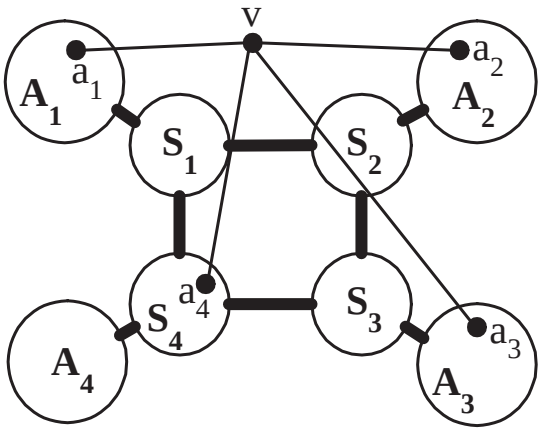


FIGURE 7.8 – Le premier cas obtenu pour la preuve du lemme 7.1

(1) est donc prouvé. Nous supposons à partir de maintenant que $v$ n'a aucun voisin dans $A_2 \cup A_4$ et nous montrons :

*(2) Si $w$ a au moins un voisin dans $S_3$ alors il existe un arbre induit couvrant $x_1$, $x_2$, $x_3$ et $x_4$.*

Soit $s_3$ un sommet de $S_3$ voisin de $w$, $s_2$ un sommet de $S_2$ et $s_4$ un sommet de $S_4$. Puisque $G$ ne contient aucun triangle, $w$ n'a aucun voisin dans $S_2 \cup S_4$. Considérons alors un arbre induit $B_3$ sous-graphe de $G[A_3 \cup \{w, s_3\}]$ et qui contient $w$, $s_3$ et $x_3$. Le graphe induit par $P_{a_1} \cup Q \cup P_{s_2} \cup B_3 \cup P_{s_4}$ est un arbre contenant les quatre terminaux.

Cela prouve (2). Nous supposons désormais que $w$ n'a aucun voisin dans $S_3$ et nous obtenons :

*(3) Si $w$ n'a aucun voisin dans $S_2 \cup S_4$ alors il existe un arbre induit couvrant $x_1$, $x_2$, $x_3$ et $x_4$.*

Si $w$ n'a pas de voisin dans $S_2 \cup S_4$ alors, par définition de $Q$, $w$ a un voisin $a_3$ dans $A_3$. D'après la première remarque donnée en début de preuve, on a donc $Q = v = w$. Soit $s_2 \in S_2$, $s_3 \in S_3$ et $s_4 \in S_4$. Soit $B_3$ un arbre induit sous-graphe de $G[A_3 \cup \{v, s_3\}]$ et contenant $v$, $s_3$ et $x_3$. Le graphe induit par $P_{a_1} \cup P_{s_2} \cup B_3 \cup P_{s_4}$ est un arbre couvant $x_1$, $x_2$, $x_3$ et $x_4$.

Cela prouve (3). Nous supposons désormais que $w$ a au moins un voisin dans $S_2 \cup S_4$, par exemple $s_2 \in S_2$ et nous avons :

*(4) Si $w$ n'a aucun voisin dans $A_3$ alors, soit il existe un arbre induit couvrant les quatre terminaux, soit $N_S(w) = S_2 \cup S_4$ et $N_A(w) \subset A_1$.*

Si $w$ n'a aucun voisin dans $A_3$, notons alors $s_3$ un sommet de $S_3$. Admet-

tons qu'il existe un sommet $s_4$ de $S_4$ non voisin de $w$ : le graphe induit par $P_{a_1} \cup Q \cup P_{s_2} \cup P_{s_3} \cup P_{s_4}$ est alors un arbre contenant les quatre terminaux. Supposons désormais que $w$ est complet à $S_4$. Par un raisonnement similaire, nous pouvons supposer que $w$ est complet à $S_2$ car sinon un arbre induit couvrant les quatre terminaux existe. Puisque le graphe ne contient aucun triangle, on a nécessairement $N_S(w) = S_2 \cup S_4$ et $N_A(w) \subset A_1$ ce qui prouve (4).

Nous supposons dorénavant que $w$ a un voisin $a_3$ dans $A_3$ et nous choisissons $a_3$ de telle sorte que la longueur de $P_{a_3}$ soit minimum. De plus, puisque $w$ a au moins un voisin dans $A \setminus A_1$, on a $Q = v = w$ et nous prouvons :

*(5) Si $v$ a un voisin dans $S_4$ alors il existe un arbre induit couvrant $x_1$, $x_2$, $x_3$ et $x_4$.*

Si $s_4$ est un sommet de $S_4$ voisin de $v$ alors le graphe induit par $P_{a_1} \cup \{v\} \cup P_{s_2} \cup P_{a_3} \cup P_{s_4}$ est un arbre contenant les quatre terminaux.

Cela prouve (5). Nous supposons désormais qu'il existe un sommet $s_4$ de $S_4$ non voisin de $v$ et nous démontrons la dernière des 6 propriétés :

*(6) Il existe un arbre induit couvrant les quatre terminaux ou il existe un cube de $G[Z \cup \{v\}]$.*

Remarquons tout d'abord qu'une fois le point (6) démontré, l'assertion 7.1 sera finalement prouvée. Soit $s_1 \in S_1$ et $s_3 \in S_3$. Les arêtes $v - s_1$ et $v - s_3$ ne peuvent exister car $G$ est sans triangle et $v$ a un voisin dans $S_2$. Si l'arête $a_1 - s_1$ n'existe pas alors le graphe induit par $P_{a_1} \cup \{v\} \cup P_{s_2} \cup P_{a_3} \cup P_{s_4} \cup \{s_1\}$ est solution de l'instance de $4 - \text{ARBREI}$ considérée. Nous supposons donc que $a_1 - s_1$ existe et que, par un raisonnement similaire, c'est également le cas pour l'arête $a_3 - s_3$ (voir la figure 7.9 pour un exemple d'une telle situation).

Nous remarquons alors que le graphe induit par $P_{a_1} \cup P_{s_2} \cup P_{a_3} \cup P_{s_4} \cup \{s_1, s_3, v\}$ est un cube et qu'une division de celui-ci est donnée par : $A_1 = P_{a_1} \setminus \{a_1\}$, $A_2 = P_{s_2} \setminus \{s_2\}$, $A_3 = P_{a_3} \setminus \{a_3\}$, $A_4 = P_{s_4} \setminus \{s_4\}$, $S_1 = \{a_1\}$, $S_2 = \{s_2\}$, $S_3 = \{a_3\}$, $S_4 = \{s_4\}$, $S_5 = \{s_3\}$, $S_6 = \emptyset$, $S_7 = \{s_1\}$, $S_8 = \{v\}$, $B = \emptyset$ et $R = \emptyset$. □

Pour terminer la preuve du lemme, il reste deux cas à examiner. Le premier, issu du deuxième point de l'assertion précédente, correspond à la situation suivante : toute chaîne induite $Q = v - ... - w$ minimale au sens de l'inclusion, telle que $Q \subset R \cup \{v\}$ et telle que $w$ possède au moins un voisin dans $(A \setminus A_1) \cup (S \setminus S_1)$, vérifient $N_S(w) = S_2 \cup S_4$ et $N_A(w) \subset A_1$.

Le second cas intervient lorsque toute chaîne induite $Q = v - ... - w$ telle que $Q \subset R \cup \{v\}$ vérifie $N_{A \cup S}(w) \subset A_1 \cup S_1$. Une telle situation advient si tous les voisins de $v$ sont dans $A_1 \cup S_1$ ou si $v$ possède des voisins dans $R$ à partir desquels on ne peut "atteindre" aucun sommet de $S \setminus S_1$.

Nous traitons simultanément ces deux cas de la manière suivante (voir la

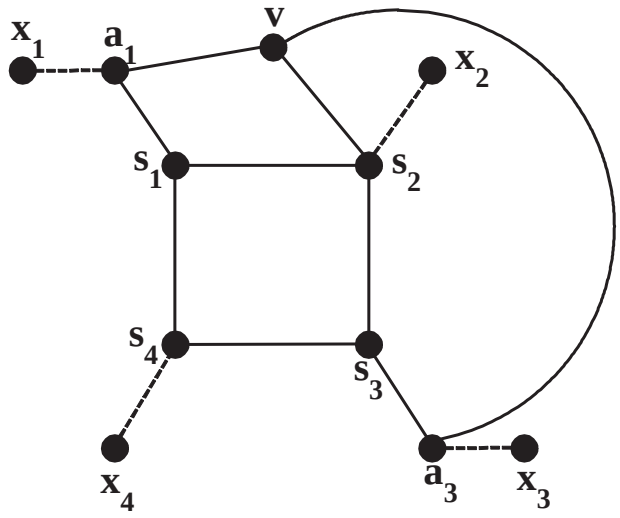


FIGURE 7.9 – Obtention d'un cube à partir de $Z$ et de $v$ (les pointillés représentent des chaînes)

figure 7.10). Soit $C$ l'ensemble (éventuellement vide) des sommets de $R \cup \{v\}$ complets à $S_2 \cup S_4$ et soit $Y$ l'ensemble des sommets $w$ de $R \cup \{v\}$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset R$. Soit $Y_1$ l'ensemble des sommets $w$ de $Y \setminus C$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset R \setminus C$. $Y_1$ est donc vide si et seulement si $v \in C$. Soit $Y_2$ l'ensemble (éventuellement vide) des sommets $w$ de $Y \cap C$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v, w\} \subset R \setminus C$. Enfin, soit $Y_3 = R \setminus (Y_1 \cup Y_2)$ ($Y_3$ pouvant également être vide). Remarquons qu'on ne peut avoir $v \in Y_3$ (car $v \in Y_1 \cup Y_2$) mais que $v \in Y_2$ est envisageable si $v$ est complet à $S_2 \cup S_4$. De plus, tout sommet de $Y_2 \setminus \{v\}$ possède nécessairement au moins un voisin dans $Y_1$ et $G[Y_1]$ est toujours connexe. D'où, $N_{Z \cup \{v\}}(Y_3) \subset Y_2 \cup S$, $N_{Z \cup \{v\}}(Y_2) \subset Y_1 \cup Y_3 \cup A_1 \cup S_2 \cup S_4$ et $N_{Z \cup \{v\}}(Y_1) \subset Y_2 \cup A_1 \cup S_1$.

De ce fait, en mettant tous les sommets de $Y_1$ dans $A_1$, tous les sommets de $Y_2$ dans $S_1$ et en laissant tous les sommets de $Y_3$ dans $R$, nous obtenons une division du carré $G[Z \cup \{v\}]$ (voir la figure 7.10).

Terminons cette section en expliquant comment transformer cette preuve en un algorithme. Tout d'abord, nous déterminons l'ensemble $C$ puis nous utilisons un parcours en profondeur à partir du sommet $v$ afin de déterminer l'ensemble $Y$ et d'obtenir une arborescence $A$ de racine $v$. De manière similaire, nous obtenons les ensembles $Y_1$, $Y_2$ et $Y_3$. Nous vérifions alors si $N_{Z \cup \{v\}}(Y_1) \subset Y_2 \cup A_1 \cup S_1$. Si tel est le cas, nous avons montré ci-dessus comment obtenir une division du carré $G[Z \cup \{v\}]$ sinon il existe un sommet $w$ de $Y_1$ qui a au moins un voisin dans $(A \setminus A_1) \cup (S \setminus S_1)$. En remontant $A$ de $w$ à $v$, nous déterminons le sommet $w'$ ayant au moins un voisin dans $(A \setminus A_1) \cup (S \setminus S_1)$ et telle que la chaîne $Q = v - \cdots - w'$ soit induite, minimale au sens de l'inclusion et vérifient $Q \setminus \{v\} \subset R$. Puisque $w' \in Y_1$, on a $N_S(w') \neq S_2 \cup S_4$ et en utilisant la vérification de certains voisinages

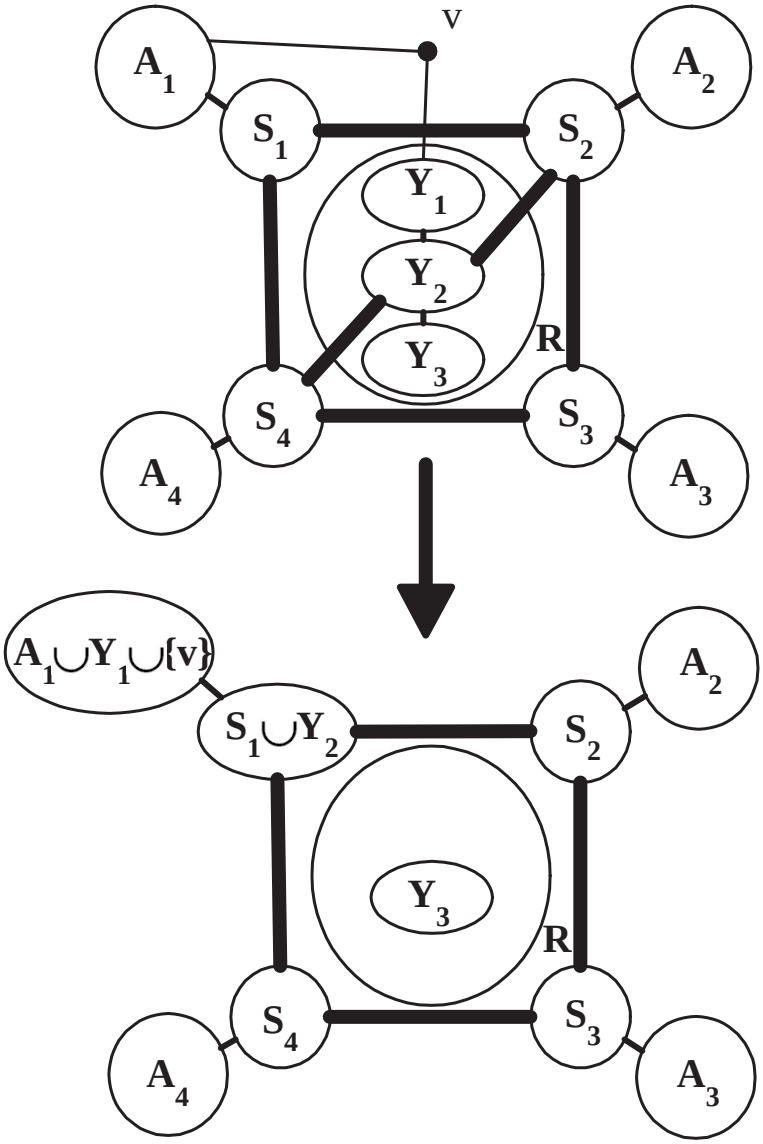


FIGURE 7.10 – Déplacement des ensembles $Y_1$ et $Y_2$ pour obtenir un nouveau carré (les sommets de $Y_2$ sont complets à $S_2 \cup S_4$ et les sommets de $Y_3$ peuvent être connectés à des sommets de $Y_2$ mais pas à des sommets de $Y_1$)

de sommets, la preuve ci-dessus décrit la manière d'obtenir soit :
- un arbre induit couvrant les quatre terminaux
- une division d'un cube de $G[Z \cup \{v\}]$

## 7.4 Ajout d'un sommet au cube

Dans cette section, nous apportons la preuve du lemme 7.2 (page 115). Cette démonstration présente de nombreuses similarités avec celle de la section précédente, tant au niveau des notations que dans la construction ou même dans l'utilisation d'une chaîne adéquate $Q$.

Soit $Z$ un cube de $G$ et soit $A_1, \ldots, A_4, B_1, \ldots, B_4, S_1, \ldots, S_8, R$ une division de ce cube. Enfin, soit $v$ un sommet de $G$ n'appartenant pas à $Z$ mais possédant au moins un voisin dans $Z$.

Nous allons prouver qu'après application de l'algorithme du lemme 7.2, nous obtenons soit un arbre induit contenant les quatre terminaux, soit une division du cube $G[Z \cup \{v\}]$. De manière similaire à la section précédente, cette preuve peut être transformée afin d'obtenir la description de l'algorithme en lui-même. Nous omettons les détails de cette transformation analogue à celle utilisée pour le lemme 7.1.

De façon identique à la partie précédente, pour $i \in \{1, 2, 3, 4\}$, si $s_i$ (resp. $a_i$) appartient à $S_i \cup A_i$ alors $P_{s_i}$ (resp. $P_{a_i}$) représente une plus courte chaîne reliant $s_i$ (resp. $a_i$) à $x_i$ dont tous les sommets, excepté $s_i$ (resp. excepté $a_i$), sont dans $A_i$.

Tout d'abord, nous montrons deux assertions intermédiaires avant de prouver le lemme dans son ensemble.

**Assertion 7.2.** *Le lemme est vérifié si $v$ a au moins un voisin dans $A$.*

*Démonstration.* Supposons que $v$ a au moins un voisin dans $A$, par exemple $a_1$ dans $A_1$ et que ce sommet ait été choisi de telle sorte que $P_{a_1}$ soit de longueur minimum. Nous montrons que :

*(7) S'il existe une chaîne induite $Q = v - \cdots - w$ minimale au sens de l'inclusion, telle que $Q \setminus \{v\} \subset B \cup R$ et telle que $w$ ait des voisins dans $A_2 \cup A_3 \cup A_4 \cup S_2 \cup S_3 \cup S_4 \cup S_5$ ; alors il existe un arbre induit couvrant $x_1$, $x_2$, $x_3$ et $x_4$.*

Soit $Q$ une telle chaîne. Notons qu'il est possible d'avoir $Q = v = w$.

Si $w$ est voisin à la fois d'un sommet $a_2$ de $S_2 \cup A_2$, d'un sommet $a_3$ de $S_3 \cup A_3$ et d'un sommet $a_4$ de $S_4 \cup A_4$ alors nous choisissons $a_2, a_3$ et $a_4$ de telle sorte que $P_{a_2}$, $P_{a_3}$ et $P_{a_4}$ soient de longueur minimum. Dans ce cas, le graphe induit par $P_{a_1} \cup Q \cup P_{a_2} \cup P_{a_3} \cup P_{a_4}$ est un arbre et contient les quatre terminaux. Nous supposons désormais que $w$ n'a aucun voisin dans $S_4 \cup A_4$.

Si $w$ est voisin à la fois d'un sommet $a_2$ de $S_2 \cup A_2$ et d'un sommet $a_3$ de $S_3 \cup A_3$ alors $Q = v = w$ car aucun sommet de $B \cup R$ ne peut avoir de voisins à la

fois dans $S_2 \cup A_2$ et dans $S_3 \cup A_3$, d'après les points 10 et 11 de la définition 7.4 (page 114). Les sommets $a_2$ et $a_3$ sont choisis de manière à minimiser la longueur des chaînes $P_{a_2}$ et $P_{a_3}$, et soit $s_2$ un sommet de $S_2$ (si $a_2 \in S_2$ alors nous posons $s_2 = a_2$), $s_4$ un sommet de $S_4$ et $s_7$ un sommet de $S_7$. Soit $D_2$ un arbre induit contenu dans $G[A_2 \cup \{v, s_2\}]$ et couvrant $x_2$, $s_2$ et $v$. $D_2$ existe nécessairement d'après le corollaire 6.4 (page 106) puisque $G[A_2 \cup \{v, s_2\}]$ est connexe. D'après le point 4, au plus un des ensembles $S_6$ et $S_7$ est vide et nous supposons alors que $S_7 \neq \emptyset$ du fait de la symétrie entre $S_2, S_7$ d'une part et $S_3, S_6$ d'autre part. Puisque les arêtes $s_2 - s_7$ et $s_7 - s_4$ existent (d'après le point 5 de la définition 7.4), le graphe induit par $P_{a_1} \cup D_2 \cup P_{a_3} \cup P_{s_4} \cup \{s_7\}$ est un arbre contenant $x_1, x_2, x_3, x_4$ sauf si l'arête $v - s_7$ existe. Dans ce cas, $a_2 \notin S_2$ car sinon les sommets $a_2, s_7, w$ induiraient un triangle. Le graphe induit par $P_{a_1} \cup P_{a_2} \cup P_{a_3} \cup P_{s_4} \cup \{s_7\}$ est alors un arbre couvrant les quatre terminaux. Dorénavant, nous supposons que $w$ n'admet aucun voisin dans $S_3 \cup A_3$.

Si $w$ est voisin d'un sommet $a_2$ de $S_2 \cup A_2$ alors $a_2$ est sélectionné de manière à minimiser la longueur de $P_{s_2}$. Tout d'abord, admettons qu'un des sommets de $Q$ ait un voisin $s_6$ dans $S_6$, et soit $s_3$ un sommet de $S_3$ et $s_4$ un sommet de $S_4$. Le graphe $G[A_1 \cup Q \cup S_2 \cup A_2 \cup \{s_6\}]$ est alors connexe et contient donc nécessairement un arbre induit $D_6$ couvrant $x_1, x_2, s_6$ puisque le graphe est sans triangle (voir le corollaire 6.4). Le graphe induit par $D_6 \cup P_{s_3} \cup P_{s_4}$ est donc un arbre contenant les quatre terminaux car les arêtes $s_3 - s_6$ et $s_4 - s_6$ existent (d'après le point 5 de la définition 7.4). Nous supposons maintenant qu'aucun sommet de $Q$ n'a de voisin dans $S_6$, et soit $s_2 \in S_2$ (si $a_2 \in S_2$ alors nous posons $s_2 = a_2$), $s_3 \in S_3$ et $s_4 \in S_4$. Soit $D_2$ un arbre induit sous-graphe de $G[A_2 \cup \{s_2, w\}]$ et couvrant $x_2, w, s_2$.

- $S_5 \neq \emptyset$ : soit $s_5$ un sommet de $S_5$. Puisque les arêtes $s_3 - s_5$ et $s_4 - s_5$ existent, le graphe induit par $P_{a_1} \cup Q \cup D_2 \cup P_{s_3} \cup P_{s_4} \cup \{s_5\}$ est un arbre contenant les quatre terminaux à moins que l'arête $w - s_5$ existe. Dans ce cas, $a_2 \in A_2$ - car si $a_2 \in S_2$, $a_2$,$s_5$ et $w$ induisent un triangle - et le graphe induit par $P_{a_1} \cup Q \cup P_{a_2} \cup P_{s_3} \cup P_{s_4} \cup \{s_5\}$ est un arbre couvrant $x_1, x_2, x_3, x_4$.
- $S_5 = \emptyset$ : d'après le point 4 de la définition 7.4, les ensembles $S_6$, $S_7$ et $S_8$ sont non vides. Si aucun sommet de $Q$ n'a de voisins dans $S_7 \cup S_8$ alors le graphe induit par $P_{a_1} \cup Q \cup D_2 \cup P_{s_3} \cup P_{s_4} \cup \{s_7, s_8\}$ est un arbre contenant les quatre terminaux avec $s_7 \in S_7$ et $s_8 \in S_8$ car les arêtes $s_2 - s_7$, $s_7 - s_4$, $s_2 - s_8$ et $s_8 - s_3$ existent. Sinon, soit $u \in Q$ le sommet le plus proche de $v$ ayant un voisin dans $S_7 \cup S_8$, par exemple $s_7 \in S_7$ (le cas avec un voisin dans $S_8$ est similaire par symétrie entre $S_7, S_4$ et $S_8, S_3$). Puisqu'aucun sommet de $Q$ n'a de voisin dans $S_6$ et que les arêtes $s_2 - s_7$, $s_7 - s_4$, $s_4 - s_6$, $s_6 - s_3$ existent, le graphe induit par $P_{a_1} \cup (v - Q - u) \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup \{s_6, s_7\}$ est un arbre couvrant $x_1, x_2, x_3, x_4$ sauf s'il existe un sommet de $v - Q - u$ ayant au moins un voisin dans $P_{s_2}$. Dans ce cas, par minimalité de $Q$, $w$ est ce sommet et $u = w$. De plus, comme $G$ ne contient pas de triangle et que les arêtes $w - s_7$ et $w - a_2$ existent, $a_2 \notin S_2$. Le graphe induit par $P_{a_1} \cup Q \cup P_{a_2} \cup P_{s_3} \cup P_{s_4} \cup \{s_6, s_7\}$ est alors un arbre contenant les quatre terminaux car les arêtes $s_7 - s_4$, $s_4 - s_6$, $s_6 - s_3$ existent.

Nous supposons désormais que $w$ n'a pas de voisin dans $S_2 \cup A_2$.

Puisque $w$ n'a pas de voisin dans $S_2 \cup S_3 \cup S_4 \cup A_2 \cup A_3 \cup A_4$, il a au moins un voisin $s_5$ dans $S_5$. Soit $s_2 \in S_2$, $s_3 \in S_3$ et $s_4 \in S_4$. Les arêtes $s_2 - s_5$, $s_3 - s_5$ et $s_4 - s_5$ existants, le graphe induit par $P_{a_1} \cup Q \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup \{s_5\}$ est un arbre couvrant $x_1, x_2, x_3, x_4$, ce qui conclut la preuve de (7).

Nous montrons à présent :

*(8) Soit une chaîne induite $Q = v - \cdots - w$ minimale au sens de l'induction telle que $Q \setminus \{v\} \subset B \cup R$ et telle que $w$ ait au moins un voisin dans $S_6 \cup S_7 \cup S_8$. On est alors dans l'un des deux cas suivants :*

- *$w$ est complet à $S_6 \cup S_7 \cup S_8$*
- *il existe un arbre induit contenant $x_1, x_2, x_3, x_4$*

Soit $Q$ une telle chaîne et par exemple $s_7 \in S_7$ un voisin de $w$. Il nous suffit de montrer que $w$ est complet à $S_6 \cup S_7 \cup S_8$ ou qu'il existe un arbre induit couvrant tous les terminaux. En utilisant (7), nous supposons qu'aucun sommet de $Q$ n'a de voisin dans $A_2 \cup A_3 \cup A_4 \cup S_2 \cup S_3 \cup S_4 \cup S_5$. Si $w$ n'est pas complet à $S_6 \cup S_7 \cup S_8$ alors il existe par exemple un sommet $s_6 \in S_6$ non voisin de $w$ (si tous les non voisins de $w$ sont dans $S_7$ alors nous choisissons $s_6 \in S_6$ voisin de $w$ et $s_7 \in S_7$ non voisin de $w$). Dans ce cas, le graphe induit par $P_{a_1} \cup Q \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup \{s_6, s_7\}$ est un arbre contenant les quatre terminaux car les arêtes $s_2 - s_7$, $s_3 - s_6$, $s_4 - s_6$ et $s_4 - s_7$ existent. (8) est ainsi prouvé.

Nous pouvons désormais conclure la preuve de l'assertion 7.2. Le seul cas restant est celui où toute chaîne induite $Q = v - \ldots - w$ minimale au sens de l'inclusion vérifiant $Q \setminus \{v\} \subset B \cup R$ et telle que $w$ possède au moins un voisin dans $S_6 \cup S_7 \cup S_8$, vérifient $N_S(w) = S_6 \cup S_7 \cup S_8$. Rappelons que d'après les points 10 et 11 de la définition 7.4, les seuls sommets de $Z$ pouvant avoir à la fois un voisin dans $S_6$, un voisin dans $S_7$ et un voisin dans $S_8$, sont ceux de $B_1$ et de $R$. De plus, ces deux ensembles sont disjoints.

Soit $C$ l'ensemble des sommets de $G[Z \cup \{v\}]$ complets à $S_6 \cup S_7 \cup S_8$. Soit $Y$ l'ensemble des sommets $w$ de $B_1 \cup R \cup \{v\}$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset B_1 \cup R$. Soit $Y_1$ l'ensemble des sommets $w$ de $Y \setminus C$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset (B_1 \cup R) \setminus C$. Soit $Y_2$ l'ensemble des sommets $w$ de $Y \cap C$ tel qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v, w\} \subset (B_1 \cup R) \setminus C$.

En utilisant (7), nous supposons qu'aucun sommet de $Y$ n'a de voisin dans $A_2 \cup A_3 \cup A_4 \cup S_2 \cup S_3 \cup S_4 \cup S_5$ (car sinon il existe un arbre induit couvrant les terminaux). $v \in Y_1 \cup Y_2$ et $v \in Y_2$ est envisageable puisque $v$ peut être complet à $S_6 \cup S_7 \cup S_8$. De plus, en utilisant (8), on peut supposer que $N_{Z \cup \{v\}}(Y_2) \subset S_6 \cup S_7 \cup S_8 \cup A_1$ et que $N_{Z \cup \{v\}}(Y_1) \subset Y_2 \cup A_1 \cup S_1$.

Ainsi, en mettant tous les sommets de $Y_1$ dans $A_1$ et tous les sommets de $Y_2$ dans $S_1$, nous obtenons une division du cube $G[Z \cup \{v\}]$ et ceci conclut la preuve de l'assertion. □

Avant de conclure la preuve du lemme, nous montrons une dernière assertion :

**Assertion 7.3.** *Le lemme est vérifié si $v$ est complet à $(S_1 \cup S_2 \cup S_3 \cup S_4) \setminus S_i$ ($i = 1, 2, 3$ ou $4$).*

*Démonstration.* Nous réalisons la preuve pour $i = 4$, les autres cas étant symétriques. $v$ est ainsi complet à $S_1 \cup S_2 \cup S_3$. De plus, d'après l'assertion 7.2, nous pouvons supposer que $v$ n'a aucun voisin dans $A$. Montrons tout d'abord que :

*(8) S'il existe une chaîne induite $Q = v - \cdots - w$ minimale au sens de l'induction vérifiant $Q \setminus \{v\} \subset B \cup R$ et telle que $w$ ait un voisin $s_4$ dans $S_4$, alors $G[Z \cup \{v\}]$ contient un arbre induit couvrant les quatre terminaux.*

Soit $Q$ une telle chaîne et soit $s_1 \in S_1$, $s_2 \in S_2$ et $s_3 \in S_3$. Tous les sommets de $Q \setminus \{v\}$ sont dans $B_4$ car cet ensemble est le seul parmi $B_1, \ldots, B_4, R$ à pouvoir avoir des voisins dans $S_4$ et les ensembles $B_1, \ldots, B_4, R$ sont disjoints deux à deux. D'où, par définition de $B_4$, aucun sommet de $Q \setminus \{v\}$ ne peut avoir des voisins dans $S_1 \cup S_2 \cup S_3$. Le graphe induit par $P_{s_4} \cup Q \cup P_{s_1} \cup P_{s_2} \cup P_{s_3}$ est alors un arbre contenant $x_1, x_2, x_3, x_4$ ; ce qui prouve (8).

Terminons la preuve de l'assertion 7.3. Soit $Y$ l'ensemble des sommets $w$ de $B \cup R$ tels qu'il existe une chaîne $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset B \cup R$. Si aucun sommet de $Y$ n'est dans $B_4$, nous plaçons $v$ dans $S_8$ et nous obtenons alors une division du cube $G[Z \cup \{v\}]$ car $v$ est complet à $S_1 \cup S_2 \cup S_3$. Cela ne fonctionne plus aussi immédiatement si certains sommets de $Y$ sont dans $B_4$ car un sommet de $S_8$ ne peut avoir de voisins dans $B_4$. Mais, d'après (8), nous pouvons supposer qu'aucun sommet de $Y$ n'a de voisin dans $S_4$ car sinon un arbre induit couvrant les quatre terminaux existe. Nous déplaçons alors les sommets de $Y \cap B_4$ dans $R$, mettons $v$ dans $S_8$ et nous obtenons finalement une division du carré $G[Z \cup \{v\}]$. □

Nous pouvons finaliser la démonstration du lemme en utilisant les deux assertions prouvées :

**Assertion 7.4.** *Le lemme 7.2 est vrai.*

*Démonstration.* D'après l'assertion 7.2, nous pouvons supposer que $v$ n'a pas de voisin dans $A$. De plus, soit deux entiers $i$ et $j$ tels que $1 \leq i < j \leq 4$ ou tels que $(i, j) \in \{(1, 5), (2, 6), (3, 7), (4, 8)\}$. Alors, d'après les points 10 et 11 de la définition 7.4, ni $R$, ni aucun des $B_i$ ne peuvent posséder à la fois un sommet ayant au moins un voisin dans $S_i$ et un sommet ayant au moins un voisin dans $S_j$. Nous montrons alors :

*(9) Soit deux entiers $i$ et $j$ tels que $1 \leq i < j \leq 4$ ou tels que $(i, j) \in \{(1, 5), (2, 6), (3, 7), (4, 8)\}$. S'il existe dans $G[B \cup R \cup \{v\}]$ une chaîne induite $Q = u - \cdots - w$ minimale au sens de l'inclusion et telle que $u$ ait un voisin dans $S_i$ et $w$ ait un voisin dans $S_j$, alors le lemme se vérifie.*

Soit $i$, $j$ et $Q$ comme définis ci-dessus. Remarquons que d'après les points 10 et 11 de la définition 7.4, on ne peut avoir $Q \subset B \cup R$ car ni $R$, ni aucun

des $B_i$ ne peuvent avoir avoir à la fois un voisin dans $S_i$ et un voisin dans $S_j$. $Q$ contient donc nécessairement $v$.

Si $u$ est voisin à la fois de $s_1 \in S_1$, de $s_2 \in S_2$, de $s_3 \in S_3$ et de $s_4 \in S_4$ alors $Q = u = v = w$ puisque $Q$ est minimale et que tout sommet de $B \cup R$ ne peut avoir à la fois un voisin dans $S_1$ et un voisin dans $S_2$. Le graphe induit par $P_{s_1} \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup \{v\}$ est alors un arbre couvant $x_1, x_2, x_3, x_4$. Nous supposons désormais que $u$ (et de manière symétrique $w$) n'a pas de voisin dans $S_4$.

Si $u$ est voisin à la fois de $s_1 \in S_1$, de $s_2 \in S_2$ et de $s_3 \in S_3$ alors, comme dans le cas précédent, $Q = u = v = w$. De plus, si $v$ est complet à $S_1 \cup S_2 \cup S_3$ alors d'après l'assertion 7.3, le lemme est vrai. Nous pouvons donc supposer ici qu'il existe un sommet $s'_1 \in S_1$ non voisin de $v$. D'après le point 4 de la définition 7.4, $S_6 \cup S_7 \neq \emptyset$ et on peut supposer qu'il existe $s_6 \in S_6$ par symétrie entre $S_2, S_7$ et $S_3, S_6$. L'arête $v - s_6$ ne peut exister puisque le graphe est sans triangle et que les arêtes $v - s_1$ et $s_1 - s_6$ existent. Le graphe induit par $P_{s'_1} \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup \{v\} \cup \{s_6\}$ est donc un arbre contenant tous les terminaux car les arêtes $s'_1 - s_6$ et $s_6 - s_4$ sont présentes. Désormais, $u$ (et symétriquement $w$) n'a pas de voisin dans $S_3$.

Si $i$ et $j$ vérifient $1 \leq i < j \leq 4$ alors par symétrie nous pouvons supposer que $i = 1$ et $j = 2$ et soit alors $s_1 \in S_1$ voisin de $u$, $s_2 \in S_2$ voisin de $w$. Soit également $s \in S_5 \cup S_6$ ($s$ existe forcément d'après le point 4 de la définition 7.4), $s_3 \in S_3$ et $s_4 \in S_4$. Aucun sommet de $Q$ ne peut avoir de voisin dans $S_5 \cup S_6$ car un tel sommet violerait la minimalité de $Q$ ou induirait un triangle ($s, s_2, w$ si $s \in S_5$ et $s, s_1, u$ si $s \in S_6$). De plus, d'après les deux paragraphes précédents, $u$ et $w$ ne possèdent aucun voisin dans $S_3 \cup S_4$ et il en est de même pour les autres sommets de $Q$ en raison de sa minimalité. D'où, le graphe induit par $P_{s_1} \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup Q \cup \{s\}$ est un arbre et contient $x_1, x_2, x_3, x_4$.

Si $(i, j) = (1, 5)$ (les cas $(2, 6)$,$(3, 7)$ et $(4, 8)$ étant similaires) notons $s_1 \in S_1$ un voisin de $u$ et $s_5 \in S_5$ un voisin de $w$. On peut supposer qu'aucun sommet de la sous-chaîne $v - \cdots - w$ de $Q$ n'a de voisin dans $S_2 \cup S_3 \cup S_4$ car cela correspond a un cas déjà étudié précédemment où $1 \leq i < j \leq 4$. De plus, $w$ est le seul sommet de cette sous chaîne ayant des voisins dans $S_5$ car sinon, cela viole la minimalité de $Q$. De même, excepté $v$, aucun sommet de la sous-chaîne $Q' = u - \cdots - v$ de $Q$ n'a de voisin dans $S_2 \cup S_3 \cup S_4 \cup S_5$ car $Q' \setminus \{v\} \subset B_1$ et $B_1$ n'a aucun voisin dans $S_2 \cup S_3 \cup S_4 \cup S_5$ (voir le point 10 de la définition 7.4). D'où, aucun sommet de $Q$ n'a de voisin dans $S_2 \cup S_4 \cup S_5$ et $w$ est le seul sommet de $Q$ ayant au moins un voisin dans $S_5$. Le graphe induit par $P_{s_1} \cup P_{s_2} \cup P_{s_3} \cup P_{s_4} \cup Q \cup \{s_5\}$ est alors un arbre couvrant les quatre terminaux. *(9) est donc finalement démontré.*

Pour terminer la preuve de l'assertion (et donc du lemme), il reste plusieurs cas à traiter. Soit $Y$ l'ensemble des sommets $w$ de $B \cup R \cup \{v\}$ tels qu'il existe une chaîne induite (pas forcément minimale) $Q$ de $v$ à $w$ avec $Q \setminus \{v\} \subset B \cup R$. Dans l'assertion 7.2, nous avons traité le cas où $N(Y) \cap A \neq \emptyset$. Dans (9), nous avons étudié les cas où $N(Y) \cap S_i \neq \emptyset$ et $N(Y) \cap S_j \neq \emptyset$ avec $i$ et $j$ tels que $1 \leq i < j \leq 4$ ou $(i, j) \in \{(1, 5), (2, 6), (3, 7), (4, 8)\}$. Il reste donc les cas où :

1. $N(Y) \subset (S_1 \cup S_6 \cup S_7 \cup S_8)$ et $N(Y) \cap S_1 \neq \emptyset$
2. $N(Y) \subset (S_2 \cup S_5 \cup S_7 \cup S_8)$ et $N(Y) \cap S_2 \neq \emptyset$

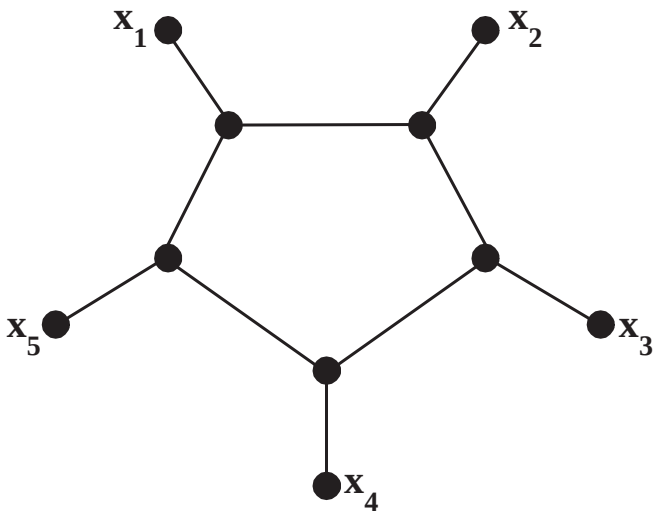


FIGURE 7.11 – Une structure en "pentagone"

3. $N(Y) \subset (S_3 \cup S_5 \cup S_6 \cup S_8)$ et $N(Y) \cap S_3 \neq \emptyset$
4. $N(Y) \subset (S_4 \cup S_5 \cup S_6 \cup S_7)$ et $N(Y) \cap S_4 \neq \emptyset$
5. $N(Y) \subset (S_5 \cup S_6 \cup S_7 \cup S_8)$

Notons qu'on ne peut avoir $N(Y) \subset B_i$ ($i = 1, 2, 3$ ou 4) ou $N(Y) \subset R$ car cela voudrait dire que $G[Z]$ n'est pas connexe ce qui est contradictoire avec le point 12 de la définition 7.4. Pour le $i$-ème cas ($i = 1, 2, 3$ ou 4), nous mettons $Y$ dans $B_i$ et nous obtenons une division du cube $G[Z \cup \{v\}]$. Pour le cinquième cas, nous plaçons $v$ dans $R$ ce qui nous donne une division du cube $G[Z \cup \{v\}]$. □

## 7.5 Quelques remarques sur 3−ArbreI et 5−ArbreI

Dans ce chapitre, nous avons montré que les seuls graphes sans triangle pour lesquels 4 − ArbreI n'admet pas de solution sont nécessairement des *carrés* ou des *cubes*. Nous nous sommes également interrogés sur la possibilité de généraliser une telle approche par exemple au cas de 5 − ArbreI dans les graphes sans triangle.

L'idée serait de déterminer de nouvelles structures pour lesquelles 4−ArbreI admet toujours une solution alors que 5 − ArbreI n'en admet aucune. A titre d'exemple, il n'existe pas de solution à l'instance de la figure 7.11 où nous considérons un pentagone et cinq terminaux connectés chacun à un des sommets du pentagone. Cependant, pour tout quadruplet de terminaux, il existe un arbre induit les contenant. Ce type de graphe peut alors être considéré comme le pendant du carré lorsque nous passons de 4 à 5 terminaux.

Considérons maintenant l'instance de la figure 7.12. Il n'existe aucun arbre induit couvrant les cinq terminaux alors que pour tout quadruplet de terminaux,

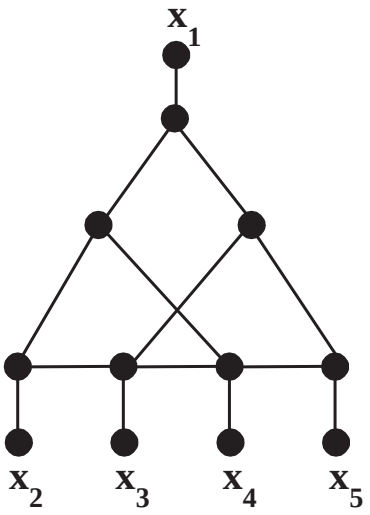


FIGURE 7.12 – Une instance de $5 - \text{ARBREI}$ n'admettant pas de solution

on peut déterminer un arbre induit les contenant. Nous nous apercevons que ce graphe, pourtant de taille modérée, semble n'avoir que peu de liens avec le type de structure élaborée pour $4 - \text{ARBREI}$. Il ne permet pas, semble-t-il, de dégager de structure plus générale pour $5 - \text{ARBREI}$ dans les graphes sans triangle.

Nous nous sommes également intéressés à $3-\text{ARBREI}$ dans le cas général (rappelons que pour les graphes sans triangle, nous avons montré dans le corollaire 6.4 page 106 qu'il existe toujours une solution). Nous savons qu'un exemple de structure "bloquante" consiste en un triangle dont chaque sommet est connecté à un terminal. Malheureusement, nous n'avons pas réussi à déterminer de manière exhaustive une famille de structures pour lesquelles $3 - \text{ARBREI}$ n'admet pas de solution et nous ne savons même pas si une telle famille existe. A titre d'exemple, la figure 7.13 présente une instance assez complexe pour laquelle il n'existe aucune solution. Cependant, dès lors que l'on supprime une seule arête (tout en conservant le graphe connexe), on peut déterminer un arbre induit couvrant les trois terminaux.

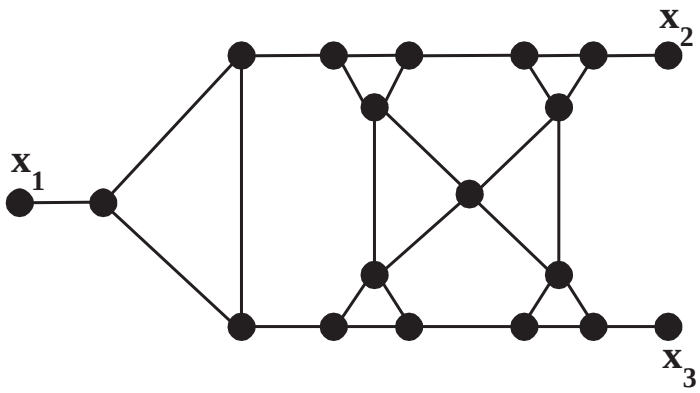


FIGURE 7.13 – Une instance de 3 – ARBREI n'admettant pas de solution

# Chapitre 8

# Conclusion

Dans cette seconde partie du mémoire, nous nous sommes focalisés sur des problèmes de recherche de sous-graphes induits particuliers tels que les arbres, les cycles et les chaînes. Nous nous sommes principalement attachés aux aspects de complexité et d'algorithmique associés à ces problèmes. Du fait du peu de résultats antérieurs à nos travaux les concernant, nous avons tenté d'apporter une vision globale de la complexité de ces problèmes. Nous avons ainsi abordé les problèmes d'existence mais également les problèmes de recherche d'un arbre, d'une chaîne ou d'un cycle induit de taille ou de poids minimum.

Le chapitre 6 contient un ensemble de résultats pour ces problèmes que avons synthétisés à l'intérieur du tableau 8. Nous avons ainsi regroupé l'ensemble des cas étudiés pour chacun des problèmes en précisant à chaque fois si l'on est confronté à un cas $\mathcal{NP}$-complet ou à un cas polynomial et en indiquant, dans ce cas, la complexité de l'algorithme élaboré. Nous en résumons ici les principaux résultats.

Nous nous sommes tout d'abord attaqués à la complexité des problèmes dans le cas général. Pour le cas des graphes pondérés, nous avons montré que $1-$ PondCycleI, $2-$ PondChaineI et $2-$ PondArbreI sont $\mathcal{NP}$-complets au sens fort dans les graphes bipartis cubiques, et même qu'il ne peut exister un algorithme approché à garantie de performance relative pour ces problèmes (à moins que $\mathcal{P} = \mathcal{NP}$). Dans un second temps, nous avons également prouvé la $\mathcal{NP}$-complétude de $k-$ CycleI, $k-$ ArbreI et $k-$ ChaineI dans les graphes bipartis cubiques planaires. Les cas généraux étant tous $\mathcal{NP}$-complets, nous avons alors considéré les problèmes non pondérés pour un nombre de terminaux fixé.

Même s'il n'existe qu'une littérature peu conséquente traitant de ces problèmes, il semble indispensable de rappeler deux résultats importants antérieurs à nos travaux : la polynomialité de $3-$ ArbreI ([16]) et la $\mathcal{NP}$-complétude de $2-$ CycleI ([8]). Ainsi, si nous avons montré que pour un seul terminal tous les problèmes sont polynomiaux, ce second résultat indique que ce n'est plus le cas dès lors que nous considérons 2 terminaux. Nous avons ainsi montré que $2-$ CycleI reste $\mathcal{NP}$-complet pour les graphes bipartis de degré maximum 4, alors que $2-$ ChaineI et $2-$ ArbreI (même sous la forme de problèmes d'optimisation) demeurent polynomiaux. Lorsque nous passons de 2 à 3 terminaux,

nous avons prouvé que $3-$ ChaineI devient $\mathcal{NP}$-complet, alors que $3-$ ArbreI est polynomial ([16]). Une grande partie des résultats du chapitre 6 fait l'objet d'un article soumis en revue ([22]).

Il semble alors naturel de se demander si $k-$ ArbreI, à son tour, devient $\mathcal{NP}$-complet à partir d'un certain nombre de terminaux ou s'il est toujours polynomial lorsque le nombre de terminaux est fixé. Cependant, nous n'avons pas réussi à répondre à cette question, qui reste totalement ouverte pour $k \geq 4$, et qui permettrait de véritablement conclure notre étude de la complexité des problèmes de cycles, de chaînes et d'arbres induits.

Dans un premier temps, nous nous sommes interrogés sur la complexité de $k-$ ArbreI lorsque nous ajoutons une condition sur la maille du graphe. En effet, les cycles représentant le principal obstacle pour la construction d'un arbre induit, nous avons décidé d'étudier la complexité de $k-$ ArbreI lorsque la maille du graphe est large. Nous avons montré que, lorsque la maille du graphe est strictement supérieure au nombre de terminaux, un arbre de Steiner de taille minimum est aussi un arbre induit. Cela nous a permis d'obtenir un ensemble de résultats de complexité puisque le problème de l'arbre de Steiner est dans ce cas $\mathcal{NP}$-complet et devient polynomial dès lors que le nombre de sommets terminaux est fixé.

Dans un second temps, nous nous sommes interrogés sur la complexité de $4-$ ArbreI. Devant notre incapacité à déterminer la complexité de ce problème dans le cas général et compte tenu de sa polynomialité lorsque le graphe ne possède pas de cycle de longueur 3 ou 4, nous avons décidé de l'étudier dans le cas "intermédiaire" des graphes sans triangle.

Les problèmes d'existence d'un arbre induit, étudiés dans le chapitre 6 sont soit $\mathcal{NP}$-complets, soit triviaux (car admettant toujours une solution). Or, nous avons montré dans le chapitre 7 que $4-$ ArbreI est polynomial dans les graphes sans triangle en élaborant un algorithme de complexité $O(nm)$. Pour cela, nous nous sommes interrogés sur les propriétés que devaient avoir un graphe sans triangle pour qu'il n'existe aucune solution à $4-$ ArbreI. Nous avons alors déterminé deux structures de graphes empêchant l'existence d'une solution : les *carrés* et les *cubes*. Puis, nous avons construit un algorithme déterminant un arbre induit couvrant les quatre terminaux ou prouvant que le graphe considéré est un carré ou un cube. Ce travail a donné lieu à la soumission d'un article en revue ([23]).

Déterminer la complexité de $k-$ ArbreI dans les graphes généraux pour $k \geq 4$ nous apparaît bien difficile. Il semble donc plus raisonnable dans un premier temps de déterminer si les résultats obtenus pour $4-$ ArbreI dans les graphes sans triangle peuvent se généraliser avec cinq terminaux ou plus.

| **Résultats de complexité** | | Cycle induit | Chaîne induite | Arbre induit |
|---|---|---|---|---|
| Graphes pondérés : | $k = 1$ | $\mathcal{NP}$-complet au sens fort | Polynomial $O(1)$ | Polynomial $O(1)$ |
| | $k = 2$ | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort | $\mathcal{NP}$-complet au sens fort |
| Graphes non pondérés : ($k$ non fixé) | cas général | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet |
| | maille $\geq k+1$ | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet | Polynomial $O(1)$, $\mathcal{NP}$-difficile pour $k-$MinArbreI |
| Graphes non pondérés : ($k$ fixé) | $k = 1$ | Polynomial | Polynomial $O(1)$ | Polynomial $O(1)$ |
| | $k = 2$ | $\mathcal{NP}$-complet [8] | Polynomial $O(1)$, $O(m)$ pour $2-$MinChaineI | Polynomial $O(1)$, $O(m)$ pour $2-$MinArbreI |
| | $k = 3$ | $\mathcal{NP}$-complet [8] | $\mathcal{NP}$-complet | Polynomial $O(n^4)$ [16] |
| | $k = 3$ sans $C_3$ | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet | Polynomial $O(1)$, $O(m)$ pour $k-$MinArbreI |
| | $k = 4$ sans $C_3$ | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet | Polynomial $O(nm)$ |
| | $k \geq 4$ | $\mathcal{NP}$-complet [8] | $\mathcal{NP}$-complet | Ouvert |
| | $k \geq 4$ maille$\geq k+1$ | $\mathcal{NP}$-complet | $\mathcal{NP}$-complet | Polynomial $O(1)$, Polynomial pour $k-$MinArbreI |

Table 8.1 – Récapitulatif des résultats obtenus pour $k-$CycleI, $k-$ChaineI et $k-$ArbreI ($k-$PondCycleI, $k-$PondChaineI et $k-$PondArbreI dans le cas des graphes pondérés). Les algorithmes dont la complexité est en $O(1)$ correspondent à des cas où le sous-graphe induit existe toujours. Une référence est fournie pour tous les résultats antérieurs à nos travaux.

# Bibliographie


[1] R. Ahuja, T. Magnanti et J. Orlin. Network Flows : Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs, New Jersey (1993)

[2] H. Aissi, C. Bazgan, D. Vanderpooten. Complexity of the min-max (regret) versions of min cut problems. Discrete Optimization **5** 66–73 (2008)

[3] T. Akiyama, T. Nishizeki, N. Saito. $\mathcal{NP}$-completeness of the Hamilton cycle problem in bipartite graphs. Journal of Information Processing **5** 73–76 (1980)

[4] A. Armon, U. Zwick. Multicriteria Global Minimum Cuts. Algorithmica **46**(1) 15–26 (2006)

[5] C. Bentz, M.-C. Costa, N. Derhy, F. Roupin. Cardinality constrained and multicriteria (multi)cut problems. A paraitre dans Journal of Discrete Algorithms.

[6] C. Berge. *Graphes et Hypergraphes.* Dunod, Paris (1969).

[7] E.R. Berkelamp (1976). Cité dans "Computers and Intractability, a Guide to the Theory of NP-completeness" par M.R. Garey et D.S. Johnson

[8] D. Bienstock. On the complexity of testing for even holes and induced odd paths. Discrete Mathematics **90** 85–92 (1991). Corrigendum by B. Reed in Discrete Mathemathics **102** 109 (1992)

[9] A. Billionnet. Different formulations for solving the heaviest $k$-subgraph problem. Information Systems and Operational Research **43** (3) 171–186 (2005)

[10] M. Bruglieri, F. Maffioli, M. Ehrgott. Cardinality constrained minimum cut problems : complexity and algorithms. Discrete Applied Mathematics **137** 311–341 (2004)

[11] M. Bruglieri, H.W. Hamacher, F. Maffioli, M. Ehrgott. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. Discrete Applied Mathematics **154** (9) (2006)

[12] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. Actes CCC (2005)

[13] J. Chen, I.A. Kanj. Constrained minimum vertex cover in bipartite graphs : complexity and parameterized algorithms. Journal of Computer and System Sciences **67** 833–847 (2003)

[14] J. Cheriyan, H. Karloff, Y. Rabani. Approximating directed multicuts. Actes FOCS 320–328 (2001)

[15] M. Chudnovsky, G. Cornuéjols, X. Liu, P.D. Seymour, K. Vušković. Recognizing Berge graphs. Combinatorica **25** 43–186 (2005)

[16] M. Chudnovsky, P.D. Seymour. The three-in-a-tree problem. Manuscrit, soumis (2006)

[17] M. Chudnovsky, N. Robertson, P.D. Seymour, R.Thomas. The strong perfect graph theorem. Annals of Mathematics **164** 51–229 (2006)

[18] M. Chudnovsky, R. Kapadia. Detecting a theta or a prism. SIAM Journal on Discrete Mathematics **22** (3) (2008)

[19] M.-C. Costa, L. Létocart, F. Roupin. A greedy algorithm for multicut and integral multiflow in rooted trees. Operations Research Letters **31**, 21–27 (2003)

[20] M.-C. Costa, L. Létocart, F. Roupin. Minimal multicut and maximal integer multiflow : a survey. European Journal of Operational Research **162**(1) 55–69 (2005)

[21] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis. The complexity of multiterminal cuts. SIAM Journal on Computing **23** 864–894 (1994)

[22] N. Derhy, C. Picouleau. Finding induced trees. Soumis à Discrete Applied Mathematics.

[23] N. Derhy, C. Picouleau, N. Trotignon. The four-in-a-tree problem in triangle-free graphs. Soumis à Graph Combinatorics.

[24] S.E. Dreyfus, R.A. Wagner. The Steiner problem in graphs. Networks **1** , 195–207 (1972)

[25] L.R. Ford, D.R. Fulkerson. Flows in networks. Princeton University Press (1962)

[26] R.L. Francis, T.J Lowe, B.C. Tansel. Location on networks : a survey-Part I : the p-center and p-median problems. Management Science **29** 482–497 (1983)

[27] M.R. Garey, D.S. Johnson. Computers and Intractability, a Guide to the Theory of NP-completeness. ed. Freeman (1979)

[28] M.R. Garey, D.S. Johnson, L. Stockmeyer. Some simplified NP-complete graph problems. Theoretical Computer Science **1** 237–267 (1976)

[29] N. Garg, V.V. Vazirani, M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. SIAM Journal on Computing **25** 235–251 (1996)

[30] N. Garg, V.V. Vazirani, M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica **18** 3–20 (1997)

[31] N. Garg, V.V. Vazirani, M. Yannakakis. Multiway cuts in node weighted graphs. Journal of Algorithms **50** 49–61 (2004)

[32] E.M. Gold. Complexity of automaton identification from given data. Manuscrit non publié (1974)

[33] D. Golovin, V. Nagarajan, M. Singh. Approximating the k-multicut problem. Actes SODA 621–630 (2006)

[34] M. Grotschel, L. Lovasz, A. Schriver. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1** 169–197 (1981)

[35] A. Gupta. Improved results for directed multicut. Actes SODA 454–455 (2003)

[36] R. Hassin, A. Tamir. Improved complexity bounds for location problems on the real line. Operations Research Letters **10** 395–402 (1991)

[37] D.S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. Discrete Applied Mathematics **6** 243–254 (1983)

[38] D.S. Hochbaum (éd.). Approximation algorithms for NP-hard problems. PWS Publishing Company, Boston (1997)

[39] F.K. Hwang, D.S. RICHARDS. Steiner tree problems. Networks **22** (1) 55–89 (1992).

[40] R.M. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations, Eds. Miller et Watcher (1972)

[41] B. Kimelfeld, Y. Sagiv. New algorithms for computing Steiner trees for a fixed number of terminals. Manuscrit non publié (2006)

[42] D. König. Graphok és matrixok. Matematikai és Fizikai Lapok **38** 116–119 (1931)

[43] Y. Kortsarts, G. Kortsarz, Z. Nutov. Greedy approximation algorithms for directed multicuts. Networks **45** 214–217 (2005)

[44] B. Lévêque, D. Lin, F. Maffray, N. Trotignon. Detecting induced subgraphs. Manuscrit, soumis (2007)

[45] A. Levin, D. Segev. Partial Multicuts in Trees. Actes WAOA, Lecture Notes in Computer Science **3879** 320–333 (2006)

[46] F. Maffray, N. Trotignon. Algorithms for perfectly contractile graphs. SIAM Journal of Discrete Mathematics **19** (3) 553–574 (2005)

[47] C. McDiarmid, B. Reed, A. Schrijver, F.B. Shepherd. Induced circuits in planar graphs. Journal of Combinatorial Theory, series B **60** 169–176 (1994)

[48] G. Mermoud, M.A. Schaub, G. Théoduloz. Partitioning CiteSeer's Citation Graph. Rapport EPFL (2005)

[49] P. Michelon. Méthodes lagrangiennes pour programmation linéaire avec variables entières. Investigación Operativa **2** 127–146 (1991)

[50] C.H. Papadimitriou, M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In Proceedings of the IEEE Symposium on Foundations of Computer Science 86–92 (2000)

[51] P.M. Pardalos, J. Xue. The maximum clique problem. Journal of Global Optimization **4** 301–328 (1994)

[52] J.-C. Picard, M. Queyranne. Selected applications of minimum cuts in networks. INFOR **20** (4) 394–422 (1982)

[53] J.L. Ramírez-Alfonsín, B.A. Reed. Perfect graphs. Ed. Wiley Interscience (2001)

[54] A. Schrijver. Theory of Linear and Integer Programming. ed. Wiley (1986)

[55] É. Tardos, V.V. Vazirani. Improved bounds for the max-flow min-multicut ratio for planar and $K_{r,r}$-free graphs. Information Processing Letters **47** 77–80 (1993)

[56] M. Yannakakis, P. Kanellakis, S. Cosmadakis, C. Papadimitriou. Cutting and partitioning a graph after a fixed pattern. Actes ICALP, Lecture Notes in Computer Science **154** 712–722 (1983)

**Title** : *Multicut and induced subgraph problems : complexity and algorithms.*

*Abstract :*

In this thesis, we consider some problems of graph theory. First, we deal with cut and multicut problems and then, we study induced subgraph problems. Nevertheless, these two parts share a common purpose : determining a general overview of the complexity of these problems by proving $\mathcal{NP}$-completeness results or by designing polynomial algorithms with low running times.

In the first part, we tackle cut and multicut problems. We study the consequences of the addition of a cardinality constraint and show the $\mathcal{NP}$-completeness of the general cases. Besides, we give complexity results for some particular graphs such as directed stars and undirected paths, and for the polynomial cases, we design several algorithms using dynamic programming or lagrangian relaxations.

Next, we generalize these problems by considering multicriteria versions of the (multi)cut problems. We obtain some $\mathcal{NP}$-completeness results in some very specific classes of graphs like undirected paths and cycles.

In the second part, we focus on the detection of specific induced subgraphs. More precisely, we look for induced paths, induced cycles or induced trees covering a given set of vertices. After proving the $\mathcal{NP}$-completeness of the general cases, we consider the cases where the number of prescribed vertices is fixed. Finally, we also give some structural results for $C_3$-free graphs.

*Keywords :*

Graph theory, multicuts, induced subgraphs, cardinality constraint, multicriterion, complexity, $\mathcal{NP}$-completness, polynomial algorithms

**Titre** : *Multicoupes et sous-graphes induits : complexité et algorithmes.*

*Résumé :*

Dans ce travail de thèse, nous nous intéressons à plusieurs problèmes de théorie des graphes. Dans un premier temps, nous étudions différents problèmes de coupes et de multicoupes puis, dans un second temps, nous nous focalisons sur des problèmes de recherche de sous-graphes induits. Néanmoins, ces deux parties suivent la même ligne directrice : donner une vue d'ensemble de la complexité des problèmes en établissant leur $\mathcal{NP}$-complétude ou en déterminant un algorithme polynomial de moindre complexité.

Dans la première partie de la thèse, nous abordons les problèmes de coupes et de multicoupes. Tout d'abord, nous étudions la conséquence de l'ajout d'une contrainte de cardinalité à ces deux types de problèmes et démontrons leur $\mathcal{NP}$-complétude dans le cas général. Puis, nous déterminons leur complexité dans plusieurs classes de graphes particuliers telles que les étoiles orientées et les chaînes en élaborant, pour les cas polynomiaux, différents algorithmes reposant principalement sur la programmation dynamique et l'utilisation de relaxations lagrangiennes.

Nous généralisons ensuite cette approche en considérant les versions multicritères des problèmes de coupes et de multicoupes. Nous prouvons que ces derniers sont $\mathcal{NP}$-complets même dans des topologies très simples comme les chaînes ou les cycles.

Dans la seconde partie de ce mémoire, nous abordons des problèmes de recherche de sous-graphes induits. Nous nous intéressons principalement à la recherche d'arbres, de chaînes et de cycles induits couvrant un ensemble $T$ de sommets donnés. Après avoir prouvé la $\mathcal{NP}$-complétude des cas généraux, nous nous focalisons davantage sur les cas où la cardinalité de $T$ est fixée. Nous donnons également plusieurs résultats structurels pour les graphes de maille suffisamment large.

*Mots-clés :*

Théorie des graphes, multicoupes, sous-graphes induits, contrainte de cardinalité, multicritère, complexité, $\mathcal{NP}$-complétude, algorithmes polynomiaux.