Algorithmes et résultats de complexité pour des problèmes de graphes avec contraintes additionnelles

Alexis Cornet

Université Clermont Auvergne

École Doctorale Sciences pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par

Alexis Cornet

pour obtenir le grade de

Docteur d'Université

Spécialité Informatique

Algorithmes et résultats de complexité pour des problèmes de graphes avec contraintes additionnelles

---

Soutenue publiquement le 5 Décembre 2018 devant le jury :

M. BAIOU Mourad	Directeur de Recherche	Examinateur
Mme COHEN Johanne	Directrice de Recherche	Examinatrice
M. LAFOREST Christian	Professeur des Universités	Directeur de thèse
Mme RUSU Iréna	Professeur des Universités	Rapporteur
M. TROTIGNON Nicolas	Directeur de Recherche	Rapporteur

Résumé

Les problèmes de domination (dominant, dominant indépendant, ...) et de couverture (vertex-cover, arbre de Steiner, ...) sont NP-complets. Pour autant, pour la plupart de ces problèmes, il existe toujours une solution constructible en temps polynomial (potentiellement de valeur objective très mauvaise), ou au moins, il est possible de déterminer facilement (en temps polynomial) l'existence ou non d'une solution.

Ces problèmes, initialement issus de situations réelles, sont des modélisations simplistes de ces situations. Nous ajoutons donc des contraintes additionnelles modélisant des contraintes pratiques plausibles : les *conflits*, des paires d'éléments ne pouvant faire simultanément partie d'une solution (modélisant des incompatibilités diverses), la *connexité dans un second graphe* (les éléments doivent pouvoir communiquer, et le graphe correspondant à ces liens de communication n'est pas forcément le même) et les *obligations*, des sous-ensembles d'éléments interdépendants devant être ajoutés simultanément à une solution.

Notre but ici n'est pas de modéliser un problème réel précis, mais d'étudier la manière dont ces contraintes modifient la complexité des problèmes étudiés. Nous verrons que dans un grand nombre de cas, déterminer l'existence même d'une solution devient difficile, même sans se préoccuper de leur optimisation.

Le problème du *firefighter* modélise des pompiers tentant de contenir un feu se propageant au tour par tour dans un graphe (potentiellement infini). Nous avons étudié ce problème en ajoutant des contraintes sur le déplacement des pompiers (une vitesse de déplacement limitée entre deux tours). Nous verrons que ces contraintes augmentent en général le nombre de pompiers nécessaires mais ne provoquent pas de changements aussi importants que dans les problèmes précédents.

Abstract

Domination problems (dominating set, independant dominating set, ...) as well as covering problems (vertex-cover, Steiner tree, ...) are NP-complete. However, for most of these problems, it is always possible to construct a (eventually bad) solution in polynomial time, or at least it is possible to determine whether a solution exists.

Those problems originally came from industry, but are simplified modelizations of the real life problems. We add additional constraints modeling plausible practical constraints : *conflicts* which are pairs of elements that cannot apear simultaneously in a solution (to modelize various incompatibilities), *connexity in a second graph* (elements of the solution must be able to communicate, and the communication links are a second graph), and *obligations* which are subsets of interdependant vertices which must be added simultaneously in a solution.

We don't aim to model a specific real-world problem, but to study how these plausible constraints affect the complexity of the studied problems. We will see that, in many cases, even determining the existence of a solution (regardless of its size) become hard.

The *firefighter* problem models firefighters aiming to contain a fire spreading turn by turn in a (eventually infinite) graph. We studied this problem with the addition of deplacement constraints for the firefighters (a limited moving speed between turns). We will see that, most of the time, this constraint increase the number of firefighters necessary to contain the fire, but does not trigger such major change as constraints studied in the others problems.

Remerciements

Je tiens à remercier particulièrement Christian Laforest, mon directeur de thèse, pour m'avoir encadré efficacement tout au long des trois années du doctorat. Ses conseils ont rendu la réalisation de ce travail possible. Les réunions de travail hebdomadaires ont toujours été un moment agréable.

Je remercie Iréna Rusu, Nicolas Trotignon, Johanne Cohen et Mourad Baiou d'avoir accepté de faire partie de mon jury de thèse, pour leurs remarques qui m'ont permis d'améliorer la qualité de ce manuscript.

Merci également aux membres du projet Géo-safe présents à Melbourne, ce changement de contexte géographique et scientifique durant la période de rédaction était plaisant.

Mes remerciements au personnel administratif et technique du Limos qui m'a fourni un cadre de travail agréable et efficace.

Merci à mon vieux père d'avoir relu et corrigé ma thèse. Merci à la chaudière d'avoir la plupart du temps bien voulu chauffer l'eau de la douche, les jours ou elle ne l'a pas fait ont été difficiles. Merci Jean d'avoir cohabité avec moi, tu as toujours été une source d'étonnement comportemental et culinaire. Merci à mes collègues de bureau, en particulier Marina et Sahar pour leur bonne humeur et leur gentillesse. Merci aux gens que je n'aimais pas d'avoir été peu nombreux.

Merci Gülşah d'avoir été présente dans les moments difficiles. Tu as toujours été là pour me distraire les jours ou je n'avais pas envie de travailler. Merci aussi d'avoir relu ma thèse. Merci Antoine et Matthieu pour la promenade du midi qui a toujours été riche en inspiration et en découvertes diverses et inattendues.

Merci aux autres personnes ayant contribué à faire de ces trois ans une période intéressante.

Merci à tout le monde, cette thèse, c'était marrant.

Table des matières

Introduction

Un graphe est un ensemble de *sommets* représentant des éléments. Ces sommets sont connectés entre eux par des *arêtes* représentant des liens entre ces éléments. Les graphes sont utilisés pour modéliser tout type de situation mettant en jeu des objets en relation les uns avec les autres. Ils sont particulièrement utilisés pour les problèmes de réseaux (réseaux informatiques, réseaux routiers, réseaux sociaux, télécommunications, ...), mais aussi dans d'autres domaines, comme par exemple la biologie ou la chimie (voir [Bal85]).

Un graphe modélise donc une situation, appelée *instance*, d'un *problème* auquel on recherche une solution, au travers d'*algorithmes*. Une solution est généralement un sous-ensemble de sommets (ou d'arêtes) vérifiant une propriété. Si un graphe modélise par exemple un réseau routier, dans lequel on cherche un plus court chemin entre deux points, une solution pourra être une séquence d'arêtes reliant ces deux points.

Les nombreux problèmes de graphes ayant vu le jour ne sont pas tous de difficulté équivalente. Certains, comme le problème du plus court chemin mentionné plus haut, sont très simples, et les algorithmes les résolvant s'exécutent rapidement. D'autres, au contraire, ne semblent pas pouvoir être résolus en temps raisonnable. Aussi, une volonté de catégoriser les problèmes suivant le temps nécessaire à leur résolution est rapidement apparue. Parmi les différentes classes de complexité, deux catégories de problèmes (supposées distinctes) ont émergé. Les problèmes *polynomiaux* sont des problèmes pour lesquels il existe des algorithmes *déterministes* s'exécutant en temps polynomial en la taille de l'instance. Ces problèmes correspondent en pratique aux problèmes pour lesquels il existe des algorithmes efficaces. Les problèmes *NP-complets* sont les problèmes les plus difficiles pour lesquels il existe des algorithmes *non déterministes* polynomiaux. En pratique, ceci correspond aux problèmes pour lesquels il n'existe pas d'algorithmes exacts s'exécutant en temps raisonnable. Pour plus de précisions sur les classes de complexité, voir par exemple [AB09].

La classification des problèmes comme étant polynomiaux ou NP-complets a donc pris une importance capitale, tant sur le plan pratique que théorique,

et de nombreux problèmes industriels se sont révélés NP-complets. Le livre *Computers and Intractability* [GJ02] est témoin de l'activité soutenue de ce domaine, et propose une liste extensive bien que non exhaustive de problèmes NP-complets.

Parmi les problèmes classiques, on peut compter les problèmes de *domination*. Un dominant est un sous-ensemble de sommets pouvant "voir" tous les sommets du graphes : chaque sommet appartient au dominant, ou est voisin d'un sommet appartenant au dominant (les définitions formelles apparaissent dans les parties concernées). Un dominant est donc une structure permettant de toucher tous les éléments d'un réseau. Des problèmes de domination sont utilisés entre autres pour la modélisation du squelette de réseaux sans fil, voir par exemple [AWF02], [WCDY08] ou encore [SRS08].

Un problème voisin est celui du *vertex cover*. Un vertex cover est un sous-ensemble de sommets pouvant "voir" toutes les arêtes du graphe : chaque arête a une extrémité faisant partie du vertex cover. On obtient donc une structure permettant de surveiller tous les liens d'un réseau. D'autres structures, comme les arbres couvrants ou les arbres de Steiner, sont utilisées pour connecter des éléments pour un coût minimal.

Ces problèmes classiques s'appliquent donc à des situations réelles. Cependant, ces modélisations sont simplistes et ne capturent en général pas toutes les contraintes rencontrées en pratique. Certains éléments peuvent être incompatibles entre eux, par exemple pour des raisons de sécurité, d'interférences, d'interfaces logicielles ou matérielles incompatibles, ou de choix mutuellement exclusifs. Nous dirons que deux éléments sont *en conflit* s'ils ne peuvent pas faire partie d'une solution simultanément. Une solution *sans conflit* sera une solution ne contenant pas de paire d'éléments en conflit. Une paire d'éléments en conflit sera modélisée par une *arête de conflit*. Ces arêtes de conflits seront interprétées comme un second graphe, le *graphe des conflits* sur les mêmes sommets que le graphe initial, appelé *graphe support*.

De premières études sur des problèmes avec conflits, appelés alors *forbidden pairs*, ont été réalisées autour de 1976, dans un article [GMO76] s'intéressant à la complexité de trouver un chemin sans conflit entre deux sommets. Ces travaux étaient motivés par la génération de chemins de tests dans des programmes, deux sommets en conflit représentant deux conditions mutuellement exclusives. Ces travaux ont été étendus en 1997 [Yin97] puis plus récemment en 2009 [KP09] et 2013 [Kov13]. La plupart des résultats sont des théorèmes montrant la NP-complétude de déterminer l'existence de tels chemins dans diverses classes d'instances. Plus récemment, Benjamin Momège a étudié dans sa thèse de doctorat [Mom15] des problèmes avec conflits sur les arêtes, appelés *transitions interdites*. Ses travaux ont donné lieu à plusieurs publications présentant des algorithmes pour des problèmes liés à la connexité (chemins, cycles, chemins Hamiltoniens) sans transitions interdites

ainsi que des résultats de complexité [KMMN15] [LM15] [LM14] [KLM13a] [KLM13b].

D'autres contraintes additionnelles peuvent être envisagées. Dans le contexte des réseaux, il est naturel de souhaiter obtenir des solutions *connexes* (pour que tous les éléments d'une solution puissent communiquer). C'est d'ailleurs pourquoi cette contrainte supplémentaire apparaît dans plusieurs problèmes, comme le problème du *vertex cover connexe* pour ne citer que lui. Cependant, en toute généralité, les liens à surveiller par le vertex cover ne sont pas nécessairement les mêmes que les liens permettant aux éléments d'une solution de communiquer entre eux. Pour modéliser cette situation, nous introduirons un nouveau problème, celui du vertex cover, connexe dans un autre graphe.

Il est également possible de rencontrer d'autres types de contraintes : des éléments peuvent être interdépendants, par exemple si un outil est distribué parmi plusieurs nœuds d'un réseau qui doivent tous être actifs pour que celui-ci fonctionne. Nous modéliserons ces situations par des *obligations*. Les sommets seront regroupés en sous-ensembles, appelés obligations, et devront être ajoutés à la solution sous-ensemble par sous-ensemble, au lieu d'un par un dans les problèmes classiques.

L'objet de ce mémoire n'est en aucun cas de résoudre un problème concret. Nous avons souhaité au cours de cette thèse définir des contraintes additionnelles plausibles sur des problèmes de graphes classiques, permettant une modélisation plus fine des situations réelles. Notre objectif était d'étudier l'effet de ces différents types de contraintes sur la complexité algorithmique de ces problèmes. L'ajout de contraintes additionnelles laisse-t-elle une possibilité de trouver une bonne solution approchée ? Existe-t-il une solution à coup sûr ? Est-t-il simplement possible de déterminer l'existence d'une solution ?

Pour tenter de répondre à ces questions, une première partie sera consacrée aux problèmes de domination sans conflit, nous y étudierons la complexité de déterminer l'existence d'une solution sans conflit dans plusieurs classes d'instances. Dans le chapitre 1, nous nous intéresserons à des paramètres du graphe support et du graphe des conflits séparément. Nous montrerons que le problème reste NP-complet dans la plupart des classes de graphes que nous avons considérées. Nous noterons tout de même certains cas polynomiaux et proposerons un algorithme paramétrique pour un cas particulier du dominant indépendant sans conflit. Au chapitre 2, nous étudierons des paramètres liant le graphe support et le graphe des conflits : plus spécifiquement, nous considèrerons des instances où l'union des deux graphes est planaire, ainsi qu'un nouveau paramètre, l'*étirement* des conflits, mesurant la localité des conflits par rapport au graphe support. Enfin, le chapitre 3 traitera de l'approximation des problèmes de domination. Nous ne parlons pas ici d'optimi-

sation de la taille des dominants sans conflit, mais de domination partielle du graphe support. Nous chercherons à savoir quelle proportion d'un graphe il est possible de dominer sans conflit.

Dans une seconde partie, d'autres problèmes avec conflits seront étudiés. Le chapitre 4 traitera d'autres problèmes classiques de graphes avec conflits, à savoir le dominant total, le vertex cover connexe, et l'arbre de Steiner. Nous montrerons encore la NP-complétude de ces problèmes dans diverses classes d'instances, et identifierons quelques cas polynomiaux. Dans le chapitre 5, nous étendrons le concept de conflits aux langages réguliers et aux automates à états finis. La contrainte "sans conflit" semble complexifier drastiquement les problèmes de graphes : qu'en est-il pour les langages ?

Dans une troisième partie, nous aborderons d'autres contraintes : la connexité, et les obligations. Le chapitre 6 sera dédié au problème du vertex cover, connexe dans un autre graphe. Nous y chercherons des algorithmes d'approximation à rapport constant. Plusieurs algorithmes seront proposés pour des cas particuliers reposant sur les relations d'inclusion entre les deux graphes de l'instance. Le chapitre 7 sera consacré aux problèmes avec obligations : certains problèmes avec obligations sur les sommets, comme le vertex cover, le vertex cover connexe, et les problèmes de dominations, et d'autres avec obligations sur les arêtes, comme le problème de l'arbre couvrant, du grave couvrant connexe, du couplage maximum ou du chemin hamiltonien. Nous nous intéresserons à la complexité de décider de l'existence d'une solution, et, quand cela est possible, à l'approximation d'une solution optimale.

Dans un dernier chapitre indépendant, nous présenterons des résultats sur le problème du *firefighter* : se rapprochant d'un jeu combinatoire au tour par tour, ce problème modélise des pompiers tentant de contenir un feu se propageant dans un graphe. Nous avons ajouté au problème classique une contrainte additionnelle limitant le déplacement des pompiers, pour rendre celui-ci plus réaliste. Nous proposerons des stratégies visant à contenir le feu en respectant les contraintes. La plupart du temps, celles-ci utiliseront un nombre plus important de pompiers que les versions non contraintes. Les résultats ont été obtenus lors d'une mobilité à Melbourne financée par GEO-SAFE [1] et constituent des résultats préliminaires à un travail toujours en cours.

Publications

Les résultats du chapitre 1 ont fait l'objet d'une publication dans *Discrete Applied Mathematics* [CL18a].

1. GEO-SAFE (*Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies*) est un projet financé par les gouvernements Européens et Australien pour la prévention et la lutte contre les feux de forêt.

Les résultats du chapitre 2 ont été présentés à la *ROADEF 2018* pour la finale du *Prix du Meilleur Article Étudiant* [CL18c].

Les résultats du chapitre 4 ont été publiés dans *Discrete Mathematics and Theoritical Computer Science* [CL17].

Les résultats des chapitres 5 et 7 n'ont pas encore été publiés mais les rapports de recherche correspondants sont disponibles sur HAL [CL16] [CL18b].

Vocabulaire de la théorie des graphes

Nous rappelons ici de manière informelle quelques notions de graphe utiles à la compréhension de ce manuscrit. Pour une introduction plus complète à la théorie des graphes, le lecteur peut se référer par exemple à [Die12].

Une *chaîne* entre deux sommets a et b, également appelée *chemin* par abus de langage dans ce document, est une suite finie d'arêtes consécutives reliant a et b.

Une arrête connectant deux sommets est dite *incidente* à ces sommets. Le *degré* d'un sommet est le nombre d'arêtes incidentes à ce sommet. Le degré maximum d'un graphe est le plus grand degré de ses sommets.

Un graphe est dit *connexe* s'il existe un chemin entre toute paire de ses sommets.

Un *graphe induit* H dans G par un sous-ensemble de sommets X est le graphe dont l'ensemble des sommets est X et où il existe une arête ab dans H si et seulement si il existe une arête ab dans G.

Un *stable* est un sous ensemble de sommets n'induisant aucune arête.

Un *couplage* est un ensemble d'arêtes deux à deux disjointes.

Un *cycle* est une chaîne dont le premier et le dernier sommets sont les mêmes.

Classes de graphes Une *classe de graphes* est une famille de graphes caractérisés par une propriété commune. Elles sont utilisées entre autre pour restreindre l'étude d'un problème à certains cas, plus proches d'une situation réelle, ou présentant des propriétés structurelles particulières.

Nous présentons ici quelques classes de graphes utilisées tout au long de ce manuscrit.

Un *arbre* est un graphe connexe sans cycle.

Un *graphe chemin*, appelé simplement *chemin* dans ce document est un arbre dont chaque sommet est de degré au plus 2. Dans ce document, un chemin à n sommet sera noté P_n.

Une *étoile* est un arbre ayant un sommet universel.

Un *caterpillar* est un arbre dans lequel tous les sommets sont à distance au plus 1 d'un chemin central.

Un *graphe complet* est un graphe dans lequel chaque paire de sommets est reliée par une arête. Le graphe complet à n sommets est noté K_n

Un *split graph* est un graphe dont les sommets peuvent être partitionnés en deux sous-ensembles, l'un induisant un stable, l'autre un graphe complet.

Un *graphe biparti* est un graphe dont les sommets peuvent être partitionnés en deux sous-ensembles induisant des stables.

Un *graphe biparti complet* est un graphe biparti dans lequel chaque élément d'une partition est voisin de tous les éléments de l'autre partition. Le graphe biparti complet à deux partitions de taille a et b est noté $K_{a,b}$.

Un *graphe triangulé* est un graphe pour lequel chaque cycle de longueur supérieure à 4 a une corde.

Un *graphe planaire* est un graphe pouvant être représenté dans le plan euclidien sans que ses arêtes se croisent.

Un *graphe de Dirac* à n sommets est un graphe dont chaque sommet est de degré au moins $n/2$.

Le *claw graph* est une étoile à 4 sommets.

Première partie

Problèmes de domination avec conflits

Introduction

Cette partie est consacrée aux problèmes de domination sans conflit. Nous y étudierons les problèmes du *Dominant sans Conflit* (DSwnC) et du *Dominant Indépendant sans Conflit*. Dans le chapitre 2, le problème du *Dominant Total sans Conflit* (TDSwnC) sera également abordé car son comportement est similaire.

Un *Dominant* de $G = (V, E)$ est un sous-ensemble S de V tel que pour tout $x \in V$, soit $x \in S$, soit il existe $xy \in E$ tel que $y \in S$. Un *Dominant Indépendant* S de $G = (V, E)$ est un dominant, c'est-à-dire que pour tout $x \in V$, soit $x \in S$, soit il existe $xy \in E$ tel que $y \in S$, et un *indépendant* de G, c'est-à-dire que pour toute arête $xy \in E$, si x appartient à S, alors y n'appartient pas à S.

Les problèmes de domination sont fondamentaux en théorie des graphes. De nombreuses variantes ont été étudiées dans la littérature, voir par exemple [HHS98]. Les problèmes d'optimisation de taille du Dominant, Dominant Indépendant et Dominant Total sont NP-complets. Il a également été prouvé qu'ils n'étaient pas approchables par une constante sauf si P = NP [ACG$^+$12]. Cependant, il est toujours possible de construire une solution réalisable en temps polynomial. Par exemple, pour tout graphe $G = (V, E)$, V est un dominant de G. Il est également possible de construire un dominant indépendant de G en temps polynomial en utilisant le petit algorithme suivant :

Algorithm 1: Trouver un dominant de G

Data: $G = (V, E)$ un graphe quelconque
Result: S un dominant indépendant de G
$S \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
    Choisir x un sommet arbitraire de V
    $S \leftarrow S \cup x$
    $V \leftarrow V - N(x)$
return S

La solution retournée par l'algorithme 1 n'est pas nécessairement bonne du point de vue de l'optimisation, mais elle existe toujours. Pour le problème du dominant total, on peut montrer qu'il existe une solution si et seulement si le graphe n'a pas de sommets isolés.

Nous définissons maintenant formellement les problèmes de domination sans conflit.
Définition 1. Dominant sans Conflit (DSwnC)

Instance : (G, C) un graphe avec conflits ou $G = (V, E)$ est le *graphe support* et $C = (V, E_2)$ est le *graphe des conflits*
Solution : S un sous-ensemble de V tel que :
— $\forall x \in V$, $x \in S$ ou il existe $xy \in E$ tel que $y \in S$ (S est un dominant de G)
— $\forall xy \in E_2$, $x \notin S$ ou $y \notin S$ (S est sans conflit dans C)

Définition 2. Dominant Indépendant sans Conflit (IDSwnC)
Instance : (G, C) un graphe avec conflits ou $G = (V, E)$ est le *graphe support* et $C = (V, E_2)$ est le *graphe des conflits*
Solution : S un sous-ensemble de V tel que :
— $\forall x \in V$, $x \in S$ ou il existe $xy \in E$ tel que $y \in S$ (S est un dominant de G)
— $\forall xy \in E$, $x \notin S$ ou $y \notin S$ (S est un indépendant de G)
— $\forall xy \in E_2$, $x \notin S$ ou $y \notin S$ (S est sans conflit dans C)

Définition 3. Dominant Total sans Conflit (TDSwnC)
Instance : (G, C) un graphe avec conflits ou $G = (V, E)$ est le *graphe support* et $C = (V, E_2)$ est le *graphe des conflits*
Solution : S un sous-ensemble de V tel que :
— $\forall x \in V$, il existe $xy \in E$ tel que $y \in S$ (S est un dominant total de G)
— $\forall xy \in E_2$, $x \notin S$ ou $y \notin S$ (S est sans conflit dans C)

L'ajout de la nouvelle contrainte *sans conflit* peut parfois rendre certaines instances sans solution. Ci-dessous, figure 1, une instance du IDSwnC n'admettant pas de solution. Le graphe support est représenté en traits pleins noirs, et le graphe des conflits en tirets rouges - ce sera également le cas pour les autres figures de cette section, sauf mention contraire.

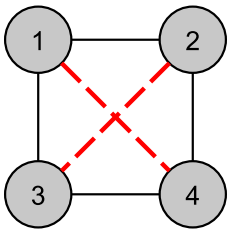


FIGURE 1 – Instance du IDSwnC n'admettant pas de solution

L'existence d'une solution n'étant plus garantie, la question de déterminer l'existence d'une solution se pose naturellement : c'est l'objet de cette partie. Par la suite, lorsque nous évoquerons le *problème du DSwnC (resp. IDSwnC, TDSwnC)* ou simplement le *DSwnC (resp. IDSwnC, TDSwnC)*, il sera question du *problème de déterminer l'existence d'un DSwnC (resp. IDSwnC, TDSwnC)*.

Dans un premier chapitre, nous nous intéresserons à la complexité des problèmes de domination sans conflit en fonction de la classe du graphe support et de la classe du graphe des conflits. Nous montrerons plusieurs résultats de NP-complétude et pointerons quelques cas polynomiaux. Ces résultats nous amèneront à un second chapitre dans lequel nous étudierons la complexité des instances selon d'autres paramètres, liés cette fois à la classe de l'union du graphe support et du graphe des conflits. La plupart des résultats obtenus seront encore des résultats de NP-complétude. La difficulté de ces problèmes mêmes dans des classes d'instances réduites nous mènera à dégrader nos solutions dans un troisième chapitre : nous étudierons des problèmes de *domination partielle* sans conflit sous l'angle de l'approximation. Ces problèmes seront définis formellement au début du chapitre.

Chapitre 1

Domination sans conflit

Dans ce chapitre, nous allons étudier en détail la complexité des problèmes DSwnC et IDSwnC selon les classes de leur graphe support et du graphe des conflits. Dans une première section, nous travaillerons sur des classes de graphes peu denses. Il est naturel de penser que les problèmes sont plus faciles à traiter dans des graphes peu denses, "simples". Nous verrons au travers de plusieurs résultats de NP-complétude que ce n'est pas le cas, sauf en de rares cas particuliers.

Dans une seconde section, nous nous intéresserons à des classes de graphes "opposées" : des graphes très denses, les *graphes de Dirac* (définis formellement plus loin) : dans des graphes suffisamment denses, les dominants minimaux et les indépendants sont plus petits, il pourrait donc être plus facile de déterminer l'existence d'une solution.

Dans une dernière section, nous proposerons un algorithme paramétrique pour une classe d'instance (NP-complète) pour le problème du dominant indépendant sans conflit.

La plupart de nos réductions seront basées sur le problème 3-SAT :
Définition 4. 3-SAT

Instance : (X, Cl) Un ensemble X de variables booléennes et un ensemble Cl de 3-clauses sur ces variables.
Question : Existe-t-il une affectation des variables de X vérifiant toutes les clauses de Cl ?

Ce problème est NP-complet, même dans le cas où chaque variable apparaît dans au plus 4 clauses (sous forme positive ou négative) [Tov84].

Le tableau 1.1 résume les résultats obtenus dans ce chapitre.

G		C		Résultat	
Classe	D	Classe	D	Existence	Thm
IDSwnC					
Chemin	2	-	1	NPC	1
Chemin	2	Dirac	-	NPC	5
Dirac	-	$\bigcup P_2P_4$	2	NPC	6
Dirac	-	Dirac	-	NPC	7
Chemin	2	épaisseur $= k$	-	$O(2^k * poly(n))$	12
DSwnC					
$\bigcup$ chemins	2	-	1	NPC	2
Chemin	2	$\bigcup P_1, P_2P_3$	2	NPC	3
Chemin	2	-	1	Toujours	4
Chemin	2	Dirac	-	NPC	8
Dirac	-	Dirac	-	NPC	9
Dirac	-	-	1	Toujours	10
*		-	1	Toujours	11

* : Séquence particulière de degrés
D : degré maximum

TABLE 1.1 – Tableau récapitulatif des résultats de complexité du chapitre 1

1.1 Graphes peu denses

Le but de cette section est de prouver la NP-complétude de déterminer l'existence d'un IDSwnC ou d'un DSwnC dans des classes de graphes très peu denses. La structure derrière les réductions pour ces deux problèmes est la même, bien que les résultats soient légèrement différents. Nous présenterons donc tout d'abord la structure de cette preuve, sous une forme générique, avant de l'instancier pour chacun des problèmes.

Nous allons réduire une restriction (NP-complète) de 3-SAT à nos problèmes. Pour cela, nous définissons des *gadgets de clause* qui simulent les clauses de 3-SAT. Nous ajoutons ensuite des conflits pour assurer la cohérence entre les clauses (un littéral et sa négation ne peuvent pas être simultanément positifs). Ceci forme la première partie de la réduction. Ensuite, dans le but de préciser nos résultats, nous utilisons un autre type de gadgets pour décomposer le graphe des conflits en un graphe encore moins dense (précisément un graphe de degré maximum 1). Finalement, un dernier gadget nous permet de connecter le graphe support et d'obtenir notre résultat le plus avancé. Pour cela nous définissons tout d'abord un problème et des gadgets abstraits et les propriétés qu'ils doivent vérifier, avant de les instancier pour chaque problème. Plus formellement :

Définition 5. Dans cette section, le problème $\mathcal{P}$ désigne un problème de

graphe avec conflits dont la solution est un ensemble de sommets, et pour lequel la famille des instances ayant une solution est fermée par union, c'est à dire que l'union de deux instances disjointes I_1 et I_2 ayant des solutions est une instance I_3 ayant une solution. De plus, nous imposons que l'union d'une solution de I_1 et d'une solution de I_2 soit une solution de I_3.

Définition 6 (Gadget de clause)**.** Un gadget de clause du problème $\mathcal{P}$ pour la clause $c_l = (x_i \vee x_j \vee x_k)$ est un graphe avec conflits (G_{c_l}, C_{c_l}) où G_{c_l} est un chemin et où le graphe des conflits C_{c_l} est une union disjointe de P_1, P_2, P_3, et x_i, x_j, x_k sont trois sommets identifiés de G_{c_l} tels que :

1. Toute solution S de $\mathcal{P}$ doit contenir au moins un des sommets x_i, x_j, x_k.
2. Pour tout sous-ensemble non vide X de x_i, x_j, x_k il doit exister une solution S sans conflit de $\mathcal{P}$ telle que $S \cap \{x_i, x_j, x_k\} = X$.
3. Aucune arête de C_{c_l} n'est incidente à x_i, x_j, x_k. (C'est à dire x_i, x_j, x_k ne sont pas en conflit dans C_{c_l})

On dit qu'il existe un gadget de clause pour le problème $\mathcal{P}$ si pour toute 3-clause, il est possible de construire un gadget de clause en temps polynomial.

Lemme 1. *Si $\mathcal{P}$ est un problème avec conflits pour lequel il existe un gadget de clause, alors $\mathcal{P}$ est NP-complet même si le graphe support est une union disjointe de chemins et que le graphe des conflits est une union de graphes bipartis complets d'au plus 4 sommets.*

Démonstration. Rappelons que le problème 3-SAT est NP-complet même dans les instances où chaque variable apparaît dans 4 clauses au maximum. Soit (X, Cl) une instance 3-SAT vérifiant ces contraintes. Construisons une instance (G, C) du problème $\mathcal{P}$: Pour chaque clause $c_l = (x_i \vee x_j \vee x_k)$ de Cl, on construit un gadget de clause (G_{c_l}, C_{c_l}) avec x_i, x_j, x_k ses trois sommets identifiés. Posons $G = \bigcup_{c_l}(G_{c_l})$ et $C' = \bigcup_{c_l}(C_{c_l})$. Construisons C en ajoutant à C' des arêtes pour créer des sous-graphes bipartis complets entre les sommets représentant x_i et les sommets représentant $\bar{x_i}$, pour toute paire $x_i, \bar{x_i}$ de littéraux opposés. Par définition, G est une union disjointe de chemins et C est une union disjointe de P_1, P_2, P_3 (venant des gadgets de clause) qui sont des bipartis complets à moins de 4 sommets, et de $K_{1,1}, K_{1,2}, K_{1,3}, K_{2,2}$ (provenant des conflits ajoutés entre les sommets représentant des littéraux opposés)

Supposons qu'il existe une solution A de l'instance 3-SAT. Construisons une solution S de $\mathcal{P}$ comme suit. Pour chaque littéral x_α étant **vrai** dans A, tous les sommets représentant x_α sont inclus dans S. Un littéral et sa négation ne pouvant être fixés à **vrai** simultanément, il n'y a pas de conflit entre les sommets sélectionnés. De plus, comme chaque clause est vérifiée, il existe au moins un sommet identifié appartenant à S dans chaque gadget de clause. D'après les propriétés 2 et 3 de la définition du gadget de clause, chaque gadget peut avoir une solution sans conflit de $\mathcal{P}$. Ainsi, comme la famille

des instances de $\mathcal{P}$ ayant une solution est fermée par union il existe une solution de $\mathcal{P}$ pour l'instance (G, C)

Supposons à présent qu'il existe une solution S au problème $\mathcal{P}$ dans l'instance (G, C). Construisons une affectation des variables de X. Pour chaque littéral x_i, x_i est **vrai** si et seulement si au moins un sommet de S représente x_i. Si la valeur de certaines variables de X n'est pas encore fixée, ces variables sont fixées à **faux**. Comme il existe des conflits entre les sommets représentant les littéraux opposés, l'affectation est bien cohérente. De plus, d'après la propriété 1 de la définition du gadget de clause, il existe dans S un sommet identifié de chaque gadget de clause, la clause correspondante est donc vérifiée. Toutes les clauses ayant un gadget, l'affectation construite est bien une solution de l'instance 3-SAT. □

Un *séparateur de conflits* est un graphe avec conflits conçu pour remplacer les sous-graphes de conflit bipartis complets par des graphes de degré maximum 1 sans changer l'existence d'une solution.

Définition 7 (Séparateur de conflits)**.** Un séparateur de conflits du problème $\mathcal{P}$ pour $K_{x,y}$ est un graphe avec conflits (G_{xy}, C_{xy}) tel que si (G, C) est une instance de $\mathcal{P}$ et $K_{x,y}$ est un sous graphe de C, alors $(G \cup G_{xy}, C - E(K_{x,y}) \cup C_{xy})$ a une solution si et seulement si (G, C) a une solution. De plus, G_{xy} et C_{xy} doivent être des graphes de degré au plus 1 , les arêtes de G_{xy} ne doivent pas être incidentes à $V(G)$, et les arêtes de C_{xy} ne doivent pas être incidentes à $V(G) - V(K_{x,y})$.

Lemme 2. *Si $\mathcal{P}$ est un problème avec conflits pour lequel il existe un gadget de clause et un séparateur de conflits pour chaque graphe biparti complet d'au plus* 4 *sommets, alors $\mathcal{P}$ est NP-complet même si le graphe support est une union disjointe de chemins et que le graphe des conflits est de degré au plus* 1*.*

Démonstration. Comme $\mathcal{P}$ a un gadget de clause, d'après le lemme 1, $\mathcal{P}$ est NP-complet même si le graphe support est une union disjointe de chemins et que le graphe des conflits est une union disjointe de graphes bipartis à au plus 4 sommets. De plus, $\mathcal{P}$ a un séparateur de conflits pour chacun de ces graphes bipartis. Ainsi, par définition, $\mathcal{P}$ est NP-complet même si le graphe support est une union disjointe de chemins et que le graphe des conflits est de degré maximum 1. □

Définition 8 (Connecteur neutre)**.** Un *connecteur neutre* pour le problème $\mathcal{P}$ est un graphe avec conflits (G_{nc}, C_{nc}) où G_{nc} est un chemin et C_{nc} son graphe des conflits tel que pour toute instance (G, C) de $\mathcal{P}$, connecter n'importe quelle paire de sommets de G par le chemin G_{nc} et ajouter les conflits C_{nc} ne change pas l'existence d'une solution.

Lemme 3. *Si $\mathcal{P}$ est un problème avec conflits pour lequel il existe un gadget de clause, un séparateur de conflit pour chaque graphe biparti complet d'au*

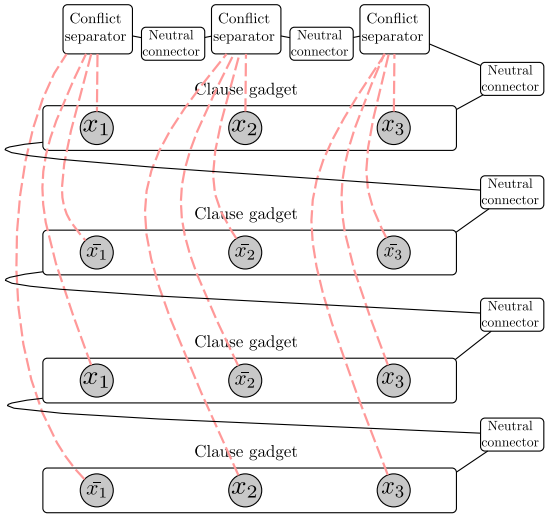


FIGURE 1.1 – Reduction de l'instance 3-SAT $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_3)$ à une instance du problème $\mathcal{P}$

plus 4 *sommets et un connecteur neutre, alors* $\mathcal{P}$ *est NP-complet même si le graphe support est un chemin, et que le graphe des conflits est l'union disjointe d'un graphe de degré maximum* 1 *et de copies de* C_{nc}

Démonstration. Comme $\mathcal{P}$ a un gadget de clause et un séparateur de conflits pour chaque graphe biparti d'au plus 4 sommets, d'après le lemme 2, $\mathcal{P}$ est NP-complet même si le graphe support est une union disjointe de chemins et que le graphe des conflits est de degré maximum 1. De plus, $\mathcal{P}$ a un connecteur neutre (G_{nc}, C_{nc}), ainsi, si (G, C) est une instance de $\mathcal{P}$ de ce type, il est possible de connecter les chemins de G pour former un chemin unique sans changer l'existence d'une solution. C devient l'union disjointe d'un graphe de degré maximum 1 et de copies de C_{nc}. □

Une illustration de cette réduction est présentée dans la figure 1.1.

1.1.1 Dominant indépendant sans conflit dans les graphes peu denses

Nous allons maintenant prouver une série de lemmes pour montrer que ID-SwnC vérifie toutes les hypothèses du lemme 3, d'où découlera le théorème 1.

Lemme 4. *La famille des instances de IDSwnC ayant une solution est fermée par inclusion, et pour toutes solutions* S_1 *et* S_2 *de deux instances* I_1 *et* I_2, $S_3 = S_1 \cup S_2$ *est une solution de* $I_3 = I_1 \cup I_2$.

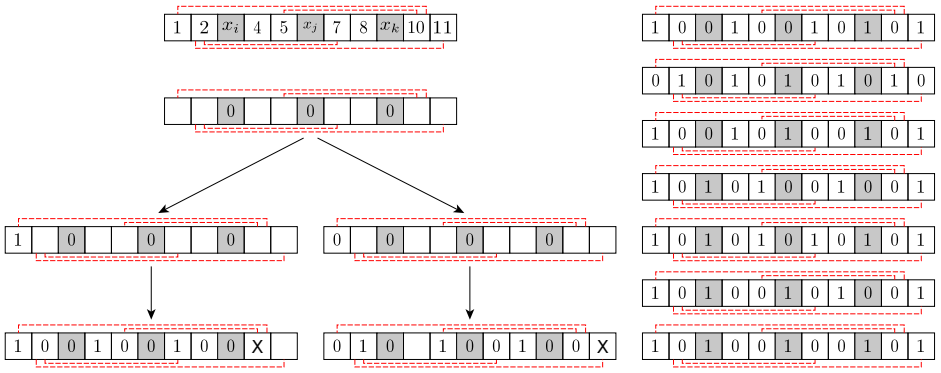


FIGURE 1.2 – Gadget de clause pour IDSwnC.

Démonstration. Soient $I_1 = (G_1, C_1)$ et $I_2 = (G_2, C_2)$ deux instances ayant des solutions S_1 et S_2, et soit $I_3 = (G_3 = G_1 \cup G_2, C_3 = C_1 \cup C_2)$ l'union de I_1 et I_2. Notons $S_3 = S_1 \cup S_2$. Supposons que S_3 ne soit pas une solution de I_3, il y a donc trois possibilités :

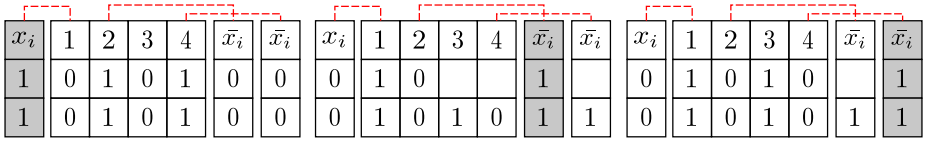
- S_3 n'est pas un dominant de G_3. Il existe donc un sommet v non dominé par S_3. Supposons sans perte de généralité que $v \in G_1$. Comme $S_1 \in S_3$, v n'est pas dominé par S_1, mais S_1 est un dominant de G_1, une contradiction.
- S_3 n'est pas un indépendant dans G_3. Il existe donc deux sommets voisins u et v de S_3. Les arêtes de G_3 proviennent de G_1 ou de G_2 uniquement. Supposons sans perte de généralité que l'arrête uv appartienne à G_1. Alors u et v appartiennent à S_1, une contradiction.
- S_3 n'est pas sans conflit dans C_3. Il existe donc deux sommets en conflit u et v de S_3. Les conflits de C_3 proviennent de C_1 ou de C_2 uniquement. Supposons sans perte de généralité que le conflit uv appartienne à C_1. Alors u et v appartiennent à S_1, une contradiction.

Tous les cas sont des contradictions, ainsi S_3 est une solution de I_3 et le lemme est vérifié. □

Lemme 5. *Il existe un gadget de clause pour IDSwnC.*

Démonstration. Un gadget pour une clause $c_l = (x_i \vee x_j \vee x_k)$ peut être (G_{c_l}, C_{c_l}) où G_{c_l} est un chemin $(1, 2, x_i, 4, 5, x_j, 7, 8, x_k, 10, 11)$ et les arêtes de C_{c_l} sont entre 1 et 10, 2 et 7, 2 et 11, 5 et 10.

De manière évidente, G_{c_l} est un chemin, C_{c_l} est une union disjointe de P_1 et P_3, et aucune arête de C_{c_l} n'est incidente à x_i, x_j, x_k.
Par recherche exhaustive et propagation de contraintes, on remarque que pour n'importe quel choix fait sur les autres sommets, il est impossible de trouver un IDSwnC de (G_{c_l}, C_{c_l}) si aucun sommet parmi x_i, x_j, x_k n'est choisi. Cette recherche exhaustive est résumée dans l'arbre de la figure 1.2.

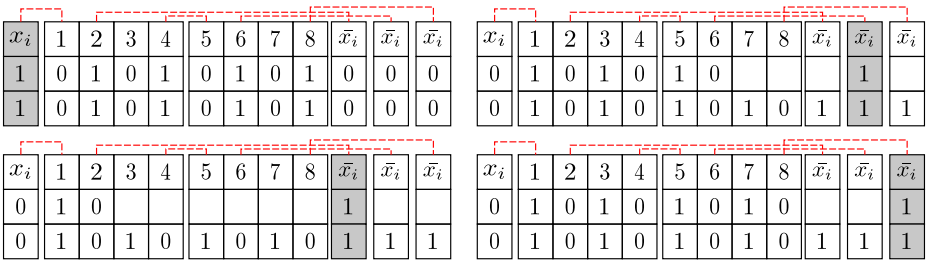


x_i	1	2	3	4	$\bar{x}_i$	$\bar{x}_i$
1	0	1	0	1	0	0
1	0	1	0	1	0	0

x_i	1	2	3	4	$\bar{x}_i$	$\bar{x}_i$
0	1	0			1	
0	1	0	1	0	1	1

x_i	1	2	3	4	$\bar{x}_i$	$\bar{x}_i$
0	1	0	1	0		1
0	1	0	1	0	1	1

FIGURE 1.3 – Séparateur de conflit pour K_{12}.



x_i	1	2	3	4	5	6	7	8	$\bar{x}_i$	$\bar{x}_i$	$\bar{x}_i$
1	0	1	0	1	0	1	0	1	0	0	0
1	0	1	0	1	0	1	0	1	0	0	0

x_i	1	2	3	4	5	6	7	8	$\bar{x}_i$	$\bar{x}_i$	$\bar{x}_i$
0	1	0	1	0	1	0				1	
0	1	0	1	0	1	0	1	0	1	1	1

x_i	1	2	3	4	5	6	7	8	$\bar{x}_i$	$\bar{x}_i$	$\bar{x}_i$
0	1	0							1		
0	1	0	1	0	1	0	1	0	1	1	1

x_i	1	2	3	4	5	6	7	8	$\bar{x}_i$	$\bar{x}_i$	$\bar{x}_i$
0	1	0	1	0	1	0	1	0			1
0	1	0	1	0	1	0	1	0	1	1	1

FIGURE 1.4 – Séparateur de conflit pour K_{13}.

Dans la partie droite de la figure 1.2, pour tout sous-ensemble X non vide de x_i, x_j, x_k nous montrons S, un IDSwnC tel que $S \cap \{x_i, x_j, x_k\} = X$. □

Lemme 6. *Il existe un séparateur de conflits pour IDSwnC pour $K_{1,2}$, $K_{2,2}$ et $K_{1,3}$.*

Démonstration. Les séparateurs de conflits pour $K_{1,2}, K_{2,2}, K_{1,3}$ sont respectivement montrés dans les figures 1.3, 1.4 et 1.5. Les sommets x_i forment une partition et les sommets $\bar{x}_i$ l'autre partition. Pour chaque tableau, la première ligne contient les labels des sommets. x_i et $\bar{x}_i$ sont les sommets de l'instance initiale, et les nombres représentent les sommets ajoutés. Les cases adjacentes représentent des sommets adjacents. Les conflits sont représentés en pointillés. La deuxième ligne montre quels sommets sont forcés d'appartenir (ou non) à la solution quand un des sommets initiaux (en gris) est choisi. Les cases vides représentent les sommets non contraints. Choisir un sommet x_i (resp. $\bar{x}_i$) doit forcer tous les sommets $\bar{x}_i$ (resp x_i) à ne pas appartenir à une solution, mais ne doit pas empêcher les autres sommets de sa partition d'être sélectionnés. La troisième ligne des tableaux montre un exemple d'une solution où tous les sommets dans la même partition que le sommet gris sont sélectionnés. Ceci permet de montrer que l'ensemble de chemins proposé vérifie les propriétés demandées pour les séparateurs de conflits. □

Lemme 7. *Il existe un connecteur neutre pour IDSwnC et son graphe des conflits est de degré au plus* 1.

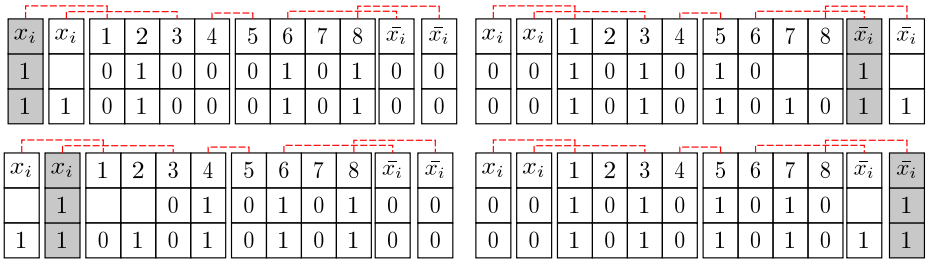


FIGURE 1.5 – Séparateur de conflit pour K_{22}.

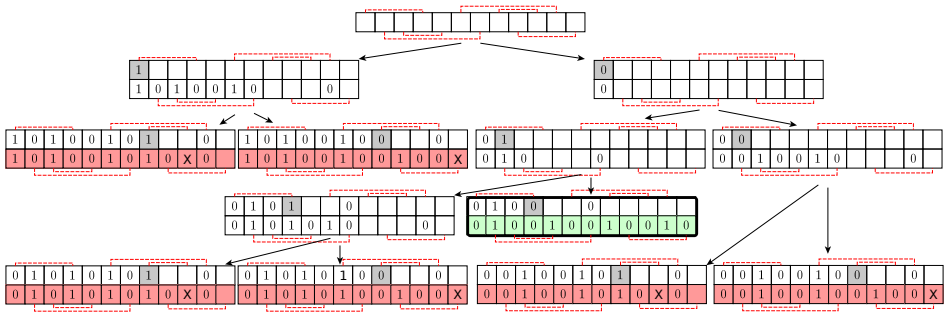


FIGURE 1.6 – Connecteur neutre pour IDSwnC

Démonstration. Un connecteur neutre possible est (G_{nc}, C_{nc}) où $G_{nc} = q_1, ..., q_{12}$ et les arêtes de C_{nc} sont $\{q_1q_4, q_2q_7, q_3q_5, q_6q_{11}, q_8q_{10}, q_9q_{12}\}$. G_{nc} est un chemin et C_{nc} un graphe de degré maximum 1. Il suffit de montrer que connecter les extrémités de G_{nc} à des sommets d'un graphe ne change pas l'existence d'un IDSwnC.

Si le premier (ou dernier) sommet de G_{nc} peut appartenir à une solution, alors ce sommet pourrait dominer son voisin à l'extérieur de G_{nc} : nous devons nous assurer que ceci est impossible. Inversement, nous devons montrer que G_{nc} peut être dominé par lui même, quel que soit son voisinage. Ceci est fait par une recherche exhaustive, exposée dans la figure 1.6. Dans la première ligne de chaque tableau, la dernière case remplie représente le choix fait pour ce sommet (1 s'il appartient à la solution, 0 sinon), et la seconde ligne montre les forçages impliqués par ce choix.

La seule solution pour dominer G_{nc} est indiquée en vert, et ne contient pas les extrémités du chemin. □

Théorème 1. *IDSwnC est NP-complet même si le graphe support est un chemin et que le graphe des conflits est de degré au plus* 1.

Démonstration. Ce théorème découle des lemmes 3, 4 5, 6 et 7. □

1.1.2 Dominant sans conflit dans les graphes peu denses

Lemme 8. *La famille des instances de DSwnC ayant une solution est fermée par inclusion, et pour toutes solutions S_1 et S_2 de deux instances I_1 et I_2, $S_3 = S_1 \cup S_2$ est une solution de $I_3 = I_1 \cup I_2$.*

Démonstration. Soient $I_1 = (G_1, C_1)$ et $I_2 = (G_2, C_2)$ deux instances ayant des solutions S_1 et S_2, et soit $I_3 = (G_3 = G_1 \cup G_2, C_3 = C_1 \cup C_2)$ l'union de I_1 et I_2. Notons $S_3 = S_1 \cup S_2$. Supposons que S_3 ne soit pas une solution de I_3, il y a donc deux possibilités :

- S_3 n'est pas un dominant de G_3. Il existe donc un sommet v non dominé par S_3. Supposons sans perte de généralité que $v \in G_1$. Comme $S_1 \in S_3$, v n'est pas dominé par S_1, mais S_1 est un dominant de G_1, une contradiction.
- S_3 n'est pas sans conflit dans C_3. Il existe donc deux sommets en conflit u et v de S_3. Les conflits de C_3 proviennent de C_1 ou de C_2 uniquement. Supposons sans perte de généralité que le conflit uv appartienne à C_1. Alors u et v appartiennent à S_1, une contradiction.

Tous les cas sont des contradictions, ainsi S_3 est une solution de I_3 et le lemme est vérifié. □

Comme pour le IDSwnC, nous prouvons une série de lemmes pour montrer que DSwnC vérifie tous les critères des lemmes 2 et 3, d'où proviendront les résultats de NP-complétude. Les propriétés des gadgets pouvant être vérifiées par recherche exhaustive de la même manière que pour le IDSwnC, cette recherche sera omise dans les preuves.

Lemme 9. *Il existe un gadget de clause pour DSwnC.*

Démonstration. Un gadget de clause possible pour la clause $c_l = (x_i \vee x_j \vee x_k)$ est (G_{c_l}, C_{c_l}) où G_{c_l} est le chemin p_1, x_i, x_j, x_k, p_2 et C_{c_l} n'a pas d'arête. □

Lemme 10. *Il existe des séparateurs de conflits pour DSwnC pour $K_{1,2}$, $K_{2,2}$, $K_{1,3}$.*

Démonstration. Un ensemble possible de séparateurs de conflits est montré à la figure 1.7. □

Théorème 2. *DSwnC est NP-complet même si le graphe support est une union disjointe de chemins et le graphe des conflits est de degré au plus* 1.

Démonstration. Le théorème découle des lemmes 2, 8, 9 et 10. □

Remarque 1. Le théorème 2 est également vrai si l'on impose au graphe des conflits d'être de degré régulier égal à 1. Il suffit par exemple d'ajouter à tout sommet s sans conflit un gadget comme montré dans la figure 1.8.

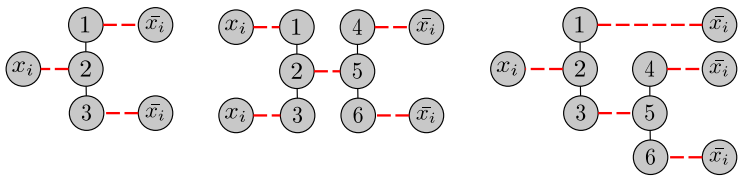


FIGURE 1.7 – De gauche à droite, des séparateurs de conflits pour $K_{1,2}, K_{2,2}$ et $K_{1,3}$ pour DSwnC

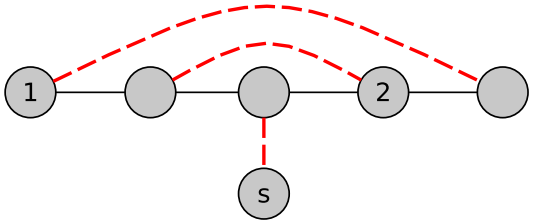


FIGURE 1.8 – Gadget pour transformer le graphe des conflits en couplage parfait.

Lemme 11. *Il existe un connecteur neutre pour DSwnC et son graphe des conflits est une union disjointe de* P_1, P_2, P_3.

Démonstration. Un connecteur neutre est montré figure 1.9. □

Théorème 3. *DSwnC est NP-complet même si le graphe support est un chemin et que le graphe des conflits est une union disjointe de* P_1, P_2, P_3.

Démonstration. Le théorème découle des lemmes 3, 9, 10 et 11. □

Les résultats de NP-complétude pour DSwnC sont un peu plus faibles que ceux obtenus pour IDSwnC. Nous montrons que ces résultats sont dans un sens les plus forts possibles, par le théorème suivant.
Théorème 4. *Si* G_P *est un chemin et* C_M *est un graphe de degré maximum* 1*, alors* (G_P, C_M) *a un DSwnC.*

Démonstration. Soit G_P un chemin et C_M un graphe de degré maximum 1. Construisons une instance (C, X) de SAT comme suit. Soit P' le chemin

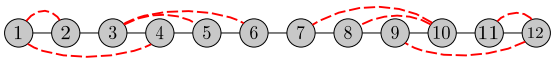


FIGURE 1.9 – Connecteur neutre pour DSwnC

composé des sommets adjacents aux arêtes de C_M, dans le même ordre que dans G_P. Posons $X = V(P')$. Pour chaque $i \in \{0, ..., |X|/2 - 1\}$ la clause $(p'_{2i+1} \vee p'_{2i+2})$ est créée. Pour chaque arête ab de C_M, la clause $(\bar{a} \vee \bar{b})$ est créée.

D'après le lemme 3.2 dans [Tov84], une instance de SAT dans laquelle toutes les clauses ont au moins 2 variables et dans laquelle chaque variable apparait exactement une fois sous forme positive et une fois sous forme négative a toujours une solution. C'est le cas de (C, X).

Ainsi, soit A une affectation sur X vérifiant toutes les clauses de C, et soit S l'ensemble des sommets correspondant à des littéraux fixés à **vrai**. Soit ab une arête de C_M. Il existe une clause $(\bar{a} \vee \bar{b})$ et donc au maximum un de ses sommets appartient à S. Ainsi, S est sans conflit. Soit a_{2k}, a_{2k+1} deux sommets consécutifs de P'. Alors la clause $(a_{2k} \vee a_{2k+1})$ existe, et un des sommets appartient à S. S est donc un DSwnC de P'.

Si $P' = G_P$, alors nous avons un DSwnC de (G_P, C_M) et le théorème est vrai. Sinon, soit $S_1 = S \bigcup_{x \notin V(P')} \{x\}$. Alors S_1 est sans conflit, car les nouveaux sommets ne sont pas en conflit. De plus, les sommets n'appartenant pas à $V(P')$ sont sélectionnés, et donc dominés, et les voisins également. Les autres sommets ont le même voisinage que dans P' et sont donc toujours dominés. Ainsi, S_1 est un DSwnC de (G_P, C_M). □

1.2 Graphes denses

Les théorèmes 1, 2 et 3 ont montré que décider de l'existence d'un IDSwnC ou d'un DSwnC était NP-complet, même dans le cas de graphes très peu denses, et avec des graphes de conflits très peu denses. Dans cette section, nous allons étudier la complexité de ces deux problèmes dans les graphes denses, et proposer plusieurs résultats pour les graphes de Dirac. Un graphe de Dirac à n sommets est un graphe dont chaque sommet est de degré au moins $n/2$.

1.2.1 Dominant indépendant dans les graphes denses

Théorème 5. *Étant donné (G_P, C_D) un graphe avec conflits, décider de l'existence d'un IDSwnC est NP-complet même si G_p est un chemin et C_D est un graphe de Dirac.*

Démonstration. Nous réduisons une instance de IDSwnC où le graphe support est un chemin et où le graphe des conflits est de degré maximum 1 à une instance de IDSwnC où le graphe support est un chemin et le graphe des

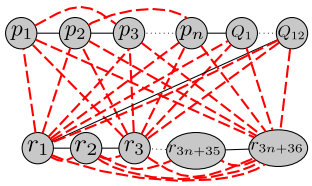


FIGURE 1.10 – Transformation depuis un chemin $p_1, ..., p_n$ et ses conflits vers un chemin avec conflits où les conflits sont un graphe de Dirac.

conflits est un graphe de Dirac. Le premier problème est NP-complet d'après le théorème 1. Soit (G_{P_0}, C_M) une instance du IDSwnC où G_{P_0} est un chemin et C_M est un graphe de degré maximum 1. Construisons (G_P, C_D) où G_P est un chemin et C_D est un graphe de Dirac tel qu'il existe un IDSwnC de (G_P, C_D) si et seulement si il existe un IDSwnC de (G_{P_0}, C_M).

Notons n la taille de G_{P_0}. Rappelons que $P_1 + P_2$ dénote la concaténation du chemin P_1 et du chemin P_2. $G_{P_1} = G_{P_0} + G_{nc}$ est de longueur $n+12$, où (G_{nc}, C_{nc}) est le connecteur neutre pour IDSwnC montré dans la figure 1.6.

Soit $C_1 = C_M \cup C_{nc}$. Créons G_{P_2} un chemin de longueur $3n + 36$ où $G_{P_2} = r_1, ..., r_{3n+36}$ et créons un graphe des conflits C_2 tel que $C_2 = \bigcup_{i=1}^{3n+36} \bigcup_{j=1}^{3n+36, j\neq 2 \pmod 3, j\neq i} \{r_i r_j\}$. Il existe S_2 un IDSwnC de (G_{P_2}, C_2), où $S_2 = \bigcup_{i=1}^{3n+36} \{r_i | i = 2 \pmod 3\}$.

Soit C_{Bip} l'ensemble d'arêtes d'un graphe biparti complet entre les sommets de G_{P_1} et les sommets de $\bigcup_{i=1}^{3n+36} \{r_i | i \neq 1 \pmod 3\}$.

Soit $G_P = G_{P_1} + G_{P_2}$ et $C_D = C_1 \cup C_2 \cup C_{Bip}$. Un schéma du graphe est montré à la figure 1.10.

Supposons qu'il existe une solution S_0 de (G_{P_0}, C_M). Alors, par définition du connecteur neutre, il existe une solution S_1 de (G_{P_1}, C_1). Soit $S = S_1 \cup S_2$. Alors, par construction, S est un indépendant dominant de G_P. De plus S est sans conflit dans C_D, ainsi S est un IDSwnC (G_P, C_D).

Supposons maintenant qu'il existe S un IDSwnC de (G_P, C_D). Soit $S_0 = S \cap G_{P_0}$. Comme $S_0 \subseteq S$, S_0 est un indépendant dans G_{P_0}, sans conflit dans C_M. De plus, comme le seul voisinage de G_{P_0} est une extrémité du connecteur neutre, on sait que S_0 est un dominant de G_{P_0}, et donc S_0 est un IDSwnC de (G_{P_0}, C_M).

Pour compléter la preuve du théorème 5, nous devons maintenant montrer que C_D est un graphe de Dirac. C_D contient $n + 12 + 3n + 36 = 4(n + 12)$ sommets. Son degré minimal doit donc être au moins $2n + 24$. Or, chaque sommet de C_D est en conflit avec au moins $2n+24$ sommets de $\bigcup_{i=1}^{3n+36} \{r_i | i \neq 1 \pmod 3\}$. Ansi, chaque sommet a bien un degré d'au moins $2n+24$. □

Théorème 6. *Étant donné (G, C) un graphe avec conflits, décider de l'existence d'un IDSwnC est NP-complet même si G est un graphe de Dirac et C est une union disjointe de P_2 et P_4 .*

Démonstration. Soit (G_1, C_1) une instance de IDSwnC. Supposons sans perte de généralité que G_1 n'a pas de sommets isolés, que C_1 est un graphe de degré au maximum 1 et que le plus petit IDSwnC est de taille au moins 2. Soit $G_1 = (V_1, E_1)$ et $V_1 = \{v_1, ..., v_n\}$.

Construisons $G' = (V', E')$ comme il suit : $V' = V_1 \cup V_2$, avec $V_2 = \{x_1, ..., x_n\}$ et $E' = E_1 \cup E_2 \cup E_3$, avec $E_2 = \bigcup_i \bigcup_{j \neq i} \{x_i v_j\}$ (les arêtes d'un graphe biparti complet moins un matching) et $E_3 = \bigcup_i \bigcup_{j \neq i} \{x_i x_j\}$ (les arêtes d'un graphe complet). Soit $C' = C_1 \bigcup_i \{v_i x_i\}$.

G' est un graphe à $2n$ sommets. Les sommets de V_1 sont de degré au moins $1 + (n-1) = n$ et les sommets de V_2 sont de degré au moins $2(n-1) \geq n$, G' est donc un graphe de Dirac. C' est une union disjointe de P_2 et P_4.

Soit S un IDSwnC de (G_1, C_1). Par construction, S a au moins 2 sommets. Aucune arête n'est ajoutée entre les sommets de V_1, donc S est un indépendant dans G'. Aucun conflit n'est ajouté entre les sommets de V_1, donc S est sans conflit dans C'. S domine V_1. Soient v_i et v_j deux sommets distincts de S. v_i domine $V_2 - \{x_i\}$ et v_j domine x_i. Ainsi S est un IDSwnC de (G', C').

Soit S un IDSwnC de (G', C'). Supposons qu'il existe $x_i \in V_2 \cap S$. Tous les sommets de G' à l'exception de v_i sont voisins de x_i et ne peuvent pas appartenir à S. De plus, $x_i v_i \in C'$ et $S = x_i$, mais x_i ne domine pas v_i : une contradiction. Ainsi $S \subseteq V_1$, donc S est un IDSwnC de (G_1, V_1). □

Théorème 7. *Étant donné (G, C) un graphe avec conflits, décider s'il existe un IDSwnC est NP-complet même si G et C sont des graphes de Dirac.*

Démonstration. Soit (G, C) avec $G = (V, E)$ et $C = (V, F)$ un graphe avec conflits comme décrit dans la preuve du théorème 6. Construisons une nouvelle instance (G', C'), avec $G' = (V, E)$ et $C' = (V, F \cup E)$. Ajouter un conflit entre deux sommets voisin ne change pas l'existence d'un IDSwnC. Le nouveau graphe des conflits est un graphe de Dirac. □

1.2.2 Dominant dans les graphes denses

Théorème 8. *Étant donné (G, C) un graphe avec conflits, décider de l'existence d'un DSwnC est NP-complet, même si G est un chemin et C est un graphe de Dirac.*

Démonstration. Soit (G_P, C_D) une instance de IDSwnC avec G_P un chemin et C_D un graphe de Dirac. Ce problème est NP-complet d'après le théorème 5. Construisons une instance (G, C) de DSwnC avec $C = C_D \cup E(G_P)$. Par construction, (G_P, C_D) a un IDSwnC si et seulement si (G, C) a un DSwnC. Après l'ajout de nouvelles arêtes, le graphe des conflits est toujours un graphe de Dirac. □

Théorème 9. *Étant donné (G, C) un graphe avec conflits, décider de l'existence d'un DSwnC est NP-complet, même si G et C sont des graphes de Dirac.*

Démonstration. Soit (G, C) un graphe avec conflits tel que décrit dans la preuve du théorème 7. Chaque arête du graphe support est également une arête du graphe des conflits, ainsi, chaque DSwnC est indépendant dans G, et donc un IDSwnC. □

Nous montrons maintenant que si le graphe support est suffisamment dense et que le graphe des conflits est suffisamment léger, alors il existe toujours un DSwnC.

Théorème 10. *Étant donné (G, C) un graphe avec conflits, si G est un graphe de Dirac et C est de degré maximum* 1*, alors il existe toujours un DSwnC de (G, C).*

Démonstration. Soit D un ensemble contenant un sommet arbitraire de chaque arête du graphe des conflits, ainsi que tous les sommets sans conflit. Par construction, D est sans conflit, et comme C est de degré maximum 1, D est de taille au minimum $n/2$, ainsi $|V - D| \leq n/2$.

Les sommets de D sont dominés. Soit x un sommet n'appartenant pas à D. Comme G est un graphe de Dirac, x a au moins $n/2$ voisins. Or, $|V - D|$ a au plus $n/2$ sommets : x a donc au moins un voisin dans D, et est donc dominé. Ainsi D est un DSwnC de (G, C). □

La preuve du théorème 10 se base sur le fait que le voisinage de chaque sommet est suffisamment grand par rapport au nombre de conflits. Nous allons voir qu'en construisant le dominant sommet par sommet, nous pouvons relâcher la contrainte sur le degré de certains sommets du graphe support et ainsi étendre le résultat à une classe de graphes support plus grande. On note $d(x)$ le degré d'un sommet x.

Théorème 11. *Étant donné (G, C) un graphe avec conflits, soit $d(x_1), ...d(x_n)$ la séquence des degrés des sommets de G triés par ordre croissant. Si pour tout i, $d(x_i) \geq min((i-1), n/2)$ et C est de degré maximum* 1*, alors il existe toujours un DSwnC de (G, C).*

Démonstration. L'algorithme 2 retourne un DSwnC pour une telle classe d'instance. Cet algorithme parcourt l'ensemble des sommets triés par ordre croissant de degré. Pour chaque sommet x non dominé, l'algorithme va l'ajouter à la solution s'il n'est pas en conflit avec un élément de la solution, et ajoutera un de ses voisins y sinon. Le tout est polynomial. Nous allons montrer que l'algorithme retourne un DSwnC de (G, C). Il s'agit ici de prouver qu'il existe toujours un tel sommet y.

Remarquons que lors du traitement du sommet x_i, l'ensemble S contient au plus $i-1$ sommets. Si x_i est dominé, rien ne se passe. Si x_i est non dominé et n'est pas en conflit avec les sommets de S, alors x_i est ajouté à S. Si x_i est non dominé et en conflit avec un sommet de S, il y a deux possibilités :

(1) $i \leq (n/2) - 1$. Dans ce cas, on a $d(x_i) \geq i - 1$, ainsi x_i a au moins $i-1$ voisins, or S a au plus $i-1$ sommets, chacun en conflit avec au plus un sommet (car C est de degré maximum 1). Comme il existe un sommet de S en conflit avec x_i, il existe au moins un voisin de x_i sans conflit avec S.

(2) $i \geq (n/2)$. Dans ce cas, on a $d(x_i) \geq (n/2)$. Or, comme C est de degré maximum 1, il y a au plus $n/2$ conflits, il existe donc au moins un voisin de x_i sans conflit avec S.

Dans tous les cas, il est possible de dominer le sommet sans conflit, ainsi l'algorithme trouve bien un DSwnC de (G, C)

Algorithm 2: Trouver un DSwnC de (G, C)

Data: (G, C) un graphe avec conflit respectant les conditions du théorème 11
Result: Un dominant sans conflit de (G, C)
Soit $V = x_1, ...x_n$ la séquence des sommets de G, classés par ordre croissant de degré.
$S \leftarrow \emptyset$
while *S ne domine pas G* **do**
  Soit x_i le premier sommet non dominé de V dans l'ordre de la séquence
  if *x_i n'est pas en conflit avec un sommet de S* **then**
    $S \leftarrow S \cup \{x_i\}$
  else
    Soit y un voisin de x_i sans conflit avec les sommets de S
    $S \leftarrow S \cup \{y\}$
return S

□

1.3 Complexité paramétrique

Dans une section précédente, nous avons prouvé la NP-complétude de IDSwnC et DSwnC quand le graphe support était un chemin, même si le graphe des conflits était très peu dense. Dans cette section, nous nous intéresserons à un paramètre : l'*épaisseur* du graphe des conflits quand le graphe support est un chemin.

Étant donné un graphe avec conflits (P, C) avec $P = (V, E)$ un chemin sur les sommets $p_1, p_2, ..., p_n$ et $C = (V, F)$, alors l'épaisseur de (P, C) est $max_i(|\{p_j p_k \in F : j \leq i < k\}|)$. Ceci représente la coupe maximum dans le graphe des conflits dans l'ordre naturel du chemin.

Dans cette section, nous prouvons que si (G, C) est un chemin avec conflits d'épaisseur bornée, alors il est possible de déterminer en temps polynomial si (G, C) a un IDSwnC ou un DSwnC, et nous donnons la description d'un algorithme permettant de trouver une solution si elle existe.

Pour cela, nous introduisons quelques notations de théorie des langages et d'automates. Σ est un ensemble fini de symboles, appelé *alphabet*. Un *mot* est une séquence finie de symboles. Un *langage* L est un ensemble de mots. Un automate à états fini déterministe (DFA) M est un quintuplet $(\Sigma, Q, q_i, Q_f, \delta)$ où Σ est l'alphabet, Q un ensemble d'états, $q_i \in Q$ l'état initial, $Q_f \subseteq Q$ l'ensemble d'états finaux et $\delta \subseteq Q \times \Sigma \times Q$ est la fonction de transition. Un mot est *accepté* par M si son traitement termine dans un état final. L'ensemble de mots acceptés par M est appelé le langage accepté par M. Le *mot vide* est noté ϵ. Une transition de l'état a à l'état b par le symbole α est notée $a \rightarrow_\alpha b$. Si L_1 et L_2 sont deux langages réguliers, alors leur intersection $L_1 \cap L_2$ est régulière et le DFA reconnaissant $L_1 \cap L_2$ est le produit cartésien du DFA reconnaissant L_1 et du DFA reconnaissant L_2. Pour plus de détails sur les automates et les langages, voir par exemple [Linon].

Nous définissons quelques langages. Si G est un graphe à n sommets avec un ordre sur ces sommets et que w est un mot binaire de longueur n, w est interprété comme un sous-ensemble de sommets de G. Un symbole 1 en position i signifie que le $i$ème sommet de G appartient au sous-ensemble, 0 signifie qu'il n'y appartient pas. L_{DOM} est l'ensemble des mots représentant les dominants des chemins de taille arbitraire. L_{IDS} est l'ensemble des mots représentant des dominants indépendants sur des chemins de longueur arbitraire. Si C est un graphe avec un ordre sur ses sommets, L_C est le langage représentant les indépendants de C, c'est-à-dire le langage des mots sans conflit dans C. Enfin, $L_{IDSwnC(P,C)}$ est le langage représentant les IDSwnC dans un chemin avec conflits (P, C) et $L_{DSwnC(P,C)}$ est le langage représentant les DSwnC dans un chemin avec conflits (P, C).

Lemme 12. *Le langage L_{IDS} est reconnu par le DFA M_{IDS}.*

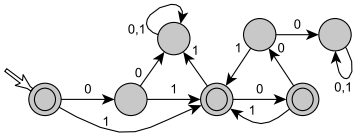


FIGURE 1.11 – DFA M_{IDS} reconnaissant le langage L_{IDS}.

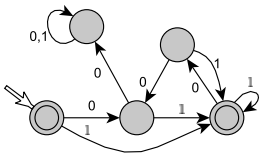


FIGURE 1.12 – DFA M_{DOM} reconnaissant le langage L_{DOM}.

Démonstration. L_{IDS} est l'ensemble des mots sans facteur 11 (indépendance) ou 000 (domination) et sans 00 comme préfixe ou suffixe (domination). De plus, $0 \notin L_{IDS}$.

Il est facile de voir que le DFA montré figure 1.11 reconnaît L_{IDS}. □

Lemme 13. *Le langage L_{DOM} est reconnu par l'automate M_{DOM}.*

Démonstration. L_{DOM} est le langage sans facteur 000 et sans 00 comme préfixe ou suffixe. De plus, $0 \notin L_{DOM}$.

Il est facile de voir que le DFA présenté figure 1.12 reconnaît L_{DOM}. □

Construction d'un automate reconnaissant les mots sans conflits Étant donné (P, C) un chemin avec conflits où $C = (V, E)$,nous allons construire un DFA M_C le plus petit possible reconnaissant L_C, le langage des mots sans conflit dans C.

L'idée derrière la construction de M_C est la suivante. Les sommets de C sont traités dans l'ordre du chemin. Pour un graphe à n sommets, les états de M_C sont divisés en $n + 1$ colonnes, correspondant au traitement des n sommets. Quand un sommet est traité, le DFA doit "mémoriser" sa valeur dans sa structure, pour pouvoir détecter d'éventuels conflits plus tard. Pour cela, dans les colonnes suivantes, le nombre d'états est doublé pour capturer les deux possibilités pour le sommet (dans le sous-ensemble ou non). Quand tous les voisins d'un sommet ont été traités, sa valeur n'a plus d'importance et ne doit plus être encodée dans la structure du DFA : dans les colonnes suivantes, le nombre d'états est réduit de moitié.

Les sommets de la colonne i sont étiquetés par des mots de longueur i

représentant les préfixes reconnus. Un 0 en position j signifie que le $j$ème sommet n'appartient pas au sous-ensemble. Un 1 en position j signifie que le $j$ème sommet appartient au sous-ensemble. Un ? en position j signifie que l'information sur le $j$ème sommet n'est plus utile.

Soit $V = v_1, ..., v_n$ les sommets de C dans l'ordre du chemin. Le DFA M_C a un état initial étiqueté ϵ et un état non final "poubelle" noté R. Les états de M_C sont divisés en $n+1$ colonnes étiquetées de 0 à n. L'état ϵ est dans la colonne 0 et l'état R dans la colonne n.

Les colonnes d'états sont construites séquentiellement de 0 à n, de gauche à droite. La première colonne contient uniquement l'état ϵ.

Supposons M_C construit jusqu'à la $(i-1)$ème colonne et introduisons quelques notations. Soit v_i le $i$ème sommet du graphe dans l'ordre du chemin. Soit $H = \{h_1, ..., h_{n_0}\}$ l'ensemble des indices des voisins de v_i qui apparaissent avant v_i dans l'ordre du chemin et dont v_i est le dernier voisin. Soit $H' = \{h'_1, ..., h'_{n_1}\}$ l'ensemble des indices des voisins de v_i qui apparaissent avant v_i dans l'ordre du chemin et pour lesquels v_i n'est pas le dernier voisin. Soit $J = \{j_1, ..., j_{n_2}\}$ l'ensemble des indices des voisins de v_i qui apparaissent après lui dans l'ordre du chemin. Soit $M = \{m_1, ..., m_{n_3}\}$ l'ensemble des états de la colonne $i-1$. Si X est un ensemble d'indices et m est un mot, soit $m[X]$ la chaine des symboles de m aux indices de X. Si a est un symbole, $m[X = a]$ est le mot m où chaque symbole à un indice de X est remplacé par le symbole a.

Nous créons maintenant des transitions depuis les états existants de M_C. La création d'une transition $a \to b$ implique la création d'un état b dans la colonne i s'il n'existe pas encore.

(1) Si J est vide, v_i n'a pas de voisin à droite, sa valeur n'est plus utile.
   (a) Pour chaque état m (de la colonne $i-1$) tel que $1 \in m[H \cup H']$, créer les transitions $m \to_0 m[H =?]?$ et $m \to_1 R$
   (b) Pour chaque état m (de la colonne $i-1$) tel que $1 \notin m[H \cup H']$, créer la transition $m \to_{0,1} m[H =?]?$

(2) Si J n'est pas vide, v_i a des voisins à droite, sa valeur doit être conservée.
   (a) Pour chaque état m (de la colonne $i-1$) tel que $1 \in m[H \cup H']$, créer les transitions $m \to_0 m[H =?]0$ et $m \to_1 R$
   (b) Pour chaque état m (de la colonne $i-1$) tel que $1 \notin m[H \cup H']$, créer les transitions $m \to_0 m[H =?]0$ et $m \to_1 m[H =?]1$

Créer la transition $R \to_{0,1} R$.

Quand tous les sommets d'un voisin sont traités, il sera noté ? dans les étiquettes suivantes. Ainsi, à la colonne n, il existera un unique état étiqueté ?.....? (en plus de l'état poubelle R) qui sera l'état final accepteur de M_C,

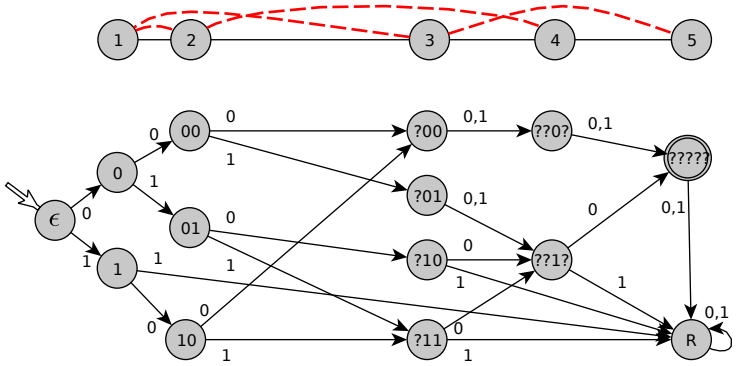


FIGURE 1.13 – Chemin avec conflits d'épaisseur 2 at DFA reconnaissant les mots sans conflit.

noté A. La figure 1.13 montre un exemple de graphe des conflits sur un chemin et de l'automate construit par l'algorithme. La table 1.2 montre comment les états du DFA sont construits séquentiellement colonne par colonne, en partant de l'état ϵ et en suivant l'ordre du chemin sur les sommets.

Nous prouvons maintenant que M_C accepte tout mot sans conflit de longueur n mais n'accepte aucun autre mot.

Notons que chaque état possède deux transitions, sur les symboles 0 et 1, et que M_C n'a pas de circuit, à l'exception de la boucle sur l'état R. Toutes les transitions se font de gauche à droite, d'une colonne i vers une colonne $i+1$, à l'exception des transitions vers R. Les seuls sommets à distance n de l'état initial ϵ sont les états A et R. Ainsi, quand un mot est traité, M_C sera exactement une fois dans un état de chaque colonne, sauf s'il effectue une transition vers R.

Soit S un ensemble de taille n sans conflit représenté par un mot I. Un mot de longueur n est accepté si et seulement si il ne suit jamais une transition $m \to R$. Supposons que durant le traitement du mot I, M_C soit dans l'état m de la colonne i. Si le $i$ème symbole de I est 0, alors il existe une transition de l'état m à un état de la colonne $i+1$ (les transitions en 0 ne sont jamais vers R). Si le $i$ème symbole est 1, comme S est sans conflit, les symboles de $m[H \cup H']$ sont tous 0, comme aucun voisin de V_i ne peut appartenir à S. Ainsi, une transition $m \to_1 m[H =?]1$ ou $m \to_1 m[H =?]?$ existe.

Comme ceci est vrai pour toute colonne, M_C n'effectuera jamais une transition $m \to R$ et s'arrêtera donc dans l'état A. Ainsi, tout mot de longueur n sans conflit est accepté.

Soit maintenant S un ensemble avec un conflit dans C représenté par un

C	S	J	H	H'	état	$m[H]$	$m[H']$	Cas	T0	T1
0	1	2, 3	$\emptyset$	$\emptyset$	ϵ	$\emptyset$	$\emptyset$	2b	0	1
1	2	4	$\emptyset$	1	0	$\emptyset$	0	2b	00	01
					1	$\emptyset$	1	2a	10	R
2	3	5	1	$\emptyset$	00	0	$\emptyset$	2b	?00	?01
					01	0	$\emptyset$	2b	?10	?11
					10	1	$\emptyset$	2a	?00	R
3	4	$\emptyset$	2	$\emptyset$	?00	0	$\emptyset$	1b	??0?	??0?
					?01	0	$\emptyset$	1b	??1?	??1?
					?10	1	$\emptyset$	1a	??0?	R
					?11	1	$\emptyset$	1a	??1?	R
4	5	$\emptyset$	3	$\emptyset$	??0?	0	$\emptyset$	1b	?????	?????
					??1?	1	$\emptyset$	1a	?????	R
5	-	-	-	-	?????	-	-	-	R	R
					R	-	-	-	R	R

TABLE 1.2 – Construction de l'automate montré figure 1.13
C : Colonne S : Sommet T0 : Transition en 0 T1 : Transition en 1

mot X. Il doit exister i et j deux indices tels que $v_i v_j$ est une arête de C et $X[i] = X[j] = 1$. Supposons sans perte de généralité que $i < j$. Supposons que M_C est dans l'état m de la $j$ème colonne (si ceci n'arrive pas, cela veut dire qu'une transition vers R a déjà été faite et que X a été rejeté). Ainsi, comme $X[i] = 1$, $m[i] = 1$. Or, $i \in H \cup H'$, donc $1 \in m[H \cup H']$ et donc la transition est $m \to_1 R$ et X est rejeté.

M_C accepte tous les mots sans conflit, et aucun mot avec conflits, il reconnaît donc le langage L_C.

Lemme 14. *Si (P, C) est d'épaisseur k, alors M_C a $O(n \cdot 2^k)$ états et est construit en temps polynomial par rapport à sa taille.*

Démonstration. À chaque étape i de la construction des colonnes, si un sommet v_h n'a plus de voisin de droite, alors les étiquettes seront ? à l'indice h pour chaque colonne $j \geq i$.

Ainsi, dans la colonne i, seuls les indices $j < i$ dont les sommets v_j ont un voisin v_l avec $l > i$ peuvent avoir un symbole 0 ou 1 dans l'étiquette. Or, l' épaisseur est au plus k, et donc au plus k tels indices existent. Ceci implique qu'il y a au plus 2^k états distincts dans chaque colonne, on obtient donc un nombre total d'états d'au plus $n \cdot 2^k$. La création de chaque nouvel état demande uniquement une recherche parmi les sommets existants, ce qui peut être fait en temps polynomial. □

Théorème 12. *Étant donné (P, C) un chemin avec conflits d'épaisseur*

bornée par une constante k, il est possible de déterminer en temps polynomial s'il existe un IDSwnC (resp. DSwnC) de (P, C).

Démonstration. Soit (P, C) un chemin avec conflits d' épaisseur k.

D'après le lemme 14, construire le DFA M_C de taille $O(n \cdot 2^k)$ en temps polynomial, pour reconnaître le langage L_C.

Construire en temps polynomial $M_{IDSwnC(P,C)}$ (resp. $M_{DSwnC(P,C)}$) le produit de M_{IDS} (resp. M_{DOM}) et M_C, reconnaissant $L_{IDSwnC(P,C)}$ (resp. $L_{DSwnC(P,C)}$). Ce DFA a $O(n \cdot 2^k)$ états.

Déterminer en temps polynomial si $L_{IDSwnC(P,C)}$ (resp. $L_{DSwnC(P,C)}$) est vide en trouvant un plus court chemin depuis ϵ vers A dans $M_{IDSwnC(P,C)}$ (resp. $M_{DSwnC(P,C)}$). □

1.4 Conclusion du chapitre

Dans ce chapitre, nous avons étudié les problèmes du dominant et du dominant indépendant sans conflit. Ces problèmes se sont révélés NP-complets même dans des classes de graphe et de graphe des conflits très peu denses, en particulier le problème du dominant indépendant sans conflit, NP-complet même quand le graphe support est un chemin et que le graphe des conflits est de degré maximum 1. Dans des classes de graphes denses, ce problème reste également NP-complet. Pour le problème du dominant sans conflit, la plupart des résultats sont également des théorèmes de NP-complétude, à l'exception de quelques cas particuliers pour lesquels il existe toujours une solution. Nous n'avons pas identifié de cas où il existe parfois des solutions et où le déterminer est polynomial.

Ces résultats nous ont motivés à chercher un algorithme paramétrique pour le dominant indépendant sans conflit quand le graphe support est un chemin, une classe très simple où le problème reste néanmoins NP-complet. Nous avons pu obtenir un algorithme paramétrique pour un nouveau paramètre que nous avons introduit : l'épaisseur du graphe des conflits par rapport à l'ordre du chemin. La formulation de cet algorithme à travers un produit d'automate semble peu exploitable pour une implémentation concrète. Il serait intéressant de chercher d'une part une formulation alternative de cet algorithme, et d'autre part une généralisation, en étendant ce paramètre d'épaisseur à l'union des deux graphes, pour relâcher la contrainte "chemin" sur le graphe support qui est très restrictive.

Chapitre 2

Problèmes de domination sans conflit : étude de la complexité sur l'union des deux graphes

Dans le chapitre précédent, nous avons étudié la complexité des problèmes IDSwnC et DSwnC dans différentes classes de graphe support et de graphe des conflits. Pour autant, même pour des classes de graphe support et de graphe des conflits restreintes, l'union des deux graphes peut être complexe : nous nous intéressons maintenant à la complexité des problèmes DSwnC, IDSwnC et TDSwnC par rapport à la classe de l'union du graphe support et du graphe des conflits. Nous introduisons également un nouveau paramètre : l'*étirement des conflits*, qui mesure la "localité" des conflits dans le graphe support. Nous souhaitons savoir si les problèmes deviennent plus simples lorsque les conflits sont entre des sommets proches.

Définition 9 (Étirement)**.** Étant donné (G, C) un graphe avec conflits, pour tout ab une arête de conflit, l'étirement de ab est la distance entre a et b dans G. L'étirement de C est le maximum de l'étirement de tous les conflits.

Les preuves qui vont suivre se basent sur une restriction du problème 3-SAT définie dans [TM16] et présentée ci-dessous.

Définition 10 (Planar 3-bounded 3-SAT)**.**

— Instance : Un ensemble X de variables et un ensemble Cl de clauses sur X, où chaque clause contient au plus trois littéraux, chaque variable apparaît dans au plus trois clauses, et où le graphe associé $G = (V, E)$ est biparti et planaire avec $V = X \cup Cl$ et où E contient exactement les arêtes entre x et cl si soit x soit $\bar{x}$ appartient à la

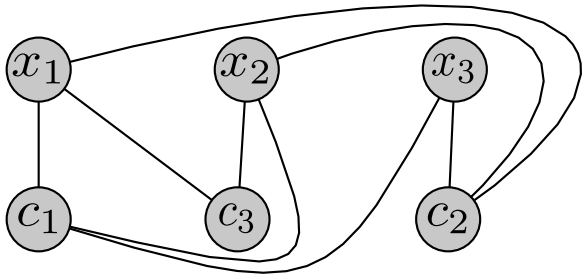


FIGURE 2.1 – Graphe associé à la formule $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2})$

G		C		$G \cup C$	Existence	Ref
Classe	D	D	Étirement	Classe		
DOMINANT SANS CONFLIT (DSwnC)						
Biparti Planaire	4	1	$= 2$	Planaire	NPC	Thm15
Quelconque	-	-	$= 1$	Quelconque	Toujours	Thm18
DOMINANT INDÉPENDANT SANS CONFLIT (IDSwnC)						
Biparti Planaire	4	1	$= 2$	Planaire	NPC	Thm16
Quelconque	-	-	$= 1$	Quelconque	Toujours	Thm18
DOMINANT TOTAL SANS CONFLIT (TDSwnC)						
Biparti Planaire	3	1	$= 2$	Planaire	NPC	Thm17
Planaire	4	1	$= 1$	Planaire	NPC	Thm19

D : Degré maximal

TABLE 2.1 – Résumé des résultats de complexité obtenus dans ce chapitre

clause *cl*.

— Question : Existe-t-il une affectation des variables de X vérifiant Cl ?

La figure 2.1 montre un graphe associé à une instance de Planar 3-bounded 3-SAT. Notons que plusieurs instances peuvent avoir le même graphe associé.

Théorème 13. *Planar 3-bounded 3-SAT est un problème NP-complet. [TM16]*

Les résultats de complexité obtenus pour les trois problèmes de domination sont très proches, leurs preuves sont également similaires. Aussi, la section 2.1 présentera la structure des preuves au travers d'un théorème générique, et la section 2.2 présentera les résultats obtenus et les spécificités des preuves pour les problèmes étudiés. La table 2.1 résume les différents résultats obtenus dans ce chapitre.

2.1 Structure des preuves

Les preuves de NP-complétude pour les trois problèmes considérés sont très similaires, aussi nous souhaitons dégager une structure commune aux trois preuves. Dans cette section, nous allons réduire une instance de Planar 3-Bounded 3-SAT au problème $\mathcal{P}$, le problème générique de décider de l'existence d'une structure sans conflit, une structure désignant ici un DSwnC, un IDSwnC où un TDSwnC. Pour cela, nous partirons du graphe (planaire) associé à une formule, et nous remplacerons ses sommets par des gadgets, jusqu'à obtenir une instance de $\mathcal{P}$ équivalente à la formule. Nous ne donnerons pas explicitement les gadgets dans cette section car ils sont différents pour chaque problème et seront présentés en section 2.2. Nous utiliserons au contraire une caractérisation des gadgets suffisante pour la structure de la preuve et commune aux trois problèmes.

Définition 11 (Gadget de variable)**.** Un gadget pour la variable x_i pour le problème $\mathcal{P}$ est un graphe avec conflits (G_{x_i}, C_{x_i}) où :

- $G_{x_i} \cup C_{x_i}$ est planaire.
- C_{x_i} est de degré maximal 1.
- $G_{x_i} \cup C_{x_i}$ contient deux sommets x_i et $\bar{x_i}$ sur sa face externe.
- x_i et $\bar{x_i}$ ne peuvent être simultanément dans une solution de $\mathcal{P}$.
- G_{x_i} est biparti et x_i et $\bar{x_i}$ sont dans la même partition.
- Il existe une solution S pour le problème $\mathcal{P}$ avec $x_i \in S$ et une solution S' avec $\bar{x_i} \in S'$.

Définition 12 (Gadget de clause)**.** Un gadget pour la clause c_i pour le problème $\mathcal{P}$ est un graphe avec conflits (G_{c_i}, C_{c_i}) où :

- $G_{c_i} \cup C_{c_i}$ est planaire.
- C_{c_i} est de degré maximal 1.
- $G_{c_i} \cup C_{c_i}$ contient un sommet c_i sur sa face externe.
- G_{c_i} est biparti.
- Il n'existe pas de solution de $\mathcal{P}$ pour (G_{c_i}, C_{c_i}).
- Si l'on ajoute un voisin x_j à c_i, alors il existe une solution de $\mathcal{P}$(contenant x_j).

Théorème 14. *S'il existe un gadget de clause et un gadget de variable pour le problème $\mathcal{P}$, alors $\mathcal{P}$ est NP-complet même si le graphe support est biparti, que le graphe des conflits est de degré maximal 1, et que l'union des deux graphes est planaire.*

Démonstration. Soit (X, Cl) une instance de Planar 3-Bounded 3-SAT. Soit $G = (V, E)$ où $V = X \cup Cl$ et où E contient exactement les arêtes entre x et cl où soit x soit $\bar{x}$ appartient à la clause cl et soit $C = (V, \emptyset)$. D'après la définition 10, G est planaire et donc $G \cup C$ est planaire. Nous allons effectuer des transformations successives sur ce graphe jusqu'à obtenir une instance de $\mathcal{P}$ telle que définie dans le théorème 14. Nous veillerons à ce que chaque

transformation conserve la planarité et la bipartition.

Chaque sommet c_i de G représentant une clause est remplacé par un gadget de clause (G_{c_i}, C_{c_i}). Les arêtes incidentes à c_i sont maintenant adjacentes au nouveau sommet c_i du gadget de clause. Comme celui-ci est sur la face externe du gadget de clause, il est facile de voir que l'union des graphes reste planaire. Comme G_{c_i} est biparti, le nouveau graphe support reste biparti. Chaque sommet x_i représentant une variable est remplacé par un gadget de variable (G_{x_i}, C_{x_i}). Les arêtes incidentes à l'ancien sommet x_i et à des sommets de clause c_j où x_i apparait de manière positive sont maintenant reliées au nouveau sommet x_i de G_{x_i}. Les arêtes incidentes à l'ancien sommet x_i et à des sommets de clause c_j où x_i apparait de manière négative sont maintenant reliées au nouveau sommet $\bar{x_i}$ de G_{x_i}. Comme x_i et $\bar{x_i}$ sont dans la même partition, le graphe support reste biparti. Il reste à montrer que l'union des graphes est planaire. Notons que chaque gadget de variable doit se connecter à au plus trois sommets de clauses, et que chacun des sommets de clauses doit être connecté à x_i où $\bar{x_i}$. Dans la figure 2.2, nous montrons tous les cas possibles à une symétrie près. Nous avons donc obtenu une instance (G, C) de $\mathcal{P}$ où G est biparti, C est de degré maximal 1, et $G \cup C$ est planaire. Nous devons maintenant prouver qu'il existe une solution de $\mathcal{P}$ dans l'instance (G, C) si et seulement si (X, Cl) est vérifiable.

Supposons qu'il existe A une affectation des variables de X vérifiant Cl. Soit A_+ l'ensemble des variables assignées à $vrai$. Construisons S une solution de $\mathcal{P}$ comme suit : pour chaque $x_i \in A_+$, on pose $x_i \in S$, et pour chaque $x_i \in A - A_+$, on pose $\bar{x_i} \in S$. Comme A est une affectation, pour chaque gadget de variable (G_{x_i}, C_{x_i}) il existe exactement un sommet dans S parmi x_i et $\bar{x_i}$. D'après les propriétés des gadgets de variable, il existe une solution pour $\mathcal{P}$ dans chacun des gadgets. De plus, comme chaque clause est vraie, chaque sommet de clause est voisin d'un sommet (x_i ou $\bar{x_i}$) appartenant à S. Ainsi, il est possible de construire une solution de $\mathcal{P}$ pour (G, C).

Supposons maintenant qu'il existe une solution de $\mathcal{P}$ dans (G, C) et construisons une affectation des variables de X vérifiant Cl. Pour chaque gadget de variable (G_{x_i}, C_{x_i}), si $x_i \in S$, alors la variable x_i est assignée à $vrai$. Les autres variables sont assignées à $faux$. Montrons maintenant que cette affectation vérifie Cl. Soit c_i une clause quelconque. Considérons le gadget de clause correspondant. Comme S est une solution, d'après les propriétés du gadget de clause, c_i doit avoir au moins un voisin appartenant à S. Si ce voisin est un sommet x_j, alors la variable x_j est $vrai$, et la clause c_i est vérifiée. Si ce voisin est un sommet $\bar{x_j}$, alors la variable x_j est $faux$, $\bar{x_j}$ est donc $vrai$, et la clause c_i est vérifiée. Ainsi, chaque clause est vérifiée et l'affectation vérifie Cl. □

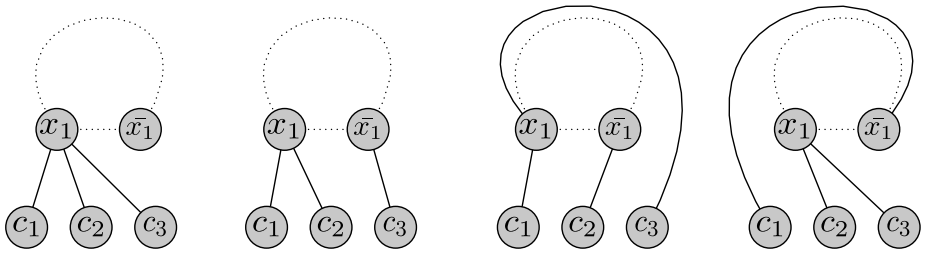


FIGURE 2.2 – Relier les sommets de clauses aux gadgets de variables de manière planaire. Les pointillés représentent la limite de la face externe du gadget de variable.

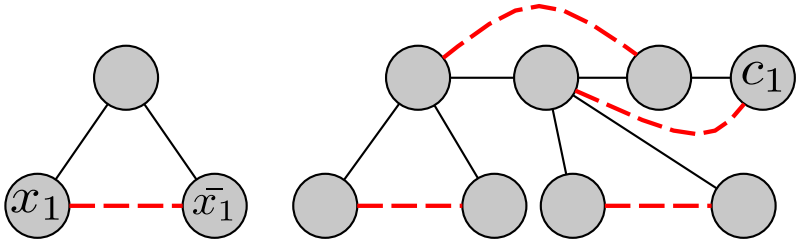


FIGURE 2.3 – Gadget de variable (à gauche) et gadget de clause (à droite) pour DSwnC.

2.2 Application aux problèmes DSwnC, IDSwnC et TDSwnC

Nous allons maintenant utiliser le théorème 14 pour prouver la NP-complétude de DSwnC, IDSwnC et TDSwnC pour des catégories d'instances spécifiques. Dans les figures de cette section, les arêtes du graphe support seront en trait plein, et les arêtes du graphe des conflits seront en tirets.

2.2.1 Dominant sans conflit

Propriété 1. *Il existe un gadget de clause et un gadget de variable pour DSwnC.*

Un gadget de variable et un gadget de clause pour DSwnC sont représentés figure 2.3. Intuitivement, le but du gadget de clause est d'empêcher que le sommet c_i appartienne à une solution et d'empêcher qu'il soit dominé par le gadget de clause. Il est facile de vérifier que ces gadgets satisfont les définitions 11 et 12.

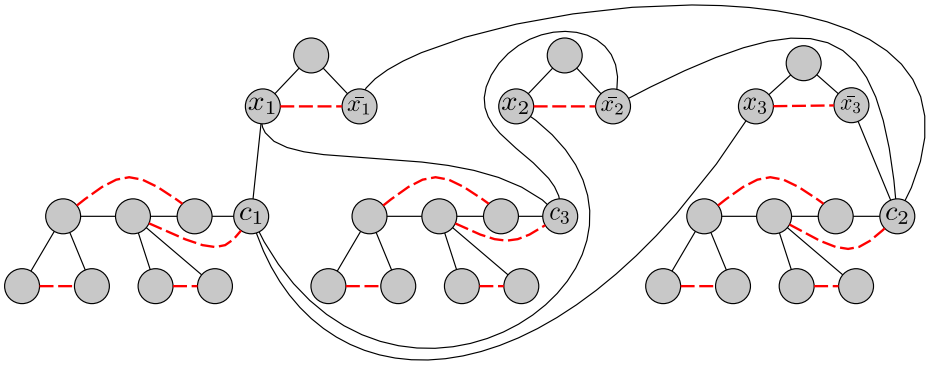


FIGURE 2.4 – Instance de DSwnC équivalent à la formule $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2})$.

Théorème 15. *Déterminer l'existence d'un DSwnC dans (G, C) est NP-complet même si G est biparti de degré maximal 4, C est de degré maximal 1, $G \cup C$ est planaire et l'étirement des conflits est exactement égal à 2.*

Démonstration. Le théorème 15 découle du théorème 14 et de la propriété 1. Nous devons simplement vérifier que la construction de l'instance de DSwnC dans la preuve du théorème 14 en utilisant les gadgets de la propriété 1 donne bien un graphe support de degré maximal 4 et des conflits d'étirement égal à 2. Les conflits sont présents uniquement dans les gadgets : on peut voir figure 2.3 que leur étirement est exactement égal à 2. Les sommets de clause sont de degré $1+3=4$ au maximum. Les sommets de variable sont de degré maximal $3+1=4$. Le graphe G est donc de degré au plus 4. □

La figure 2.4 présente un exemple d'instance de DSwnC équivalent à une formule de Planar 3-Bounded 3-SAT. Ce graphe a été obtenu en remplaçant les sommets de clause et de variables du graphe associé à la formule par les gadgets présentés propriété 1 comme expliqué dans le théorème 14. La construction étant similaire pour les autres problèmes, les instances de IDSwnC et TDSwnC ne seront pas illustrées.

2.2.2 Dominant Indépendant sans conflit

Propriété 2. *Il existe un gadget de clause et un gadget de variable pour IDSwnC.*

Un gadget de variable et un gadget de clause pour IDSwnC sont représentés figure 2.5. Intuitivement, le but du gadget de clause est d'empêcher que le sommet c_i appartienne à une solution et d'empêcher qu'il soit dominé

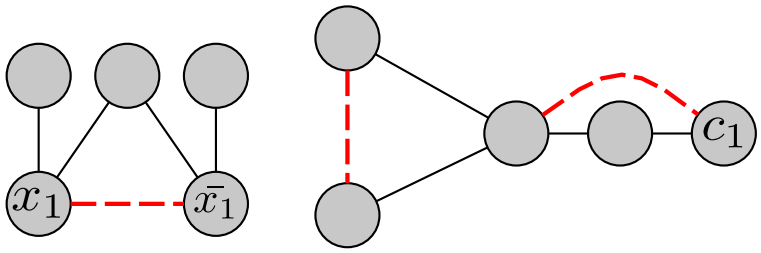


FIGURE 2.5 – Gadget de variable (à gauche) et gadget de clause (à droite) pour IDSwnC.

par le gadget de clause. Il est facile de vérifier que ces gadgets satisfont les définitions 11 et 12.

Théorème 16. *Déterminer l'existence d'un IDSwnC dans (G, C) est NP-complet même si G est biparti de degré maximal 4, C est de degré maximal 1, $G \cup C$ est planaire et l'étirement des conflits est exactement égal à 2.*

Démonstration. Le théorème 16 découle du théorème 14 et de la propriété 2. Nous devons simplement vérifier que la construction de l'instance de IDSwnC dans la preuve du théorème 14 en utilisant les gadgets de la propriété 2 donne bien un graphe support de degré maximal 4 et des conflits d'étirement égal à 2. Les conflits sont présents uniquement dans les gadgets : on peut voir que leur étirement est exactement égal à 2. Les sommets de clause sont de degré $1+3 = 4$ au maximum. Les sommets de variables (ou leur négation) sont reliés à deux sommets du gadget de variable, et aux clauses dans lesquelles elles apparaissent. Nous savons que chaque variable apparaît au plus dans trois clauses : on peut supposer sans perte de généralité qu'elle apparaît au moins une fois de manière positive et une fois de manière négative (sinon il est possible de fixer directement la valeur de cette variable et de la retirer de l'instance). Ainsi, chaque sommet x_i ou $\bar{x}_i$ est voisin d'au plus deux sommets de clause, ce qui lui donne un degré maximal de $2 + 2 = 4$. Le graphe G est donc de degré au plus 4. □

2.2.3 Dominant Total sans conflit

Propriété 3. *Il existe un gadget de clause et un gadget de variable pour TDSwnC.*

Un gadget de variable et un gadget de clause pour TDSwnC sont représentés figure 2.6. Il est facile de vérifier que ces gadgets satisfont les définitions 11 et 12.

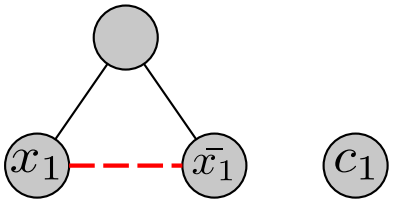


FIGURE 2.6 – Gadget de variable (à gauche) et gadget de clause (à droite) pour TDSwnC.

Théorème 17. *Déterminer l'existence d'un TDSwnC dans (G, C) est NP-complet même si G est biparti de degré maximal* 3*, C est de degré maximal* 1*, $G \cup C$ est planaire et l'étirement des conflits est exactement égal à* 2*.*

Démonstration. Le théorème 17 découle du théorème 14 et de la propriété 3. Nous devons simplement vérifier que la construction de l'instance de TDSwnC dans la preuve du théorème 14 en utilisant les gadgets de la propriété 3 donne bien un graphe support de degré maximal 3 et des conflits d'étirement égal à 2. Les conflits sont présents uniquement dans les gadgets de variable : on peut voir que leur étirement est exactement égal à 2. Les sommets de clause sont de degré 3 au maximum. Les sommets de variables (ou leur négation) sont reliés à un sommet du gadget de variable, et aux clauses dans lesquelles elles apparaissent. Par le même argument que dans la preuve du théorème 16, on peut dire que chaque sommet x_i ou $\bar{x_i}$ est voisin d'au plus 2 sommets de clause, ce qui lui donne un degré maximal de $1 + 2 = 3$. Le graphe G est donc de degré au plus 3. □

2.3 Étirement égal à 1

Les réductions précédentes utilisent des instances où l'étirement des conflits est égal à 2, mais qu'advient-il si l'étirement des conflits est de 1 ? Cela signifie que pour toute arête ab du graphe des conflits, il existe une arête ab du graphe support.

Théorème 18. *Si (G, C) est un graphe avec conflits d'étirement égal à* 1*, alors il existe toujours un IDSwnC et un DSwnC, et il peut être construit en temps polynomial.*

Démonstration. Il est possible de construire en temps polynomial S un dominant indépendant de $G = (V, E)$. Pour cela, on choisit arbitrairement un sommet de G que l'on place dans S. On retire ensuite ses voisins de V. L'opération est répétée tant que V n'est pas vide. Quand V est vide, S

est un dominant indépendant de G. Comme $C \subseteq G$, S est également un indépendant de C : S est sans conflit. S est donc un IDSwnC (et donc un DSwnC) de (G, C). □

S'il existe toujours une solution pour IDSwnC et DSwnC quand l'étirement des conflits est égal à 1, ceci permet de répondre en temps constant "existe-il un IDSwnC (resp. DSwnC)" dans ce type d'instance. Ce n'est pas le cas pour TDSwnC, où le problème de déterminer l'existence d'une solution reste NP-complet.

Théorème 19. *Déterminer l'existence d'un TDSwnC dans* (G, C) *est NP-complet même si* G *est de degré maximal* 4*,* C *est de degré maximal* 1*,* $G \cup C$ *est planaire et l'étirement des conflits est exactement égal à* 1*.*

Démonstration. Considérons le graphe construit dans la démonstration du théorème 17. Nous pouvons observer que l'ajout d'une arête de G entre x_i et $\bar{x_i}$ ne change pas l'existence d'une solution. Le graphe support n'est plus biparti, son degré maximal augmente d'un, mais les autres propriétés de l'instance sont inchangées. □

2.4 Conclusion du chapitre

Dans ce chapitre, nous avons étudié des paramètres liant les deux graphes entre eux : quand l'union des deux graphes est planaire, et quand l'étirement des conflits est faible. Pour les problèmes considérés (Dominant, Dominant Indépendant et Dominant Total sans Conflit), il est NP-complet de déterminer l'existence d'une solution, même quand l'union des deux graphes est planaire, avec un étirement faible des conflits (égal à 2). Ces paramètres ne permettent donc pas d'isoler des cas plus simples (polynomiaux). Quand l'étirement des conflits est égal à 1, le problème du Dominant Total sans Conflit reste NP-complet. Pour les problèmes du Dominant sans Conflit et du Dominant Total sans conflit, il existe toujours une solution : en effet, le problème est réduit à une relaxation du problème du Dominant Indépendant classique, mais dès que l'étirement dépasse ce seuil, le problème redevient NP-complet. Il serait intéressant pour des travaux futurs d'étudier des paramètres de graphes classiques sur l'union des deux graphes.

Chapitre 3

Résultats d'approximation sur les problèmes de domination sans conflit

Nous avons vu dans les chapitres précédents que les problèmes IDSwnC et DSwnC étaient NP-complets, et ce pour un grand nombre de classes de graphes. Déterminer l'existence d'une solution en temps polynomial étant compromis, nous nous penchons à présent sur l'étude de solutions partielles pour les problèmes de domination. Nous souhaitons déterminer quelle proportion d'un graphe il est possible de dominer sans conflit. Les définitions 13 et 14 introduisent formellement les notions de domination partielle.

Définition 13. Étant donné (G, C) un graphe avec conflits, un *Dominant Partiel sans Conflit* (PDSwnC) de (G, C) est un sous-ensemble de sommets de G sans conflit dans C. Un PDSwnC optimal est un PDSwnC qui maximise le nombre de sommets de G dominés.

Définition 14. Étant donné (G, C) un graphe avec conflits, un *Dominant Indépendant Partiel sans Conflit* (PIDSwnC) de (G, C) est un sous-ensemble indépendant de sommets de G sans conflit dans C. Un PIDSwnC optimal est un PIDSwnC qui maximise le nombre de sommets de G dominés.

Les problèmes de décision associés appartiennent trivialement à NP. Les preuves de NP-complétude de cette section se concentreront donc uniquement sur l'aspect NP-difficile des problèmes.

La table 3.1 résume les différents résultats obtenus dans ce chapitre.

Problème	G	C	Proportion	Existence	Thm.
PIDSwnC	-	degré max k	$n/(k+1)$	toujours	20
PDSwnC	-	couplage	$(3/4)n$	toujours	21
PIDSwnC	chemin	couplage	$(3/4)n-1$	toujours	23
PDSwnC	$\bigcup$ chemins	couplage	$(1/2+\epsilon)n$	NPC	24
PIDSwnC	$\bigcup$ chemins	couplage	$(1/2+\epsilon)n$	NPC	25
PDSwnC	arbre	couplage	$(3/4+\epsilon)n$	NPC	26

TABLE 3.1 – Résumé des résultats de complexité obtenus dans ce chapitre

3.1 Algorithmes polynomiaux

Nous montrons tout d'abord qu'il est toujours possible de dominer une grande partie des sommets lorsque le graphe des conflits est de faible degré. Nous présentons ensuite un algorithme spécifique pour le PIDSwnC quand le graphe support est un chemin (il est dans ce cas NP-complet de déterminer l'existence d'un IDSwnC).

Théorème 20. *Étant donné (G, C) un graphe avec conflits, si C est un graphe de degré maximum k, il est toujours possible de trouver un PIDSwnC dominant au moins $n/(k+1)$ sommets, et cet ensemble peut être construit en temps polynomial. En particulier, si C est un couplage parfait, il est possible de trouver un PIDSwnC dominant au moins $n/2$ sommets.*

Algorithm 3: $1/k$ Approximation PIDSwnC

Data: $(G = (V, E_1), C = (V, E_2))$
Result: Un PIDSwnC dominant au moins $n/(k+1)$ sommets
Construire en temps polynomial un dominant indépendant S de $G \cup C$
return S

Démonstration. Étant donné $(G = (V, E_1), C = (V, E_2))$ une instance de PIDSwnC, notons S le résultat de l'algorithme 3, S_1 le voisinage de S dans G et $S_2 = V - (S \cup S_1)$. Notons que S est un indépendant sans conflit dominant $S \cup S_1$. Soit x un sommet de S_2. Alors par définition, $N_G(x) \cap S = \emptyset$. S étant un dominant de $G \cup C$, x doit être en conflit avec au moins un sommet de S or C est de degré maximum k. On en déduit que $|S_2| \leq k|S|$ d'où $|S| + |S_1| \geq n/(k+1)$. □

Corollaire 1. *Étant donné (G, C) un graphe avec conflits, si C est un graphe de degré maximum k, il est toujours possible de trouver un PDSwnC dominant au moins $n/(k+1)$ sommets, et cet ensemble peut être construit en temps polynomial.*

Démonstration. Tout PIDSwnC étant un PDSwnC, l'algorithme 3 construit l'ensemble voulu. □

L'algorithme présenté ne fait aucune hypothèse quant au graphe support. Nous montrons maintenant que si le graphe support n'a pas de sommet isolé, il est possible de construire un PDSwnC dominant au moins trois quarts des sommets.

Lemme 15. *Si (G, C) est un graphe avec conflits à $n = 2p$ sommets avec C un couplage parfait, alors il existe toujours une solution optimale du PDSwnC comportant p sommets.*

Démonstration. Comme C est un couplage parfait, il n'existe pas d'ensemble sans conflit de plus de la moitié des sommets. Ainsi, une solution est composée d'au plus p sommets.

Supposons qu'il existe S une solution optimale comportant strictement moins de p sommets. Alors, comme C est un couplage parfait, pour chaque arête de C dont aucune extrémité ne se trouve dans S, on ajoute une extrémité arbitraire à la solution. La solution S' construite comportera bien p sommets et dominera autant de sommets que S. □

Théorème 21. *Si (G, C) est un graphe avec conflits à n sommets où G n'a pas de sommet isolé et C est un couplage parfait, alors il existe un PDSwnC dominant au moins $(3/4)n$ sommets.*

Démonstration. Soit (G, C) une instance du PDSwnC à $n = 2p$ sommets où G est un graphe sans sommet isolé et où C est un couplage parfait. Soit S^* une solution optimale.

Nous prouvons le théorème 21 par l'absurde. Supposons que S^* couvre strictement moins de $(3/4)n = p + p/2$ sommets.

D'après le lemme 15, nous pouvons supposer sans perte de généralité que $|S^*| = p$.

Posons $V_1 = S^*$ et $V_2 = V - S^*$. Soit X l'ensemble des sommets de V_2 dominés par S^*, et X' les sommets en conflit avec X (notons que $X' \subseteq V_1$). Notons Y les voisins de X dans V_1 et $Z = V_2 - X$. Soient x, y, z les tailles respectives de ces ensembles.

S^* domine exactement $|S^*| + |X| = p + x$ sommets. Par hypothèse, on a donc $x < p/2$. Comme $z + x = p$ et $x < p/2$, on obtient $z > p/2$.

Supposons sans perte de généralité que $y \leq x$. Dans le cas contraire $y > x$ et V_2 domine au moins $p + y > p + x = |S^*|$ sommets, une contradiction.

V_1 contient :
- x sommets de X',

— y sommets de Y ,
— α sommets n'appartenant ni à X' ni à Y.

On en déduit que $p \leq x + y + \alpha \leq 2x + \alpha < p + \alpha$ et donc $\alpha > 0$. Soit r' un sommet quelconque parmi ces α sommets. Comme G n'a pas de sommet isolé, r' a au moins un voisin u. Or $u \notin Z$, car les sommets de Z ne sont pas dominés, et $u \notin X$ car $r' \notin Y$, ainsi, $u \in V_1$. Soit $r \in V_2$ le sommet en conflit avec r'. On sait que $r \in Z$ car $r' \notin X'$. En échangeant r et r', on obtient $S' = S^* - r' \cup r$ une solution qui domine les sommets de $S' \cup X \cup r'$. S' domine donc $p + x + 1$ sommets, soit plus que S^*, une contradiction. □

Il est possible de dériver un algorithme constructif de la preuve du théorème 21.

Algorithm 4: PDSwnC 3/4

Data: $(G = (V, E), C)$ un graphe avec conflit à $n = 2p$ sommets où G n'a pas de sommet isolé et C est un couplage parfait
Result: Un PDSwnC dominant au moins $(3/4)n$ sommets
$S \leftarrow$ un ensemble sans conflit à p sommets
continuer $\leftarrow$ **vrai**
while *continuer* **do**
  if *$V - S$ domine plus de sommets que S* **then**
    $S \leftarrow V - S$
  else
    Choisir $r' \in V$ et r en conflit avec r' tel que $S - r' \cup r$ domine strictement plus de sommets que S.
    if *un sommet r' à été trouvé* **then**
      $S \leftarrow S - r' \cup r$
    else
      continuer $\leftarrow$ **faux**
return S

Théorème 22. *L'algorithme 4 retourne un PDSwnC dominant au moins* 3/4 *des sommets en temps polynomial : cet algorithme est* 3/4 *approché.*

Démonstration. Chaque opération de la boucle principale de l'algorithme est exécutée en temps polynomial. De plus, à chaque tour de boucle, le nombre de sommets dominés par la solution courante augmente, la boucle est donc exécutée au plus n fois. Ainsi, l'algorithme 4 se termine en temps polynomial.

Soit S la solution retournée par l'algorithme. L'algorithme est sorti de la boucle principale, ce qui signifie qu'il n'existe plus de sommet pouvant être permuté de manière à augmenter le nombre de sommets dominés. Or, nous savons d'après la preuve du théorème 21 que si une solution domine moins

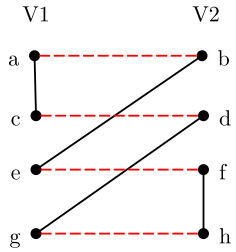


FIGURE 3.1 – Ratio de 3/4 atteint par l'algorithme 4.

de 3/4 des sommets, alors il existe au moins un sommet r' de S pouvant être permuté avec son sommet en conflit r de manière à améliorer la solution. Ainsi, la solution S retournée pour l'algorithme domine au moins $(3/4)n$ sommets et l'algorithme 4 est bien 3/4 approché. □

Remarque 2. Il existe une infinité d'instances pour lesquelles le rapport 3/4 est atteint. Un exemple est montré à la figure 3.1. Si l'algorithme choisit comme solution initiale $S = V_1$, aucun sommet ne peut être permuté, et S domine 6 des 8 sommets. L'ensemble $\{a, d, e, h\}$ est un PDSwnC dominant tous les sommets, le ratio de 3/4 est donc atteint. De plus, des copies disjointes de ce graphe peuvent obtenir le même ratio pour une famille infinie de graphes.

Nous présentons maintenant un algorithme pour le PIDSwnC dans les chemins quand le graphe des conflits est un couplage parfait. L'algorithme parcourt le chemin dans l'ordre naturel, et tente de construire un IDSwnC en sélectionnant un sommet sur trois. Si un sommet ne peut pas être sélectionné à cause d'un conflit, l'algorithme tentera de choisir le sommet suivant.

Théorème 23. *L'algorithme 5 appliqué à (G, C) une instance de taille n où G est un chemin et C un couplage parfait retourne en temps polynomial un PIDSwnC dominant au moins $(3/4)n - 1$ sommets.*

Démonstration. Soit S l'ensemble retourné par l'algorithme. S contient les sommets marqués par 1. Or, quand un sommet x est marqué par 1, le sommet en conflit est soit marqué par 0, s'il fait partie des deux sommets suivant x, soit par X dans le cas contraire. Dans tous les cas, le sommet en conflit sera marqué par autre chose que 1 et ne pourra donc pas appartenir à S. De plus, quand un sommet est marqué par 1, son successeur est marqué par 0, ainsi, deux sommets voisins ne peuvent appartenir à S. Ainsi, S est bien un PIDSwnC de (G, C).

Étudions maintenant le nombre de sommets dominés. On remarque que le marquage d'un sommet par X ne peut se produire que dans une seule

Algorithm 5: $01X$

Data: $(G = (V, E_1), C = (V, E_2))$ avec G un chemin et C un couplage parfait.

Result: Un PIDSwnC dominant au moins $(3/4)n$ sommets

Soient $x_1, ..., x_n$ les sommets du chemin dans l'ordre naturel.

Tous les sommets sont non marqués.

$i \leftarrow 0$

while $i \leq n - 2$ **do**

  if *x_i est non marqué* **then**

    Marquer x_i par 1

    Marquer x_{i+1} et x_{i+2} par 0

    if *le sommet en conflit avec x_i est non marqué* **then**

      Le marquer par X

    $i \leftarrow i + 3$

  else

    $i \leftarrow i + 1$

if *x_{n-1} est non marqué* **then**

  Marquer x_{n-1} par 1

  Marquer x_n par 0

if *x_n est non marqué* **then**

  Marquer x_n par 1

return L'ensemble des sommets marqués par 1

condition : après le marquage d'un sommet par 1 et le marquage de deux sommets par 0. On en déduit que le nombre de sommets marqués par X est inférieur ou égal à $n/4$.

Soit maintenant u un sommet non dominé quelconque. u n'est donc pas marqué par 1, sinon il serait dominé. Si le successeur de u est marqué par 1, u est voisin d'un sommet de S et est donc dominé : le successeur de u n'est donc pas marqué par 1. Supposons que le successeur de u soit marqué par 0. Les seuls sommets marqués par 0 dans l'algorithme sont le ou les deux premiers successeurs d'un sommet marqué par 1. Ainsi, si le successeur de u est marqué par 0, soit u est marqué par 1, soit le prédécesseur de u est marqué par 1 : les deux cas sont des contradictions.

Si un sommet u n'est pas dominé, son successeur n'est marqué ni par 0 ni par 1. Comme tous les sommets sont marqués, cela signifie que soit le successeur de u est marqué par X, soit u n'a pas de successeur. On en déduit qu'il y a au plus $n/4 + 1$ sommets non dominés, et donc S domine bien au moins $(3/4)n - 1$ sommets. □

3.2 Résultats de NP-complétude

Comme nous l'avons vu dans la section précédente, il est possible dans certains cas de dominer une grande proportion des sommets. Nous montrons maintenant qu'il est en général impossible de déterminer s'il existe un PDSwnC (ou PIDSwnC) dominant plus de la moitié des sommets. Il n'est donc pas possible d'obtenir un algorithme polynomial garantissant une proportion de couverture plus grande que l'algorithme 3 présenté à la section précédente. Ceci n'est cependant pas une preuve qu'il n'existe pas d'algorithme ayant un rapport d'approximation meilleur que 2. Nous proposons ensuite une preuve de NP-complétude dans le cas où le graphe support est un arbre.
Remarque 3. Si (G, C) est un graphe avec conflit à n sommets avec G un ensemble indépendant et C un couplage parfait, alors il n'existe pas de PDSwnC ou de PIDSwnC de (G, C) dominant plus de $n/2$ sommets.
Théorème 24. *Étant donné (G, C) un graphe avec conflits à n sommets, il est NP-Complet de déterminer s'il existe un PDSwnC dominant au moins $(1/2 + \epsilon)n$ sommets, avec ϵ arbitrairement petit tel que $n/4\epsilon$ soit entier, même si C est un couplage parfait et G est une union de chemins.*

Démonstration. Nous effectuons une réduction du problème de déterminer l'existence d'un DSwnC dans $(G = (V, E_1), C = (V, E_2))$ à n sommets où C est un couplage parfait et G est une union de chemins. Ce problème est NP-complet d'après le théorème 2 et la remarque 1 du chapitre 1.

Soit ϵ arbitrairement petit et tel que $n/4\epsilon$ soit entier. Soit I un ensemble de $2k = 2(n/4\epsilon - n/2)$ sommets et M un couplage parfait sur les sommets de I. Soit $(G' = (V \cup I, E_1), C' = (V \cup I, E_2 \cup M))$ une instance à $N = n + 2k$ sommets du PDSwnC.

Supposons qu'il existe S un PDSwnC de (G', C') dominant au moins $(1/2 + \epsilon)N$ sommets. Cet ensemble domine au moins $(\frac{1}{2} + \epsilon)(2k + n) = k + \frac{n}{2} + 2k\epsilon + \epsilon n = k + \frac{n}{2} + 2\epsilon\frac{n}{4\epsilon} - \frac{n}{2}) = k + \frac{n}{2} + \frac{n}{2} - \epsilon n + \epsilon n = k + n$ sommets. Or, comme I est un indépendant dont les sommets sont en conflit deux à deux, il n'est possible de dominer qu'au maximum k sommets de I. Il faut donc dominer n sommets de G. Comme G n'a pas de voisin, il doit donc exister un DSwnC de (G, C).

Supposons maintenant qu'il existe un DSwnC S de (G, C). Construisons $S' = S \cup S_1$ où S_1 est un sous-ensemble sans conflit de I à k sommets. Alors S' est sans conflit et domine $k + n = (1/2 + \epsilon)N$ sommets. □

Théorème 25. *Étant donné (G, C) un graphe avec conflits, il est NP-complet de déterminer s'il existe un PIDSwnC dominant au moins $(1/2+\epsilon)n$ sommets, avec ϵ arbitrairement petit et tel que $n/4\epsilon$ soit entier, même si C est un couplage parfait et G est une union de chemins.*

Démonstration. Nous effectuons une réduction identique à celle de la preuve du Théorème 24 à partir du problème IDSwnC.

Soit $(G = (V, E_1), C = (V, E_2))$ une instance à n sommets du IDSwnC où C est un couplage parfait et G est un chemin. Soit ϵ arbitrairement petit et tel que $n/4\epsilon$ soit entier. Soit I un ensemble de $2k = 2(n/4\epsilon - n/2)$ sommets et M un couplage parfait sur les sommets de I. Soit $(G' = (V \cup I, E_1), C' = (V \cup I, E_2 \cup M))$ une instance à $N = n + 2k$ sommets du PIDSwnC.

Supposons qu'il existe S un PIDSwnC de (G', C') dominant au moins $(1/2 + \epsilon)N$ sommets. Cet ensemble domine au moins $(\frac{1}{2} + \epsilon)(2k + n) = k + \frac{n}{2} + 2k\epsilon + \epsilon n = k + \frac{n}{2} + 2\epsilon\frac{n}{4\epsilon} - \frac{n}{2}) = k + \frac{n}{2} + \frac{n}{2} - \epsilon n + \epsilon n = k + n$ sommets. Or, comme I est un indépendant dont les sommets sont en conflit deux à deux, il n'est possible de dominer qu'au maximum k sommets de I. Il faut donc dominer n sommets de G. Comme G n'a pas de voisin, il doit donc exister un IDSwnC de (G, C).

Supposons maintenant qu'il existe un IDSwnC S de (G, C). Construisons $S' = S \cup S_1$ où S_1 est un sous-ensemble sans conflit de I à k sommets. Alors S' est un indépendant sans conflit et domine $k + n = (1/2 + \epsilon)N$ sommets. □

Les preuves précédentes se basaient sur l'existence d'un grand ensemble de sommets isolés dans le graphe support. Nous abordons maintenant le cas où le graphe support est connexe.

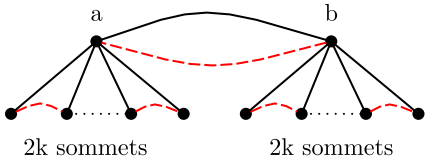


FIGURE 3.2 – Instance à $4k + 2$ sommets où il est impossible de dominer plus de $3k + 2$ sommets.

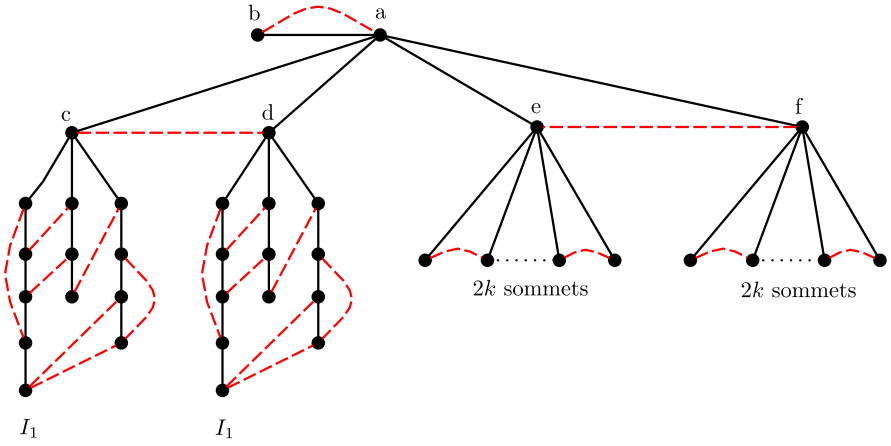


FIGURE 3.3 – Réduction de DSwnC à PDSwnC dans les arbres.

Remarque 4. Il existe des instances connexes où il est impossible de dominer asymptotiquement plus de 3/4 des sommets. Un exemple d'une telle instance est montré figure 3.2. Le graphe support est composé de deux étoiles de centre a et b reliées par une arête ab. Supposons que le sommet a appartienne à une solution. Alors b n'appartient pas à cette solution, et seulement k des $2k$ feuilles de l'étoile de centre b peuvent être dominées.

Nous montrons maintenant que même dans le cas d'un graphe support étant un arbre, il est NP-complet de déterminer s'il existe un PDSwnC dominant plus de 3/4 des sommets.

Théorème 26. *Étant donné (G, C) un graphe avec conflits, il est NP-complet de déterminer s'il existe un PDSwnC dominant au moins $(3/4+\epsilon)n$ sommets, avec ϵ arbitrairement petit de la forme $\epsilon = \frac{1}{8p+4}$ avec p un entier, même si C est un couplage parfait et G est un arbre.*

Démonstration. Cette preuve suit le même principe général que la preuve du Théorème 24, tout en utilisant un graphe support connexe (ici un arbre).

Soit $I_1 = (G = (V, E_1), C = (V, E_2))$ une instance du DSwnC à n sommets où C est un couplage parfait et G est une union de chemins. Choisissons ϵ arbitrairement petit et de la forme $\epsilon = \frac{1}{8p+4}$ et posons $k = np + 3p$.

Soit I_2 une instance construite comme illustré dans la figure 3.3. Cette instance comprend deux copies de I_1 chacune connectée par un sommet additionnel (c et d) eux mêmes en conflit, ainsi que deux étoiles de centre e et f (en conflit) à $2k$ feuilles, dont les feuilles sont en conflit deux à deux. Les sommets c, d, e, f ainsi qu'un sommet additionnel b sont connectés par un sommet a. L'instance ainsi construite est un arbre à $N = 2n + 4k + 6$ sommets et le graphe des conflits est un couplage parfait.

Soit S un PDSwnC de I_2 dominant au moins $(\frac{3}{4} + \epsilon)N = 3np + \frac{3n}{2} + 9p + \frac{9}{2} + \frac{4np}{8p+4} + \frac{2n}{8p+4} + \frac{12p}{8p+4} + \frac{6}{8p+4} = 3np + 2n + 9p + 6 = 3k + 2n + 6$ sommets. Remarquons qu'il est impossible de dominer sans conflit plus de $3k$ feuilles des étoiles. Ainsi, S domine $3k$ feuilles des étoiles, les sommets $a, ..., f$ et les $2n$ sommets des deux copies de I_1. Or, c et d étant en conflit, il existe une copie de I_1 dominée uniquement par ses sommets internes. Ainsi, il existe un DSwnC de I_1.

Supposons maintenant qu'il existe S un DSwnC de I_1. Soit S' un DSwnC de la copie de I_1. Alors $S_2 = S \cup S' \cup a \cup e \cup X$ où X est un ensemble sans conflit de k feuilles de l'étoile de centre f couvre $3k + 2n + 6$ sommet, soit $(3/4 + \epsilon)N$ sommets. □

3.3 Conclusion du chapitre

Nous avons étudié dans ce chapitre une version relâchée des problèmes de domination sans conflit, où le but est de trouver un dominant (indépendant) partiel sans conflit, dominant un maximum de sommets. Nous nous sommes restreints à un graphe des conflits de degré limité (en général un couplage), ce qui a permis de montrer l'existence d'un dominant (indépendant) partiel sans conflit, dominant au moins la moitié des sommets. Des algorithmes constructifs ont été proposés ayant un rapport d'approximation de 2.

Toutefois, nous avons également montré que pour un graphe quelconque, même quand le graphe des conflits est un couplage parfait, il est NP-complet de déterminer s'il existe un dominant (indépendant) partiel sans conflit, dominant strictement plus que cette moitié de sommets. Ceci n'est cependant pas suffisant pour affirmer qu'il n'existe pas d'algorithme d'approximation de rapport meilleur que 2, de futures recherches peuvent être envisagées dans ce sens.

Deuxième partie

Autres problèmes avec conflits

Introduction

Dans la partie précédente, nous avons pu observer que les problèmes de Dominant et de Dominant indépendant voient leur complexité drastiquement augmentée avec l'ajout de la contrainte *sans conflit* qui rend la détermination de l'existence d'une solution NP-complète, même dans des sous-cas très simples.

Nous allons maintenant étendre cette contrainte à d'autres problèmes : dans le chapitre 4, nous étudierons une contrainte identique sur d'autres problèmes de couverture de graphes classiques, comme le problème du Dominant Total (brièvement mentionné dans la partie précédente), le problème du Vertex Cover Connexe, et le problème de l'Arbre de Steiner.

Dans le chapitre 5, nous introduirons deux notions de conflits proches portant sur les langages et les automates. Nous nous intéresserons à la classe des langages réguliers avec conflits (selon ces deux notions), ainsi qu'à la taille des automates nécessaire à leur reconnaissance.

Chapitre 4

Autres problèmes de graphes avec conflits

Nous nous intéressons ici à d'autres problèmes de couverture de graphes avec la contrainte *sans conflit*. Nous montrerons qu'à l'image des problèmes de domination étudiés dans la partie précédente, les problèmes de Dominant Total, Vertex Cover Connexe et Arbre de Steiner sans conflit sont NP-complets même dans des cas très restreints. Le paramètre d'étirement des conflits, introduit dans la définition 9 de la partie précédente sera également regardé. La base de la plupart des réductions est encore une fois une restriction du problème 3-SAT que nous rappelons ici.

Définition 15. 3-SAT

Instance : (X, Cl) Un ensemble X de variables booléennes et un ensemble Cl de 3-clauses sur ces variables.

Question : Existe-t-il une affectation des variables de X vérifiant toutes les clauses de Cl ?

Rappelons également que ce problème est NP-complet, même dans le cas où chaque variable de X apparaît dans au plus trois clauses (sous forme positive ou négative)[Tov84].

La table 4.1 résume les résultats présentés dans ce chapitre. Notons que le problème du Vertex Cover n'y apparaît pas : il a été montré dans [DLP16] qu'il était polynomial de déterminer l'existence d'un vertex cover sans conflit. De plus, un algorithme d'approximation de rapport 2 a été proposé : le ratio d'approximation est le même que celui des meilleurs algorithmes connus pour le Vertex Cover classique. En effet, le problème du Vertex Cover sans conflit est équivalent au problème 2-SAT contrairement aux autres problèmes étudiés, équivalents au problème 3-SAT.

G		C		Complexité	Ref.
Classe	Degré	Classe	Étirement		
DOMINANT TOTAL SANS CONFLIT (TDSwnC)					
Biparti	≤ 4	Degré ≤ 1	$= 2$	NPC	Thm 27
Caterpillar	≤ 3	Degré ≤ 1	-	NPC	Thm 28
-	≤ 2	-	-	P	Thm 29
VERTEX COVER CONNEXE SANS CONFLIT(CVCwnC)					
Biparti	≤ 4	Degré ≤ 1	$= 2$	NPC	Thm 30
Arbre	-	-	-	P	Rmq 5
Split	-	-	-	P	Thm 31
-	$\geq (1/2 - \epsilon)n$	-	-	NPC	Thm 32
ARBRE DE STEINER SANS CONFLIT (STwnC)					
Biparti	≤ 4	Degré ≤ 1	$= 2$	NPC	Thm 33
Biparti Planaire	≤ 3	Degré ≤ 1	-	NPC	Thm 34
Triangulé Planaire	≤ 4	$\bigcup$ bipartis complets	-	NPC	Thm 35
Split	-	Degré ≤ 1	$= 1$	NPC	Thm 36
Dirac	$\geq n/2$	Degré $\bigcup$ chemins	-	NPC	Thm 37
Dirac	$\geq n/2$	Dirac	-	NPC	Thm 38
Arbre	-	-	-	P	Rmq 6

TABLE 4.1 – Résumé des résultats de complexité obtenus dans ce chapitre

4.1 Dominant total

Étant donné (G, C) où $G = (V, E)$ est le graphe support et C est le graphe des conflits, un *dominant total sans conflit* (TDSwnC) est un sous-ensemble de sommets $S \subseteq V$ tel que :

- pour tout $x \in V$ $\exists y \in S$ avec $xy \in E$

- pour tout $xy \in C$, $x \notin S$ ou $y \notin S$

Nous prouvons tout d'abord un résultat de NP-complétude quand l'étirement est exactement égal à 2. Ce résultat montre que le problème est difficile même quand les conflits sont très locaux.

Théorème 27. *Étant donné (G, C) un graphe avec conflits, déterminer s'il existe un TDSwnC est NP-complet même si G est un graphe biparti de degré maximum 4 et C est un graphe de degré maximum 1 et d'étirement exactement égal à 2.*

Démonstration. Soit (X, Cl) une instance de 3-SAT où chaque variable apparaît dans au plus 4 clauses. Supposons sans perte de généralité que chaque littéral est dans au plus 3 clauses. Construisons (G, C) une instance du TDSwnC comme suit : pour chaque variable $x_i \in X$ les sommets $x_i, \bar{x_i}$ et r_i sont créés ainsi que les arêtes $x_i r_i$ et $\bar{x_i} r_i$, et le conflit $x_i \bar{x_i}$. Pour chaque clause $c_m = (a \vee b \vee c)$ où a, b, c sont des littéraux, le sommet c_m et les arêtes $c_m a, c_m b, c_m c$ sont créés. Les sommets r_i sont de degré 2, les sommets c_m de degré 3 et les sommets x_i et $\bar{x_i}$ sont de degré au plus 4 (ils sont voisins de r_i et d'au plus 3 clauses). Un exemple est donné dans la figure 4.1.

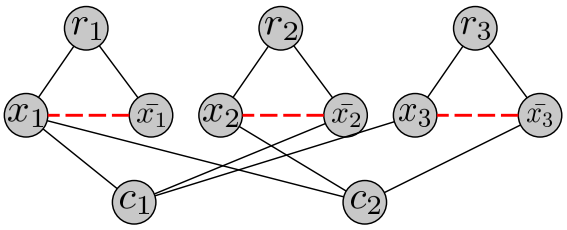


FIGURE 4.1 – Réduction de l'instance 3-SAT $(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x_3})$ à une instance du TDSwnC. Les tirets représentent les arêtes de conflit.

Soit A une affectation sur X vérifiant Cl. Construisons S un TDSwnC de (G, C). Pour chaque i, $r_i \in S$. Ainsi, les seuls sommets de G pas encore dominés sont les sommets c_m de chaque clause, et les sommets r_i. Pour chaque $x_i = 1$ de A, $x_i \in S$. Pour chaque $\bar{x_i} = 1$ de A, $\bar{x_i} \in S$. Comme A est une affectation, ceci n'engendre pas de conflits dans S. De plus, chaque clause est vérifiée, et donc chaque c_m a un voisin dans S. Pour chaque variable x_i, soit x_i soit $\bar{x_i}$ est fixé à **vrai**, et r_i est dominé. Ainsi S est un TDSwnC de (G, C).

Soit S un TDSwnC de (G, C) et soit A l'affectation suivante sur X : $x_i = 1$ si $x_i \in S$ et $x_i = 0$ si $\bar{x_i} \in S$. Pour chaque sommet c_i il existe un sommet $x_j \in S$ connecté à c_i pour le dominer, ainsi la clause correspondant est vérifiée par A. De plus, comme x_i et $\bar{x_i}$ sont en conflit, l'affectation est consistante. □

Nous prouvons maintenant la NP-complétude pour les caterpillars de degré maximum 3 quand le graphe des conflits est de degré maximum 1. Pour obtenir ceci, nous prouvons tout d'abord le résultat plus faible du lemme 16 puis dans les lemmes 17 et 18, nous présentons des gadgets pour simplifier le graphe support et le graphe des conflits.

Lemme 16. *Étant donné (G, C) un graphe avec conflits, déterminer s'il existe un TDSwnC est NP-complet même si G est une union disjointe de claws et C est une union disjointe de graphes bipartis complets à au plus 4 sommets.*

Démonstration. Soit (X, Cl) une instance de 3-SAT dans laquelle chaque variable apparaît dans au plus 4 clauses. Construisons (G, C) une instance de TDSwnC. Pour chaque clause $c_i = (a \vee b \vee c)$ où a, b, c sont des littéraux, construire une étoile de centre c_i avec 3 feuilles a, b, c. Pour chaque paire $\{a, \bar{a}\}$ de sommets, créer le conflit $a\bar{a}$. Le graphe G est une union de claws (qui sont des caterpillar de degré maximum 3) et le graphe des conflits est une union de graphes bipartis complets à au plus 4 sommets (car chaque variable est dans au plus 4 clauses). Un exemple de cette réduction est montré à la figure 4.2.

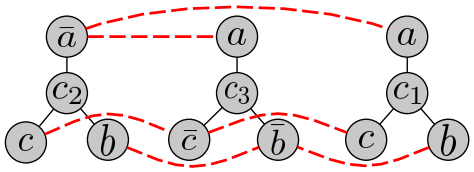


FIGURE 4.2 – Graphe équivalent à la formule 3-SAT suivante : $(\bar{a} \vee b \vee c) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (a \vee b \vee c)$. Les tirets représentent les conflits.

Soit A une affectation sur X vérifiant Cl. Construisons $S = \bigcup_i \{c_i\} \cup P$ où P est l'ensemble des sommets correspondant aux littéraux positifs de A. Comme A est une affectation, un sommet correspondant à un littéral et un sommet correspondant à sa négation ne peuvent appartenir simultanément à S. Ainsi, S est sans conflit. Pour chaque claw, les feuilles sont dominées par le centre. De plus, chaque clause est vérifiée et donc pour chaque claw, une feuille appartient à S et le centre est dominé. Ainsi, S est un TDSwnC de (G, C).

Soit S un TDSwnC de (G, C). Soit A l'affectation sur X suivante : $l_i =$ **vrai** si $l_i \in S$, $l_i =$ **faux** sinon. Un conflit existe pour chaque paire $a, \bar{a}$, ainsi un littéral et sa négation ne peuvent pas être simultanément fixés à **vrai**, et l'affectation est consistante. De plus, pour chaque clause c_i, il existe une claw dont le centre ne peut être dominé que par un sommet représentant un de ses littéraux. Ainsi, chaque clause est vérifiée. □

Dans la réduction ci-dessus, le graphe des conflits est une union disjointe de petits graphes bipartis complets. Le lemme suivant présente des gadgets pour décomposer ces bipartis en graphes de degré au plus 1.

Lemme 17. *Étant donné (G, C) un graphe avec conflits où G est une union disjointe de caterpillars de degré maximum $\delta > 1$ et C une union de graphes bipartis complets à au plus 4 sommets, il est possible de construire (G', C') où G' est une union disjointe de caterpillars de degré maximum δ et C un graphe de degré maximum 1 tel que (G, C) a un TDSwnC si et seulement si (G', C') a un TDSwnC.*

Démonstration. Soit (G, C) un graphe avec conflits où G est une union disjointe de caterpillars de degré maximum $\delta > 1$ et C une union disjointe de graphes bipartis complets à au plus 4 sommets. Décomposons les $K_{1,2}$, $K_{1,3}$ et $K_{2,2}$ du graphe des conflits en graphes de degré maximum 1 en utilisant les gadgets de la figure 4.3. On peut voir par une recherche exhaustive que le choix d'un sommet a empêche le choix de tous les sommets $\bar{a}$ (et réciproquement), et que les nouveaux chemins peuvent être dominés quel que soit le choix de a ou $\bar{a}$. Seuls des chemins ont été créés dans cette transformation, ainsi le degré maximum du graphe support n'a pas changé et le

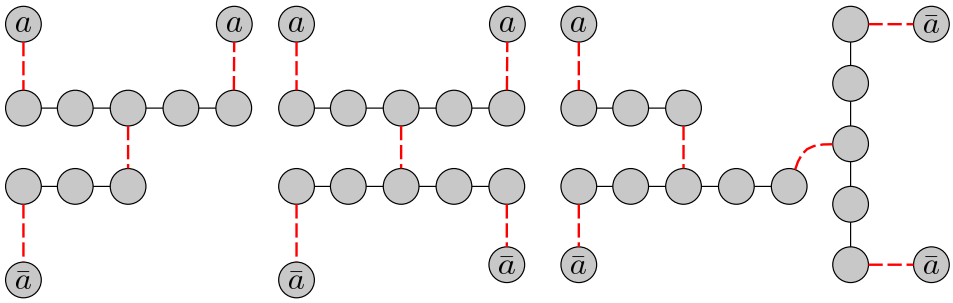


FIGURE 4.3 – Gadgets utilisés pour décomposer les $K_{1,2}$(gauche), $K_{2,2}$ (centre) et $K_{1,3}$ (droite) du graphe des conflits en graphes de degré au plus 1. Les tirets représentent les conflits.

graphe des conflits est maintenant de degré maximum 1. □

Le graphe des conflits est maintenant de degré maximum 1, mais le graphe support n'est pas connecté. Le résultat suivant montre comment connecter les caterpillars en utilisant des gadgets.

Lemme 18. *Étant donné (G, C) un graphe avec conflits où G est une union disjointe de caterpillars de degré maximum $\delta > 2$ et C est un graphe de degré maximum 1, il est possible de construire (G', C') où G' est un caterpillar (connexe) de degré maximum δ et C' est un graphe de degré maximum 1 tels que (G, C) a un TDSwnC si et seulement si (G', C') a un TDSwnC.*

Démonstration. Soit (G, C) un graphe avec conflits où G est une union disjointe de caterpillars de degré maximum $\delta > 2$ et C est un graphe de degré maximum 1. Si G est un seul caterpillar, le lemme est vrai. Sinon, soit p_1 et p_2 deux caterpillars de G et soit (G_1, C_1) le graphe avec conflits où p_1 et p_2 sont connectés par p, où p est le carterpillar montré à la figure 4.4. Plus précisément, on connecte les sommets 1 et 4 de p aux extrémités des plus longs chemins des caterpillars p_1 et p_2. Ainsi, le graphe est toujours une union de caterpillars. Les extrémités des chemins étant des feuilles, leur degré change de 1 à 2, le degré maximum du graphe ne change donc pas. De plus, p peut être dominé si et seulement si les sommets 1 et 4 n'appartiennent pas à la solution. Ceci peut être vu par une recherche exhaustive des TDSwnC du gadget. Ainsi, il peut être utilisé pour connecter deux caterpillars sans changer l'existence d'une solution. De plus, G_1 a un caterpillar de moins que G. On répète cette transformation jusqu'à ce qu'il ne reste plus qu'un seul caterpillar. □

Théorème 28. *Étant donné (G, C) un graphe avec conflits, déterminer s'il existe un TDSwnC est NP-complet, même si G est un caterpillar de degré maximum 3 et si C est un graphe de degré maximum 1.*

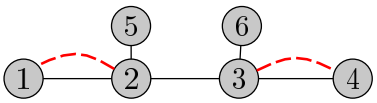


FIGURE 4.4 – Gadget utilisé pour connecter deux composantes connexes sans changer l'existence d'une solution. Les tirets représentent les conflits.

Démonstration. Comme les claws sont des caterpillars de degré maximum 3, le théorème 28 découle des lemmes 16, 17 et 18. □

Les résultats précédents ont montré que TDSwnC était NP-complet dans les caterpillars de degré maximum 3. Nous prouvons maintenant que ce résultat est, en un sens, le plus fort possible. En effet, le problème devient polynomial quand le degré maximum de G est 2.

Théorème 29. *Si G est un graphe de degré maximum 2, alors déterminer si (G, C) a un TDSwnC peut être fait en temps polynomial.*

Démonstration. Nous réduisons ce problème à 2-SAT pour lequel des algorithmes polynomiaux sont connus (voir [APT79]). Soit $(G = (V, E), C)$ une instance de TDSwnC où G est de degré maximum 2. Supposons sans perte de généralité qu'il n'existe pas de sommet isolé (dans le cas contraire, G n'a pas de TDSwnC). Construisons (X, Cl) une instance de SAT correspondante : $X = V$ et $Cl = \bigwedge_{x \in V}(\bigvee_{y \in N(x)} y) \bigwedge_{ab \in C}(\bar{a} \vee \bar{b})$. Chaque sommet a au plus deux voisins, on obtient donc une instance 2-SAT.

Soit A une affectation sur X vérifiant Cl. Alors $S = \{x \mid x = 1\}$ est sans conflit dans C car pour chaque conflit ab il existe une clause $\bar{a} \vee \bar{b}$. De plus, pour chaque $x \in V$, il existe une clause $\bigvee_{y \in N(x)} y$ et donc x est dominé par un de ses voisins. Ainsi S est un TDSwnC.

Soit S un TDSwnC de (G, C). Construisons A une affectation sur X. x est **vrai** si $x \in S$, x est **faux** sinon. Pour chaque clause c, il existe un sommet de V qui ne peut être dominé que par les sommets représentant des littéraux de cette clause. Un de ces sommets appartient à S, la clause est donc vérifiée. Pour chaque clause $\bar{a} \vee \bar{b}$, il existe un conflit ab, ainsi au plus un des sommets appartient à S, et la clause est vérifiée. Ainsi A vérifie Cl. □

4.2 Connected Vertex Cover

Étant donné (G, C) où $G = (V, E)$, un *vertex cover connexe sans conflit* (CVCwnC) est un sous-ensemble de sommets $S \subseteq V$ tel que :

- pour tout $xy \in E$, $x \in S$ ou $y \in S$

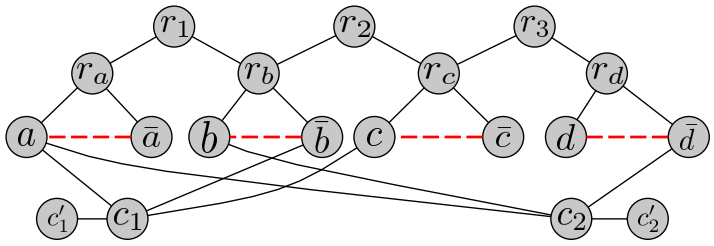


FIGURE 4.5 – Graphe équivalent à la formule 3-SAT $(a \vee \bar{b} \vee c) \wedge (a \vee b \vee \bar{d})$. Les tirets représentent les conflits.

- pour tout $xy \in C$, $x \notin S$ ou $y \notin S$

- $G[S]$ est connexe.

Théorème 30. *Étant donné (G, C) un graphe avec conflits, déterminer s'il existe un CVCwnC est NP-complet même si G est un graphe biparti de degré maximum 4 et si C est un graphe de degré maximum 1 et d'étirement 2.*

Démonstration. Soit (X, Cl) une instance 3-SAT où chaque variable est dans au plus 4 clauses. Construisons (G, C) une instance du CVCwnC. Pour chaque variable α, les sommets $\alpha, \bar{\alpha}$ et r_α sont créés, ainsi que les arêtes αr_α et $\bar{\alpha} r_\alpha$ et le conflit $\alpha\bar{\alpha}$. Les sommets r_α sont connectés par des sommets intermédiaires r_i. Pour chaque clause $c_i = (a \vee b \vee c)$, deux sommets c_i et c'_i sont créés, ainsi que des arêtes $c_i c'_i, c_i a, c_i b, c_i c$. Ainsi G est un graphe biparti de degré maximum 4 et C est de degré maximum 1 et d'étirement 2. Un exemple de cette construction est présenté figure 4.5.

Soit A une affectation sur X vérifiant Cl. Soit S l'ensemble des sommets correspondant aux littéraux positifs de A. Les sommets r_α, r_i et c_i sont également ajoutés à S. S est donc un vertex cover. De plus, S est sans conflit car un littéral et sa négation ne peuvent pas être **vrai** simultanément. Les sommets r_α, r_i et les sommets correspondant aux littéraux sont connectés, et pour chaque clause $c_i = (a \vee b \vee c)$, a, b ou c appartient à S, ainsi c_i est connecté au reste du graphe et S est connexe.

Soit S un CVCwnC de (G, C). Les littéraux correspondant aux sommets de S sont fixés à **vrai**, les autres variables sont fixées à **faux**. Comme des conflits sont présents entre les littéraux et leurs négations, on obtient bien une affectation sur X. De plus, les sommets c_i appartiennent nécessairement à S, et chacun doit être connecté par un sommet correspondant à un littéral de la clause. Ainsi, chaque clause est vérifiée. □

Nous présentons maintenant deux classes de graphes dans lesquelles déterminer l'existence d'un CVCwnC peut être fait en temps polynomial.

Remarque 5. Déterminer l'existence d'un CVCwnC est polynomial dans les arbres. En effet, il existe un unique vertex cover connexe minimal pour l'inclusion : l'ensemble des sommets internes de l'arbre. Il suffit de tester si cet ensemble est sans conflit.

Théorème 31. *Étant donné (G, C) un graphe avec conflits, déterminer s'il existe un CVCwnC peut être fait en temps polynomial si G est un* split graph.

Démonstration. Soit $G = (V, E)$ un split graph où $V = K \cup I$ avec K une clique et I un stable, et C le graphe des conflits.

- si K est sans conflit, alors K est un CVCwnC de (G, C).

- si $C[K]$ est une étoile de centre a, a, alors si $S = (K - a) \cup N_G(a)$ est sans conflit, S est un CVCwnC. (Il y a deux possibilités pour le centre a si l'étoile est une arête). Si $(K - a) \cup N_G(a)$ a un conflit, alors (G, C) n'a pas de CVCwnC.

- si $C[K]$ n'est pas une étoile, alors au moins 2 sommets a et b de K ne peuvent pas être dans une solution : l'arête ab ne sera pas couverte et il n'y a pas de CVCwnC. □

Dans le théorème 30, nous avons prouvé la NP-complétude pour les graphes bipartis. Nous nous intéressons maintenant aux graphes denses, et prouvons des résultats de difficulté dans des classes de graphes denses.

Théorème 32. *Pour tout $\epsilon > 0$, étant donné (G, C) un graphe avec conflits, déterminer s'il existe un CVCwnC est NP-complet même si G est de degré minimum $(1/2 - \epsilon)n$.*

Démonstration. Soit (X, Cl) une instance 3-SAT à m clauses sur n variables. Construisons (G, C) une instance du CVCwnC. Soit d le plus petit entier supérieur à $(m+2n)/2\epsilon$. Pour chaque variable $x \in X$, créer les sommets $x, \bar{x}$. Pour chaque clause c_i créer le sommet c_i. Soit G_X l'ensemble des sommets représentant les littéraux. Le graphe G_X a $2n$ sommets. Soit G_{Cl} l'ensemble des m sommets représentant les clauses. Construire G_K une clique de taille d et G_I un stable de taille d. Ajouter des arêtes pour que chaque sommet de G_K soit connecté à chaque sommet de G_X, chaque sommet de G_{Cl} soit connecté à chaque sommet de G_I et chaque sommet de G_K soit connecté à chaque sommet de G_I. Pour chaque clause $c_i = (a \vee b \vee d)$, créer les arêtes $c_i a, c_i b, c_i d$. Construisons maintenant C. Chaque paire $\{x, \bar{x}\}$ est en conflit. Chaque sommet de G_I est en conflit avec tous les autres sommets de G. Un schéma du graphe support utilisé dans cette construction est présenté dans la figure 4.6. Dans un souci de lisibilité, les conflits n'ont pas été représentés.

Le graphe G a $m+2n+2d$ sommets, et $d \geq (m+2n)/2\epsilon$. Grâce aux bipartis complets, les sommets de G_X, G_{Cl}, G_I et G_K sont de degré au moins d.

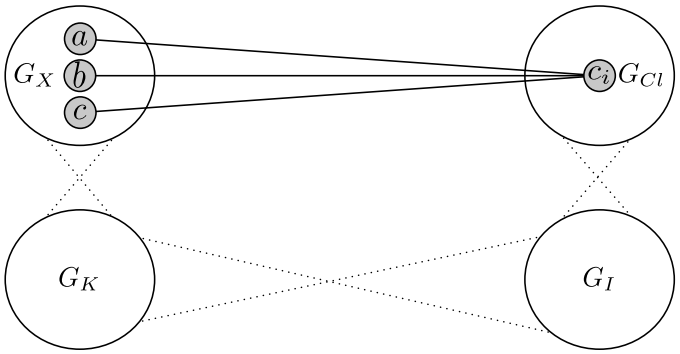


FIGURE 4.6 – Construction du graphe dense. Les pointillés représentent les bipartis complets entre les sous-ensembles de sommets.

Par des opérations arithmétiques basiques, on peut déduire que $d > (1/2 - \epsilon)(m + 2n + 2d)$, le graphe satisfait donc les hypothèses du théorème.

Soit A une affectation sur X vérifiant Cl, et soit A_1 l'ensemble des littéraux positifs de A. Soit $VC = A_1 \cup G_K \cup G_{Cl}$. L'ensemble VC ne contient pas simultanément de sommets représentant un littéral et sa négation, ni de sommets de G_I, c'est donc un ensemble sans conflit dans C. Les arêtes entre G_K et G_X, G_{Cl} et G_I, G_K et G_I, G_X et G_{Cl} sont couvertes. Ainsi VC est un vertex cover de G. Les sommets de G_K et G_X sont connectés. De plus, pour chaque sommet $c_i \in Cl$ il existe un sommet x de A_1 correspondant à un littéral positif de c_i. Ainsi VC est connexe.

Soit VC un CVCwnC de (G, C). Soit A l'affectation suivante : pour chaque littéral x, si x est représenté par un sommet de VC, alors x est **vrai**, sinon x est **faux**. L'ensemble VC n'est pas réduit à un seul sommet, ce qui implique que VC ne peut pas contenir de sommets de G_I, car chaque sommet de G_I est en conflit avec tous les autres sommets. L'ensemble VC doit donc contenir le voisinage de G_I : $G_{Cl} \cup G_K$. De plus, chaque sommet de G_{Cl} doit être connecté à G_K. Ceci ne peut être fait que par des sommets de G_X. Pour chaque sommet représentant une clause $(a \vee b \vee c)$, il existe un sommet représentant un de ses littéraux dans VC. Ainsi, A vérifie Cl. De plus, un sommet représentant un littéral et un sommet représentant sa négation ne peuvent pas être **vrai** tous les deux, l'affectation est donc consistante. □

4.3 Arbre de Steiner

Un arbre de Steiner de $(G = (V, E), M)$ où $M \subseteq V$ est un sous-arbre de G qui inclut tous les sommets de M.

Étant donné (G, M, C) où $G = (V, E)$ et $M \subseteq V$, un *arbre de Steiner sans conflit* (STwnC) est un sous-ensemble de sommets $S \subseteq V$ tel que :

- pour tout $xy \in C$, $x \notin S$ or $y \notin S$

- $M \subseteq S$

- $G[S]$ est connexe.

Si $G[S]$ est connexe et sans conflit, il est facile d'en extraire un arbre couvrant de S, c'est pourquoi nous nous intéressons ici uniquement à l'ensemble de sommets constituant l'arbre.

Tout d'abord, nous prouvons la NP-complétude quand les conflits sont locaux.

Théorème 33. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer s'il existe un STwnC est NP-complet même si G est un graphe biparti de degré maximum 4 et C un graphe de degré maximum 1 et d'étirement 2.*

Démonstration. Soit (X, Cl) une instance de 3-SAT dans laquelle chaque variable apparaît dans au plus 4 clauses. Supposons sans perte de généralité que chaque littéral apparaît dans au plus 3 clauses. Construisons (G, M, C) une instance de STwnC. Pour chaque variable α, les sommets $\alpha, \bar{\alpha}$ et r_α sont créés, ainsi que les arêtes αr_α et $r_\alpha \bar{\alpha}$ et le conflit $\alpha\bar{\alpha}$. Les sommets r_α sont connectés par des sommets intermédiaires m_i. Pour chaque clause $c_i = (a \vee b \vee c)$, un sommet c_i est créé, ainsi que les arêtes $c_i a, c_i b, c_i c$. Le graphe est de degré maximum 4, et le graphe des conflits est de degré maximum 1 et d'étirement 2. L'ensemble M est composé de tous les sommets m_i et c_i. Un exemple est montré à la figure 4.7.

Soit A une affectation sur X vérifiant Cl. Soit $S = M \bigcup_\alpha \{r_\alpha\} \cup P$ où P est l'ensemble de sommets représentant les littéraux positifs de A. Comme une variable et sa négation ne peuvent être **vrai** simultanément, S est sans conflit. Les sommets m_i et r_α sont connectés. De plus, pour chaque sommet c_i, il existe une clause c_i dans laquelle un des littéraux est **vrai** dans A, ainsi c_i est connecté aux autres sommets de S. Ainsi, S est connexe dans G.

Soit S un STwnC de (G, M, C). Les littéraux de X dont les sommets sont dans S sont fixés à **vrai**. Les variables non encore fixées sont fixées à **faux**. Comme des conflits existent entre les sommets représentant des variables et leurs négations, on obtient une affectation consistante. De plus, pour chaque sommet c_i représentant une clause $a \vee b \vee c$, un des sommets a, b, c doit appartenir à S pour assurer la connectivité avec les sommets m_j de M, la clause est donc vérifiée. □

Les théorèmes suivants sont des résultats de NP-complétude pour des classes

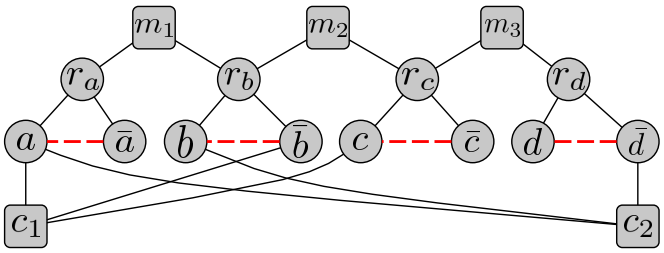


FIGURE 4.7 – Graphe équivalent à la formule 3-SAT $(a \vee \bar{b} \vee c) \wedge (a \vee b \vee \bar{d})$. Les sommets de M sont carrés. Les tirets représentent les conflits.

de graphe support plus restreintes, mais les conflits ne sont plus locaux (l'étirement n'est pas borné)

Théorème 34. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer s'il existe un STwnC est NP-complet même si G est un graphe planaire biparti de degré maximum 3 et C est un graphe de degré maximum 1.*

Démonstration. Soit (X, Cl) une instance de 3-SAT où chaque variable apparaît dans au plus 4 clauses. Construisons (G, M, C) une instance du STwnC où G est un graphe planaire biparti de degré maximum 3 et C un graphe de degré maximum 1 tels qu'il n'existe une affectation sur X vérifiant Cl si et seulement si il existe un STwnC de (G, M, C).

Pour chaque clause, créer un gadget de 15 sommets composés de trois chemins non disjoints ayant les mêmes extrémités, chaque chemin correspondant à un littéral. Ce gadget est présenté dans la figure 4.8. Ces gadgets sont connectés linéairement, et l'ensemble M, composé de seulement deux sommets, est le premier sommet du premier gadget et le dernier sommet du dernier gadget. Pour chaque paire de littéraux $\{x, \bar{x}\}$, ajouter un conflit entre un sommet sans conflit du chemin représentant x et un sommet sans conflit du chemin représentant $\bar{x}$. Le graphe est biparti planaire de degré maximum 3 et le graphe des conflits est de degré maximum 1.

Soit A une affectation sur X vérifiant Cl. Construisons S un sous-ensemble de G. Pour chaque gadget représentant une clause $(a \vee b \vee c)$, choisir dans S un chemin représentant un littéral positif. Chaque gadget est traversé, les sommets de M sont donc connectés. L'ensemble S est le chemin connectant les deux sommets de M. De plus, ce chemin ne passe pas par des chemins représentant des variables et leur négation, il est donc sans conflit.

Soit S un STwnC de (G, M, C). Supposons sans perte de généralité que S est minimal pour l'inclusion (sinon, le minimaliser). Construisons A une affectation sur X vérifiant Cl. Pour chaque gadget représentant une clause $(a \vee b \vee c)$, S contient un chemin représentant un de ses littéraux. Celui-

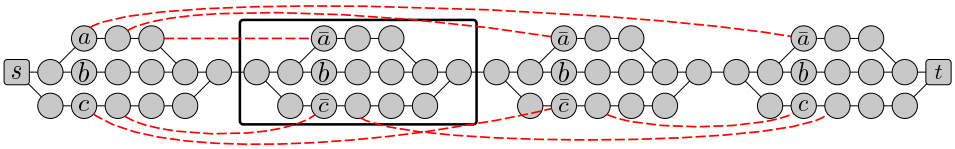


FIGURE 4.8 – Graphe équivalent à la formule 3-SAT $(a \vee b \vee c) \wedge (\bar{a} \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee c)$. Les sommets de M sont carrés. Dans le rectangle noir, le gadget correspondant à la clause $(\bar{a} \vee b \vee \bar{c})$. Les tirets représentent les conflits.

ci est fixé à **vrai** dans A. Comme il existe des conflits entre les chemins représentant les variables et leur négation, l'affectation sur A est consistante. De plus, pour chaque clause, un littéral est **vrai**, ainsi, A vérifie Cl. □

Théorème 35. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer si il existe un STwnC est NP-complet même si G est un graphe triangulé planaire de degré maximum 4 et que C est une union disjointe de graphes bipartis complets à au plus 4 sommets.*

Démonstration. Soit (X, Cl) une instance de 3-SAT dans laquelle chaque variable apparaît dans au plus 4 clauses. Construisons (G, M, C) une instance du STwnC. Pour chaque clause $c_i = (a \vee b \vee c)$, on crée les sommets c_i, c'_i, a, b, c, m_i et les arêtes $c'_i c_i$,$c_i a$,$c_i b$, $c_i c$, ab, ac, am_i, bm_i, bc, cm_i. L'ensemble M est l'ensemble de tous les sommets m_i et c'_i. Les sommets c'_i sont connectés pour former un chemin. Pour chaque paire de sommets $\{\alpha, \bar{\alpha}\}$ représentant un littéral et sa négation, on crée le conflit $\alpha\bar{\alpha}$. Le graphe est triangulé planaire de degré maximum 4 et le graphe des conflits est une union de graphes bipartis complets à au plus 4 sommets. Un exemple est montré figure 4.9.

Soit A une affectation sur X vérifiant Cl. Construisons $S = M \bigcup_i \{c_i\} \cup P$ où P est l'ensemble de sommets représentant des littéraux positifs de A. Alors $\bigcup_i \{c_i\} \bigcup_i \{c'_i\}$ est connexe. De plus, pour chaque clause c_i, un sommet x correspondant à un littéral positif appartient à S. Ainsi, le sommet m_i est connecté et S est connexe dans G. De plus, les conflits apparaissent uniquement entre les littéraux et leurs négations, qui ne peuvent être **vrai** simultanément dans A. Ainsi S est sans conflit.

Soit S un STwnC de (G, M, C). Les littéraux correspondant aux sommets de S sont fixés à **vrai**, les autres variables sont fixées à **faux**. Comme il existe des conflits entre les sommets représentant des littéraux et les sommets représentant leurs négations, on obtient une affectation consistante. De plus, chaque sommet m_i doit être connecté aux autres sommets de S par un sommet représentant un littéral de la clause associée, qui est donc vérifiée. □

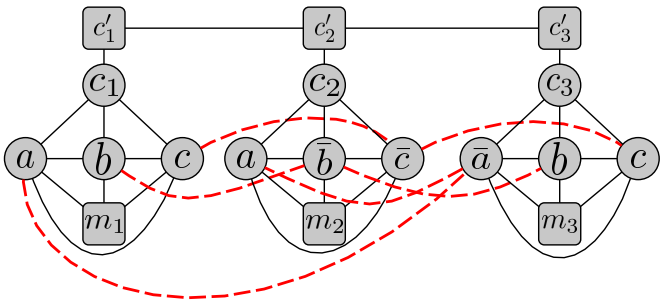


FIGURE 4.9 – Graphe équivalent à la formule 3-SAT $(a \vee b \vee c) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b \vee c)$. Les sommets de M sont carrés. Les tirets représentent les conflits.

Remarque 6. Déterminer l'existence d'un STwnC est polynomial dans les arbres. Il existe un unique arbre de Steiner minimal pour l'inclusion : il suffit de tester si celui-ci est sans conflit.

Théorème 36. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer s'il existe un STwnC est NP-complet même si G est un split graph et C un graphe de degré maximum 1 et d'étirement 1.*

Démonstration. Soit (X, Cl) une instance de 3-SAT. Supposons sans perte de généralité que Cl contient plusieurs clauses. Créons $(G = (K, I, E), M, C)$ une instance du STwnC. Chaque littéral devient un sommet de K et chaque clause devient un sommet de I. Des conflits sont ajoutés entre les littéraux et leurs négations, et des arêtes entre les clauses et leurs littéraux. Le graphe ainsi construit est un split graph, et le graphe des conflits est de degré maximum 1 et d'étirement 1. Posons $M = I$. Un exemple est montré figure 4.10.

Soit A une affectation sur X vérifiant Cl. Construisons $S = I \cup P$ avec P l'ensemble des sommets correspondant à des littéraux positifs de A. Comme les littéraux et leurs négations ne peuvent être **vrai** simultanément, S est sans conflit. De plus, pour chaque $c_i \in I$, la clause associée est vérifiée, il existe donc un voisin dans $K \cap S$. Ainsi, chaque sommet de I est connecté à $K \cap S$ et comme K est une clique, S est connecté.

Soit S un STwnC de $(G = (K, I, E), M, C)$. Les littéraux correspondant aux sommets de S sont fixés à **vrai**, les autres variables sont fixées à **faux**. Comme il existe des conflits entre les littéraux et leurs négations, on obtient une affectation consistante. De plus, chaque sommet c_i doit être connecté à K par un sommet représentant un littéral de la clause associée, qui est donc vérifiée.

□

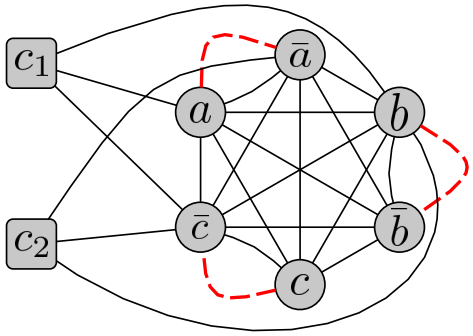


FIGURE 4.10 – Graphe équivalent à la formule 3-SAT. $(a \vee \bar{c} \vee b) \wedge (\bar{a} \vee b \vee \bar{c})$. Les sommets de M sont carrés. Les tirets représentent les conflits.

Les précédents théorèmes montraient la NP-complétude dans des graphes peu denses (ou split) pour le graphe support et le graphe des conflits. Nous prouvons maintenant que ce problème reste NP-complet dans une classe de graphes denses : les graphes de Dirac (qui sont des graphes de degré minimum $n/2$).

Théorème 37. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer s'il existe un STwnC est NP-complet même si G est un graphe de Dirac et C est une union disjointe de P_1, P_2 et P_3.*

Démonstration. Soit $(G_1 = (V_1, E_1), M_1, C_1)$ une instance du STwnC où C_1 est un graphe de degré maximum 1. Ce problème est NP-complet d'après le théorème 33. Construisons $(G_2 = (V_2, E_2), M_2, C_2)$ un graphe avec conflits tel qu'il existe un STwnC de (G_2, M_2, C_2) si et seulement si il existe un STwnC de (G_1, M_1, C_1). Soit $n = |V_1|$. Soit $V_2 = V_1 \cup A \cup B$ avec A un chemin de longueur n et B un indépendant de taille $2n$. $M_2 = M_1 \cup A$. A est connecté à un sommet arbitraire de M_1. Un graphe biparti complet est ajouté entre B et $V_2 - B$. Les conflits de C_1 sont ajoutés à C_2. De plus, chaque sommet de A est en conflit avec deux sommets distincts de B. Par construction, G_2 est un graphe de Dirac et C_2 une union disjointe de P_1, P_2 et P_3.

Soit T_1 un STwnC de G_1. Alors $T_2 = T_1 \cup A$ est un STwnC de (G_2, M_2, C_2).

Supposons maintenant qu'il existe T_2 un STwnC de (G_2, M_2, C_2). Alors $T_2 \cap B = \emptyset$. De plus, les sommets de A ne connectent pas les sommets de V_1. Ainsi, $T_1 = T_2 - A$ est un STwnC de (G_1, M_1, C_1). □

Théorème 38. *Étant donné (G, M, C) un graphe avec conflits et un sous-ensemble de sommets, déterminer s'il existe un STwnC est NP-complet même si G et C sont des graphes de Dirac.*

Démonstration. Soit $(G_1 = (V_1, E_1), M_1, C_1)$ une instance de STwnC. Nous construisons (G_2, M_2, C_2) avec $G_2 = (V_2, E_2)$ un graphe de Dirac avec conflits tel qu'il existe un STwnC de (G_2, M_2, C_2) si et seulement si il existe un STwnC de (G_1, M_1, C_1). Soit $n = |V_1|$. Soit $V_2 = V_1 \cup A$ où A est un indépendant de taille n. $M_2 = M_1$. Un graphe biparti complet est créé entre V_1 et A. Les conflits de C_1 sont ajoutés à C_2. De plus, chaque sommet de A est en conflit avec tous les sommets de V_1. Par construction, G_2 et C_2 sont des graphes de Dirac.

Soit T un STwnC de (G_1, M_1, C_1). Alors c'est également un STwnC de (G_2, M_2, C_2).

Supposons maintenant qu'il existe T un STwnC de (G_2, M_2, C_2). Alors $T \cap A = \emptyset$. Ainsi, T est un STwnC de (G_1, M_1, C_1). □

4.4 Conclusion du chapitre

Nous observons à nouveau que les problèmes étudiés restent NP-complets pour des cas assez restreints même si les résultats sont moins tranchés que pour les problèmes de domination étudiés dans la partie précédente. À l'exception du problème du dominant total sans conflit dans les graphes de degré maximum 2 (qui se réduit au problème 2-SAT), les autres résultats de polynomialité sont assez triviaux : il suffit de tester un nombre constant de cas pour déterminer l'existence d'une solution.

Notons que le résultat de NP-complétude pour le problème de l'arbre de Steiner dans les graphes bipartis planaires utilise un groupe composé de seulement deux sommets : le résultat est donc également vrai pour le problème de déterminer l'existence d'un chemin entre deux sommets.

Nous avons montré la NP-complétude de ces problèmes pour des conflits très locaux (étirement égal à 2). Pour les problèmes de l'Arbre de Steiner et du Dominant Total (voir partie précédente), le problème est également NP-complet dans le cas d'un étirement égal à 1. Cette question reste ouverte pour le problème du vertex cover connexe.

Chapitre 5

Automates avec conflits

5.1 Introduction et notations

Dans ce chapitre, nous étudions une notion voisine de conflits, appliquée aux automates et aux langages.

Nous commencerons par rappeler quelques concepts de base sur les langages formels nécessaires à la compréhension de ce chapitre (voir [Linon] pour plus de détails). Σ, un ensemble fini de *symboles*, est appelé un *alphabet*. Un *mot* est une séquence finie de symboles. Le *mot vide* est noté λ. Un *langage* $\mathcal{L}$ est un ensemble de mots. Un *AFD* (Automate Fini Déterministe) M est composé d'un ensemble fini Q d' *états* incluant un unique état *initial*. L'ensemble des états *finaux* (ou *accepteurs*) est noté F ($F \subseteq Q$).

La *fonction de transition* δ est dite *déterministe* car pour tout $p \in Q$ et tout $c \in \Sigma$ il existe un *unique* état $q \in Q$ (p et q peuvent être égaux) tel que $\delta(p, c) = q$. Si $w = a_1 \dots a_k$ est un mot et p un état, on note $\delta^*(p, w)$ l'unique état dans lequel M est après avoir traité la séquence de symboles dans l'ordre de w en partant de l'état p. Un mot w est *accepté* par M d'état initial q_0 si $\delta^*(w, q_0) \in F$. L'ensemble des mots acceptés par M est appelé le langage accepté par M : $\tau(M) = \{w : \delta^*(w, q_0) \in F\}$. Un résultat classique dit qu'il existe un AFD M reconnaissant $\mathcal{L}$ si et seulement si $\mathcal{L}$ est *régulier*.

Introduction des conflits Soit M un AFD, son alphabet Σ et son langage accepté $\mathcal{L}$. Nous ajoutons de nouvelles contraintes, les *conflits* G, de deux sortes.

Dans la section 5.2, G est un *graphe* sur les symboles de Σ : un mot contient un conflit si il contient deux symboles reliés par une arête de G. Un conflit dénote ici une paire de symboles interdits, ne pouvant pas être présents dans

le même mot. Nous montrons que le langage $\mathcal{L}^{\mathcal{C}}$ composé des mots de $\mathcal{L}$ sans conflit est régulier. Pour prouver ceci, nous transformons M en un nouvel AFD $\mathcal{M}^{\mathcal{C}}$ acceptant $\mathcal{L}^{\mathcal{C}}$. Cependant, la taille de $\mathcal{M}^{\mathcal{C}}$ est exponentielle en la taille de l'instance initiale (M et G). Une question naturelle est donc de se demander s'il existe un AFD de taille polynomiale acceptant $\mathcal{L}^{\mathcal{C}}$. Dans la section 5.3 nous prouvons que ce n'est pas le cas en analysant une instance particulière $\mathcal{L}$ et $G(\Sigma)$, et prouvant qu'un AFD acceptant $\mathcal{L}^{\mathcal{C}}$ doit contenir un nombre exponentiel d'états.

Dans la section 5.4 les conflits ne sont plus entre les symboles de Σ mais entre les états de M. Nous définissons le nouveau langage $\mathcal{L}^{\mathcal{Q}}$ composé des mots de $\mathcal{L}$ dont le traitement par M ne passe pas par deux états en conflits.(Une définition plus précise sera donnée section 5.4). Nous prouvons maintenant que $\mathcal{L}^{\mathcal{Q}}$ est régulier.

5.2 Les langages réguliers avec conflits sur les symboles restent réguliers

Soit $\mathcal{L}$ un langage régulier quelconque sur l'alphabet Σ. Soit M un AFD acceptant $\mathcal{L}$ ($\tau(M) = \mathcal{L}$). Dans cette section nous introduisons $G(\Sigma)$ que nous appelons *graphe des conflits*, un graphe non orienté dont les sommets sont les symboles de Σ. Chaque arête uv de $G(\Sigma)$ est appelée un*conflit* entre les symboles u et v de Σ.

Étant donné $\mathcal{L}$ et $G(\Sigma)$ on note $\mathcal{L}^{\mathcal{C}}$ le langage composé de tous les mots de $\mathcal{L}$ qui ne contiennent pas deux symboles en conflits dans $G(\Sigma)$:

$$\mathcal{L}^{\mathcal{C}} = \{w = a_1 \dots a_k : w \in \mathcal{L} \text{ and } a_i a_j \notin G(\Sigma)(\forall i, j)\}$$

Le problème auquel nous nous intéressons ici est : étant donné un AFD M acceptant $\mathcal{L}$ et un graphe des conflits $G(\Sigma)$, $\mathcal{L}^{\mathcal{C}}$ est-il un langage régulier ? Les données de notre problème dans cette section sont un AFD M sur un alphabet Σ, reconnaissant un langage $\mathcal{L}$ et un graphe des conflits $G(\Sigma)$. On note Q l'ensemble des états de$M$, q_0 son unique état initial, F son ensemble d'états finaux et δ sa fonction de transition. De cette instance, nous allons construire un AFD $\mathcal{M}^{\mathcal{C}}$ acceptant $\mathcal{L}^{\mathcal{C}}$, et ainsi prouver que$\mathcal{L}^{\mathcal{C}}$ est régulier.

Construction de $\mathcal{M}^{\mathcal{C}}$. Soit $\mathcal{STAB}$ la famille des *stables* de $G(\Sigma)$, c'est à dire des ensembles de symboles sans conflit. Créons un nouvel état initial q_0' pour $\mathcal{M}^{\mathcal{Q}}$, qui est également final si et seulement si q_0 est final dans M. Maintenant, pour chaque stable S de $\mathcal{STAB}$, on crée une copie notée M_S de M. Chaque copie de l'état initial q_0 dans chaque M_S est maintenant

non initiale (seul q'_0 est initial dans $\mathcal{M}^{\mathcal{C}}$). Chaque état final de M_S reste final dans la construction. On ajoute aussi un état non final TRASH. Nous modifions maintenant les fonctions de transition de ces copies pour obtenir $\delta^{\mathcal{C}}$, la fonction de transition finale de $\mathcal{M}^{\mathcal{C}}$.

Pour tout $c \in \Sigma$, $\delta^{\mathcal{C}}(\text{TRASH}, c) = \text{TRASH}$ (Une boucle est créée sur l'état TRASH sans possibilité d'en sortir).

Concernant le nouvel état initial q'_0 nous avons : pour chaque $c \in \Sigma$, $\delta^{\mathcal{C}}(q'_0, c) = p$ où p est la copie de l'état $\delta(q_0, c)$ dans la copie $M_{\{c\}}$ (L'ensemble $\{c\} \in \mathcal{STAB}$ est un stable à un seul élément).

L'idée globale et informelle de la transformation est que le traitement d'un mot w par $\mathcal{M}^{\mathcal{C}}$ sera dans un état de M_S quand l'ensemble de symboles de w déjà traités sera S. Ainsi, la copie M_S est utilisée comme "mémoire" des symboles déjà traités. Ce mécanisme permet de sortir dans l'état TRASH quand un symbole en conflit est rencontré. Sinon, le traitement de w continue dans les copies M_S de $\mathcal{M}^{\mathcal{Q}}$ avec une logique similaire à celle de M.

$\mathcal{M}^{\mathcal{C}}$ reste dans la copie M_S tant que le symbole traité actuellement est dans S : pour tout $c \in S$ et tout état p de M_S, $\delta^{\mathcal{C}}(p, c) = q$ où q est la copie de l'état $\delta(p, c)$ dans M_S.

Si l'automate est dans l'état p et que le symbole courant c est en conflit avec un symbole de S, alors l'automate doit aller dans l'état TRASH (ceci représente la détection d'un conflit entre c et un précédent symbole déjà traité) : pour tout c en conflit avec n'importe quel symbole de S et pour tout état p de M_S, $\delta^{\mathcal{C}}(p, c) = \text{TRASH}$.

Le dernier cas à prendre en compte est celui où le symbole lu c n'est pas dans S et n'est pas en conflit avec les éléments de S. Dans ce cas, $S' = S \cup \{c\}$ est un stable ($S' \in \mathcal{STAB}$). Ainsi, pour tout état p de M_S, et pour tout symbole c avec $S' = S \cup \{c\} \in \mathcal{STAB}$ la transition est $\delta^{\mathcal{C}}(p, c) = q$ où q est la copie de l'état $\delta(p, c)$ (de M) dans $M_{S'}$.

Ceci termine la construction de $\mathcal{M}^{\mathcal{C}}$ qui est un AFD (en effet $\delta^{\mathcal{C}}(p, c)$ existe et est unique pour chaque état de $\mathcal{M}^{\mathcal{C}}$ et chaque symbole de $c \in \Sigma$. De plus, $\mathcal{M}^{\mathcal{C}}$ a un état initial q'_0). Nous devons maintenant prouver que $\mathcal{M}^{\mathcal{C}}$ accepte $\mathcal{L}^{\mathcal{C}}$.

$\mathcal{M}^{\mathcal{C}}$ accepte $\mathcal{L}^{\mathcal{C}}$. Si $\lambda \in \mathcal{L}^{\mathcal{C}}$, alors $\lambda \in \mathcal{L}$ (comme $\mathcal{L}^{\mathcal{C}} \subseteq \mathcal{L}$) et donc λ est accepté par M, i.e. q_0 est un état final de M et par construction q'_0 est aussi final dans $\mathcal{M}^{\mathcal{C}}$. Ainsi λ est aussi accepté par $\mathcal{M}^{\mathcal{C}}$.

Soit $w = a_1 \dots a_k \neq \lambda$ un mot sur l'alphabet Σ et S son ensemble de symboles On note R_i l'ensemble de symboles du préfixe de longueur i de w : $a_1 \dots a_i$.

Si $w \in \mathcal{L}^{\mathcal{C}}$ alors, par définition, S est un stable dans le graphe des conflits. Ainsi, par contruction, le traitement de w par $\mathcal{M}^{\mathcal{C}}$ se termine dans un état de la copie M_S de $\mathcal{M}^{\mathcal{C}}$: en effet, après le traitement de a_1 l'état courant p_1 est un de $M_{R_1} = M_{\{a_1\}}$, ..., après le traitement de a_i l'état courant p_i est un de M_{R_i}, ..., après le traitement du dernier symbole a_k de w, l'état courant p_k est un de$M_{R_k} = M_S$. Comme il n'y a pas de symboles en conflit dans w, l'automate ne va jamais dans l'état TRASH. Notons q_i l'état de M correspondant à l'état p_i de la copie M_{R_i}. Si w est traité par M alors $\delta^*(q_0, w) \in F$ car $w \in \mathcal{L}^{\mathcal{C}} \subseteq \mathcal{L}$. Durant le traitement de w, la séquence d'états est $q_1, \ldots, q_k$, ainsi q_k est final dans M et, par construction, sa copie p_k dans M_S est finale dans $\mathcal{M}^{\mathcal{C}}$. Le mot w est donc accepté par $\mathcal{M}^{\mathcal{C}}$.

Considérons maintenant la situation inverse. Supposons que w soit accepté par $\mathcal{M}^{\mathcal{C}}$. Cela signifie que S est un stable du graphe des conflits (sinon $\delta^{\mathcal{C}*}(q'_0, w) =$ TRASH serait non final). Ainsi, par construction, l'état final $\delta^{\mathcal{C}*}(q'_0, w)$ est dans M_S. En utilisant les mêmes notations que dans le précédent paragraphe et par construction de $\mathcal{M}^{\mathcal{C}}$, si w est traité par M, la séquence d'états du traitement est $q_1, \ldots, q_k$. Comme $p_k = \delta^{\mathcal{C}*}(q'_0, w)$ est final et est la copie de q_k dans M_S, q_k est également final dans M, ainsi w est accepté par M et donc $w \in \mathcal{L}$. De plus, comme il n'y a pas de symboles en conflits, $w \in \mathcal{L}^{\mathcal{C}}$.

Taille de $\mathcal{M}^{\mathcal{C}}$. Analysons la taille de $\mathcal{M}^{\mathcal{C}}$ par rapport à la taille de l'instance (M et $G(\Sigma)$). $\mathcal{M}^{\mathcal{C}}$ est composé de deux nouveaux états q'_0, TRASH et $|\mathcal{STAB}|$ copies de M, où certaines transitions sont redirigées entre les copies. La taille est alors $O(|\mathcal{STAB}|.|M|)$. Comme chaque élément de $\mathcal{STAB}$ est un sous ensemble de Σ, $|\mathcal{STAB}| \leq 2^{|\Sigma|}$. La taille de $\mathcal{M}^{\mathcal{C}}$ est alors $O(|M|.2^{|\Sigma|})$. Cette construction générique mène donc à un automate de taille exponentielle.

Notons que cette construction peut facilement être améliorée pour obtenir un automate plus petit dans certaines situations (mais toujours non polynomial en général). Pour cela, on peut simplement considérer dans les stables l'ensemble de symboles de Σ qui sont dans au moins un conflit (les autres sont "compatibles" avec tous les symboles de Σ) pour faire moins de copies de M.

5.3 La taille de $\mathcal{M}^{\mathcal{C}}$ doit parfois être exponentielle

Dans la section 5.2 nous avons prouvé que $\mathcal{L}^{\mathcal{C}}$ est régulier en construisant un AFD le reconnaissant, mais cet automate a en général une taille exponentielle. Dans cette section, nous prouvons que cet inconvénient ne peut

pas être évité. Pour certaines instances, i.e. $\mathcal{L}$ et $G(\Sigma)$, le plus petit AFD $\mathcal{M}^{\mathcal{C}}$ pour $\mathcal{L}^{\mathcal{C}}$ a nécessairement une taille exponentielle.

Soit Σ_k un alphabet sur $2k$ symboles représentés ici par des entiers $\Sigma_k = \{1, 2, \ldots, 2k\}$. Le graphe des conflits $G(\Sigma_k)$ que nous considérons ici a un ensemble de sommets Σ_k et les conflits constituent un *couplage parfait* : il existe une arête entre le symbole $2i-1$ et $2i$ pour tout $i = 1, \ldots, k$, i.e. un conflit entre 1 et 2, un conflit entre 3 et 4, ..., un conflit entre $2k-1$ et $2k$. Le langage initial considéré ici, $\mathcal{L}_k$, sera l'ensemble de *tous les mots possibles* sur l'alphabet Σ_k. $\mathcal{L}_k$ est régulier car il est accepté par un AFD simple M_k composé d'un unique état q_0 (initial et final) avec : $\delta(q_0, c) = q_0$ pour tout $c \in \Sigma_k$. On note $\mathcal{L}^{\mathcal{C}}$ le langage résultant de $\mathcal{L}_k$ et $G(\Sigma_k)$.

On décrit un AFD $\mathcal{M}^{\mathcal{C}}$ pour $\mathcal{L}^{\mathcal{C}}$ dont la fonction de transition est $\delta^{\mathcal{C}}$. Créons un état q_S pour chaque stable $S \in \mathcal{STAB}$; cet état q_S est final. On crée un état initial q_0 qui est aussi final (car $\lambda \in \mathcal{L}_k$). Ceci veut dire que tous les états de $\mathcal{M}^{\mathcal{C}}$ sont finaux, à l' exception du nouvel état TRASH avec : $\delta^{\mathcal{C}}(\text{TRASH}, c) = \text{TRASH}$ pour tout $c \in \Sigma_k$.

La construction des transitions est naturelle et suit l'idée de la construction générique de la section 5.2. Pour chaque état q_S :

- pour tout $c \in S$, $\delta^{\mathcal{C}}(q_S, c) = q_S$.
- pour tout $c \notin S$ et c en conflit avec un symbole de S, $\delta^{\mathcal{C}}(q_S, c) =$ TRASH.
- pour tout $c \notin S$ et c en conflit avec aucun symbole de S, $\delta^{\mathcal{C}}(q_S, c) = q_{S\cup\{c\}}$ ($S \cup \{c\}$ est un stable).

Une illustration de $\mathcal{M}^{\mathcal{C}}$ pour $k = 2$ est montrée figure 5.1.

Il n'est pas difficile de voir que $\mathcal{M}^{\mathcal{C}}$ est un AFD acceptant $\mathcal{L}^{\mathcal{C}}$. En effet, si un mot w est dans $\mathcal{L}^{\mathcal{C}}$ alors il ne contient aucune paire de symboles en conflit et son traitement par $\mathcal{M}^{\mathcal{C}}$ ne finira pas dans TRASH, et w sera accepté. Inversement, si w est accepté par $\mathcal{M}^{\mathcal{C}}$, cela veut dire que son traitement s'arrête dans n'importe quel état, excepté TRASH, et donc ne contient aucune paire de symboles en conflit, et est donc dans $\mathcal{L}^{\mathcal{C}}$. De plus $\mathcal{M}^{\mathcal{C}}$ contient plus de $|\mathcal{STAB}|$ états, et $|\mathcal{STAB}|$ contient au moins les 2^k ensembles consistant à sélectionner une extrémité de chacune des k arêtes de $G(\Sigma_k)$. Ainsi, la taille de $\mathcal{M}^{\mathcal{C}}$ n'est pas polynomiale.

On pourrait penser qu'un automate plus petit, et potentiellement de taille polynomiale acceptant $\mathcal{L}^{\mathcal{C}}$ puisse exister. Nous montrons maintenant que ce n'est pas le cas, en prouvant que $\mathcal{M}^{\mathcal{C}}$ est minimal, c'est à dire que c'est un AFD de taille minimale acceptant $\mathcal{L}^{\mathcal{C}}$.

Pour cela, nous devons rappeler quelques notations. Deux états p et q sont *distinguishable* dans $\mathcal{M}^{\mathcal{C}}$ si il existe un mot w sur Σ_k tel que $\delta^{\mathcal{C}*}(p, w)$ est final et $\delta^{\mathcal{C}*}(q, w)$ est *non* final, ou inversement. D'après des résultats

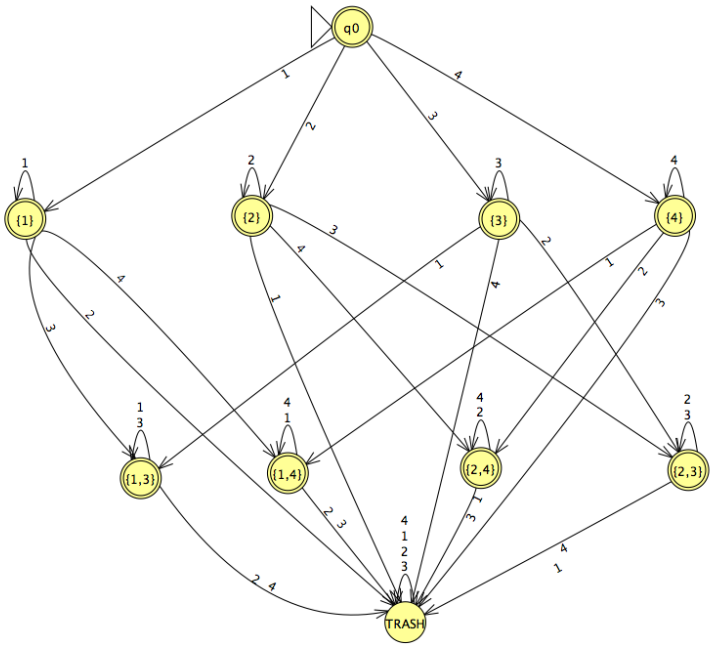


FIGURE 5.1 – $\mathcal{M}^{\mathcal{C}}$ pour $k = 2$

classiques (voir par exemple ce livre [Linon]), $\mathcal{M}^{\mathcal{C}}$ est minimal si chaque paire d'états est distinguishable. Dans ce qui suit, on notera $F^{\mathcal{C}}$ l'ensemble des états finaux de $\mathcal{M}^{\mathcal{C}}$, c'est-à-dire tous les états à l'exception de TRASH.

Tout d'abord, TRASH est distinguishable de n'importe quel autre état, car tous les autres états sont finaux. En effet, $\delta^{\mathcal{C}}(q_0, 1) = q_{\{1\}} \in F^{\mathcal{C}}$ and $\delta^{\mathcal{C}}(\text{TRASH}, 1) = \text{TRASH} \notin F^{\mathcal{C}}$. Pour n'importe quel autre état q_S, soit c un symbole de S ; nous avons : $\delta^{\mathcal{C}}(q_S, c) = q_S \in F^{\mathcal{C}}$ et $\delta^{\mathcal{C}}(\text{TRASH}, c) = \text{TRASH} \notin F^{\mathcal{C}}$.

Considérons maintenant le cas de q_0. Soit q_S n'importe quel autre état. Soit c un symbole en conflit avec un symbole de S. Alors $\delta^{\mathcal{C}}(q_0, c) = q_{\{c\}} \in F^{\mathcal{C}}$ et $\delta^{\mathcal{C}}(q_S, c) = \text{TRASH} \notin F^{\mathcal{C}}$.

Soit q_S un état quelconque (différent de q_0 et TRASH). Les résultats précédents ont montré que q_S est distinguishable de q_0 et TRASH. Soit q_R n'importe quel autre état. Premier cas : si S contient un symbole c et R contient un symbole en conflit avec c alors $\delta^{\mathcal{C}}(q_S, c) = q_S \in F^{\mathcal{C}}$ et $\delta^{\mathcal{C}}(q_R, c) = \text{TRASH} \notin F^{\mathcal{C}}$. Second cas : les symboles de S ne sont pas en conflit avec ceux de R. Comme $S \neq R$, supposons que $|R| < |S|$ (la preuve pour le cas $|S| < |R|$ est similaire) et soit c un symbole de S n'appartenant pas à R. Soit d l'unique symbole en conflit avec c. Cela signifie que $d \notin R$, ainsi $R \cup \{d\}$ est un stable et $\delta^{\mathcal{C}}(q_R, d) = q_{R\cup\{d\}} \in F^{\mathcal{C}}$ et $\delta^{\mathcal{C}}(q_S, d) = \text{TRASH} \notin F^{\mathcal{C}}$.

Nous avons prouvé que $\mathcal{M}^{\mathcal{C}}$ ne peut pas être réduit car chaque paire de sommets est distinguishable. Ainsi, il n'y a pas de plus petit AFD acceptant $\mathcal{L}^{\mathcal{C}}$.

5.4 Les langages reconnus par des AFD avec conflits sont réguliers.

Dans cette section, nous nous intéressons à un autre problème. Une instance est un AFD M avec un ensemble fini d'états Q, acceptant un langage régulier $\mathcal{L}$, et un graphe $G(Q)$ dont les sommets sont les états de Q. Une arête pq entre p et q dénote un *conflict* entre les états p et q. Soit $w = a_1 \dots a_k$ un mot. Soit $p_1, \dots, p_k$ la séquence d'états rencontrés lors du traitement de w par M : $p_i = \delta^*(q_0, a_1 \dots a_i)$ $(i = 1, \dots, k)$. Cette séquence $p_1, \dots, p_k$ est appelée la *séquence de traitement* de w et est notée $\mathrm{SEQ}(w)$.

Le langage associé à M et $G(Q)$, noté $\mathcal{L}^{\mathcal{Q}}$, est l'ensemble de mots de $\mathcal{L}$ pour lesquels $\mathrm{SEQ}(w)$ ne contient pas de paire d'états en conflits :

$$\mathcal{L}^{\mathcal{Q}} = \{w = a_1 \dots a_k : \mathrm{SEQ}(w) = p_1 \dots p_k \text{ and } p_i p_j \notin G(Q)(\forall i \neq j)\}$$

La question est ici : est-ce que $\mathcal{L}^{\mathcal{Q}}$ est un langage régulier ? Nous donnons une réponse positive dans cette section. Pour cela, nous construisons un nouvel AFD $\mathcal{M}^{\mathcal{Q}}$ acceptant $\mathcal{L}^{\mathcal{Q}}$. Cette construction est similaire à celle de la section 5.2.

Construction de $\mathcal{M}^{\mathcal{Q}}$. On note $\mathcal{STAB}$ la famille d'ensembles de sommets qui ne sont pas en conflit : $\mathcal{STAB} = \{S \subseteq Q : pq \notin G(Q)(\forall q, q \in S)\}$, c'est-à-dire l'ensemble des *stables* de $G(Q)$. Pour chaque $S \in \mathcal{STAB}$, on construit une copie notée M_S de M. Chaque copie d'un état final de M reste finale dans M_S. On crée un nouvel état initial noté q_0' qui est l'unique état initial de $\mathcal{M}^{\mathcal{Q}}$ (les copies de q_0 dans les M_S ne sont plus initiales). Cet état q_0' est final si et seulement si q_0 est final dans M. Créons un nouvel état non final TRASH. La fonction de transition $\delta^{\mathcal{Q}}$ est définie pour chaque état q de chaque copie M_S ; l'état correspondant à q dans M est noté p (ainsi q est la copie de p dans M_S).

- Pour chaque symbole $c \in \Sigma$ tel que $\delta(p, c) \in S$, $\delta^{\mathcal{Q}}(q, c) = r$ où r est la copie de $\delta(p, c)$ dans M_S.
- Pour chaque symbole $c \in \Sigma$ tel que $\delta(p, c) \notin S$ et $\delta(p, c)$ pas en conflit avec un autre état de S, $\delta^{\mathcal{Q}}(q, c) = r$ avec r la copie de $\delta(p, c)$ dans $M_{S \cup \{\delta(p,c)\}}$ (comme $S \cup \{\delta(p, c)\}$ est un stable).
- Pour chaque symbole $c \in \Sigma$ tel que $\delta(p, c) \notin S$ et $\delta(p, c)$ est en conflit avec un état de S, $\delta^{\mathcal{Q}}(q, c) = \mathrm{TRASH}$.

Pour TRASH, les transitions sont : $\delta^{\mathcal{Q}}(\text{TRASH}, c) = \text{TRASH}$ pour chaque $c \in \Sigma$. Pour q'_0 les transitions sont : $\delta^{\mathcal{Q}}(q'_0, c) = r$ où r est une copie de l'état $p = \delta(q_0, c)$ dans $M_{\{p\}}$. L'automate $\mathcal{M}^{\mathcal{Q}}$ est un AFD.

$\mathcal{M}^{\mathcal{Q}}$ accepte $\mathcal{L}^{\mathcal{Q}}$. Soit $w = a_1 \dots a_k$ un mot non vide et $\text{SEQ}(w) = p_1, \dots, p_k$ sa séquence de traitement dans M. Soit $R_i = \{p_1, \dots, p_i\}$ l'ensemble des i premiers états de $\text{SEQ}(w)$. Soit $q_1, \dots, q_k$ la séquence de traitement de w dans $\mathcal{M}^{\mathcal{Q}}$. Par construction, q_i est soit la copie de l'état p_i dans M_{R_i}, soit TRASH.

Si w est accepté par $\mathcal{M}^{\mathcal{Q}}$ alors montrons que $w \in \mathcal{L}^{\mathcal{Q}}$. Comme w est accepté par $\mathcal{M}^{\mathcal{Q}}$, pour tout i, $q_i \neq \text{TRASH}$ et q_k est final dans $\mathcal{M}^{\mathcal{Q}}$. Par construction, l'état q_i est la copie de p_i dans M_{R_i}. Ainsi, p_k est final dans M, et w est accepté par M (et est dans $\mathcal{L}$) et $\text{SEQ}(w)$ ne contient pas de paire d'états en conflits. Ceci prouve que $w \in \mathcal{L}^{\mathcal{Q}}$.

Si $w \in \mathcal{L}^{\mathcal{Q}}$, on doit montrer que w est accepté dans $\mathcal{M}^{\mathcal{Q}}$. Comme $w \in \mathcal{L}^{\mathcal{Q}} \subseteq \mathcal{L}$, p_k est final (dans M) par construction, q_k est final dans $\mathcal{M}^{\mathcal{Q}}$.

Notons pour finir le cas particulier du mot vide $w = \lambda$: par construction de l'état initial q'_0, w est accepté par M si et seulement si w est accepté par $\mathcal{M}^{\mathcal{Q}}$.

5.5 Conclusion du chapitre

Les problèmes de graphe des sections précédentes voyaient leur complexité exploser sous l'effet de la contrainte *sans conflit* : le problème de l'existence d'une solution, auparavant trivial, devient NP-Complet. L'ajout de conflits à un langage régulier (selon les deux définitions proposées dans ce chapitre) ne change pas sa classe : le langage reste régulier. Cependant, l'explosion combinatoire provoquée par les conflits se retrouve dans la description du langage : la taille de l'automate minimal reconnaissant un langage sans conflit peut être exponentielle (en la taille de l'automate reconnaissant le langage "classique").

Troisième partie

Autres types de contraintes additionnelles

Introduction

Dans les parties précédentes, nous avons étudié les problèmes avec *conflits*, une contrainte additionnelle modélisant une incompatibilité entre des éléments d'une solution. Ces contraintes étaient vues comme un second graphe sur le même ensemble de sommets.

Nous allons maintenant nous intéresser à d'autres contraintes plausibles pouvant s'appliquer à des problèmes de graphes. Deux types de contraintes seront considérés : la connexité *dans un second graphe*, dans le chapitre 6 et les *obligations* dans le chapitre 7. Ces contraintes seront introduites dans leurs chapitres respectifs.

Dans un troisième chapitre indépendant, nous aborderons le problème du *firefighter* avec contraintes sur le déplacement des pompiers.

Chapitre 6

Vertex cover, connexe dans un autre graphe

6.1 Introduction et cas général

Dans les parties précédentes, nous étudiions les problèmes avec conflits. Une instance était une paire de graphes (G, C) sur un même ensemble de sommets V, et une solution devait être *sans conflit dans* C, c'est à dire être un indépendant dans C. Nous souhaitons maintenant étudier un autre type de contrainte additionnelle : la connexité. Une instance sera donc une paire de graphe (G_1, G_2), et toute solution devra être connexe dans G_2. Nous appellerons G_2 le graphe de connexité et G_1 sera le graphe support. Nous étudions dans ce chapitre l'effet de cette contrainte additionnelle sur le problème du vertex cover.

Le vertex cover modélise les problèmes de surveillance des liens d'un réseau. Le problème du vertex cover connexe capture une nécessité supplémentaire : le système de surveillance mis en place doit être connecté. Il est cependant possible et naturel que les liens à surveiller (à couvrir par le vertex cover) soient différents des liens permettant de connecter les systèmes de surveillance entre eux (pour la connexité). Aussi, il nous parait intéressant d'étudier le problème du VC-C qui modélise ces situations.

Soit (G_1, G_2) une paire de graphes avec $G_1 = (V, E_1)$ et $G_2 = (V, E_2)$. On appelle VC-C un sous-ensemble $S \subseteq V$ tel que S est un vertex cover de G_1 et S est connexe dans G_2.
Remarque 7. Étant donné $S \subseteq V$ un sous-ensemble de sommets, il est polynomial de vérifier si S est un VC-C de (G_1, G_2). Déterminer s'il existe une solution est donc polynomial : pour chaque composante connexe S de G_2, déterminer si S est un VC-C de G_1. Si oui, S est une solution. Si au-

cune composante connexe de G_2 n'est un VC-C de G_1, alors il n'y a pas de solution.

De plus, si $G_1 = G_2$, le problème du VC-C est équivalent au problème du vertex cover connexe. Il est donc NP-complet de trouver un VC-C de taille minimale.

Notre première interrogation était de savoir si il était possible d'approcher le VC-C de taille minimale avec un rapport constant. Pour cela, nous nous sommes intéressés au problème du *Group Steiner Tree* qui s'en rapproche.

Une instance du Group Steiner Tree est un graphe $G = (V, E)$ pondéré sur les arêtes et $C = C_1, ..., C_k$ une famille de sous-ensembles de V, chacun appelé un *groupe*. Une solution S est un sous-arbre de G de V tel que pour tout i, $S \cap C_i \neq \emptyset$. Une solution optimale est une solution de poids minimal.
Théorème 39. *S'il existe un algorithme d'approximation de rapport kp pour le* Group Steiner Tree *où p est la taille maximale d'un groupe, alors il existe un algorithme d'approximation de rapport* $2k$ *pour le problème du VC-C.*

Démonstration. Soit (G_1, G_2) une instance du VC-C ayant au moins une solution. Construisons (G, C) une instance du Group Steiner Tree. Posons $G = G_2$ et pour toute arête ab de G_1, on a $\{a, b\} \in C$. Toutes les arêtes de G ont un poids unitaire.

Soit S une solution de poids s du problème du Group Steiner Tree sur l'instance (G, C). Construisons S' une solution du VC-C. Si ab est une arête de S, alors $a \in S'$ et $b \in S'$. S est de poids s, les poids étant unitaires, S a donc s arêtes et $s + 1$ sommets (car S est un arbre). Ainsi, S' est de taille $s + 1$. S est un sous-arbre de $G = G_2$, S' est donc connexe dans G_2. De plus, un sommet de chaque C_i est contenu dans S, S' contient donc un sommet de chaque arête de G_1, et est donc un VC-C de (G_1, G_2).

Soit maintenant S' une solution du VC-C de taille s'. Construisons S une solution du Group Steiner Tree. S' est connexe dans G_2, on construit donc en temps polynomial un sous-arbre S de G_2 de taille s' couvrant S'. Comme $G = G_2$, S est bien un sous-arbre de G. De plus, comme S' est un vertex cover de G_1, S contient bien un élément de chaque C_i et est une solution du Group Steiner Tree. De plus, S a $s' - 1$ arêtes et est donc de poids $s' - 1$.

Ainsi, à toute solution de taille s' du VC-C correspond une solution de poids $s' - 1$ du Group Steiner Tree et réciproquement. Notons OPT_{VC-C} et OPT_{ST} les valeurs optimales respectives d'une instance du VC-C et du Group Steiner Tree. On a donc $OPT_{VC-C} = OPT_{ST} + 1$.

On peut ainsi trouver une $2k$-approximation du VC-C optimal comme ceci :
— Construire l'instance du Group Steiner Tree correspondante.

Relation entre G_1 et G_2	Rapport	Complexité	Ref.
$G_1 \subseteq G_2$, G_1 connexe	2	Polynomial	Thm 40
$G_1 \subseteq G_2$	5	Polynomial	Thm 41
$G_2 \subseteq G_1$	3	Polynomial	Thm 42
$G_1 \cap G_2$ est connexe	3	Polynomial	Thm 44
$\|E(G_2) - E(G_1)\| = f$	3	Poly$*2^f$	Thm 43

TABLE 6.1 – Rapports d'approximations pour le VC-C dans différents types d'instances

— Calculer un Group Steiner Tree S de poids $s \leq 2kOPT_{ST}$ (en effet, la taille maximale d'un groupe est ici 2).
— En déduire une solution S' du VC-C de taille $s+1 \leq 2kOPT_{ST}+1 \leq 2kOPT_{VC-C}$.

□

Dans un rapport de recherche [Sla97], Petr Slavik propose une méthode de résolution approchée pour le Group Steiner Tree. L'algorithme proposé permet d'obtenir une p-approximation d'un Group Steiner Tree optimal en temps polynomial, où p est la taille maximale d'un groupe. Toutefois, ce résultat n'a jamais été publié, puis apparaît sous une forme dégradée dans sa thèse de doctorat [Sla98] dans laquelle il propose un rapport d'approximation de $2p$. Ce nouveau résultat n'a (à notre connaissance) jamais été publié non plus.

Si ses résultats sont exacts, d'après le théorème 39, il existe un algorithme 4-approché pour le problème du VC-C. Cependant, cet algorithme utilise la programmation linéaire et un nombre potentiellement exponentiel de contraintes : il est donc assez lourd à mettre en place.

Nous avons donc cherché des algorithmes ad hoc pour tenter d'obtenir de meilleurs ratios d'approximation. Nous y sommes parvenus pour certaines classes d'instances, caractérisées par la relation d'inclusion entre G_1 et G_2. Ces résultats sont rassemblés dans la table 6.1 et sont développés dans les sections suivantes.

6.2 Si G_1 est inclus dans G_2

Dans cette section, le graphe support est inclus dans le graphe de connexité. Nous étudions tout d'abord le cas particulier où G_1 est connexe avant de nous intéresser au cas général.

6.2.1 Si G_1 est connexe

Théorème 40. *Soit (G_1, G_2) une instance du VC-C où $G_1 \subseteq G_2$ et G_1 est connexe. Il est alors possible de calculer une 2-approximation du VC-C optimal de (G_1, G_2) en temps polynomial.*

Démonstration. Dans le cas où G_1 est inclus dans G_2, alors tout vertex cover connexe de G_1 est un VC-C de (G_1, G_2). Or, comme G_1 est connexe, il existe toujours un vertex cover connexe de G_1. Posons OPT_{VC} la valeur d'une solution optimale du vertex cover dans G_1 et OPT_{VC-C} la valeur d'une solution optimale du VC-C dans (G_1, G_2). Comme tout VC-C de (G_1, G_2) est un vertex cover de G_1, on a $OPT_{VC} \leq OPT_{VC-C}$. Construisons grâce à l'algorithme de Savage [Sav82] un vertex cover S 2-approché de G_1, ayant la propriété d'être connexe.

On a :

1. $OPT_{VC} \leq OPT_{VC-C}$
2. S est un vertex cover de G_1 d'après l'algorithme de Savage
3. $|S| \leq 2OPT_{VC}$ d'après l'algorithme de Savage
4. S est connexe dans G_1 d'après l'algorithme de Savage
5. S est connexe dans G_2 d'après 4. et car $G_1 \subseteq G_2$
6. S est un VC-C de (G_1, G_2) d'après 2. et 5.

S est donc bien un VC-C 2-approché, d'après 1. et 6. □

6.2.2 Cas général

Nous relâchons maintenant l'hypothèse que G_1 est connexe. Nous ne pouvons donc plus supposer que G_1 a toujours un vertex cover connexe.

Théorème 41. *Si (G_1, G_2) est une instance du VC-C avec $G_1 \subseteq G_2$, alors l'algorithme 6 retourne un VC-C 5-approché de (G_1, G_2) s'il existe, $faux$ sinon.*

Démonstration. Supposons que l'algorithme retourne $faux$. Alors, il existe deux composantes connexes non triviales de G_1 non connectées par G_2 : il n'existe donc pas de VC-C de (G_1, G_2).

Soit S un sous-ensemble de sommets retourné par l'algorithme. Un exemple de résolution est montré à la figure 6.1. Montrons que S est un VC-C de (G_1, G_2). Par construction, S_1 est un vertex cover de G_1. Montrons maintenant que S est connexe dans G_2.

Par construction, St est connexe dans G_3. Soient A et B deux composantes connexes voisines de St et montrons que $X = S \cap (A \cup B)$ est

Algorithm 6: 5-approximation du VC-C si $G_1 \subseteq G_2$

Data: $(G_1 = (V, E_1), G_2 = (V, E_2))$ une instance du VC-C où $G_1 \subseteq G_2$
Result: S un VC-C s'il existe, $faux$ sinon
$S_1 = \emptyset$
forall *C une composante connexe de G_1* **do**
  $S_1 = S_1 \cup CVC(C)$ où $CVC(C)$ désigne un CVC 2-approché de C dans G_1.
Construire $G_3 = (V_3, E_3)$ avec V_3 l'ensemble des composantes connexes de G_1 et pour tout $A, B \in V_3$ $AB \in E_3$ si et seulement si il existe une arête de G_2 entre un sommet de A et un sommet de B.
Construire M l'ensemble des sommets de G_3 représentant des composantes connexes non triviales (non réduites à un seul sommet).
Construire St un arbre de Steiner 2-approché de (G_3, M)
if *St n'existe pas* **then**
  return $faux$
$S_2 = \emptyset$
forall $AB \in St$ **do**
  Choisir arbitrairement a et b des sommets de A et B tel que $ab \in E_2$
  $S_2 = S_2 \cup \{a\} \cup \{b\}$
return $S = S_1 \cup S_2$

connexe dans G_2. En utilisant les notations de l'algorithme, $X = CVC(A) \cup CVC(B) \cup \{a\} \cup \{b\}$. Par construction, a et b sont voisins et $CVC(A), CVC(B)$ sont connexes (ou vides).

Supposons que $CVC(A) \cup \{a\} \cup \{b\}$ ne soit pas connexe. Alors $a \notin CVC(A)$ et a n'est pas voisin de $CVC(A)$. Ceci implique que soit $CVC(A) \cup \{a\}$ n'est pas connexe dans G_2, soit il existe un chemin de longueur ≥ 2 entre a et $CVC(A)$ ne contenant pas de sommet de $CVC(A)$. Ces deux cas sont des contradictions, ainsi $CVC(A) \cup \{a\} \cup \{b\}$ est connexe. En utilisant le même argument pour b, on obtient que X est connexe. Ceci étant vrai pour toute paire de composantes connexes voisines de St, on obtient que S est connexe.

Étudions maintenant le poids d'une solution retournée par l'algorithme. On note respectivement $vc^*, st_e^*, st_v^*, st_+^*$ le poids d'un vertex cover optimal de G_1, le nombre d'arêtes dans l'arbre de Steiner optimal de (G_3, M), le nombre optimal de sommets, et le nombre minimal de sommets isolés de G_1 à ajouter à un vertex cover de G_1 pour le connecter dans G_2. On note également vc, st_e, st_v, st_+ le poids de S_1, le nombre d'arêtes de St, le nombre de sommets de St, et le nombre de sommets ajoutés pour la connexité de la solution ($|S_2 - S_1|$).

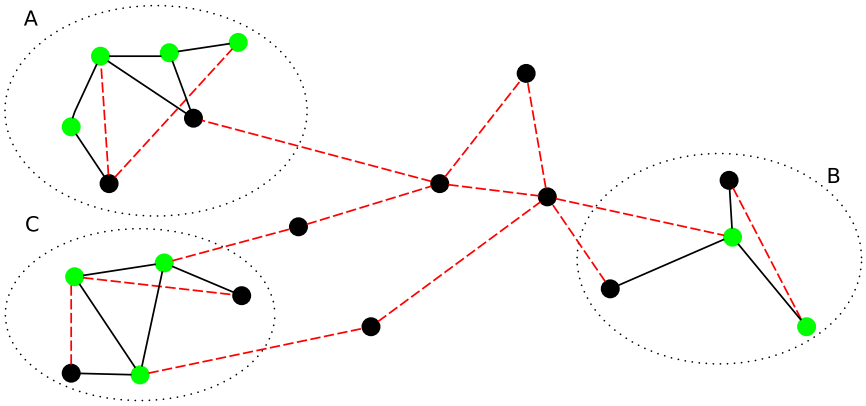
Il est facile de voir que $st_e^* = st_v^* - 1$ et $st_e = st_v - 1$. En utilisant des algorithmes d'approximation standard, on peut garantir $vc \leq 2vc^*$ [Sav82] et $st_e \leq 2st_e^*$ [RZ00], d'où $st_v \leq 2st_v^*$. Par définition, on a $st_+^* = st_v^* - k$. On peut également voir que $st_+ \leq st_v + k$. Ainsi, on dérive $st_+ \leq 2st_+^* + 3k$.

Le poids d'une solution optimale est de $opt \geq vc^* + st_+^*$, et le poids d'une solution retournée par l'algorithme est de $alg = vc + st_+$. En utilisant les inégalités précédentes, on obtient $alg \leq 2vc^* + 2st_+^* + 3k$. Comme $vc^* \geq k$, on a $alg \leq 5opt$. L'algorithme 6 est donc une 5-approximation. □

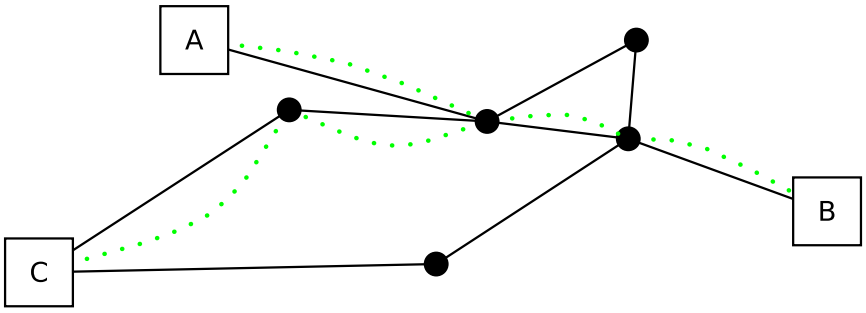
6.3 Si G_2 est inclus dans G_1

Dans cette section, le graphe de connexité est inclus dans le graphe support. Nous définissons un autre problème pour résoudre ce cas particulier. Dans [DLP15], les auteurs introduisent le problème du *Vertex Cover with Forbidden and Required Vertices (VCwFRV)*, une version sur-contrainte du problème du vertex cover, où certains sommets apparaissent obligatoirement dans la solution, et où d'autres ne peuvent pas y apparaître. Plus formellement :

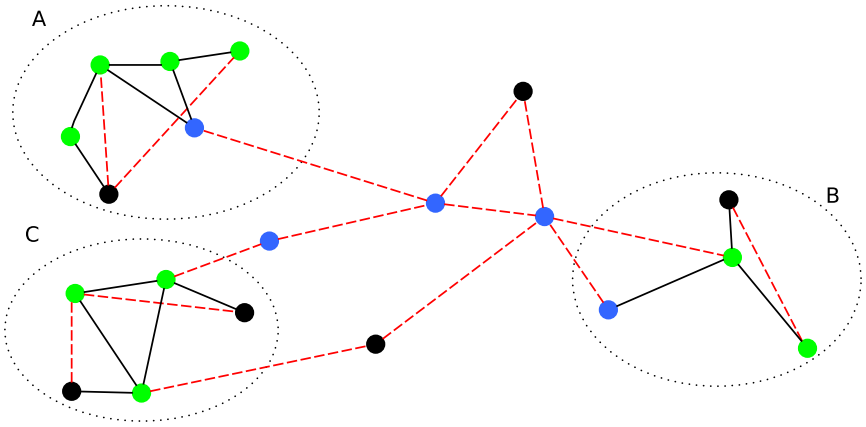
Définition 16 (VCwFRV)**.** Étant donné $(G = (V, E), F, R)$ où F et R sont des sous-ensembles de V, un VCwFRV S est un vertex cover de G (pour chaque arête $e = uv$ de G, $u \in S$ ou $v \in S$) tel que $F \cap S = \emptyset$ et $R \subseteq S$.



(a) Instance initiale : les arêtes noires représentent $G_1 \cap G_2$ et les arêtes en tirets rouges les arêtes exclusives à G_2. Les pointillés représentent les composantes connexes de G_1 et les sommets verts sont les sommets de S_1.



(b) Graphe G_3. Les sommets de M sont carrés, St est représenté en pointillés verts.



(c) Solution du problème : en vert les sommets ajoutés pour la couverture, en bleu les sommets ajoutés pour la connexité.

FIGURE 6.1 – Résolution d'une instance du VC-C par l'algorithme 6.

Les auteurs proposent un algorithme 2-approché pour le problème du VCwFRV.

Théorème 42. *Soit (G_1, G_2) une instance du VC-C où $G_2 \subseteq G_1$. Il est alors possible de calculer une 3-approximation du VC-C optimal de (G_1, G_2) en temps polynomial s'il existe.*

Algorithm 7: 3-approximation du VC-C quand $G_2 \subseteq G_1$

Data: (G_1, G_2) tel que $G_2 \subseteq G_1$, G_2 n'a qu'une seule composante connexe non triviale et celle-ci forme un vertex cover de G_1
Result: Un VC-C 3-approché
$F \leftarrow$ l'ensemble des sommets isolés de G_2
$V_2 \leftarrow$ l'ensemble des sommets de la composante non triviale de
S_1 un VCwFRV 2-approché de $(G_1, F, \emptyset)$ en temps polynomial.
Soit C l'ensemble des $k \leq |S_1|$ composantes connexes de S_1 dans G_2.
Soit C_1 une composante connexe arbitraire de C.
while $C_1 \neq V$ **do**
  Soit C_i la composante connexe la plus proche de C_1 dans G_2.
  Soit P le plus court chemin de G_2 entre C_1 et C_i
  Connecter C_1 et C_i en utilisant P, c'est à dire $C_1 \leftarrow C_1 \cup P \cup C_i$
return C_1

Démonstration. Une telle solution est donnée par l'algorithme 7. Remarquons que les conditions sur l'entrée de l'algorithme correspondent aux conditions d'existence d'une solution (vérifiable polynomialement).

Remarquons que les sommets de F ne peuvent faire partie d'une solution : ils ne pourraient pas être connectés dans G_2. Remarquons que F est aussi un indépendant de G_1.

Notons OPT la solution optimale du VC-C de (G_1, G_2), de taille opt et s_1 la taille de S_1. Notons S^* la solution optimale du VCwFRV de $G_1, F, \emptyset)$ de taille s^*.

Nous avons les inégalités suivantes : $s_1 \leq 2s^*$ et $s^* \leq opt$, d'où $s_1 \leq 2opt$. S_1 est bien un vertex cover de G_1, mais n'est pas encore forcément connexe.

L'algorithme connecte ensuite les k composantes connexes de C.

Nous montrons tout d'abord que connecter les deux composantes C_1 et C_i dans G_2 coûte au plus un sommet. Supposons que cela ne soit pas le cas. Comme V_2 est connexe et que $S_1 \subseteq V_2$, il existe un chemin de G_2 entre C_1 et C_i. Supposons que le plus court chemin entre C_1 et C_i comporte plus d'un sommet hors de C_1 et C_i. Alors ce chemin comporte une arête de G_2 hors de C_1 et C_i. Comme $G_2 \subseteq G_1$, cette arête appartient à G_1 également : soit cette arête n'est pas couverte, soit elle est couverte par une autre composante

connexe C_j, plus proche que C_i. Ces deux cas sont des contradictions, C_1 et C_i peuvent donc être connectées en ajoutant un unique sommet. On a donc ajouté au total $p \leq k$ sommets pour connecter C.

Montrons maintenant qu'il existe un couplage de taille p dans G_1. Notons $P_1, ...P_p$ les p sommets ajoutés pour connecter les p composantes connexes $C_1, ...C_p$ et pour tout $0 < i \leq p$, a_i est le sommet de C_i connecté à P_i dans $G_1 \cap G_2$. Alors, toutes les arêtes $P_i a_i$ appartiennent à G_1, et tous les sommets sont deux à deux disjoints : on a donc construit un couplage de taille p.

C_1 est de taille $s_1 + p$. De plus, G_1 contient un couplage de taille p. Or, la taille d'un couplage est toujours inférieure à la taille d'un vertex cover : on peut en déduire que $opt \geq p$. Ainsi, $|S| \leq 3opt$. □

6.3.1 Si G_2 est presque inclus dans G_1

Nous proposons maintenant un algorithme d'approximation pour les situations où le graphe de connexité est "presque" inclus dans le graphe support, c'est-à-dire que le nombre d'arêtes de $G_2 - G_1$ est faible. Nous proposons dans ce cas un algorithme 3-approché d'une complexité exponentielle en la taille de $E(G_2) - E(G_1)$.

Si A est un ensemble d'arêtes, on note G^A le graphe G dans lequel on a ajouté l'ensemble A d'arêtes.

Nous nous plaçons dans le cas où il existe une solution (détectable polynomialement : une des composantes connexes de G_2 doit induire un vertex cover de G_1).

Théorème 43. *Soient G_1 et G_3 deux graphes connexes avec $G_3 \subseteq G1$, et $F = e_1, ...e_f$ un ensemble d'arêtes n'appartenant pas à G_1. On pose $G_2 = G_3 \cup F$. Soit $I = (G_1, G_2)$ une instance du VC-C. Il est alors possible de calculer un VC-C 3-approché de I en temps $O(2^f Poly(n))$.*

Algorithm 8: 3-approximation pour le VC-C

Data: $(G_1, G_2 = G_3 \cup F)$
Result: Un VC-C 3-approché
forall $R \subseteq F$ **do**
  calculer S^R un VC-C 3 approché de (G_1^R, G_3^R) en temps polynomial en utilisant l'algorithme 7.
return $S = min_R(S_R)$

Démonstration. Une telle approximation est retournée par l'algorithme 8.

Chaque S^R est un vertex cover de G_1 (car il couvre $G_1^R = G_1 \cup R$) et est connexe dans G_2 (car il est connexe dans $G_3^R \subseteq G_2$). L'algorithme retourne donc bien un VC-C de (G_1, G_2)

Notons $S^*(I)$ la solution optimale de I, de valeur $OPT(I)$. Soit $M \subseteq F$ l'ensemble des arêtes de F couvertes par $S^*(I)$

Lors de l'exécution de l'algorithme, R prendra la valeur M. L'algorithme calculera donc une 3-approximation du VC-C (G_1^M, G_3^M). Or, $S^*(I)$ est un VC-C optimal de G_1, G_2^F. Comme $S^*(I)$ est également un VC-C de (G_1^M, G_3^M) et que (G_1^M, G_3^M) correspond à l'instance I avec des contraintes additionnelles (moins de possibilités pour la connexité, plus d'arêtes à couvrir), $S^*(I)$ est un VC-C optimal de (G_1^M, G_3^M).

On a donc $|S^M| \leq 3OPT(I)$. Or l'algorithme retourne S^M ou un VC-C de taille inférieure : on a donc bien une 3-approximation. □

6.4 Si $G_1 \cap G_2$ est connexe

Nous nous intéressons maintenant au cas où l'intersection des deux graphes est connexe.

Théorème 44. *Soit $I = (G_1, G_2)$ une instance du VC-C telle que $G_1 \cap G_2$ est connexe. Il est alors possible de calculer un VC-C 3-approché de I en temps polynomial.*

Algorithm 9: 3-approximation du VC-C quand G_1, G_2 et $G_1 \cap G_2$ sont connexes

Data: (G_1, G_2)
Result: Un VC-C 3-approché.
Calculer S_1 un vertex cover 2-approché de G_1 en temps polynomial.
Soit C l'ensemble des $k \leq |S_1|$ composantes connexes de S_1 dans G_2.
Soit C_1 une composante connexe arbitraire de C.
while $C_1 \neq V$ **do**
  Soit C_i la composante connexe la plus proche de C_1 dans G_2.
  Soit P le plus court chemin de $G_1 \cap G_2$ entre C_1 et C_i.
  Connecter C_1 et C_i en utilisant P, c'est-à-dire $C_1 \leftarrow C_1 \cup P \cup C_i$
return C_1

Démonstration. Une telle approximation est donnée par l'algorithme 9. L'algorithme 9 s'exécute en temps polynomial. Remarquons tout d'abord que l'ensemble C_1 retourné par l'algorithme est un vertex cover de G_1, et connexe dans G_2 : c'est donc un VC-C de (G_1, G_2).

Nous montrons que connecter les deux composantes C_1 et C_i dans G_2 coûte au plus un sommet. Supposons que cela ne soit pas le cas. Comme $G_1 \cap G_2$ est connexe, il existe un chemin de $G_1 \cap G_2$ entre C_1 et C_i. Supposons que le plus court chemin entre C_1 et C_i comporte plus d'un sommet hors de C_1 et C_i. Alors ce chemin comporte une arête hors de C_1 et C_i : soit cette arête n'est pas couverte, soit elle est couverte par une autre composante connexe C_j, plus proche que C_i. Ces deux cas sont des contradictions, C_1 et C_i peuvent donc être connectées en ajoutant un unique sommet. On a donc ajouté $p \leq k$ sommets pour connecter C.

Montrons maintenant qu'il existe un couplage de taille p dans G_1. Notons $P_1, ...P_p$ les p sommets ajoutés pour connecter les p composantes connexes $C_1, ...C_p$ et pour tout $0 < i \leq p$, a_i est le sommet de C_i connecté à P_i dans $G_1 \cap G_2$. Alors, toutes les arêtes $P_i a_i$ appartiennent à G_1, et tous les sommets sont deux à deux disjoints : on a donc construit un couplage de taille p, or la taille d'un couplage est toujours inférieure à celle d'un vertex cover.

Notons OPT_{VC} la valeur d'un vertex cover optimal de G_1 et OPT_{VC-C} la valeur du VC-C optimal de (G_1, G_2). Nous avons : $p \leq OPT_{VC} \leq OPT_{VCC}$ et $|S_1| \leq 2OPT_{VC} \leq 2OPT_{VCC}$. Ainsi, $|C_1| = |S_1| + p \leq 3OPT_{VC-C}$ et l'algorithme retourne bien une 3-approximation du VC-C optimal. □

6.5 Conclusion du chapitre

Si l'algorithme de Slavik est juste, le problème du VC-C est approchable par un rapport constant de 4 en utilisant un algorithme d'approximation générique du Group Steiner Tree. L'utilisation de cet algorithme ne nous permet cependant pas de comprendre le problème plus en détail et d'appréhender la difficulté d'approximation selon les différentes instances et il reste complexe à mettre en place. Les algorithmes ad hoc que nous avons présentés permettent d'obtenir pour certains cas d'inclusion des graphes des résultats légèrement meilleurs (ou légèrement moins bons) que l'algorithme générique. Pour atteindre et confirmer un résultat aussi général (c'est-à-dire obtenir un algorithme d'approximation à rapport constant pour le problème du VC-C), il faut encore résoudre le cas où il n'existe pas de relation d'inclusion entre G_1 et G_2, et que $G_1 \cap G_2$ n'est pas connexe.

La contrainte de connexité dans un autre graphe pourrait être étudiée pour d'autres problèmes d'optimisation de graphes, notamment pour le problème du Dominant, pour lequel il existe déjà une variante connexe (dans le même graphe).

Chapitre 7

Problèmes avec obligations

7.1 Obligations

Les systèmes (de production, de distribution, les réseaux,...) sont composés d'éléments (usines, véhicules, logiciels, nœuds, personnes) et doivent fournir un produit, résultat ou service. Ces éléments sont connectés (pour communiquer, échanger des ressources, etc) et ces liens forment un réseau modélisé par un graphe $G = (V, E)$. Pour fonctionner et compléter leur tâche, ces éléments doivent être organisés : par exemple, un arbre de Steiner permet de diffuser l'information à un ensemble de nœuds donné, un vertex cover surveille les liens de G, et un dominant surveille les éléments de G. Dans certaines situations, des ensembles d'éléments doivent être actifs simultanément. Par exemple, si le traitement d'une tâche implique un outil distribué sur plusieurs nœuds, ou quand les nœuds sont des personnes membres d'équipes : si un membre de l'équipe est mobilisé pour la tâche, alors tous les membres sont mobilisés.

Nous modélisons cette interdépendance entre deux éléments a et b comme suit : si l'élément a est sélectionné alors b est sélectionné également. Nous notons cette dépendance $< a, b >$. Cette relation est *symétrique* : si $< a, b >$, alors $< b, a >$. Cette relation est en quelque sorte opposée aux *conflits* étudiés dans les parties précédents : si deux éléments étaient en conflits, ils ne *pouvaient pas* être sélectionnés simultanément, si deux éléments sont dans une relation d'obligation, alors ils *doivent* être sélectionnés simultanément. Par sa nature, cette relation est transitive : si $< a, b >$ et $< b, c >$, alors nécessairement $< a, c >$. Cette relation est également réflexive, on a $< a, a >$. Ainsi, la relation $< ., . >$ est une relation d'équivalence, et forme donc des *classes d'équivalences*, et donc une *partition* des éléments. Dans ce chapitre, chaque partie / classe d'équivalence sera appelée *obligation*.

Nous ne traitons pas ici un problème pratique spécifique, mais nous donnons un cadre général et étudions les problèmes d'optimisation combinatoire sous-jacents. Dans ce chapitre nous nous intéresserons donc à des problèmes de graphe classiques avec contraintes additionnelles. Soit $G = (V, E)$ un graphe. On appelle *système d'obligations sur les sommets de* G une partition $\Pi_V = V_1, ..., V_k$ de V et *système d'obligation sur les arêtes de* G une partition $\Pi_E = E_1, ..., E_k$ de E. Étant donné G et Π_V (resp. Π_E) un système d'obligation sur les sommets (resp. les arêtes) associé , toute solution S à un problème sur G doit *respecter* les obligations, c'est à dire doit avoir la propriété suivante : si $u \in S$ (resp. $e \in S$) et $u \in V_i$ (resp. $u \in E_i$), alors tous les éléments de V_i (resp. E_i) doivent appartenir à la solution, c'est-à-dire $V_i \subseteq S$ (resp. $E_i \subseteq S$).

Comme mentionné plus haut, les obligations peuvent être utiles pour modéliser des situations dans lesquelles des ensembles d'éléments (capteurs, ordinateurs, logiciels, personnes, etc...) sont interdépendants et où la présence d'un élément implique la présence de tous les autres. D'un point de vue algorithmique, il est clair qu'introduire les contraintes d'obligations dans un problème de graphe $\mathcal{P}$ mène à une généralisation directe de $\mathcal{P}$ (où les obligations sont des singletons). Nous verrons que dans la plupart des cas, les problèmes avec obligations deviennent beaucoup plus difficiles que les problèmes originaux.

7.2 Vertex Cover avec obligations sur les sommets

Soit $G = (V, E)$ un graphe et $\Pi_V = V_1, ..., V_k$ un système d'obligations sur les sommets de G. Un *vertex cover avec obligations* (VCO) S de (G, Π_V) est un sous-ensemble de sommets de G tel que :

- S est un vertex cover de G : chaque arête $e = uv \in E$ est couverte par S ($u \in S$ ou $v \in S$).
- $\forall u \in S$, si $u \in V_i$, alors $V_i \subseteq S$.

Remarquons que pour toute instance $(G = (V, E), \Pi_V)$ il existe au moins un VCO : V.

Un VCO est dit *optimal* et noté VCO_{OPT} s'il est de taille minimale. Construire un VCO_{OPT} est NP-difficile, en effet, dans le cas particulier où chaque partie de Π_V est un singleton, le problème est équivalent à celui du vertex cover classique. Dans ce qui suit nous proposerons un algorithme pour approcher un VCO_{OPT}. Tout d'abord, nous effectuons un pré-traitement sur les instances. Remarquons que si $e = uv \in E$ et u et v sont dans la même partie V_i, alors tout VCO (et donc tout VCO_{OPT}) doit contenir V_i, car l'arête uv doit être couverte par u ou par v, et par conséquent, tout V_i appartient à la solution. Nous incluons donc dans toute solution toutes les parties V_i

de Π_V induisant des arêtes de G. Les sommets de ces parties sont ensuite supprimés de l'instance. Ce pré-traitement peut être effectué en temps polynomial. Supposons que ceci ait été effectué, que G ne contient plus les sommets concernés, et que Π_V ne contient plus les parties concernées.

Une 2-approximation pour le VCO_{OPT} Une instance est maintenant $(G = (V, E), \Pi_V = V_1, ...V_k)$ où chaque V_i est un stable de G.

Algorithm 10: 2-approx VCO_{OPT}

Data: $(G = (V, E), \Pi_V = V_1, ...V_k)$ où chaque V_i est un stable de G
Result: Un VCO 2-approché
Construire un nouveau graphe pondéré $G_C = (V_C, E_C)$ appelé *graphe contracté* :
- Chaque partie V_i de Π_V est associée à un sommet v_i de G_C.
- Le poids de v_i est le nombre de sommets de V_i ($|V_i|$).
- Il existe une arête entre v_i et v_j si et seulement si il existe au moins une arête de G entre un sommet de V_i et un sommet de V_j.

Construire un vertex cover connexe pondéré 2-approché S_C de G_C en temps polynomial.
return $S = \bigcup_{i|v_i \in S_C} V_i$

Théorème 45. *L'algorithme 10 retourne une* 2*-approximation du* VCO_{OPT}.

Démonstration. L'algorithme est polynomial. Il construit un vertex cover de G qui satisfait les obligations.

Notons que pour respecter les obligations, tout VCO est une union de parties de Π_V. Nous construisons une bijection respectant les poids et tailles entre les VCO de (G, Π_V) et les vertex cover de G_C.

- Soit S un VCO quelconque de (G, Π_V). L'ensemble $S_c = \{v_i : V_i \subseteq S\}$ est un vertex cover de G_C de poids $|S|$.
- Réciproquement, soit $S_C = \{v_1, ...v_l\}$ un vertex cover pondéré quelconque de G_C. Dans ce cas, $S = \{V_i : v_i \in S_C\}$ est un VCO de (G, Π_V) dont la taille est égale au poids de S_C

Une 2-approximation d'un vertex cover optimal pondéré de G_C correspond donc à une 2-approximation d'un VCO_{OPT} de (G, Π_V). Ainsi l'algorithme proposé est 2-approché pour le problème du VCO_{OPT}. □

7.3 Vertex Cover Connexe avec obligation sur les sommets

Dans cette section, $G = (V, E)$ est un graphe *connexe* non pondéré. Comme dans la section 7.2, les *obligations* sont une partition $\Pi_V = V_1, \ldots, V_k$ de V. Un *vertex cover connexe avec obligations* (VCCO) S d'une instance (G, Π_V) est un sous-ensemble de sommets tel que :

- S est un vertex cover de G : chaque arête $e = uv \in E$ est couverte par S ($u \in S$ ou $v \in S$).
- S est un ensemble de sommets connectés, c'est-à-dire $G[S]$ est connexe.
- $\forall u \in S$, si $u \in V_i$, alors $V_i \subseteq S$.

Un $VCCO_{OPT}$ est VCCO de taille minimale. Construire un $VCCO_{OPT}$ est un problème NP-complet, en effet, si Π_V est une partition en singleton, le problème est équivalent au problème du vertex cover connexe classique [GJ02]).

Théorème 46. *Tout algorithme α-approché pour le $VCCO_{OPT}$ peut être transformé en algorithme 2α-approché pour le problème du* Set Cover.

Démonstration. Soit (A, X) une instance quelconque du Set Cover. $A = \{a_1, \ldots, a_n\}$ est un ensemble de n éléments et $X = X_1, \ldots, X_k$ est une famille de k sous-ensembles de A ($X_i \subseteq A$) couvrant A : $A = \cup_{i=1}^{k} X_i$. Un set cover optimal est une sous-famille de X, de taille minimale et couvrant A. On note t^* la taille d'une solution optimale de (A, X).

Construisons une instance de notre problème à partir de (A, X). Chaque élément a_i est associé à un sommet, également noté a_i. Chaque sous-ensemble X_i de X est associé à un ensemble V_i de $n + 1$ nouveaux sommets, formant un stable. Chacun des $n + 1$ sommets de l'ensemble V_i est connecté au sommet a_j si et seulement si l'ensemble X_i contient a_j. Un sommet r est créé et connecté à tous les sommets des k ensembles V_i. Le degré de r est donc $k(n + 1)$. On note $G = (V, E)$ le graphe final, qui est biparti.

Les obligations sont les suivantes. Chaque V_i est une obligation contenant exactement $n + 1$ sommets indépendants. Nous ajoutons une obligation V_0 contenant r et les n sommets de A. V_0 est alors un stable de G composé de $n + 1$ sommets. $\Pi_V = V_0, V_1 \ldots, V_k$ est une partition de l'ensemble V des sommets de G et sera le système d'obligations que nous considérerons. Ici, chaque V_i est un stable de $n+1$ sommets de G. L'instance (G, Π_V) peut être construite en temps polynomial. Considérons maintenant la bijection entre les VCCO de (G, Π_V) et les set cover de (A, X).

Soit $S_X = X_{i_1}, \ldots, X_{i_t}$ un set cover quelconque de (A, X) de taille t. Soit S l'ensemble de sommets de G suivant :

$$S = V_0 \cup \bigcup_{j=1}^{t} V_{i_j}$$

S est un vertex cover de G (toutes les arêtes de G sont couvertes par les sommets de V_0), $G[S]$ est connexe (car les sommets de V_{i_j} sont connectés *via* r et chaque a_i est connecté à au moins tous les sommets d'un ensemble V_{i_j} car S_X est une couverture) et satisfait les obligations de Π_V (S est composé de l'union d'obligations de Π_V). La taille de S est : $|S| = n + 1 + t(n+1) = (n+1)(t+1)$.

Soit maintenant S un VCCO de (G, Π_V). Comme S satisfait les obligations, S est composé d'une union d'obligations. Comme $G[S]$ est connecté et que G est biparti, S doit contenir au moins une obligation $V_i, i \geq 1$. Comme S doit contenir r ou un sommet a_i pour assurer la connexité, S doit contenir V_0. Notons $V_0, V_{i_1}, \ldots, V_{i_t}$ la famille d'obligations contenues dans S : $S = V_0 \cup V_{i_1} \cup \ldots \cup V_{i_t}$. Soit $S_X = X_{i_1}, \ldots, X_{i_t}$ la sous-famille de X associée à ce VCCO S. Comme $V_0 \subseteq S$, chaque sommet a_i est connecté aux autres sommets de S via les sommets d'au moins un V_{i_j}. Ainsi, S_X est un set cover de (A, X). Nous avons : $|S_X| = t$ et $|S| = (t+1)(n+1)$.

Cette bijection associe tout set cover de taille t à un VCCO de taille $(t+1)(n+1)$ et réciproquement. Le calcul de cette bijection peut être effectué dans les deux directions en temps polynomial.

Supposons que le $VCCO_{OPT}$ puisse être approché en temps polynomial avec un ratio α. Alors, pour toute instance (A, X) il est possible de construire l'instance associée (G, Π_V) en temps polynomial, puis utiliser cet algorithme d'approximation pour construire un VCCO α-approché S tel que $(t+1)(n+1) = |S| \leq \alpha|S^*|$. Alors, grâce à la bijection, il est possible de construire le set cover associé S_X, de taille t. Ce calcul se fait en temps polynomial. Soit S_X^* un set cover optimal, de taille t^*. Par la bijection, cela correspond à un VCCO de taille $(t^*+1)(n+1)$. Ce VCCO est optimal (sinon il serait possible d'en construire un plus petit avec la bijection). Ainsi,

$$|S| = (t+1)(n+1) \leq \alpha(t^*+1)(n+1)$$

donc, $t + 1 \leq \alpha(t^*+1)$ et $t \leq \alpha t^* + (\alpha - 1) \leq \alpha(t^*+1) \leq 2\alpha t^*$ (car $1 \leq t^*$). L'algorithme décrit ci-dessus est donc 2α-approché pour le problème du set cover. □

Corollaire 2. *Le problème du VCCO ne peut pas être approché avec un rapport meilleur que* $c\log(n)/2$ *sauf si* $P = NP$.

Démonstration. Le théorème 46 montre que le problème du VCCO ne peut pas être approché par un rapport meilleur que $c\log(n)/2$ car le problème du set cover ne peut pas être approché avec un rapport meilleur que $c\log(n)$ pour un certain $c > 0$, sauf si $P = NP$. Voir par exemple [ACG+12]. □

7.4 Arbre couvrant avec obligations sur les arêtes

Dans cette section, une instance est $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ où G est un graphe connexe et Π_E, les obligations, sont une partition de E.

Étant donné une instance $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$, on cherche à déterminer s'il existe un *arbre couvrant de G respectant les obligations* (ACO) $T = (V, E_T)$ qui est un arbre couvrant de G tel que pour chaque $e \in E_T$, si $e \in E_i$ alors toutes les arêtes de E_i doivent appartenir à T.

Théorème 47. *Déterminer si $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ a un ACO est NP-complet, même si :*

- *G est biparti de degré maximum 4 et*
- *chaque E_i induit une étoile à exactement 3 arêtes ($|E_i| = 3$).*

Démonstration. Le problème est dans NP.

Soit $(X = \{x_1, \ldots, x_{3q}\}, Z_1, \ldots, Z_k)$ une instance de X3C (exact cover by 3 sets) où X est un ensemble de $3q$ éléments et chaque Z_i est un sous-ensemble de 3 éléments de X ($Z_i \subseteq X$ et $|Z_i| = 3$) avec la propriété $X = \bigcup_{i=1}^{k} Z_i$ (les ensembles Z_i couvrent X). Le problème X3C consiste à déterminer si une instance contient une *couverture exacte* de X, i.e., s'il existe $Z_{i_1}, \ldots, Z_{i_q}$ une famille de sous-ensembles deux à deux disjoints tels que $X = \bigcup_{j=1}^{q} Z_{i_j}$. Ce problème est NP-complet même si chaque élément x_i est dans au plus 3 sous-ensembles, voir par exemple [GJ02]. Nous utiliserons ici cette restriction.

Construisons le graphe G. Pour chaque élément x_i de X, un sommet est créé, également noté x_i. Pour chaque ensemble Z_i un nouveau sommet est créé, également noté Z_i. Des arêtes sont ajoutées entre chaque sommet Z_i et les 3 sommets représentant les éléments contenus dans Z_i. Créons maintenant un arbre T_r pour connecter les k sommets Z_i qui deviendront les feuilles de T_r. Les Z_i sont connectés deux à deux par de nouveaux sommets. Ces $\lceil k/2 \rceil$ sommets sont ensuite connectés deux à deux par de nouveaux sommets, et ainsi de suite jusqu'à ce qu'il n'y ait plus qu'un sommet noté r, la racine de T_r.

Chaque sommet u de T_r, à l'exception des feuilles, possède un ou deux

fils. Pour chacun de ces sommets, on ajoute un nouveau sommet l_u (ou deux si nécessaire) uniquement connecté à u. Tout ceci forme le graphe final $G = (V, E)$ qui est biparti, et grâce à la restriction des instances de X3C, G est de degré maximum 4. Une illustration de cette construction est donnée à la figure 7.1 : les sommets de la dernière ligne sont les éléments de x, les sommets carrés sont les éléments Z_i, les sommets noirs sont les fils supplémentaires et les sommets colorés sont les nœuds internes de T_r. Les ellipses en pointillés représentent les obligations.

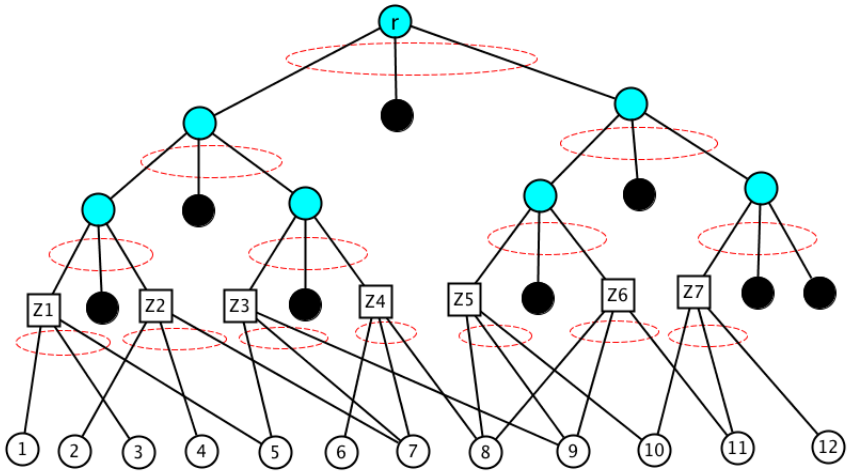


FIGURE 7.1 – Construction de G à partir de l'instance de X3C

Pour chaque sommet Z_i une obligation E_i est créée, regroupant les 3 arêtes connectant Z_i aux 3 sommets représentant les éléments contenus dans Z_i. Pour chaque sommet interne u de T_r, une obligation est créée contenant les trois arêtes reliant u à ses fils. Ces dernières obligations sont appelées *obligations de l'arbre*. Chaque arête est maintenant dans exactement une obligation, et l'ensemble de ces obligations est Π_E, composé d'étoiles à 3 arêtes.

La construction décrite ci-dessus peut être faite en temps polynomial.

Supposons que l'instance X3C ait une solution $Z_{i_1}, \ldots, Z_{i_q}$. Dans ce cas, nous pouvons sélectionner les obligations suivantes : toutes les obligations de l'arbre et toutes les obligations $E_{i_1}, \ldots, E_{i_q}$. Ceci donne un arbre couvrant de G. (chaque sommet x_i est une feuille car il est voisin d'exactement un sommet Z_{i_j} et chaque sommet Z_l est connecté aux autres *via* l'arbre T_r). Cet arbre respecte les obligations de Π_E et est donc un ACO de (G, Π_E).

Réciproquement, supposons que l'instance (G, Π_E) a un ACO noté T. Comme

T respecte les obligations, il doit donc contenir toutes les obligations de l'arbre pour connecter les sommets l_u. T doit également contenir d'autres obligations. Mais chaque sommet x_i est une feuille de T. Sinon, s'il est voisin de 2 sommets, Z_a et Z_b, ceci formerait un cycle avec certaines des arêtes de l'arbre, ce qui est interdit.

Comme T couvre tous les $3q$ sommets x_i il doit contenir exactement q sommets de type Z_i, notés $Z_{i_1}, \ldots, Z_{i_q}$, et leurs 3 arêtes incidentes des obligations associées $E_{i_1}, \ldots, E_{i_q}$. Les ensembles $Z_{i_1}, \ldots, Z_{i_q}$ couvrent X et sont deux à deux disjointes : c'est donc une solution à l'instance de X3C. □

7.5 Graphe couvrant avec obligations sur les arêtes

Dans cette section, $G = (V, E)$ est un graphe pondéré connexe : chaque arête $e \in E$ a un poids $w(e) > 0$. Les obligations forment une partition $\Pi_E = E_1, \ldots, E_k$ de E. L'objectif est d'extraire de G un sous-graphe connexe couvrant V, ayant un poids minimum et respectant les contraintes d'obligations. Un tel objet est appelé $GCCO_{OPT}$ (*Graphe Couvrant de Poids Minimum avec Obligations*). On appelle GCCO un *Graphe Couvrant avec Obligations* (C'est à dire qu'un $GCCO_{OPT}$ est un GCCO de poids minimum).

Remarquons qu'à cause des obligations, un GCCO n'est pas nécessairement un arbre. En effet, si chaque obligation induit un cycle par exemple, aucun arbre couvrant n'est possible. Remarquons également que comme G est connexe, G est lui même un GCCO de (G, Π_E) (le problème a donc toujours une solution) et que si Π_E ne contient que des singletons, le problème est celui de l'arbre couvrant de poids minimal, un problème classique pouvant être résolu en temps polynomial, par exemple par l'algorithme de Prim.

Théorème 48. *Soit $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ une instance où G est un graphe pondéré connexe. Déterminer s'il existe un GCCO de poids au plus $|V| - 1$ est NP-complet même si :*

- *G est biparti,*
- *tous les poids sont unitaires et*
- *chaque obligation induit une étoile à 3 arêtes.*

Démonstration. Tout graphe couvrant contient au moins $n - 1$ arêtes, où $n = |V|$. Ainsi, dans le cas où chaque arête a un poids de 1, il n'existe pas de GCCO de poids strictement inférieur à $n - 1$. Déterminer s'il existe un GCCO de poids au plus $n - 1$ revient donc exactement à déterminer s'il existe un ACO de cette instance, ce qui est NP-complet, même si G est biparti et si les obligations induisent une étoile à 3 arêtes, comme le montre le théorème 47. □

Le théorème 48 nous montre qu'il est en toute généralité NP-complet de trouver un $GCCO_{OPT}$, même dans le cas où les pondérations sont unitaires. Nous allons maintenant montrer que dans le cas de pondérations quelconques, le problème n'est pas approchable par un rapport constant.

Théorème 49. *Tout algorithme α-approché pour le problème du GCCO dans les graphes bipartis où chaque obligation induit une étoile peut être transformé en un algorithme α-approché pour le problème du* set cover minimum.

Démonstration. Soit $(X = x_1, ...x_n, F = F_1, ...F_k)$ une instance du set cover non pondéré. Construisons une instance du GCCO. V contient n sommets $x_1, ...x_n$, k sommets $F_1, ...F_k$ et un sommet r. r est relié à chacun des sommets F_i et chaque sommet F_i est relié aux sommets x_j tels que $x_j \in F_i$. Le graphe ainsi obtenu est biparti. Toutes les arêtes incidentes à r forment une obligation O_0 et ont un poids ϵ/k arbitrairement petit. Pour chaque F_i, toutes les arêtes incidentes n'appartenant pas à O_0 (c'est à dire les arêtes incidentes aux sommets x_j) forment une obligation O_i, et ont un poids de $1/|F_i|$. Ainsi, le poids total de chaque obligation est de 1, sauf O_0 qui est de poids ϵ. Chaque obligation induit bien une étoile. Un exemple de cette construction est montré figure 7.2. Nous allons montrer qu'à chaque solution du set cover correspond une solution de taille équivalente du GCCO, et inversement.

Soit S une solution du set cover de taille t. On construit C l'ensemble d'arêtes suivant : $O_0 \in C$, et pour tout $0 < 1 \leq k$, $O_i \in C$ si et seulement si $F_i \in S$. Les sommets r et F_i sont connectés dans C (par O_0). Chaque élément x_j est couvert par un ensemble F_i du set cover : le sommet correspondant x_j est connecté au sommet F_i par l'obligation O_i, c'est donc bien un GCCO. C contient O_0 et t autres obligations, son poids est donc de $t + \epsilon$.

Soit maintenant C un GCCO. C contient nécessairement O_0 pour connecter r, ainsi qu'un nombre entier t d'autres obligations, et est donc de poids $t+\epsilon$. On construit S une solution du set cover. Pour tout $0 < i \leq k$, $F_i \in S$ si et seulement si $O_i \in C$. Soit x_j un élément de X. Le sommet correspondant est connecté par une arête appartenant à une obligation O_i, ainsi l'ensemble F_i correspondant appartient à S et l'élément est couvert, S est donc un set cover. De plus, S contient t sous-ensembles. □

Corollaire 3. *Le problème du GCCO de poids minimal n'est pas approchable par un rapport constant, même si G est un graphe biparti et que chaque obligation induit une étoile.*

Démonstration. Le théorème 49 montre qu'il n'est pas possible d'approcher le problème du GCCO avec un rapport meilleur que celui du problème du set cover minimum. Or, ce problème ne peut être approché avec un rapport

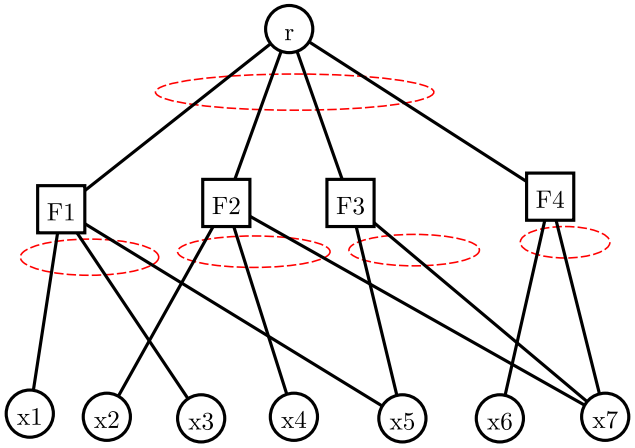


FIGURE 7.2 – Construction de l'instance du GCCO. Les ellipses en pointillés représentent les obligations.

meilleur que $clog(n)$ pour un certain c, sauf si $P = NP$ d'après [ACG+12]. □

Un problème reste ouvert : qu'en est-il de l'approximation du GCCO dans le cas de pondérations unitaires ?

7.6 Couplage avec obligations sur les arêtes

Dans cette section, une instance est $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ où G est un graphe et Π_E, les obligations, est une partition de E.

Un *couplage avec obligations* (CO) M d'une instance (G, Π_E) est un couplage de G (un ensemble d'arêtes deux à deux disjointes) tel que pour toute arête e de M, si $e \in E_i$ alors $E_i \subseteq M$.

Il est polynomial de déterminer si (G, Π_E) contient un CO non vide. En effet, il existe un CO non vide si et seulement si au moins une partie E_i induit un couplage. Nous pouvons donc simplifier une instance (G, Π_E) comme ceci : si une partie E_i de Π_E induit un graphe dans lequel un sommet a plus d'un voisin, alors E_i peut être supprimé de Π_E et les arêtes de E_i peuvent être supprimées de G. Ce pré-traitement peut être effectué en temps polynomial. Nous supposerons maintenant que $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ est une instance où chaque E_i induit un couplage de G et donc contient un CO (potentiellement vide). Un *CO de taille maximale* est noté CMO.

Théorème 50. *Soit* $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ *où chaque* E_i *induit un couplage de* G*. Tout algorithme* α*-approché pour le problème du CMO peut être transformé en un algorithme* α*-approché pour le problème du* stable maximum*.*

Démonstration. Soit $H = (V_H, E_H)$ un graphe quelconque, instance du stable maximum.

Notons $V_H = \{h_1, \ldots, h_n\}$ les n sommets de H. On construit une instance de notre problème à partir de H.

Pour chaque arête h_ih_j de H, on crée un nouveau P_3 (chemin à 3 sommets) associé à cette arête. L'union de ces $|E_H|$ chemins deux à deux disjoints forme un graphe noté Q (qui n'est pas encore le graphe final G). Pour chaque i, $1 \leq i \leq n$, on crée D_i un sous-ensemble d'arêtes de Q comme ceci : pour chaque arête h_ih_j de H, mettre une arête du P_3 associé dans D_i et l'autre dans D_j. Ces n ensembles $D_1, \ldots, D_n$ forment une partition des arêtes de Q et chaque D_i est un couplage. La figure 7.3 donne un exemple de cette construction.

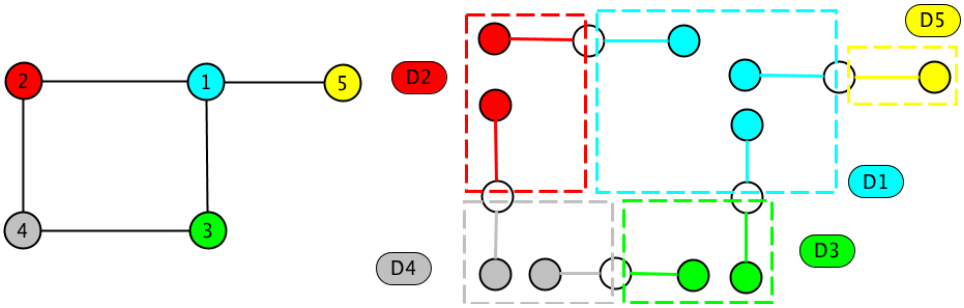


FIGURE 7.3 – Construction de Q à partir de H

Les ensembles D_i peuvent avoir des tailles différentes. Soit D_a celui de taille maximale. Les étapes suivantes consistent à ajouter de nouvelles arêtes indépendantes, entre de nouveaux sommets, à chaque D_i de telle sorte que les n ensembles aient la même taille $|D_a|$.

Notons $G = (V, E)$ le graphe obtenu à partir de Q par l'ajout de ces nouveaux sommets et de ces nouvelles arêtes. On note $\Pi_E = E_1, \ldots, E_n$ les ensembles $D_1, \ldots, D_n$ Nous avons maintenant les propriétés suivantes :

- tous les ensembles E_i ont la même taille notée t,
- $E_1, \ldots, E_n$ est une partition de l'ensemble E des arêtes de G,
- chaque E_i induit un couplage dans G,
- G est un graphe biparti.
- $E_i \cup E_j$ est un couplage de G si et seulement si $h_ih_j \notin E$.

Cette instance (G, Π_E) peut être construite en temps polynomial à partir de l'instance H du stable maximum : Q contient exactement $n + |E_H|$ arêtes. Pour obtenir G, on ajoute au plus $n|E_H|$ arêtes. G contient donc au maximum $O(n^3)$ arêtes. Les parties de Π_E peuvent se construire facilement.

Soit $S = \{h_{i_1}, \ldots, h_{i_q}\}$ un stable quelconque de taille q, dans H. Considérons les obligations associées à S : $E_{i_1}, \ldots, E_{i_q}$. Comme S est un stable, $M_S = \bigcup_{j=1}^{q} E_{i_j}$ est un couplage de G de taille qt.

Réciproquement, soit M un couplage quelconque de G, composé des obligations $E_{i_1}, \ldots, E_{i_q}$. Comme M est un couplage de G de taille qt, $S = \{h_{i_1}, \ldots, h_{i_q}\}$ est un stable de taille q dans H.

Il existe une bijection entre les CO de (G, Π_E) et les stables de H. Les tailles sont les mêmes, à un facteur multiplicatif constant t près. Ainsi, s'il existe un algorithme d'approximation de rapport α pour le CMO, il est possible de construire un algorithme d'approximation pour le problème du stable maximum de même rapport, via les transformations précédemment décrites : transformer H en une instance (G, Π_E), appliquer l'algorithme d'approximation sur cette instance, puis transformer son résultat en un stable de H. La conservation des tailles (à un facteur constant t près) garantit le ratio d'approximation. □

Corollaire 4. *Le problème du CMO ne peut pas être approché avec un rapport meilleur que $|V|^{1/2-\epsilon}$ sauf si $P = NP$.*

Démonstration. Le théorème 50 montre qu'il n'est pas possible d'approcher le problème du CMO avec un rapport meilleur que celui du problème du stable maximum. Or, ce problème ne peut être approché avec un rapport meilleur que $|V|^{1/2-\epsilon}$ pour tout $\epsilon > 0$, sauf si $P = NP$ d'après [ACG+12]. □

7.7 Chemin hamiltonien avec obligations sur les arêtes

Dans cette section, une instance sera $(G = (V, E), \Pi_E = E_1, \ldots, E_k)$ avec G un graphe connexe et Π_E une partition de E.

Un *chemin hamiltonien avec obligations* (CHO) de (G, Π_E) est un chemin hamiltonien de G vérifiant les obligations de Π_E (si une arête e fait partie

du chemin, alors toutes les arêtes appartenant à cette obligation doivent faire partie du chemin).

Théorème 51. *Déterminer si* (G, Π_E) *contient un CHO est NP-complet même si* G *est un graphe complet.*

Démonstration. Le problème est dans NP. Soit $H = (V, E)$ un graphe connexe quelconque, instance du problème du chemin hamiltonien, qui est un problème NP-Complet (voir [GJ02]). Soit $n = |V|$. On suppose ici que $n \geq 4$ (si n est plus petit, le problème peut être facilement résolu en temps constant). Le graphe de l'instance de notre problème sera K_n, le graphe complet sur les n sommets de H. Les obligations sont les suivantes : pour chaque arête uv de H, l'arête uv de K_n est le seul élément de cette obligation. Toutes les arêtes uv n'étant pas présentes dans H ($uv \notin E$) forment une unique obligation E_0. Cette instance (K_n, Π_E) peut être construite en temps polynomial. Nous séparons notre étude en deux cas.

Cas 1 : E_0 induit un graphe de degré supérieur ou égal à 3. Dans ce cas, les arêtes de E_0 ne peuvent pas être dans un CHO de K_n. Ainsi, H contient un chemin hamiltonien si et seulement si (K_n, Π_E) contient CHO.

Cas 2 : E_0 induit un graphe de degré au plus 2. Dans ce cas, chaque sommet u a un degré au moins $n-2$ dans H. Par hypothèse, $n \geq 4$, ce qui implique que le degré de chaque sommet de H est au moins $n/2$. Ceci est la condition (voir [Die12] par exemple) de Dirac suffisante pour H pour avoir un cycle hamiltonien, et donc également un chemin hamiltonien. Ainsi, H a un chemin hamiltonien et (K_n, Π_E) a un CHO.

Dans les deux cas, H a un chemin hamiltonien si et seulement si (K_n, Π_E) a un CHO. □

7.8 Dominant avec obligations sur les sommets

Dans cette section, une instance sera $(G = (V, E), \Pi_V = V_1, \ldots, V_k)$ avec G un graphe et Π_V une partition de V.

Un *dominant avec obligations* (DSO) de (G, Π_V) est un dominant de G vérifiant les obligations de Π_V (si un sommet u fait partie du dominant, alors tous les sommets appartenant à cette obligation doivent faire partie du dominant).

Remarque 8. Il existe toujours un dominant respectant les obligations : l'ensemble V des sommets du graphe.

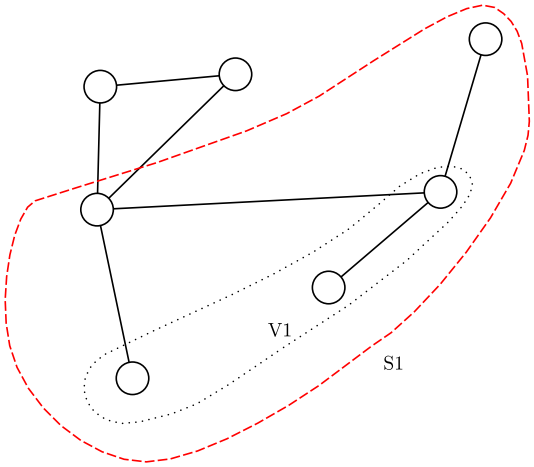


FIGURE 7.4 – Construction de S_1 à partir de V_1.

Le problème de la minimisation de la taille du dominant avec obligations est NP-complet et inapprochable par un rapport meilleur que $c \cdot log|V|$ pour un $c > 0$: en effet, dans le cas où les obligations sont des singletons, on retombe sur le problème du dominant classique, qui a ces caractéristiques [RS97].

Nous montrons maintenant qu'il est possible de construire un algorithme polynomial $\mathcal{O}(log(|V|))$-approché pour le problème du dominant avec obligations de taille minimale.

Pour cela, nous réduisons notre problème au problème du *Set Cover Pondéré* pour lequel il existe un algorithme d'approximation de rapport $\mathcal{O}(log(|V|))$ [Chv79].

Théorème 52. *Étant donné (G, Π_V), il est possible de trouver en temps polynomial un DSO $\mathcal{O}(log(|V|))$-approché.*

Démonstration. Étant donné $(G = (V, E), \Pi_V)$, on construit (U, S, w) une instance du set cover pondéré. Soit $U = V$. Pour toute obligation $V_i \in \Pi_V$, on construit un ensemble S_i composé de l'union des voisinages fermés des sommets de V_i (le voisinage fermé d'un sommet x est l'ensemble des voisins de x et x lui même). Remarquons qu'un ensemble S_i contient exactement les éléments correspondant à un sommet dominé par V_i. Le poids de cet ensemble sera la taille de l'obligation (qui sera en général différente de la taille de S_i) c'est à dire $w(S_i) = |V_i|$. La famille S des ensembles de l'instance du set cover est alors composée de tous ces S_i. La figure 7.4 montre un exemple de construction de S_1 à partir de V_1. Ici, l'ensemble construit sera de poids 3 (la taille de V_1) et dominera V_1 et ses voisins.

Nous construisons maintenant une bijection entre l'ensemble des dominants avec obligations de (G, Π_V) et l'ensemble des set cover de (U, S, w).

Soit D un dominant avec obligations de (G, Π_V). Comme D respecte les obligations, D est une union d'obligations. Posons $C = \bigcup_{i|V_i \in D} S_i$. Comme D est un dominant de G, chaque sommet de V est dominé et donc chaque élément de U est couvert par C. On constate également que $|D| = \sum_{i|V_i \in D} |V_i| = \sum_{i|V_i \in D} w(S_i) = w(C)$.

Inversement, si C est un set cover de (U, S, w), on pose $D = \bigcup_{i|S_i \in C} V_i$. Comme C est un set cover, chaque élément est couvert par au moins un S_1, et donc chaque sommet correspondant est dominé, par lui même s'il est dans V_i, ou par un de ses voisins dans V_i, D est bien un dominant de G. De plus, par construction, D respecte les obligations. Comme précédemment, on constate que $w(C) = \sum_{i|V_i \in D} w(S_i) = \sum_{i|V_i \in D} |V_i| = |D|$. □

Dans le problème du dominant, l'ajout d'obligations ne semble donc pas changer fondamentalement la difficulté du problème : en effet, le rapport d'approximation ne change pas.

7.9 Dominant total avec obligations sur les sommets

Dans cette section, une instance sera $(G = (V, E), \Pi_V = V_1, \ldots, V_k)$ avec G un graphe et Π_V une partition de V.

Un *dominant total avec obligations* (TDSO) de (G, Π_V) est un dominant total de G vérifiant les obligations de Π_V (si un sommet u fait partie du dominant total, alors tous les sommets appartenant à cette obligation doivent faire partie du dominant).

L'ajout des obligations au problème du dominant total produit les mêmes effets que pour le problème du dominant. Les réductions sont extrêmement similaires. Dans un souci de complétude, nous reproduisons la preuve intégrale dans cette section.
Remarque 9. Si G est un graphe sans sommet isolé, il existe toujours un dominant total respectant les obligations : l'ensemble de sommets de G. Si G a un sommet isolé, alors G n'a pas de dominant total.

Le problème de la minimisation de la taille du dominant total avec obligations est NP-complet et inapprochable par un rapport meilleur que $c \cdot log|V|$ pour un $c > 0$: en effet, dans le cas où les obligations sont des singletons, on retombe sur le problème du dominant total classique, qui a ces caractéristiques, voir par exemple [CC04].

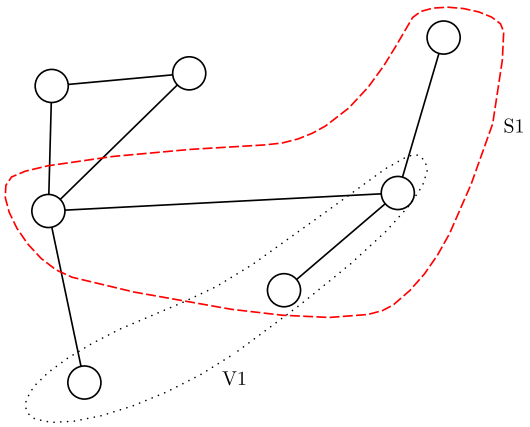


FIGURE 7.5 – Construction de S_1 à partir de V_1.

Grâce à une réduction au problème du Set Cover, nous montrons maintenant qu'il est possible de construire un algorithme polynomial $\mathcal{O}(log(|V|))$-approché pour le problème du dominant total avec obligations de taille minimale.

Théorème 53. *Étant donné (G, Π_V), il est possible de trouver en temps polynomial un TDSO $\mathcal{O}(log(|V|))$-approché.*

Démonstration. Soit $(G = (V, E), \Pi_V)$ une instance du TDSO. Supposons sans perte de généralité que l'instance a une solution (c'est à dire que G n'a pas de sommet isolé) et construisons (U, S, w) une instance du set cover pondéré. Soit $U = V$. Pour toute obligation $V_i \in \Pi_V$, on construit un ensemble S_i composé de l'union des voisinages *ouverts* des sommets de V_i (le voisinage ouvert d'un sommet x est l'ensemble des voisins de x, et x n'est *pas* inclus). Remarquons qu'un ensemble S_i contient exactement les éléments correspondant à un sommet dominé par V_i Le poids de cet ensemble sera la taille de l'obligation (qui sera en général différente de la taille de S_i) c'est-à-dire $w(S_i) = |V_i|$. La famille S des ensembles de l'instance du set cover est alors composée de tous ces S_i.

La figure 7.5 montre un exemple de construction de S_1 à partir de V_1. Ici, l'ensemble construit sera de poids 3 (la taille de V_1) et dominera l'union des voisinages ouverts des sommets de V_1. Remarquons que tous les sommets de V_1 n'appartiennent pas à S_1 car certains de ces sommets ne sont pas voisins d'autres sommets de V_1.

Nous construisons maintenant une bijection entre l'ensemble des dominants totaux avec obligations de (G, Π_V) et l'ensemble des set cover de (U, S, w).

Soit D un dominant total avec obligations de (G, Π_V). Comme D respecte les obligations, D est une union d'obligations. Posons $C = \bigcup_{i|V_i \in D} S_i$. Comme D est un dominant total de G, chaque sommet de V est dominé par un de ses voisins et donc chaque élément de U est couvert par C. On constate également que $|D| = \sum_{i|V_i \in D} |V_i| = \sum_{i|V_i \in D} w(S_i) = w(C)$.

Inversement, si C est un set cover de (U, S, w), on pose $D = \bigcup_{i|S_i \in C} V_i$. Comme C est un set cover, chaque élément est couvert par au moins un S_i, et donc chaque sommet correspondant est dominé par un de ses voisins dans V_i, D est bien un dominant total de G. De plus, par construction, D respecte les obligations. Comme précédemment, on constate que $w(C) = \sum_{i|V_i \in D} w(S_i) = \sum_{i|V_i \in D} |V_i| = |D|$. □

Comme pour le problème du dominant, l'ajout d'obligations au problème du dominant total ne semble pas modifier la difficulté du problème : le rapport d'approximation ne change pas.

7.10 Dominant indépendant avec obligations sur les sommets

Dans cette section, une instance sera $(G = (V, E), \Pi_V = V_1, \ldots, V_k)$ avec G un graphe et Π_V une partition de V.

Un *dominant indépendant avec obligations* (IDSO) de (G, Π_V) est un dominant indépendant de G vérifiant les obligations de Π_V (si un sommet u fait partie du dominant total, alors tous les sommets appartenant à cette obligation doivent faire partie du dominant).

Contrairement aux problèmes du dominant ou du dominant total, l'ajout des obligations peut rendre certaines instances sans solution. Nous allons prouver qu'il est en fait NP-complet de déterminer l'existence d'une solution.
Théorème 54. *Déterminer si (G, Π_V) contient un IDSO est NP-complet.*

Démonstration. Soit (X, Z) une instance de X3C. Pour chaque élément x de X, un P_3 (chemin à trois sommets) est créé dont une extrémité sera le *sommet représentant l'élément*. Pour chaque sous-ensemble z de Z, un P_2 est créé dont une extrémité sera le *sommet représentant le sous-ensemble*. Des arêtes sont ajoutées entre :

— les sommets représentant un sous-ensemble et les sommets représentant un élément leur appartenant.
— toute paire de sommets représentant des sous-ensembles dont l'intersection est non-vide.

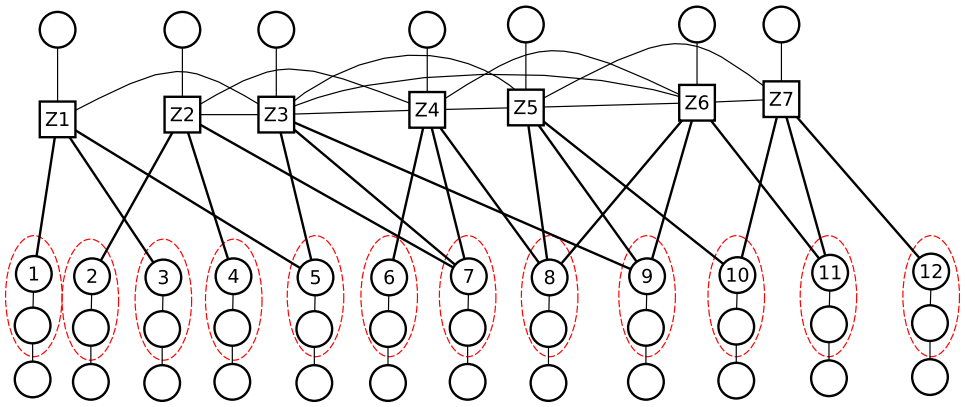


FIGURE 7.6 – Construction de (G, Π_V) à partir de (X, Z).

Pour chaque élément x, une obligation contenant le sommet représentant x et son voisin dans le P_3 est créée, on les appelle *obligations d'élément*. Les autres obligations sont des singletons. Un exemple de cette construction est présenté figure 7.6.

Soit D un dominant indépendant respectant les obligations de (G, Π_V). On sait que D ne contient aucune obligation d'éléments, car ces obligations contiennent des sommets voisins. Ainsi, chaque sommet d'élément ne peut être dominé que par les sommets de sous-ensembles voisins. Soit S la famille de sous-ensembles correspondants aux sommets de sous-ensemble appartenant à D. Alors, comme chaque sommet d'élément est dominé, chaque élément est couvert. De plus, comme deux sous-ensembles ayant une intersection non nulle sont voisins et que D est un indépendant, les sous-ensembles de S sont deux à deux disjoints : S est bien une couverture exacte de (X, Z).

Soit maintenant S une couverture exacte de (X, Z). Construisons D un sous-ensemble de V. Pour chaque Z_i, le sommet de sous-ensemble correspondant appartient à D si et seulement si $Z_i \in S$. Sinon, le voisin de Z_i dans le P_2 appartient à D. Ajouter à D tous les sommets des extrémités opposées aux sommets d'éléments de chaque P_3.

Remarquons tout d'abord que D est un indépendant. De plus, D respecte les obligations : chaque sommet appartenant à D est dans une obligation singleton. Enfin, chaque élément est couvert par S : chaque sommet d'élément est donc dominé par un sommet de sous-ensemble. Les P_2 sont chacun dominés par le sommet de sous-ensemble ou son voisin, et pour chaque P_3, l'extrémité opposée au sommet d'élément appartient à D et se domine, ainsi que son voisin. D est donc bien un dominant indépendant respectant les obligations de (G, Π_V). □

7.11 Conclusion

Dans ce chapitre, nous avons montré que, comme les conflits, ajouter des obligations augmente drastiquement la complexité de problèmes de graphes classiques. C'est le cas pour le vertex cover connexe avec obligations qui n'admet pas d'algorithme d'approximation à rapport constant (alors qu'il existe une 2-approximation du vertex cover connexe) et du graphe couvrant connexe de poids minimum. Pour d'autres problèmes, la situation est encore pire : il devient NP-complet de déterminer s'il existe une solution, indépendamment de sa taille (ce qui pouvait être fait trivialement ou en temps polynomial dans le problème initial). Ceci concerne les problèmes du dominant indépendant avec obligations, de l'arbre couvrant avec obligations, et du chemin hamiltonien dans les graphes complets. Pour le dominant et le dominant total, les rapports d'approximation sont presque les mêmes avec ou sans les obligations, mais ces ratios ne sont pas constants. Seul le problème du vertex cover garde le même rapport constant de 2.

On pourrait imaginer qu'une perspective serait de raffiner nos résultats en étudiant des instances plus spécifiques et restreintes, malheureusement, dans certains cas, le problème est "équivalent" à un autre problème difficile (set cover, stable maximal,...) ayant déjà reçu beaucoup d'attention, apporter des améliorations sur ces problèmes est difficile en soi. Dans d'autres cas, les instances pour lesquelles nos problèmes sont difficiles sont basiques : des graphes bipartis de degré borné avec de très petites obligations pour le problème de l'arbre couvrant, graphe complet pour le problème du chemin hamiltonien, etc.

D'autres problèmes combinatoires pourraient être étudiés avec des obligations, mais nos résultats montrent que cela conduit à des problèmes très complexes pouvant être insolvables en temps polynomial. Concevoir des systèmes concrets avec obligations sera d'une grande complexité.

Chapitre 8

Firefighter et contraintes de déplacement

Ce chapitre est le fruit d'un séjour scientifique d'un mois au RMIT (Royal Melbourne Institute of Technology) financé par le projet GEO-SAFE (Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies). Ce travail a été commencé en fin de thèse, et sa thématique diffère légèrement du reste de ce manuscrit, le problème traité se rapprochant d'un jeu combinatoire. Aussi, ce chapitre ne prétend pas proposer un grand nombre de résultats, mais présenter le problème du *Firefighter*, des résultats préliminaires et des perspectives de recherche pour un travail encore en cours.

Le problème du *Firefighter* a été introduit en 1995 par Bert Hartnell [Har95]. Ce problème modélise la propagation d'un feu que des pompiers tentent de contenir. L'espace dans lequel le feu se propage est modélisé par un graphe, la propagation du feu et les actions des pompiers se font tour par tour.

Un feu se déclare dans un ou plusieurs sommets d'un graphe. À chaque tour, chaque pompier peut protéger (définitivement) un sommet non encore brûlé. À la fin de chaque tour, le feu se propage à tous les sommets voisins non protégés.

Le problème a depuis été étudié sous de nombreux angles : est-il possible de contenir le feu dans une grille infinie ? Avec combien de pompiers par tour ? Est-il possible de sauver un sous-ensemble de sommets donné dans un graphe fini ? Pour un aperçu des questions considérées et des résultats existants, voir par exemple [FM09].

Nous nous intéresserons dans ce chapitre à plusieurs versions contraintes du problème, dans lesquelles les pompiers ne peuvent pas être déployés librement. Dans une première variante, les pompiers auront une vitesse limitée :

chaque pompier pourra parcourir au plus v arêtes entre le sommet protégé à un tour et le sommet protégé au tour suivant, sans traverser de sommets en feu. Au premier tour, les pompiers peuvent être placés sans contraintes dans le graphe.

Dans une seconde variante, les pompiers n'auront pas de contraintes de déplacement, mais devront opérer par paires de pompiers adjacents (afin que les équipes déployées puissent s'entraider rapidement en cas de problème).

Dans un graphe infini, on dira qu'un feu est *contenu* si la suppression d'un nombre fini de sommets protégés déconnecte le graphe et que chaque composante connexe contenant un sommet en feu est finie.

Dans ce chapitre, pour les figures représentant des grilles, les sommets protégés par des pompiers seront représentés en vert (gris clair) et les sommets en feu seront représentés en rouge (gris foncé). Les cases brûlées ou protégées seront numérotées en fonction du tour auquel elles ont été brûlées ou protégées. Les sommets en feu numérotés 0 correspondant aux sommets initialement en feu.

8.1 Grilles infinies avec contraintes de déplacement

Dans cette section, une instance est un triplet (F, p, v) où F est un ensemble fini de sommets initialement en feu, p le nombre de pompiers disponibles à chaque tour et v la vitesse de déplacement maximale des pompiers. Wang et Moeller montrent dans [WM02] qu'un pompier par tour ne suffit pas à contenir le feu, même sans contrainte de déplacement. Dans [DH07], Develin et Hartke montrent que dans le cas où F est limité à un sommet, sans contrainte de déplacement, deux pompiers sont suffisant pour contenir le feu. Ils proposent également une solution (présentée figure 8.1) en 8 tours et 18 sommets brûlés et prouvent son optimalité. Ce résultat est ensuite généralisé par Ng et Raff qui montrent que 2 pompiers par tour suffisent à contenir n'importe quel nombre fini de départs de feu (sans contrainte de déplacement) [NR08].

Toutefois, les stratégies proposées ne peuvent pas être appliquées dans les cas de déplacements limités, en effet, les déplacements des pompiers entre deux tours induits par ces stratégies sont non bornés.

8.1.1 Grille à vitesse 1

À vitesse 1, 4 pompiers sont nécessaires et suffisants pour arrêter un unique feu. Pour arrêter le feu, il suffit simplement de placer un pompier de chaque

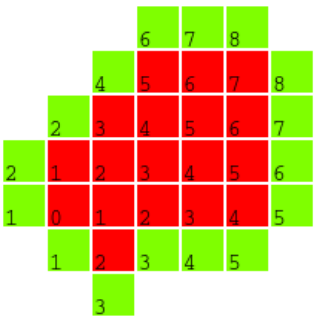


FIGURE 8.1 – Solution optimale de Develin et Hartke avec 2 pompiers, sans contrainte de déplacement.

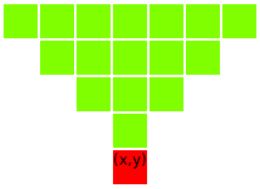


FIGURE 8.2 – Positions *pouvant contenir* (x, y) au nord.

coté au premier tour. Nous montrons maintenant que 3 pompiers ne peuvent pas contenir le feu.

Lemme 19. *Pour qu'un sommet en feu aux coordonnées* (x, y) *soit contenu, il doit exister au moins* 4 *sommets protégés aux coordonnées* (x, a), (x, b), (c, y), (d, y) *avec* $a < y < b$ *et* $c < x < d$.

Démonstration. Supposons qu'un de ces sommets n'existe pas, par exemple (x, a) (les autres cas sont symétriques). Alors la suite infinie de sommets $(x, y), (x, y-1), (x, y-2)$... contient un sommet en feu et n'est pas déconnectée par la suppression des sommets protégés. Le feu n'est donc pas contenu. □

Étant données (x, y) les coordonnées d'un sommet en feu, on dit que la position (a, b) *peut contenir* (x, y) au nord (symétriquement sud, est, ouest) si $b - y > |a - x|$. Cette notion est illustrée à la figure 8.2.

Lemme 20. *À vitesse* 1, *s'il existe un sommet en feu* (x, y) *tel qu'aucun pompier ne peut le contenir par le nord (symétriquement sud, est, ouest) au tour* t, *alors, quel que soit le déplacement des pompiers, il existera un sommet en feu tel qu'aucun pompier ne peut le contenir par le nord (symétriquement sud, est, ouest) au tour* $t + 1$.

Démonstration. Soit (x, y) un sommet en feu tel qu'aucun pompier ne puisse le contenir par le nord au tour t (les autres cas sont symétriques). Alors, au

tour $t+1$, le sommet $(x, y+1)$ est en feu (car il n'était pas protégé). Chaque pompier en position (a, b) au tour $t-1$ vérifiait $b - y \leq |a - x|$ et peut être placé aux positions $(a+1, b), (a-1, b), (a, b+1)$ ou $(a, b-1)$. Or :

$$b - y \leq |a - x| \Rightarrow \begin{cases} b - (y+1) \leq |a + 1 - x| \\ b - (y+1) \leq |a - 1 - x| \\ b + 1 - (y+1) \leq |a - x| \\ b - 1 - (y+1) \leq |a - x| \end{cases}$$

Ainsi, au tour $t+1$, le sommet $(x, y+1)$ ne peut être contenu. □

Théorème 55. *À vitesse* 1*,* 3 *pompiers ne suffisent pas à contenir un unique feu dans la grille infinie.*

Démonstration. Notons que les quatre sommets du lemme 19 peuvent contenir respectivement le feu initial au nord, sud, est, ouest. Ainsi, d'après le lemme 20, il est nécessaire d'avoir dès le premier tour un pompier pouvant contenir le feu initial au nord, un pompier pouvant le contenir au sud, un à l'est, un à l'ouest. On remarque qu'un pompier ne peut être en position de pouvoir contenir un feu que d'un coté à la fois. 4 pompiers sont donc nécessaires pour contenir un feu à vitesse 1. □

Nous nous intéressons maintenant au cas où plusieurs feux se sont déclarés. Notons que tout ensemble fini de feux est inscrit dans un carré correspondant à un feu s'étant propagé pendant un nombre fini de tours. Nous considérerons donc uniquement les feux de cette forme.

Théorème 56. *À vitesse* 1*,* 8 *pompiers sont suffisants pour arrêter tout feu de taille finie.*

Démonstration. Supposons sans perte de généralité que le feu soit inscrit dans un carré correspondant à un feu initial centré en $(0, 0)$ laissé se propager pendant k tours. Le feu aura atteint les sommets $(0, k), (0, -k), (k, 0), (-k, 0)$. Au tour 1, on place les pompiers aux extrémités du carré formé par le feu aux emplacements $(0, k+1), (1, k+1), (k+1, 0), (k+1, -1), (0, -k-1), (-1, -k-1), (-k-1, 0), (-k-1, 1)$. Chaque pompier se déplace ensuite dans la direction opposée au pompier lui étant adjacent, jusqu'à ce que le feu soit contenu. Cette stratégie demande $k+1$ tours pour contenir le feu et un total de $(2k)^2$ sommets sont brûlés. La figure 8.3 présente un exemple de cette stratégie. □

Comme le montre le théorème 56, 8 pompiers sont suffisants. Mais sont-ils nécessaires ? Un complément intéressant à ce théorème serait de déterminer s'il existe des feux pour lesquels 7 pompiers sont insuffisants.

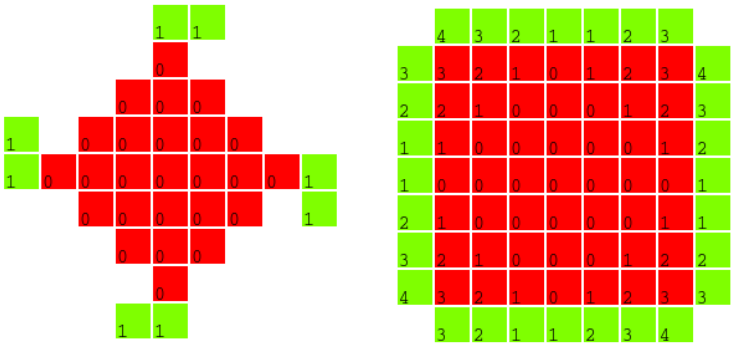


FIGURE 8.3 – À gauche, le déploiement initial des pompiers, à droite la situation finale.

8.1.2 Grille à vitesse 2

A vitesse 2, un pompier pourra traverser jusqu'à deux cases mais ne protégera que sa case d'arrivée.

Théorème 57. *A vitesse 2, 3 pompiers sont suffisants pour contenir tout feu fini.*

Démonstration. On suppose sans perte de généralité que le feu initial forme un carré de coté k centré en $(0,0)$. Au tour 1, le pompier p_1 est placé à l'extrémité nord du feu, en $(0,k)$. Le pompier p_2 est placé immédiatement à l'ouest, en $(-1,k)$ et le pompier p_3 au sud de p_2, en $(-1,k-1)$. À chaque tour, p_1 se déplacera d'une case vers l'est et protégera sa case d'arrivée. Le feu se déplaçant à la même vitesse, p_1 sera toujours à la même distance du front est du feu. p_1 construit donc un mur de protection contre le feu. Pendant ce temps, p_2 et p_3 vont encercler le feu à vitesse 2 au nord-ouest, puis sud-ouest, sud-est, nord-est. Un exemple de cette stratégie est proposé à la figure 8.4. p_3 va se déplacer à vitesse 2 autour du feu en protégeant à chaque tour son sommet d'arrivée. p_2 empruntera le même chemin que p_3 avec une case de retard, et protégera les sommets non protégés par p_3, formant ainsi un chemin contenant le feu.

Plus précisément, le front ouest du feu (première case non brûlée) se situe initialement en $(-k,0)$, à distance $2k-2$ de p_3. À chaque tour, ce front se déplace d'une case vers l'est. Ainsi, à vitesse 2, il faudra $2k-2$ tours pour que p_3 atteigne le front ouest du feu, alors en position $(-3k+2,0)$. Notons que p_3 et p_2 n'ont pas un unique chemin pour joindre ce point : n'importe quel plus court chemin (hors du feu) fait l'affaire. Ainsi, un chemin au plus proche du feu minimisera la surface brûlée, alors qu'un chemin plus loin du feu pourra être moins dangereux ou plus facile d'accès.

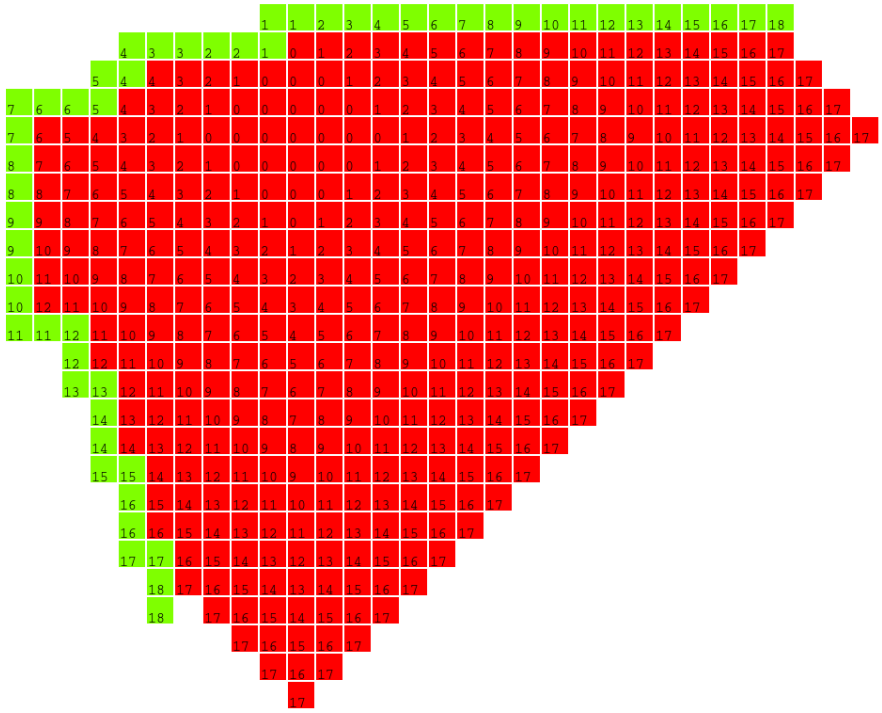


FIGURE 8.4 – 3 pompiers contenant un feu à vitesse 2. Le front ouest a déjà été contourné, les pompiers sont maintenant en train de se diriger vers le front sud.

À ce point, p_3 est en position $(-3k+2, 0)$ et le front sud du feu en position $(0, -3k+2)$, à distance $6k-4$ et se déplace vers le sud à vitesse 1. De la même manière que précédemment, $6k-4$ tours seront nécessaires pour rejoindre le front en protégeant un chemin ininterrompu. À ce point, p_3 sera aux coordonnées $(0, -9k+6)$ et le front est en $(9k-6, 0)$. $18k-12$ tours sont nécessaires pour le rattraper en $(27k-18, 0)$.

À cet instant, p_1 sera en position $(26k-18, k)$, à distance $2k$ et se déplacera vers l'est à vitesse 1. p_1 continuera d'avancer vers l'est pendant k tours et p_3 avancera au nord pendant k tours, au terme desquels les pompiers se rejoindront en $(27k-18, k)$ et le feu est contenu au tour $27k-18$. La surface brûlée dépend des chemins empruntés par les pompiers, mais est inscrite dans un rectangle de $10k$ de haut par $30k$ de large et n'excède donc pas $300k^2$. □

Il serait ici aussi intéressant de savoir si les trois pompiers du théorème 57 sont nécessaires ou si deux sont suffisants. Si deux pompiers sont du même coté du feu, celui-ci se propagera de l'autre coté sans pouvoir être rattrapé.

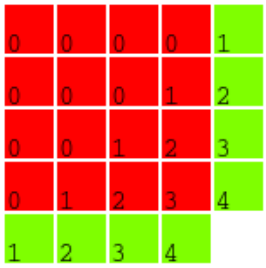


FIGURE 8.5 – 2 pompiers contenant un feu à vitesse 1 dans un quart de grille.

Si les pompiers se séparent, chacun ne pourra que construire un mur de protection de son coté et jamais prendre de l'avance sur le feu pour l'encercler. Ces arguments semblent indiquer que trois pompiers seront nécessaires, ils ne constituent cependant pas une preuve suffisante.

8.1.3 Quart de grille

Les instances en quart de grille modélisent des situations où le feu est bloqué dans certaines directions, par des éléments naturels (rivières, falaises, zone non propice à la propagation du feu) ou climatiques (vent). Dans cette première sous section, on s'intéresse à un quart de grille correspondant à un quadrant (nord-est, nord-ouest, sud-ouest ou sud-est). Nous traiterons le cas sud-ouest, les autres sont symétriques.

Théorème 58. *Quelle que soit la vitesse de déplacement des pompiers,* 2 *pompiers suffisent à arrêter tout feu fini.*

Démonstration. Supposons sans perte de généralité que le feu soit un triangle d'extrémités $(0,0),(0,k),(k,0)$. Au tour 1, placer les pompiers en position $(k+1,0)$ et $(0,k+1)$. À chaque tour, les pompiers vont ensuite se déplacer à vitesse 1 en direction de la case $(k+1,k+1)$. Le feu sera donc contenu en k tours et $k+1$ sommets seront brûlés au total. Cette stratégie est illustrée à la figure 8.5 pour $k=4$.

□

8.1.4 Quart de grille (pivoté)

Ici, on s'intéresse à un quart de grille (nord, sud, est ou ouest) délimité par des diagonales. Nous développerons ici le cas sud, c'est à dire les cases (x,y) telles que $y<0$ et $|x|<|y|$. Dans les figures de cette partie, les cases telles

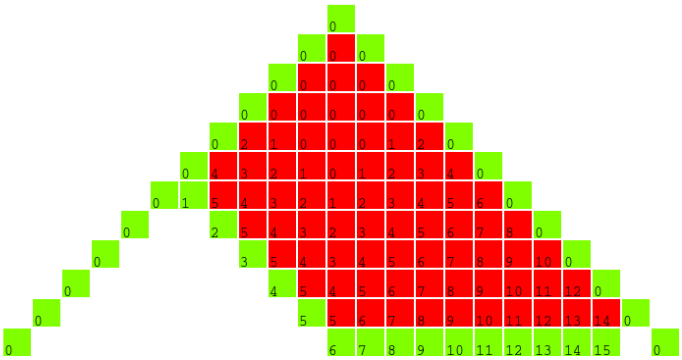


FIGURE 8.6 – Pompier contenant un feu à vitesse 2 dans un quart de grille pivoté.

que $|x| = |y|$ seront montrées comme protégées au tour 0 pour faciliter la visualisation de la zone concernée. Les autres cas sont symétriques.

Théorème 59. *À vitesse 2 ou plus, un pompier est suffisant pour contenir n'importe quel feu initial de taille finie.*

Démonstration. Supposons sans perte de généralité que le feu initial soit un carré d'extrémités $(0, -1), (k, -k-1), (0, -2k-1), (-k, -k-1)$. Le pompier est initialement déployé en $(-2k-1, -2k-2)$. Le pompier se déplace ensuite en diagonale vers le sud-est pendant $2k+1$ tours jusqu'à arriver en position $(0, -4k-3)$. Le front sud du feu atteint alors la case $(0, -4k-2)$. Le pompier se déplace maintenant à vitesse 1 vers l'est pendant $4k+2$ tours jusqu'à atteindre le bord est du quart de grille. Le feu est donc encerclé en un total de $6k+3$ tours. Cette stratégie est illustrée à la figure 8.6 pour $k = 2$. □

À vitesse 1, la stratégie proposée dans la preuve du théorème 59 n'est plus possible, à cause des déplacements en diagonale du pompier (qui sont donc à vitesse 2). Nous proposons maintenant une stratégie à vitesse 1 utilisant 2 pompiers.

Théorème 60. *À vitesse 1, deux pompiers sont suffisants pour contenir n'importe quel feu initial de taille finie.*

Démonstration. Supposons sans perte de généralité que le feu initial soit un carré d'extrémités $(0, -1), (k, -k-1), (0, -2k-1), (-k, -k-1)$. Les pompiers sont initialement déployés en $(0, -2k-2)$ et $(1, -2k-2)$. Les pompiers se déplacent ensuite respectivement vers l'ouest et l'est, à vitesse 1, jusqu'à atteindre le bord du quart de grille et contenir le feu en $2k+1$ tours. □

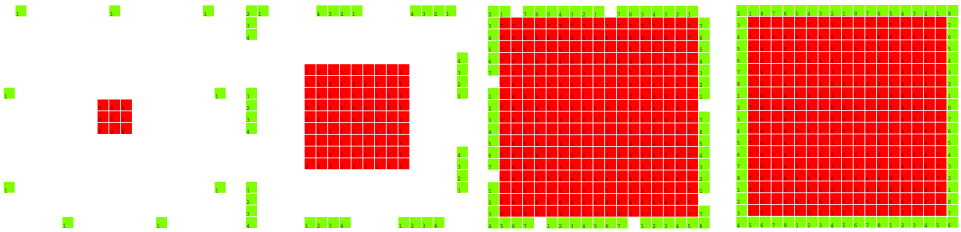


FIGURE 8.7 – 9 pompiers contenant un feu à vitesse 1 dans la grille diagonale.

8.1.5 Grille diagonalisée

Nous cherchons maintenant à contenir le feu dans une grille diagonalisée. Dans la grille diagonalisée, chaque case est voisine des quatre cases nord, sud, est, ouest mais également des quatre cases en diagonale. Le feu pouvant se propager dans plus de directions, il est donc en général plus difficile à contenir.

Théorème 61. *À vitesse* 1*,* 9 *pompiers sont suffisants pour contenir tout ensemble de feu fini.*

Démonstration. Tout ensemble fini de feu est inscrit dans un carré. On s'intéressera donc sans perte de généralité au cas où le feu est un carré de coté $k \geq 3$ impair. On supposera également que ce carré est centré en $(0, 0)$. On propose ici une stratégie utilisant 9 pompiers et arrêtant le feu en $4k - 4$ tours.

Durant ces $4k-4$ tours, les pompiers pourront protéger un total de $9(4k-4)$ cases, réparties sur la frontière d'un carré de coté $9k - 8$ centré en $(0, 0)$. Au premier tour, un pompier est placé sur un sommet arbitraire du carré, puis les autres pompiers sont placés régulièrement avec un espacement de $4k - 4$ cases sur le carré. À chaque tour, chaque pompier se déplacera d'une case sur la frontière du carré, dans le sens trigonométrique. À la fin du tour $4k - 5$, le feu aura un coté de $k + 2(4k - 5) = 9k - 10$ et sera donc inscrit dans le carré de coté $9k - 8$ en train d'être protégé par les pompiers. Au début du tour $4k - 4$, les pompiers finissent de protéger le carré et le feu ne peut plus se propager. Cette stratégie est illustrée à la figure 8.7. □

Conjecture 1. *À vitesse* 1*,* 8 *pompiers ne peuvent pas contenir un feu suffisamment étendu.*

La frontière d'un feu de coté initial k laissé se propager librement pendant t tours est de taille $4k + 4 + 8t$. Avec 8 pompiers, on peut protéger $8t$ sommets en t tour, c'est donc insuffisant. Il faut donc que le feu soit d'abord partiellement contenu sur un coté pour limiter son extension. Ceci parait difficile dans le cas de la grille diagonale à vitesse 1.

Il a été montré dans [Mes11] que sans contrainte de vitesse, 4 pompiers étaient nécessaires et suffisants pour contenir tout feu fini. Avec une vitesse limitée, 4 pompiers seront donc nécessaires. La stratégie proposée dans [Mes11] pour contenir le feu demande des déplacements de pompier arbitrairement grands. Nous proposons donc une stratégie alternative permettant de contenir le feu avec 4 pompiers à vitesse 2.

Théorème 62. *À vitesse 2, 4 pompiers suffisent à contenir tout feu fini dans la grille diagonale.*

Démonstration. Supposons sans perte de généralité que le feu initial forme un carré d'extrémités $(0,0),(0,k),(k,k),(k,0)$. Les 4 pompiers p_1, p_2, p_3 et p_4 seront placés initialement en $(-1,1),(0,1),(1,1)$ et $(2,1)$. p_1 va construire un mur de protection au nord (ouest) du feu : durant toute la durée de l'opération, il avancera à vitesse 1 vers l'ouest en protégeant toutes les cases de sa trajectoire. Les autres pompiers vont contourner le feu dans l'autre sens.

Dans un premier temps, ils vont rattraper le front est du feu. Pour cela, tant que p_4 n'a pas dépassé le front est du feu, les trois pompiers se déplaceront ainsi : $(x,y) \rightarrow (x+2, y+1)$. Les pompiers se déplaçant vers l'est plus rapidement que le feu, ceux-ci pourront le rattraper en un temps fini.

Quand p_4 aura dépassé le front est du feu, les pompiers vont entamer une manœuvre de contournement de l'angle du feu comme décrit dans le tableau suivant donnant les coordonnées des pompiers durant la manœuvre, en supposant que p_4 commence en position initiale (x,y) :

Pompier	t	t+1	t+2
p_2	$(x-2,y)$	(x,y)	$(x+2,y-1)$
p_3	$(x-1,y)$	$(x+1,y)$	$(x+2,y-2)$
p_4	(x,y)	$(x+1,y-1)$	$(x+2,y-3)$

Après cette manœuvre, les pompiers se retrouvent en position de longer le front est de la même manière qu'ils viennent de longer le front nord. Ils le longent donc jusqu'à rattraper le front sud, effectuent une manœuvre de contournement du coté sud-est, longent le front sud jusqu'à rattraper le front ouest, contournent le coin sud ouest, puis remontent le front ouest jusqu'à rejoindre p_1 et contenir le feu. Un exemple de cette stratégie est montrée figure 8.8. □

8.1.6 Quart de grille diagonale

Comme dans la grille classique, nous étudions maintenant le cas du quart de grille diagonale. Le cas développé sera ici le cas sud-est, les autres étant symétriques.

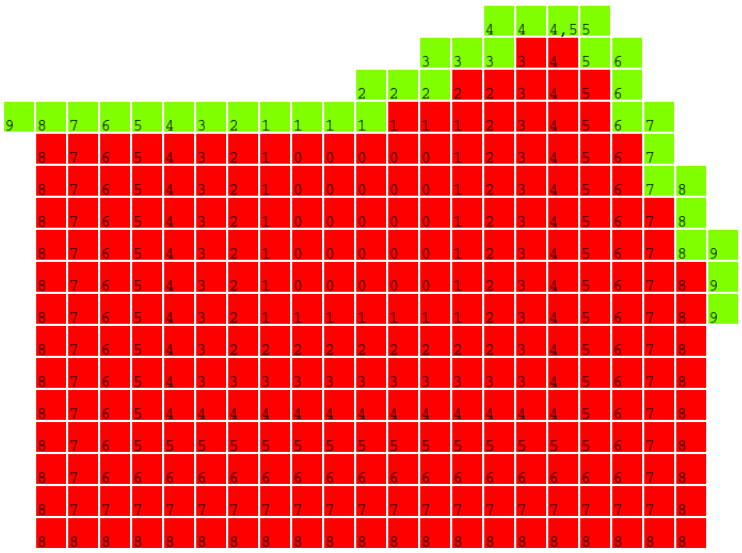


FIGURE 8.8 – 4 pompiers contenant un feu à vitesse 2 dans la grille diagonale. le coin nord-est a été contourné, les pompiers longent maintenant le front est en direction du de l'angle sud-est.

Théorème 63. *À vitesse* 2 *ou plus,* 2 *pompiers sont suffisants pour contenir tout feu fini dans le quart de grille diagonale.*

Démonstration. Supposons sans perte de généralité que le feu soit un carré d'extrémités $(0,0),(k,0),(k,-k),(0,-k)$. Les pompiers sont déployés initialement en $(0,-2k)$ pour p_1 et en $(1,-2k)$ pour p_2. Pendant k tours, ils se déplaceront vers l'est à vitesse 2 en formant un mur de protection au sud du feu. Au tour $k+1$, p_1 se déplacera de deux cases vers l'est et $p2$ d'une case au nord-est. Pendant $2k-2$ tours, p_1 et p_2 se déplacerons ensuite au nord-est d'une case par tour, jusqu'à atteindre le bord de la grille, au tour $3k-1$. Une illustration de cette stratégie est présentée figure 8.9. □

Cette stratégie n'est pas possible dans le cas de déplacement à vitesse 1, nous proposons donc une stratégie à vitesse 1 utilisant 3 pompiers.
Théorème 64. *À vitesse* 1*,* 3 *pompiers sont suffisants pour contenir tout feu fini dans le quart de grille diagonale.*

Démonstration. Supposons sans perte de généralité que le feu soit un carré d'extrémités $(0,0),(k,0),(k,-k),(0,-k)$. Les pompiers sont déployés initialement en $(0,-3k)$ pour p_1, $(2k+1,-3k)$ pour p_2 et $(3k,-2k)$ pour p_3. p_1 se déplacera à l'est à vitesse 1 pendant $2k$ tours jusqu'à atteindre la position initiale de $p2$. p_2 se déplacera vers l'est à vitesse 1 pendant k tours, puis vers le nord pendant k tours, jusqu'à rejoindre la position initiale de p_3. p_3

FIGURE 8.9 – 2 pompiers contenant un feu à vitesse 2 dans le quart de grille diagonale.

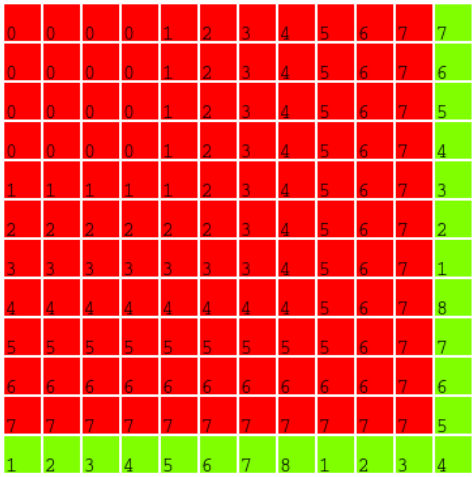


FIGURE 8.10 – 3 pompiers contenant un feu à vitesse 1 dans le quart de grille diagonale.

se déplacera vers le nord pendant $2k$ tours jusqu'à atteindre le bord de la grille. Le feu sera contenu en $2k$ tours et le nombre de sommets brûlés sera de $3k^2$. Une illustration de cette stratégie est présentée figure 8.10. □

8.1.7 Quart de grille diagonale (pivoté)

Ici, on s'intéresse à un quart de grille (nord, sud, est ou ouest) délimité par des diagonales. Nous développerons ici le cas sud, les autres cas sont symétriques.

Théorème 65. *Quelle que soit la vitesse, 3 pompiers sont suffisants pour contenir le feu dans le quart de grille diagonale pivoté.*

Démonstration. Supposons sans perte de généralité que le feu soit un triangle d'extrémités $(0,-1),(k,-k-1),(-k,-k-1)$. Les pompiers sont déployés initialement en $(-3k,-3k-1),(-k+1,-3k-1),(k+2,-3k-1)$. Pen-

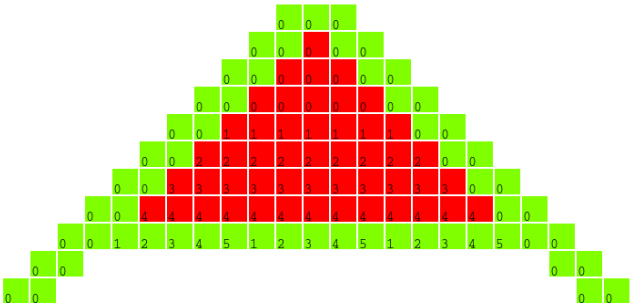


FIGURE 8.11 – 3 pompiers contenant un feu à vitesse 1 dans le quart de grille diagonale.

dant les $2k+1$ tours de l'opération, les pompiers se déplaceront à vitesse 1 vers l'est. Au tour $2k+1$, quand le feu atteindra la ligne $-3k-1$, celle-ci sera entièrement protégée, et le feu sera contenu. Une illustration de cette stratégie est présentée figure 8.11. □

8.2 Résultats de complexité dans les graphes finis

Nous nous intéressons maintenant au cas des graphes quelconques finis. Dans le cas d'un graphe fini, la question n'est plus de déterminer si il est possible de contenir le feu (il suffirait d'attendre qu'il se propage dans tout le graphe avant de ne plus pouvoir se propager...), mais de sauver un maximum de sommets, ou sauver certains sommets. Nous étudierons ici le problème de déterminer s'il est possible de sauver un ensemble de sommets prédéterminés.

Dans cette section, une instance sera (G, F, S, p, v) où G est un graphe fini, F un ensemble de sommets en feu, S un ensemble de sommets à sauver, p le nombre de pompiers et v la vitesse maximale de déplacement.

Remarquons que dans un graphe à n sommets, le feu peut se propager pendant au plus n tours. Nous limiterons donc également les déplacements des pompiers aux n premiers tours. Nous restreignons également le nombre de pompiers, qui doit être polynomial en n. Une solution pourra donc être exprimée en espace polynomial, par exemple par la liste des positions des pompiers à chaque tour. La vérification de solutions pourra donc se faire facilement en temps polynomial, et les problèmes étudiés sont donc dans NP : les preuves de NP-complétudes qui suivrons traiteront uniquement de la NP-difficulté.

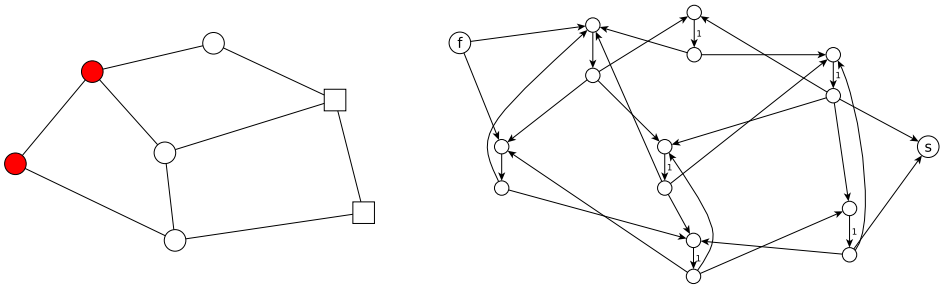


FIGURE 8.12 – A gauche, une instance de firefighter. Les sommets rouges sont initialement en feu et les sommets carrés doivent être sauvés. A droite, une instance de fs-coupe minimale équivalente. Les arêtes non marquées sont de capacité M.

8.2.1 Est-il possible de sauver les sommets de S en un tour ?

Théorème 66. *Il est polynomial de déterminer si l'on peut sauver les sommets de S en un tour.*

Démonstration. Soit (G, F, S, p, v) une instance de firefighters avec $G = (V, E)$. Nous allons réduire cette instance à un problème de coupe minimale. Soit $G_2 = (V_2, A)$ un graphe orienté pondéré avec : $V_2 = \{a_{in}, a_{out} | a \in V\} \cup \{f\} \cup \{s\}$ et $A = A_1 \cup A_2 \cup A_3 \cup A_4$ avec $A_1 = \{(a_{out}, b_{in})| = \{a, b\} \in E\}$ l'ensemble des arcs correspondant aux arêtes de G (deux arcs par arêtes), $A_2 = \{(a_{in}, a_{out}) | a \in V\}$ l'ensemble des arcs traversants correspondant aux sommets de G, $A_3 = \{(f, a_{in}) | a \in F\}$ l'ensemble des arcs reliant le nouveau sommet f aux sommets de F et $A_4\{(a_{out}, s) | a \in S\}$ l'ensemble des arcs reliant les sommets de S au nouveau sommet s. Les arcs de A_1, A_3 et A_4 ont des capacités de $M > p$ et les arcs (a_{in}, a_{out}) de A_2 une capacité de M si $a \in F$, 1 sinon. Un exemple de cette construction est montré figure 8.12.

Il est polynomial de calculer une coupe minimale séparant f de s dans G_2, en utilisant par exemple un algorithme de flot [FF56].

Supposons qu'il existe une fs-coupe de capacité inférieure ou égale à p. Les arêtes de capacité M ne peuvent en faire partie : seules au plus p arêtes de A_2 de capacité 1 correspondant à des sommets de G et n'appartenant pas à F peuvent en faire partie. Il est donc possible de protéger ces sommets en un tour en respectant le nombre de pompiers assignés. Soit $p_1, ...p_i$ un chemin d'un sommet de F vers un sommet de S. Supposons qu'il n'existe aucun sommet protégé sur ce chemin. Alors le chemin $f, p_{1in}, p_{1out}, ..., p_{iin}, p_{iout}, s$ ne contient aucune arrête de la coupe : une contradiction. Ainsi pour tout chemin entre un sommet de F et un sommet de S, il existe un sommet protégé : les sommets de S sont sauvés en 1 tour.

Supposons maintenant que les sommets de S soient sauvés en 1 tour et soit P l'ensemble des sommets protégés. Soit $C = \{(p_{in}, p_{out}) | p \in P\}$. Toutes les arêtes de C sont de capacité 1 (elles appartiennent à A_2 et correspondent à des sommets n'étant pas en feu), ainsi C est de capacité inférieure ou égale à p. Montrons maintenant que C sépare f de s. Remarquons que tous la seule arête sortante d'un sommet a_{in} est dirigée vers le sommet a_{out} et que la seule arête entrante d'un sommet a_{out} provient du sommet a_{in}. Soit P un chemin quelconque de f à s. P est donc de la forme $f, p_{1in}, p_{1out}, ..., p_{iin}, p_{iout}, s$. Supposons qu'il n'existe aucune arête de C dans ce chemin. Alors le chemin $p_1, ...p_i$ correspondant dans G ne contiens pas de sommet protégé, une contradiction. Ainsi, C sépare bien f de s. □

Nous nous intéressons maintenant au cas ou les pompiers doivent être déployés par paires. On dit que les pompiers sont déployés par paires si au tour 1, les pompiers sont identifiés et répartis par paires et qu'à chaque tour, les deux pompiers de chaque paire sont sur des sommets à distance au plus 1 (deux pompiers pouvant se trouver sur la même case).

Théorème 67. *Il est NP-complet de déterminer s'il est possible de sauver les sommets de S en un tour dans le cas ou les pompiers doivent être déployés par paires, même si G est biparti planaire, et que le feu n'atteint les sommets de degré supérieur à 2 et les sommets à protéger qu'au tour 3.*

Démonstration. Soit (X, C) une instance de planar 3-bounded 3-SAT. Rappelons que ce problème est NP-complet et que dans une telle instance, chaque variable apparaît dans au plus 3 clauses, et le *graphe associé* à la formule est planaire (le graphe associé est le graphe dans lequel chaque variable et chaque clause sont représentés par un sommet, et chaque sommet de variable est relié aux sommets des clauses dans lesquels elle apparaît) [TM16]. Construisons une instance de firefighter telle qu'il existe une affectation des variables de X vérifiant C si et seulement si il est possible de sauver les sommets de S en un tour en déployant les pompiers par paire. On pose $p = 2k = 2|X| + 4|C|$ le nombre de pompiers disponibles. Pour chaque variable x_i, on construit un gadget à $9p + 12$ sommets. Un sommet S_i est connecté à deux sommets x_i et $\bar{x_i}$. Chacun de ces trois sommets est relié à $p + 1$ chemins de longueur 3 dont l'extrémité opposée est en feu. Une illustration de ce gadget est proposée figure 8.13. Le sommet $s_i \in S$ (il fait partie des sommets à sauver).

Pour chaque clause $c_i = (l_j \vee l_k \vee l_l)$, on construit un gadget à $2p + 11$ sommets. Un sommet c_{ia} est relié aux sommets l_j et l_k et un sommet c_{ib} est relié aux sommets l_k et l_l. Les sommets c_{ia} et c_{ib} sont chacun reliés à $p + 1$ chemins de longueur 3 dont l'extrémité opposée est en feu. Les sommets c_{ia}, c_{ib}, l_j, l_k et l_l font partie des sommets à sauver. Une illustration de ce gadget est proposée figure 8.14.

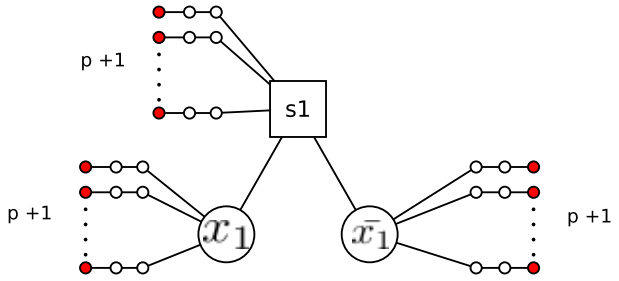


FIGURE 8.13 – Gadget pour la variable x_1. Les sommets initialement en feu sont rouges, les sommets à sauver sont carrés.

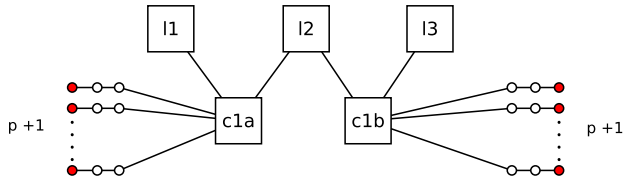


FIGURE 8.14 – Gadget pour la clause $c_1 = (l_1 \vee l_2 \vee l_3)$. Les sommets initialement en feu sont rouges, les sommets à sauver sont carrés.

Enfin, pour chaque variable x_i, le sommet x_i (respectivement $\bar{x_i}$) du gadget correspondant est relié à tous les sommets l_j tels que $l_j = x_i$ (respectivement tels que $l_j = \bar{x_i}$) des gadgets de clause. Remarquons que le graphe ainsi construit est de taille polynomiale par rapport à l'instance de 3-SAT, biparti, et que le feu n'atteindra les sommets de degré supérieur à 2 qu'au tour 3.

Le graphe associé à la formule 3-SAT est par définition planaire. Pour que le graphe construit ici le soit, il suffit de montrer qu'il est possible de séparer les sommets de variables en deux sommets de littéraux connectés et les sommets de clauses en trois sommets de littéraux connectés sans affecter la planarité. La subdivision d'arêtes ou l'ajout de chemins connectés à un seul sommet n'ayant pas d'influence sur la planarité. Nous montrons dans les figures 8.15 et 8.16 que quelque soit la position relative des sommets, il est possible de séparer les sommets de variables puis les sommets de clauses de manière planaire.

Supposons qu'il existe A une affectation sur les variables de X vérifiant C. Pour chaque variable x_i fixée à **vrai**, on déploie deux pompiers voisins en s_i et x_i dans le gadget de variable correspondant. Pour chaque variable x_i fixée à **faux**, on déploie deux pompiers voisins en s_i et $\bar{x_i}$ dans le gadget de variable correspondant. Ainsi, pour chaque variable, le sommet s_i est protégé et donc sauvé.

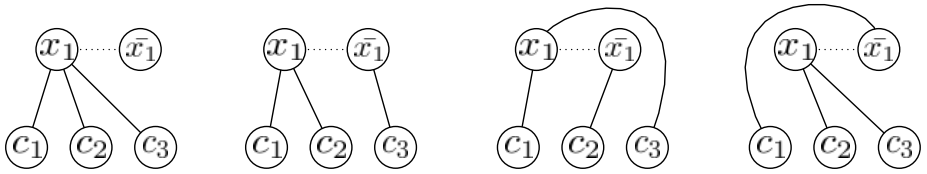


FIGURE 8.15 – Quelle que soient les positions relatives des sommets, il est possible de séparer les sommets de variable et les relier aux sommets de clause de manière planaire.

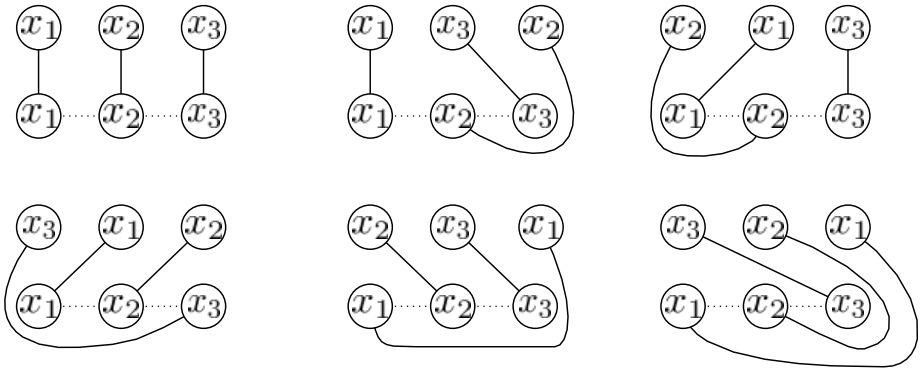


FIGURE 8.16 – Quelle que soient les positions relatives des sommets, il est possible de séparer les sommets de clause et les relier aux sommets de littéraux de manière planaire.

Chaque clause $c_i = (l_j \vee l_k \vee l_l)$ est vérifiée. Supposons sans perte de généralité que le littéral l_j soit **vrai**. Alors on déploie deux paires de pompiers voisins en c_{ia}, l_k et c_{ib}, l_l dans le gadget de clause correspondant. Les sommets c_{ia}, c_{ib}, l_k et l_l sont protégés et donc sauvés. Le sommet l_j a deux voisins : c_{ia}, qui est protégé, et un sommet x_i tel que $x_i = l_j$ dans un gadget de variable. Or, par hypothèse, ce littéral est **vrai**, le sommet du gadget de variable correspondant est donc protégé : l_j est sauvé.

Ainsi, tous les sommets de S sont sauvés en 1 tour. Les pompiers sont toujours déployés par paires. Le nombre total de pompiers déployés est de 4 par clause et 2 par variable, ce qui corresponds au nombre de pompiers disponibles.

Supposons maintenant qu'il soit possible de sauver les sommets de S en un tour en déployant au plus p pompiers, par paires. On construit A l'affectation suivante sur les variables de X : une variable x_i est fixée à **vrai** si le sommet x_i du gadget de variable correspondant est protégé, et à **faux** sinon.

Remarquons que chaque sommet s_i, c_{ia}, c_{ib} doit être sauvé. Chacun de ces sommets est relié à $p + 1$ chemins dont l'extrémité est en feu : le nombre de pompiers étant de p, il est impossible de sauver ces sommets en un tour sans les protéger. Ainsi, tous ces sommets seront protégés. Notons que ceci nécessite de déployer $p/2$ pompiers, dont aucun ne sont sur des sommets voisins, les pompiers restants devront donc être déployés pour "compléter" les paires : chaque pompier devra donc être déployé à coté d'un pompier isolé. Ainsi, pour chaque gadget de variable, les sommets x_i et $\bar{x_i}$ n'ont qu'un voisin s_i sur lequel est un pompier isolé : un seul d'entre eux pourra être protégé.

Soit $c_i = (l_j \vee l_k \vee l_l)$ une clause. Les sommets l_j, l_k et l_l du gadget de clause correspondant n'ont que deux voisins (c_{ia} et c_{ib}) sur lesquels des pompiers isolés ont été déployés. Au moins un d'entre eux ne peut donc être protégé : supposons sans perte de généralité que ce soit l_j. Par construction, le sommet l_j du gadget de clause est voisin d'un sommet x_i (ou $\bar{x_i}$) d'un gadget de variable tel que $x_i = l_j$ (ou $\bar{x_i} = l_j$) : ce sommet est lui même connecté à $p + 1$ chemins en feu : il est donc protégé, sinon l_j ne pourrait être sauvé. Si l_j est un littéral positif, alors le sommet correspondant x_i est protégé, et par construction x_i est fixé à **vrai** : la clause est donc vérifiée. Si l_j est un littéral négatif, alors $\bar{x_i}$ est protégé, ce qui implique que x_i ne l'est pas, par construction, x_i est fixé à **faux** et la clause est vérifiée.

Ainsi, toutes les clauses sont vérifiées. □

8.2.2 Est-il possible de sauver les sommets de S en deux tours ?

Théorème 68. *A vitesse* 1*, il est NP-complet de déterminer s'il est possible de sauver les sommets de* S *en deux tours même si* G *est biparti planaire, et que le feu n'atteint les sommets de degré supérieur à* 2 *et les sommets à protéger qu'au tour* 3*.*

Démonstration. Soit (X, C) une instance de planar 3-SAT. Construisons une instance de firefighter telle qu'il existe une affectation des variables de X vérifiant C si et seulement si il est possible de sauver les sommets de S en deux tours à vitesse 1. L'instance construite est identique à celle construite dans la preuve du théorème 67, à l'exception du nombre de pompiers par tour qui sera $k = 2|X| + 4|C|$.

Supposons qu'il existe A une affectation des variables de X vérifiant les clauses de C. Soit P l'ensemble de k paires de pompiers déployées dans la preuve du théorème 67. On remarque qu'aucun pompier n'est déployé proche du feu. Au tour 1, on protège un sommet arbitraire de chaque paire en utilisant k pompiers. Au tour 2, chaque pompier protège le second sommet de la paire (ce sommet n'est pas brûlé car il n'était pas proche du feu), ainsi, tous les sommets de S sont protégés en 2 tours.

Supposons maintenant qu'il soit possible de protéger tous les sommets de S en deux tours avec k pompiers à vitesse 1. Alors, chaque pompier protège deux sommets voisins, il est donc possible de protéger les mêmes sommets au tour 1 avec $2k$ pompiers déployés par paires. Ainsi, d'après la preuve du théorème 67, il existe une affectation des variables de X vérifiant les clauses de C. □

Nous montrons maintenant que cette classe d'instances est polynomiale sans contraintes de déplacement.

Théorème 69. *Sans contraintes de déplacement, il est possible de déterminer en temps polynomial s'il est possible de sauver les sommets de* S *en deux tours si le feu n'atteint les sommets de degré supérieur à* 2 *et les sommets à protéger qu'au tour* 3*.*

Démonstration. Nous avons montré dans le théorème 66 qu'il était polynomial de déterminer s'il était possible de sauver les sommets à protéger en un tour.

Soit (G, F, S, p, ∞) une instance de firefighter sans contraintes de déplacement tel que le feu n'atteint les sommets de degré supérieur à 2 qu'au tour 3. Construisons $(G, F \cup N(F), S, 2p, \infty)$ l'instance comportant deux fois plus de pompiers et ou le feu s'est propagé pendant 1 tour. Il est polynomial de

déterminer s'il est possible d'arrêter le feu en un tour avec $2p$ pompiers dans cette nouvelle instance.

Supposons qu'il soit possible de sauver les sommets de S en un tour dans la nouvelle instance en utilisant $2p$ pompiers, joués sur un ensemble de sommets X. Il est alors possible de sauver les sommets de S en deux tours en utilisant p pompiers par tour dans la première instance. Pour cela, au tour 1, p pompiers sont joués sur des sommets arbitraires de X. Au second tour, p pompiers sont joués sur les sommets restant de X. Ceux-ci ne sont pas en feu car $X \cup N(F) = \emptyset$. Les sommets de S sont donc ainsi protégés en 2 tours.

Supposons maintenant qu'il soit possible de sauver les sommets de S en deux tours dans l'instance initiale en utilisant p pompiers par tour. Soit X l'ensemble des $2p$ positions des pompiers, soit X_f le sous ensemble de X voisin de F, correspondant aux sommets voisins du feu et V_n le sous ensemble de $V - F$ de sommets voisins de X_f. Par construction $|V_n| \leq |X_f|$, ainsi, $|X - X_f| \cup |V_n| \leq 2p$. De plus, tout chemin d'un sommet de F à un sommet de S passant par un sommet u de X_f passe également par un sommet v de V_n. Ainsi, il est possible de sauver les sommets de S en un tour dans $(G, N(F), S, 2p, \infty)$ en protégeant les sommets de $X - X_f \cup V_n$. □

8.3 Conclusion du chapitre et perspectives de recherche

Dans ce chapitre, nous avons étudié des versions sur-contraintes du problème du firefighter, correspondant à des contraintes sur le déploiement des pompiers, pour modéliser, par exemple, des déplacements limités.

Dans le cas des graphes finis, nous nous sommes concentrés sur des questions consistant à arrêter le feu en un nombre très faible de tours (un ou deux). Nous avons montré que déterminer si le feu était stoppable en un tour était facile dans le cas non contraint, mais NP-complet si l'on impose aux pompier de travailler par paires (pour ne pas laisser de pompier isolé, pour des raisons de sécurité). Avec une vitesse de déplacement limitée (vitesse 1), il est NP-complet de déterminer s'il est possible d'arrêter le feu en 2 tours, même dans une classe particulière de graphes planaires. Il est au contraire polynomial de déterminer s'il est possible d'arrêter le feu en deux tours sans contraintes de déplacement dans cette même classe d'instances particulières.

Pour compléter ces travaux dans les graphes finis, il serait intéressant d'obtenir un résultat de complexité dans le cas général, pour le problème de déterminer s'il est possible de stopper le feu en deux tours, sans contraintes de déplacement. Il serait également intéressant de travailler dans un graphe

ou les arêtes n'ont pas des longueurs unitaires. Enfin, il est envisageable d'étudier des graphes particuliers, comme par exemple les arbres.

Dans le cas de grilles infinies, nous avons proposé des stratégies visant à contenir le feu avec un nombre limité de pompiers sujets à des contraintes de déplacement. Nos stratégies utilisent en général un nombre de pompiers plus élevé que les stratégies n'imposant pas de limite au déplacement d'un pompier entre deux tours : ce résultat n'est pas surprenant. Cependant, pour la plupart des stratégies proposées, nous n'avons pas prouvé que le nombre de pompier utilisés était minimal : il serait intéressant de le faire pour compléter les résultats obtenus, ou, le cas échéant, proposer des stratégies utilisant moins de pompiers.

Notre contrainte de vitesse de déplacement vise à se rapprocher d'un modèle (un peu) plus réaliste des problèmes, cependant, au tour 1, les pompiers peuvent être déployés n'importe où : une prochaine étape pourrait être d'imposer la position initiale des pompiers (celle-ci correspondant par exemple aux positions des casernes de pompiers). Ceci créerait une plus grande diversité des instances : dans la grille infinie sans position initiale des pompiers, il est possible de ramener tout feu à un feu carré d'une certaine taille. Ceci n'est plus possible quand la position initiale des pompiers est imposée : en effet, si ces positions initiales sont encerclées par le feu, les pompiers ne pourront pas le contenir. Une question préliminaire apparaît : les pompiers peuvent-ils échapper au feu ?

Dans nos problèmes, tous les pompiers avaient une même vitesse, constante. Il est possible d'avoir des pompiers se déplaçant à des vitesses différentes, ou ayant deux vitesses de déplacement (une vitesse de "travail", ou le pompier peut se déplacer de manière limitée et protéger un sommet, et une vitesse de "déplacement" ou le pompier peut se déplacer beaucoup plus loin, mais ne peut pas protéger de sommet à ce tour). De nombreuses variations de la modélisation sont envisageables.

Chapitre 9

Conclusion générale

L'objectif de cette thèse était d'étudier l'effet de plusieurs types de contraintes additionnelles sur la complexité de problèmes combinatoires classiques.

Trois types de contraintes ont été étudiés pour les problèmes de graphes : les conflits, la connexité, et les obligations ; ainsi que des contraintes de déplacement pour le problème du firefighter.

Les conflits sont les contraintes ayant augmenté de manière la plus drastique la complexité des problèmes étudiés. En effet, à l'exception du problème du vertex cover sans conflit, il est devenu NP-complet de déterminer l'existence d'une solution sans conflit pour l'ensemble des problèmes de graphes concernés. Pour les problèmes de domination, la contrainte "sans conflit" est tellement forte qu'elle "écrase" complètement la structure du graphe support : déterminer l'existence d'une solution sans conflit est NP-complet même dans des graphes supports triviaux, comme par exemple des chemins. Déterminer l'existence d'une solution de base n'étant pas possible en temps polynomial, la question de l'optimisation de ces solutions ne s'est pas posée. Nous nous sommes par la suite intéressés à des solutions dégradées, en cherchant des dominants partiels : ce problème a été étudié pour une classe de conflits restreinte : les couplages parfaits. Ceci a permis d'obtenir quelques résultats d'approximation à rapport constant, et nous avons proposé des algorithmes dominant au moins la moitié des sommets. Ces résultats sont toutefois assez faibles, mais nous avons montré qu'il est NP-complet de déterminer s'il est possible de dominer plus de la moitié des sommets, une amélioration semble donc peu probable. De plus, cette approximation n'est possible que grâce au nombre réduit de conflits de ces instances, et n'est pas généralisable. De manière quelque peu étonnante, les langages réguliers avec conflits (sur les états ou les symboles) restent réguliers. Toutefois, la taille des automates minimaux les reconnaissant augmente exponentiellement.

Nous avons vu dans une autre partie qu'il était facile de déterminer l'existence d'une solution du vertex cover, connexe dans un autre graphe. Sous réserve de la correction des résultats de Slavik, il existe un algorithme d'approximation à rapport constant pour ce problème. Celui-ci est cependant complexe à mettre en œuvre. Nous avons proposé divers algorithmes d'approximation à rapport constant pour différents cas d'inclusions entre le graphe support et le graphe des conflits.

Parmi les problèmes avec obligations étudiés, certains, comme le problème du chemin hamiltonien dans les graphes complets, le problème de l'arbre couvrant, ou du dominant indépendant ne permettent pas de déterminer l'existence d'une solution en temps polynomial. Pour d'autres, comme le problème du vertex cover connexe, du graphe couvrant connexe ou du couplage de taille maximal, déterminer l'existence d'une solution est facile, mais il est NP-complet d'approcher une solution optimale avec un rapport constant, contrairement au problème classique. Les problèmes de dominant et dominant total sont approchables avec un rapport logarithmique proche de celui du problème classique. Seul le problème du vertex cover avec obligation garde un rapport d'approximation identique et constant de 2.

Dans le problème du firefighter, l'ajout de contraintes de déplacement sur les pompiers n'a pas fondamentalement changé la nature du problème comme l'ont pu faire les conflits sur des problèmes de domination. L'ajout de ces contraintes a dans certains cas augmenté la complexité de déterminer s'il était possible de stopper le feu rapidement dans des graphes finis, et, dans le cas des grilles infinies, augmente le nombre de pompiers nécessaires pour contenir le feu, mais les changements ne sont pas aussi drastiques que pour les autres problèmes étudiés. Par exemple, il reste toujours possible de contenir tout feu fini avec un nombre constant de pompiers, quelle que soit la vitesse (non nulle) de déplacement des pompiers.

Perspectives À court terme, peu de perspectives semblent encore ouvertes pour les problèmes avec conflits tant ceux-ci sont difficiles. La voie des solutions partielles pourrait toutefois être explorée à l'aide d'heuristiques, ainsi que d'algorithmes probabilistes.

Il serait également intéressant de trouver un algorithme de graphe permettant d'obtenir un rapport constant pour le problème du vertex cover, connexe dans un autre graphe dans le cas général, pour unir les résultats obtenus pour les cas particuliers.

Enfin, les résultats sur les problèmes avec obligations pourraient être raffinés. Il serait intéressant de déterminer si le problème du graphe couvrant connexe avec obligations peut être approché par un rapport constant dans le cas de pondérations uniformes.

À plus long terme, la contrainte “connexe dans un autre graphe” pourrait être étendue et étudiée pour d'autres problèmes classiques, comme les problèmes de domination.

Le travail effectué sur le problème du firefighter n'étant pas encore abouti, les perspectives de recherche à plus ou moins long terme sont nombreuses et détaillées dans le chapitre 8.

Bibliographie

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity : a modern approach*. Cambridge University Press, 2009.

[ACG+12] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation : Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[APT79] Bengt Aspvall, Michael F Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3) :121–123, 1979.

[AWF02] K. Alzoubi, P.-J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9*, HICSS '02, pages 297–, Washington, DC, USA, 2002. IEEE Computer Society.

[Bal85] Alexandru T Balaban. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3) :334–343, 1985.

[CC04] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems. In Susanne Albers and Tomasz Radzik, editors, *Algorithms – ESA 2004*, pages 192–203, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Chv79] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3) :233–235, 1979.

[CL16] Alexis Cornet and Christian Laforest. Note : Regular languages with no conflicts (forbidden pairs) are regular but have exponential size DFA. Research report, LIMOS (UMR CNRS 6158), université Clermont Auvergne, France , November 2016.

[CL17] Alexis Cornet and Christian Laforest. Total Domination, Connected Vertex Cover and Steiner Tree with Conflicts. *Discrete Mathematics and Theoretical Computer Science*, Vol. 19 no. 3, December 2017.

[CL18a] Alexis Cornet and Christian Laforest. Domination problems with no conflicts. *Discrete Applied Mathematics*, 244 :78 – 88, 2018.

[CL18b] Alexis Cornet and Christian Laforest. Graph problems with obligations. Research report, LIMOS (UMR CNRS 6158), université Clermont Auvergne, France, May 2018.

[CL18c] Alexis Cornet and Christian Laforest. Problèmes de domination avec conflits dans les graphes planaires. In *19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, 2018.

[DH07] Mike Develin and Stephen G Hartke. Fire containment in grids of dimension three and higher. *Discrete Applied Mathematics*, 155(17) :2257–2268, 2007.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[DLP15] Francois Delbot, Christian Laforest, and Raksmey Phan. Graphs with Forbidden and Required Vertices. In *ALGOTEL 2015 - 17èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Beaune, France, June 2015.

[DLP16] Francois Delbot, Christian Laforest, and Raksmey Phan. Hardness Results and Approximation Algorithms for Discrete Optimization Problems with Conditional and Unconditional Forbidden Vertices. Research report, HAL, https ://hal.archives-ouvertes.fr/hal-01257820, January 2016.

[FF56] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3) :399–404, 1956.

[FM09] Stephen Finbow and Gary MacGillivray. The firefighter problem : a survey of results, directions and questions. *Australasian Journal of Combinatorics*, 43(57-77) :6, 2009.

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*. wh freeman New York, 2002.

[GMO76] Harold N Gabow, Shachindra N Maheshwari, and Leon J Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, (3) :227–231, 1976.

[Har95] B Hartnell. Firefighter. In *An application of domination. Presentation at the 25th Manitoba Conference on Combinatorial Mathematics and Computing, University of Manitoba, Winnipeg, Canada*, 1995.

[HHS98] Teresa W Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

[KLM13a] Mamadou Moustapha Kanté, Christian Laforest, and Benjamin Momège. An exact algorithm to check the existence of (elementary) paths and a generalisation of the cut problem in graphs with forbidden transitions. In *SOFSEM 2013*, volume LNCS 7741, pages 257–267, 2013.

[KLM13b] Mamadou Moustapha Kanté, Christian Laforest, and Benjamin Momège. Trees in graphs with conflict edges or forbidden transitions. In *TAMC 2013, Hong Kong*, volume LNCS 7876, pages 343–354, 2013.

[KMMN15] Mamadou Moustapha Kanté, Fatima Zahra Moataz, Benjamin Momège, and Nicolas Nisse. Finding paths in grids with forbidden transitions. In *Graph-Theoretic Concepts in Computer Science - 41st International Workshop, WG 2015*, volume LNCS 9224, pages 154–168, 2015.

[Kov13] Jakub Kováč. Complexity of the path avoiding forbidden pairs problem revisited. *Discrete Applied Mathematics*, 161(10) :1506–1512, 2013.

[KP09] Petr Kolman and Ondřej Pangrác. On the complexity of paths avoiding forbidden pairs. *Discrete Applied Mathematics*, 157(13) :2871–2876, 2009.

[Linon] Peter Linz. *An introduction to formal languauges and automata*. Jones and Barrlett, 2006 (fourth edition).

[LM14] Christian Laforest and Benjamin Momège. Some hamiltonian properties of one-conflict graphs. In *IWOCA 2014*, volume LNCS 8986, pages 262–273, 2014.

[LM15] Christian Laforest and Benjamin Momège. Nash-Williams-type and Chvátal-type conditions in one-conflict graphs. In *SOFSEM 2015*, volume LNCS 8939, pages 327–338, 2015.

[Mes11] Margaret-Ellen Messinger. Firefighting on the strong grid. *Preprint*, 2011.

[Mom15] Benjamin Momège. *Autour de la connexité dans les graphes avec conflits. (On the Connectivity of Graphs with Conflicts)*. PhD thesis, Blaise Pascal University, Clermont-Ferrand, France, 2015.

[NR08] K.L. Ng and P. Raff. A generalization of the firefighter problem on zxz. *Discrete Applied Mathematics*, 156(5) :730 – 745, 2008.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

[RZ00] Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *SODA*, pages 770–779. Citeseer, 2000.

[Sav82] Carla Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14(5) :233 – 235, 1982.

[Sla97] Petr Slavik. Errand scheduling problem. Technical report, 1997.

[Sla98] Petr Slavik. *Approximation Algorithms for Set Cover and Related Problems*. PhD thesis, Buffalo, NY, USA, 1998. AAI9833643.

[SRS08] Christian Scheideler, Andrea Richa, and Paolo Santi. An o(log n) dominating set protocol for wireless ad-hoc networks under the physical interference model. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pages 91–100, New York, NY, USA, 2008. ACM.

[TM16] Simon Tippenhauer and Wolfgang Muzler. *On Planar 3-SAT and its Variants*. PhD thesis, Fachbereich Mathematik und Informatik der Freien Universitat Berlin, 2016.

[Tov84] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1) :85 – 89, 1984.

[WCDY08] Jie Wu, M. Cardei, Fei Dai, and Shuhui Yang. Extended dominating set and its applications in ad hoc networks using cooperative communication. *IEEE Transactions on Parallel and Distributed Systems*, 17(8) :851–864, Aug. 2008.

[WM02] Ping Wang and Stephanie A Moeller. Fire control on graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 41 :19–34, 2002.

[Yin97] Hananya Yinnone. On paths avoiding forbidden pairs of vertices in a graph. *Discrete Applied Mathematics*, 74(1) :85–92, 1997.