L'indépendant faiblement connexe : études algorithmiques et polyédrales

Djelloul Mameri

N° d'ordre : D. U : 2513
E D S P I C : 673

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

Djelloul MAMERI

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPÉCIALITÉ : INFORMATIQUE

Titre de la thèse :

L'indépendant faiblement connexe : Études algorithmiques et polyédrales

Soutenue publiquement le 25 novembre 2014 devant le jury :

M^{me}. Fatiha BENDALI-MAILFERT	Directrice de thèse
M. Jean MAILFERT	Encadrant
M. Alain QUILLIOT	Président
M. Arnaud PÊCHER	Rapporteur et examinateur
M. Ali Ridha MAHJOUB	Rapporteur et examinateur
M. Mohamed DIDI BIHA	Examinateur
M. Alexandre GUITTON	Invité

Remerciements

En tout premier lieu, je tiens à remercier ma directrice de thèse Madame Fatiha Bendali-mailfert et mon encadrant Monsieur Jean Mailfert pour la confiance qu'ils m'ont accordée durant ces quatre années de thèse. Leur disponibilité et leur constant soutien m'ont permis de mener à bien ces travaux. Ils ont su me transmettre leur passion pour la recherche et l'enseignement en recherche opérationnelle, en mathématiques et en informatique. Pour tout cela, je leur témoigne ma plus sincère reconnaissance.

Je tiens à remercier les rapporteurs et membres du jury. Je remercie le professeur Ali Ridha Mahjoub, rapporteur du mémoire, qui a suivi ce travail depuis ses débuts en participant au premier comité de thèse. Je remercie le professeur Arnaud Pêcher, rapporteur du mémoire, pour sa patience et l'intérêt qu'il a montré pour notre sujet. Je remercie également le professeur Alain Quilliot pour avoir accepté de présider le jury de cette thèse. Je remercie aussi le professeur Mohamed Didi Biha d'avoir bien voulu examiner mes travaux et participer au jury. Je remercie enfin Monsieur Alexandre Guitton qui m'a fait l'honneur de participer à ma soutenance. Je remercie à cette occasion le professeur Francisco Barahona qui a assisté au deuxième comité de thèse.

Je remercie tout particulièrement mes amis et collègues de bureau Marie, Thomas, Sahar, Marina, Ibrahim, Lakhdar, Yahya, Karima, Kawther, Saber, Wajdi, Slim, Rahimeh, Benjamin, Maxime, Salsabil, Diyé, Heitor, Raksmey, Libo, Nathalie, Philippe, Vincent et Hervé, grâce à qui travailler et faire de la recherche est un bonheur. Je désire remercier mon amie Kahina pour les bons moments partagés durant toutes ces années. Plus généralement, je souhaite remercier tout le personnel du Limos et de l'Isima. Je tiens également à remercier tous mes amis de l'association WorldTop.

Je désire aussi témoigner toute ma reconnaissance envers mes enseignants de l'Université de Béjaia, en particulier les membres du département de recherche opérationnelle. Je remercie spécialement Rachid Djijli professeur de mathématiques au Lycée de Tassift qui m'a donné l'envie et le plaisir de faire des mathématiques. Mes remerciements vont vers mes ami(e)s d'Algérie, Rachid, Sonia, Walid, Assia, Djamel, Salem, Soufiane, Samir, Mohand, Farid, Mouloude, Selim, Khelef, Nassir, Yazid, Hamide, Nadir et mon ami Xin de Chine.

Enfin, je tiens à remercier ma famille en Algérie qui m'a encouragé durant toutes ces années. Cela n'a pas dû être facile tous les jours, mais ils ont toujours été là pour me soutenir. Et un grand merci pour ma femme Lynda de l'avoir à côté de moi, pour toujours.

Résumé

Dans ce travail, nous nous intéressons à une topologie pour les réseaux de capteurs sans fil. Un réseau de capteurs sans fil peut être modélisé comme un graphe non orienté $G = (V, E)$. Chaque sommet de V représente un capteur et une arête $e = \{u, v\}$ dans E indique une transmission directe possible entre deux capteurs u et v. Contrairement aux dispositifs filaires, les capteurs sans fil ne sont pas a priori agencé en réseau. Une topologie doit être créée en sélectionnant des nœuds "dominants" qui vont gérer les transmissions. Les architectures qui ont été examinées dans la littérature reposent essentiellement sur les ensembles dominants connexes et les ensembles dominants faiblement connexes.

Cette étude est consacrée aux ensembles *indépendants faiblement connexes*. Un indépendant $S \subset V$ est dit faiblement connexe si le graphe $G_S = (V, [S, V \backslash S])$ est connexe, où $[S, V \backslash S]$ est l'ensemble des arêtes $e = \{u, v\}$ de E avec $u \in S$ et $v \in V \backslash S$. Une topologie basée sur les ensembles faiblement connexes permet de partitionner l'ensemble des capteurs en trois groupes, les esclaves, les maîtres et les intermédiaires. Les premiers effectuent les mesures, les seconds rassemblent les données collectées et les troisièmes assurent les communications inter-groupes.

Nous donnons d'abord quelques propriétés de cette structure combinatoire lorsque le graphe non orienté G est connexe. Puis nous proposons des résultats de complexité pour le problème de la recherche de l'indépendant faiblement connexe de cardinalité minimale (MWCISP).
Nous décrivons également un algorithme d'énumération exact de complexité $O^*(1.4655^{|V|})$ pour le MWCISP. Des tests numériques de cette procédure exacte sont présentés.

Nous formulons ensuite le $MWCISP$ comme un programme linéaire en nombres entiers. Le polytope associé aux solutions de ce problème est complètement caractérisé lorsque G est un cycle impair. Nous étudions des opérations de composition de graphes et leurs conséquences polyédrales. Nous introduisons des inégalités valides notamment les contraintes dites de multibord. Par la suite, nous développons un algorithme de coupes et branchement sous CPLEX pour résoudre ce problème en utilisant des heuristiques pour la séparation de nos familles de contraintes. Des résultats expérimentaux de ce programme sont exposés.

Mots clés : ensemble dominants, indépendant faiblement connexe, réseaux sans fil, algorithme d'énumération exact, polytope, algorithme de coupes et branchement.

Abstract

In this work, we focus on a topology for Wireless Sensor Networks (WSN). A wireless sensor network can be modeled as an undirected graph $G = (V, E)$. Each vertex of V represents a sensor and an edge $e = \{u, v\}$ in E implies a direct transmission between the two sensors u and v. Unlike wired devices, wireless sensors are not a priori arranged in a network. Topology should be made by selecting some sensor as dominators nodes who manage transmissions. Architectures that have been studied in the literature are mainly based on connected dominating sets and weakly connected dominating sets.

This study is devoted to weakly connected independent sets. An independent set $S \subset V$ is said Weakly Connected if the graph $G_S = (V, [S, V \backslash S])$ is connected, where $[S, V \backslash S]$ is the set of edges with exactly one end in S. A sensor network topology based on weakly connected sets is partition into three groups, slaves, masters and bridges. The first performs the measurements, the second gathers the collected data and the later provides the inter-group communications.

We first give some properties of this combinatorial structure when the undirected graph G is connected. Then we provide complexity results for the problem of finding the minimum weakly connected independent set problem (MWCISP). We also describe an exact enumeration algorithm of complexity $O^*(1.4655^{|V|})$ (for the (MWCISP)). Numerical tests of this exact procedure are also presented.

We then present an integer programming formulation for the minimum weakly connected independent set problem and discuss its associated polytope. Some classical graph operations are also used for defining new polyhedra from pieces. We give valid inequalities and describe heuristical separation algorithms for them. Finally, we develop a branch-and-cut algorithm and test it on two classes of graphs.

Keywords : dominating set, weakly connected independent set, wireless networks, exact algorithm, branch-and-cut algorithm.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

De nombreuses applications civiles et militaires utilisent des réseaux de capteurs sans fil [1, 28]. Les capteurs peuvent être déployés pour recueillir des mesures météorologiques telles que la température et la pression. Ils peuvent aussi détecter des catastrophes naturelles comme les tremblements de terre et aider les unités d'intervention d'urgence à retrouver les survivants.

Un réseau de capteurs sans fil (WSN) est généralement constitué d'un ensemble de composants autonomes qui collectent des données et diffusent des messages vers une station de base. Chaque appareil est doté d'une mémoire et d'une capacité de calcul limitées et fonctionne à l'aide d'une batterie. Les communications sont réalisées via une bande passante partagée, directement si les appareils sont assez proches ou par des relais assurés par des capteurs intermédiaires. La Figure 1.1 nous donne une représentation générale d'un réseau de capteurs sans fil.

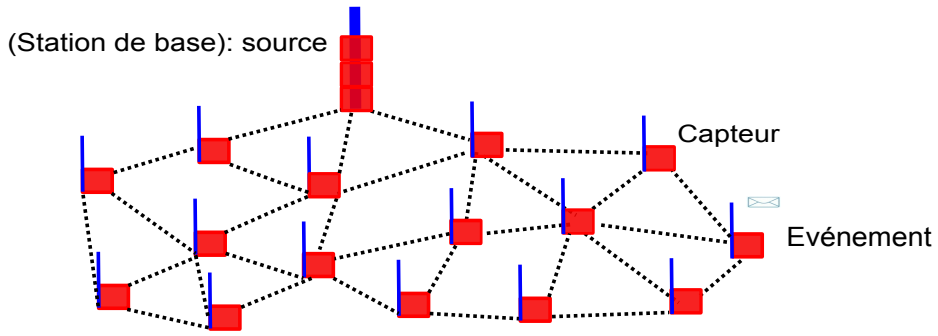


Figure 1.1 – Réseau de capteurs sans fil.

Comme il n'existe aucune infrastructure physique, contrairement aux réseaux filaires, une topologie virtuelle doit être créée. Si l'environnement observé est bien maillé par les détecteurs, les données relevées seront fortement corrélées. La formation de "clusters" (groupes) d'appareils permet alors de centraliser des mesures voisines en un seul message et de diminuer ainsi la fréquence des communications. Il est donc important d'organiser l'ensemble des capteurs dans le but de sélectionner des éléments "maîtres" qui rassemblent l'information collectée par les capteurs "esclaves" et qui la transmettent, en utilisant éventuellement d'autres capteurs "intermédiaires", vers la station de base où elles seront traitées et interprétées par l'utilisateur. Dans ce travail, nous étudions une topologie générant un "clustering" (répartition) des

appareils dans des ensembles ayant des rôles différents.

Un réseau de capteurs sans fil peut être modélisé par un graphe non orienté $G = (V, E)$ [15, 28]. Chaque sommet de V représente un capteur situé dans la région contrôlée et une arête $e = \{u, v\}$ de E symbolise une communication possible entre deux capteurs u et v. Ce lien dépend en général de la distance euclidienne entre u et v qui influe sur l'énergie à déployer pour établir cette connexion. Le graphe G des communications correspond, en première approximation, à un graphe d'intersection de disques dans le plan euclidien [15]. La Figure 1.2 nous montre la manière de le construire en connaissant la position des capteurs sur le plan euclidien et leur puissance P d'émission. L'envoi d'une série de messages HELLO permet à chaque nœud de connaître son voisinage physique. La connexité de G est indispensable pour acheminer toutes les informations collectées par les capteurs vers la station de base.

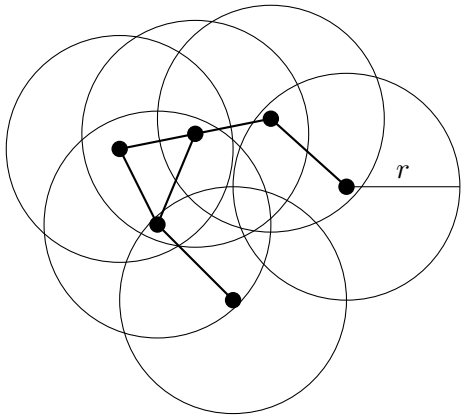


FIGURE 1.2 – Graphe des communications avec un rayon r fonction de P.

Les *ensembles dominants connexes* [36] ont été proposées comme solution pour la structure virtuelle d'un réseau de capteurs par de nombreux auteurs [12, 40, 62, 65]. Un ensemble de sommets D est un *ensemble dominant connexe* noté *cds*, si chaque sommet de G est dans D ou adjacent à au moins un des sommets de D et, si le sous-graphe induit par D est connexe. Ainsi, les communications sont assurées entre tous les sommets grâce à l'ensemble D. Comme on veut réduire le nombre de messages échangés et éviter une consommation d'énergie inutile, D doit être de petite taille. Cependant, déterminer un Dominant Connexe Minimum (*mcds*) dans un graphe est NP-Difficile [29, 40]. Par conséquent, de nombreux algorithmes d'approximation et heuristiques ont été présentés pour ce problème [2, 34, 43, 59, 62]. La majorité de ces algorithmes suivent une procédure à deux phases. La première phase construit un ensemble dominant, les nœuds dans cet ensemble sont appelés dominateurs. La deuxième phase sélectionne des nœuds supplémentaires, appelés connecteurs, qui avec les dominateurs assurent la connexité. Ces algorithmes diffèrent dans la manière de sélectionner les dominateurs et les connecteurs. L'algorithme décrit dans [18] sélectionne les dominateurs en utilisant l'algorithme de Chvàtal [14] pour le problème de la couverture de sommets. Guha et Khuller [34] ont proposé un algorithme d'approximation glouton qui détermine un ensemble dominant connexe dont la cardinalité est au plus $(3 + \ln \Delta(G))$ celle d'un dominant connexe minimum, où $\Delta(G)$ est le degré maximum de G. Pour construire un *cds*, les algorithmes décrits dans [2, 59] sélectionnent initialement un indépendant maximal (*mis*) quelconque

et les algorithmes issus de [12, 43, 62, 63] choisissent un *mis* particulier comme ensemble dominant. Rappelons qu'un sous-ensemble de sommets $S \subset V$ est un indépendant ou stable de G si aucune arête de E ne relie deux sommets de S et un indépendant S est maximal (mis) si aucun indépendant de G ne le contient strictement.

Une structure plus faible que celle de l'ensemble dominant connexe, est l'*ensemble dominant faiblement connexe* ou *wcds* [11, 21]. Un ensemble dominant $S \subset V$ est dit faiblement connexe si le graphe partiel $G_S = (V, E(S) \cup [S, V \backslash S])$ est connexe. Toutefois, le problème de détermination d'un *wcds* de cardinalité minimum est aussi NP-Difficile [29]. Dans [11] les auteurs ont proposé des algorithmes d'approximation avec une garantie en $O(\ln \Delta(G))$. Dans [3], les auteurs utilisent des algorithmes qui construisent un dominant faiblement connexe à partir d'un ensemble indépendant maximal S de G en lui ajoutant des sommets de $V \setminus S$ pour réaliser un *wcds* de G. Notons qu'il est facile d'obtenir un ensemble indépendant maximal dans un graphe G. En outre, la recherche d'un ensemble indépendant dominant minimum est polynomiale pour certaines classes de graphes comme les graphes d'intervalles [10] et les graphes triangulés [25] alors que le problème est NP-difficile dans les graphes bipartis et de comparabilité [17]. Toutefois, comme cela est souligné dans [54], il semble souhaitable de conserver l'indépendance des sommets "dominateurs", tout en recherchant à connecter le réseau. A cet effet, les auteurs de [2] montrent que des *mis* particuliers peuvent être faiblement connectés. Les ensembles sont tels que les "sommets maîtres" i.e. issus du *mis* sont reliés grâce à des sommets dominés.

Les problèmes des ensembles indépendants, dominants connexes et dominants faiblement connexes ont été abordés aussi sous un angle polyédral. Beaucoup d'articles ont traité des contraintes valides pour le polytope des indépendants. Un ensemble de facettes a été décrit dans Chvátal [13], Nemhauser et Trotter [47], Padberg [48] et Trotter [61]. Dans [47, 48, 49] les auteurs prouvent que l'inégalité $\sum_{u \in K} x(u) \leq 1$ est facette pour le polytope des indépendants si et seulement si le sous-graphe induit par K est une clique maximale de G. Dans [47, 48] les auteurs montrent que l'inégalité $\sum_{u \in V} x(u) \leq \frac{(|V|-1)}{2}$ est facette pour le polytope des indépendants lorsque le graphe $G = (V, E)$ est un cycle impair. L'article [23] donne une preuve de la conjecture de Ben Rebea sur la description complète de polytope des stables dans les quasi-line graphs par les inégalités de cliques. Pêcher et Wagler [50] ont donné une famille de facettes dans les graphes sans griffes. Concernant les dominants, Bouchakour et Mahjoub [7] ont présenté des résultats sur les techniques de composition et décomposition de graphes pour le polytope des dominants. Dans [6], les auteurs ont donné une caractérisation complète de ce polytope dans un cycle. L'article [32] propose trois algorithmes exacts pour le problème du dominant connexe de cardinalité minimum. Le premier algorithme utilise la décomposition de Benders, le second est basé sur la méthode de coupes et branchement et le troisième combine les deux techniques. Dans [57], on trouve une formulation en nombres entiers pour le problème du dominant connexe de cardinalité minimum, ainsi qu'un algorithme de coupes et branchement. Saxena [55] propose enfin une famille de formulations pour les problèmes dominants, dominants connexes et dominants faiblement connexes dans un graphe. Farber [26] décrit un algorithme en $O(|V| + |E|)$ pour trouver un indépendant dominant minimum dans les graphes fortement triangulés. Dans [45] Mahjoub et Mailfert ont donné une description complète du polytope des indépendants dominants dans un cycle.

Ce mémoire s'intéresse aux indépendants faiblement connexes d'un graphe connexe $G =$

(V, E). Un indépendant S de G est dit faiblement connexe si le graphe $G_S = (V, [S, V \backslash S])$ est connexe. Nous suggérons d'utiliser en anglais l'expression *weakly connected independent set* (*wcis*) qui est plus concise que *weakly connected independent dominating set*, d'autant plus qu'un indépendant faiblement connexe est nécessairement dominant. Un tel ensemble peut servir comme topologie de base pour une architecture en blocs (clusters) d'un réseau de capteurs. En effet, les nœuds de V sont partitionnés en trois catégories. Les esclaves sont chargés de détecter des événements. Les maîtres (les sommets de l'indépendant faiblement connexe) synthétisent les données et les émettent en direction de la station de base. Enfin, les sommets intermédiaires relient les différents blocs entre eux et sous-tendent la connexité du graphe. Les communications dans le graphe G_S se font donc de la manière suivante : les esclaves échangent seulement avec leur maître et les maîtres à leur tour communiquent entre eux grâce aux sommets intermédiaires.

Sur la Figure 1.3 nous positionnons l'ensemble des parties de V, $\mathcal{P}(\mathcal{V})$, l'ensemble des ensembles indépendants $\mathcal{IS}$, l'ensemble des indépendants maximaux $\mathcal{MIS}$, l'ensemble des dominants connexes $\mathcal{CDS}$, l'ensemble des dominants faiblement connexes $\mathcal{WCDS}$ et l'ensemble des ensembles indépendants faiblement connexes de G, $\mathcal{W}(G)$. $\mathcal{W}(G)$ est l'intersection des ensembles $\mathcal{WCDS}$ et $\mathcal{MIS}$.

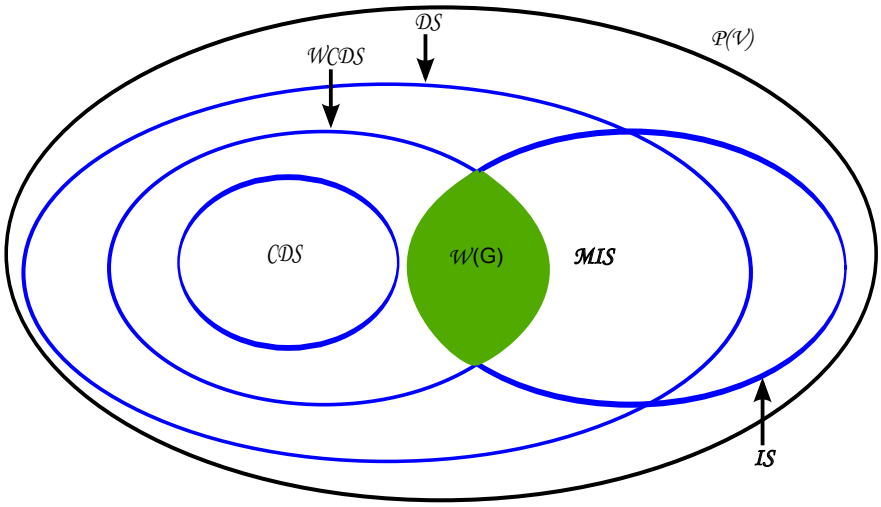


FIGURE 1.3 – Localisation de $\mathcal{W}(G)$ dans l'ensemble des parties de V.

Ce travail de thèse s'inscrit dans le cadre des méthodes d'énumération exacte et d'optimisation combinatoire polyédrale. Le chapitre 2 est consacré à la présentation des notions, définitions de la théorie des graphes et de l'optimisation combinatoire qui seront utiles tout au long de cette thèse. Dans le chapitre 3, nous définissons la structure d'indépendant faiblement connexe (*wcis*), donnons quelques-unes de ses propriétés et des résultats de complexité. Le chapitre 4 est consacré à un algorithme d'énumération exacte. Des résultats numériques sur des graphes générés aléatoirement dans le plan euclidien et sur des classes particulières de graphes sont aussi présentés.

Nous étudions ensuite l'enveloppe convexe des indépendants faiblement connexes. Une description linéaire complète de ce polytope est donnée pour les cycles impairs. Nous proposons

aussi des opérations de composition de graphes et étudions leurs conséquences polyédrales. Nous présentons par la suite un algorithme de coupes et branchement ainsi que des heuristiques de séparation des contraintes dites de bord et multi-bords. Le chapitre 6 est consacré à la présentation des résultats expérimentaux de l'algorithme de coupes et branchement.

Enfin quelques perspectives à ce travail sont exposées.

Chapitre 2

Notions préliminaires

Sommaire

Introduction

Ce chapitre est consacré aux définitions et concepts de la théorie des graphes, de la théorie de la complexité et de la théorie polyédrale utilisés tout au long du mémoire. Dans la section 2.1, nous introduisons des notations et objets de la théorie des graphes. Nous présentons ensuite des notions liées à l'algorithmique dans les sections 2.2 et 2.3. La section 2.4 est dévolue à des rappels sur les polyédres.

2.1 Théorie des graphes

2.1.1 Notations préliminaires

Soit $G = (V, E)$ un graphe non orienté où V est l'ensemble des sommets et E l'ensemble des arêtes. Le voisinage ouvert $N(v)$ d'un sommet v est l'ensemble des sommets adjacents à v.

Le voisinage fermé $N[v]$ d'un sommet v est l'ensemble $N(v) \cup \{v\}$. $\Delta(G)$, (resp. $\delta(G)$) est le degré maximum (resp. degré minimum) de G. Pour tout sous-ensemble S de V, le voisinage $N(S)$ est l'union des ensembles $N(v)\backslash S$ pour tout $v \in S$, $N(S) = (\cup_{v \in S} N(v))\backslash S$. Si $S \subset V$, notons par $E(S)$ l'ensemble des arêtes de G ayant leurs deux extrémités dans S. Soient S et $S^{'}$ deux sous-ensembles de sommets disjoints de V, $[S, S^{'}]$ est l'ensemble d'arêtes de E avec exactement une extrémité dans S et l'autre dans $S^{'}$. L'ensemble $[S, V\backslash S]$ est aussi noté $\delta(S)$.

2.1.2 Définitions et notations

Definition 2.1.1. *(chaîne élémentaire) Une chaîne μ entre deux sommets u_0 et u_k de V, est une séquence alternée de sommets et d'arêtes $u_0e_1u_1e_2u_2, ..., e_ku_k$, telle que pour $1 \leq i \leq k$, les extrémités de e_i sont u_{i-1} et u_i et les sommets $u_0, u_1, ..., u_k$ sont tous différents.*

Definition 2.1.2. *(graphe connexe) Un graphe G est connexe s'il existe une chaîne entre toute paire de sommets.*

Definition 2.1.3. *(distance) Pour deux sommets u et v dans G, la distance $d_G(u, v)$ est la plus courte chaîne de G qui connecte u et v. Si $S \subset V$ et $u \notin S$, la distance de u à S, $d_G(u, S)$ est égale à $\min_{v \in S}\{d_G(u, v)\}$.*

Definition 2.1.4. *(voisinage à distance 2) Le voisinage à distance 2 d'un sommet u, noté $N^2(u)$, est l'ensemble des sommets de V à distance 2.*

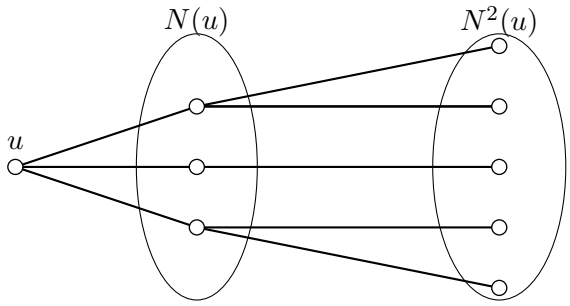


FIGURE 2.1 – u, $N(u)$ et $N^2(u)$

Definition 2.1.5. *(sous-graphe) Pour un sous-ensemble de sommets S de V, le sous-graphe de G induit par S est le graphe $G(S) = (S, E(S))$.*

Definition 2.1.6. *(sous-graphe partiel) $H = (V(H), E(H))$ est un sous-graphe partiel de G si les sommets de H forment un sous-ensemble S de V et $E(H) \subseteq E(S)$.*

Definition 2.1.7. *(graphe partiel) $G^{'} = (V(G^{'}), E^{'})$ est un graphe partiel de G si $V(G^{'}) = V$ et $E^{'} \subseteq E$. Autrement dit, on obtient $G^{'}$ en enlevant une ou plusieurs arêtes du graphe G.*

Definition 2.1.8. *(graphe complet) Un graphe est dit complet si tous les sommets sont deux à deux adjacents.*

Definition 2.1.9. *(Clique) $S \subseteq V$ forme une clique de G si $G(S)$ est complet.*

Definition 2.1.10. *(Indépendant ou stable) Un stable est un ensemble de sommets qui ne contient aucune arête de E. La taille d'un stable est égale au nombre de sommets qu'il contient.*

Definition 2.1.11. *(Dominant) Un ensemble dominant dans un graphe G est un sous-ensemble de sommets $S \subseteq V$ tel que tout sommet qui n'appartient pas à S est adjacent à un sommet de S.*

Definition 2.1.12. *(Dominant connexe) Un dominant S de G est connexe si le graphe $G(S) = (S, E(S))$ est connexe.*

Definition 2.1.13. *(Indépendant maximal ou indépendant dominant) Un indépendant S de V est maximal si aucun indépendant de V ne le contient.*

Definition 2.1.14. *(Dominant faiblement connexe) Un dominant S est faiblement connexe si le graphe $G_S = (V, E(S) \cup [S, V \backslash S])$ est connexe.*

Definition 2.1.15. *(Indépendant faiblement connexe) Un indépendant W de G tel que le graphe partiel $G_W = (V, [W, V \backslash W])$ est connexe est appelé indépendant faiblement connexe (*Weakly Connected Independent Set*).*

On remarque que un ensemble faiblement connexe est aussi un dominant du graphe.

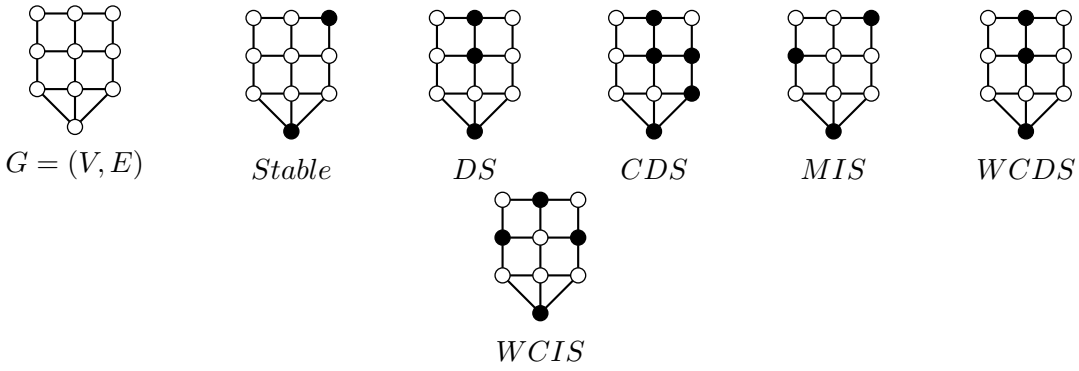


FIGURE 2.2 – Les différentes structures.

Definition 2.1.16. *(Point d'articulation) Un point d'articulation dans un graphe connexe G est un sommet dont la suppression déconnecte G.*

Definition 2.1.17. *(Ensemble d'articulation) Dans un graphe connexe $G = (V, E)$, un ensemble d'articulation $U \subset V$ est un sous-ensemble de sommets de V tel que le graphe $G(V \backslash U) = (V \backslash U, E(V \backslash U))$ est non connexe.*

Definition 2.1.18. *(Jumeau) Soient x, y dans V. x et y sont jumeaux si $N(x) = N(y)$. Si $\{x, y\} \in E$, alors x et y sont des* vrais jumeaux.

Definition 2.1.19. *(Module d'un graphe) Un module d'un graphe est un ensemble de sommets $M \subseteq V$ non vide tel que pour chaque sommet v n'appartenant pas à M, tous les sommets de M sont adjacents à v, ou aucun sommet de M n'est adjacent à v.*

Une présentation générale sur la théorie des graphes se trouve dans [5].

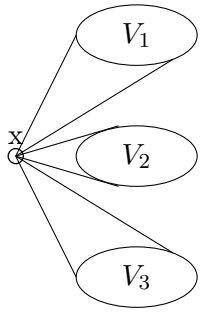


FIGURE 2.3 – $G\backslash\{x\}$ a trois composantes connexes.

2.2 Complexité

Les premiers travaux sur la théorie de la complexité ont été faits par Edmonds [22] et Cook [16]. L'idée principale de cette théorie est de classer les problèmes selon leur difficulté. Dans cette section, nous rappelons quelques définitions de base sur la théorie de la complexité.

2.2.1 Taille des données

Pour mesurer les ressources critiques (temps processeur et espace mémoire) utilisées par un algorithme, nous allons évaluer les performances d'un algorithme en fonction de la taille des données. Par exemple la taille des données d'un graphe est une fonction du nombre de sommets dans le graphe et du nombre de ses arêtes. La taille d'une donnée correspond au codage en mémoire de cette donnée exprimée en bits. Par exemple : un entier A se code en $\log_2(A)$ bits. Pour un exposé plus complet, on peut se référer à Michael R. Garey et David S. Johnson [30].

2.2.2 Problème, instance d'un problème et problème de décision

Definition 2.2.1. *(Problème) Un problème est une question générale possédant des paramètres dont la valeur n'est pas connue. Un problème est décrit en donnant :*

1. *une description générale de tous ces paramètres.*
2. *une solution satisfaisant ces paramètres.*

Exemple 2.1. *Considérons le problème classique du voyageur de commerce ou TSP. Les paramètres sont un ensemble de villes $V = \{v_1, v_2, ..., v_n\}$ et pour chaque paire de villes v_1, v_2 dans V on donne une distance euclidienne $d(v_1, v_2)$. Une solution du problème est un ordre de villes $< v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)} >$. Elle est optimale si elle minimise $(\sum_{i=1}^{n-1} d(v_{\pi(i)}, v_{\pi(i+1)})) + d(v_{\pi(n)}, v_{\pi(1)})$.*

Definition 2.2.2. *(Instance d'un problème) Une instance d'un problème est obtenue en spécifiant la valeur de chaque paramètre du problème.*

Definition 2.2.3. *(Problème de décision) Un problème de décision est un problème ayant deux réponses possibles : oui ou non.*

Definition 2.2.4. *(Taille d'un problème) La taille d'un problème est la taille des données nécessaire pour décrire une instance.*

Exemple 2.2. *Considérons le problème* ***plus-courte-chaîne*** *qui étant donné un graphe non orienté $G = (V, E)$, et deux sommets $x, y \in V$, calcule la plus courte chaîne (en nombre d'arêtes) entre x et y.*
Une instance pour le problème ***plus-courte-chaîne*** *est un triplet formé d'un graphe $G = (V, E)$ et deux sommets $x, y \in X$, noté $i =< G, x, y >$.*
Le problème de décision ***CHAÎNE*** *associé au problème* ***plus-courte-chaîne*** *est défini comme suit :*

Données : Étant donné un graphe non orienté $G = (V, E)$, deux sommets $x, y \in V$ et k un entier positif.

Question : Existe-t-il une chaîne de x à y dont la longueur est au plus k ?

Si $i =< G, x, y, k >$ est une instance de ***CHAÎNE****, alors CHAÎNE(i) =1 s'il existe une chaîne de x à y de longueur inférieure à k et CHAÎNE(i)=0 sinon.*

2.2.3 Notations asymptotiques $\mathcal{O}$, Ω et Θ

Cette notation permet une analyse sans tenir compte de facteurs constants dus aux détails de la machine informatique utilisée. Toutes les fonctions manipulées sont du type : $g : \mathbb{N} \longrightarrow \mathbb{N}$.

- $g \in \mathcal{O}(f)$, s'il existe deux constantes strictement positives c et n_0 telles que $cf(n)$ est une borne *supérieure* de $g(n)$ pour tout $n > n_0$, $g(n) \leq cf(n), \forall n > n_0$.

- $g \in \Omega(f)$, s'il existe deux constantes strictement positives c et n_0 telles que $cf(n)$ est une borne *inférieure* de $g(n)$ pour tout $n > n_0$, $g(n) \geq cf(n), \forall n > n_0$.

- $g \in \Theta(f)$ si $g \in \mathcal{O}(f)$ et $g \in \Omega(f)$.

Pour compléter ces trois notations, Woeginger a introduit dans [64] la notation O^* qui autorise la suppression de tous les termes bornés par un facteur polynomial dépendant de la taille de l'entrée. Ainsi une complexité de la forme $O(f(n).poly(n))$, où n est la taille de l'entrée et poly(n) un polynôme dépendant de n, se notera $O^*(f(n))$. Cela nous permettra en particulier d'insister sur la croissance exponentielle de la fonction qui borne le temps d'exécution.

2.3 Algorithmes

Un algorithme est une procédure qui pour chaque instance du problème produit une réponse. Un algorithme est constitué par un ensemble de règles logiques (nombre donné d'opérations élémentaires) et chronologiques (un ordre bien déterminé) qu'on doit suivre pour aboutir à la résolution du problème. Un algorithme peut chercher l'optimum ou trouver au moins une valeur approchée, ou nous fournir une solution simplement réalisable, en un temps, au mieux, polynomial.

2.3.1 Algorithmes exacts

Les premiers algorithmes exacts exponentiels pour résoudre des problèmes $NP-difficiles$ datent des années soixante et soixante-dix. Ils sont motivés par les nombreuses applications de problèmes réputés difficiles et qui, sous l'hypothèse $P \neq NP$, n'admettent pas d'algorithme polynomial calculant une solution exacte. Des algorithmes pour le problème SAT sont alors publiés[20, 19]. Held et Karp, en 1962[37] donnent également un algorithme pour résoudre le problème du voyageur de commerce en temps $O(2^n)$ alors que l'algorithme était en $O(n!)$. Plus tard, Horowitz et Sahni[38] proposent un algorithme en $O(2^{\frac{n}{2}})$ pour résoudre le problème du sac à dos binaire, améliorant l'algorithme naïf en $O(2^n)$. La signification d'un tel résultat est que si pendant un certain temps, on est capable de traiter une entrée de taille k avec l'algorithme en $O(2^n)$, alors, grâce à un algorithme en $O(2^{\frac{n}{2}})$, dans le même temps, on peut traiter une entrée de taille $2k$. En 1977, Tarjan et Trojanowski [60] donnent un algorithme en $O(2^{\frac{n}{3}})$ pour résoudre le problème de l'ensemble stable maximum.

2.3.2 Algorithmes d'approximation

Un algorithme d'approximation est une approche de résolution en temps polynomial, dont on arrive à garantir la qualité des solutions trouvées. On dit qu'un algorithme A est c-approximation s'il retourne des solutions de taille au plus c fois celle de la solution optimale pour un problème de minimisation et de taille au mois $\frac{1}{c}$ fois la taille de la solution optimale pour un problème de maximisation. La variable c appelée facteur d'approximation. Pour montrer qu'un algorithme est une c-approximation il faut que :

1. l'algorithme soit polynomial ;
2. l'algorithme renvoie des solutions réalisables au problème posé ;
3. le rapport d'approximation soit égal à la constante positive c.

Un problème qui admet un algorithme d'approximation de facteur constant c est de la classe APX. Un problème qui admet un algorithme d'approximation de facteur $1 \pm \varepsilon$, avec $\varepsilon \in]0, 1[$ est de la classe $PTAS$ (Polynomial Time Approximation Scheme).

2.3.3 Heuristiques

Pour certaines instances d'un problème, les résolutions exactes ont un temps d'exécution pratique beaucoup trop grand. On est donc amené à rechercher une solution la plus proche possible d'une solution optimale en procédant par essais successifs en exploitant certains choix algorithmiques. Puisque toutes les combinaisons ne peuvent être essayées, certains choix stratégiques doivent être faits. Ces choix, généralement très dépendants du problème traité, constituent ce qu'on appelle une heuristique. Le but d'une heuristique est de trouver une solution approchée convenable (qui peut être exacte dans certains cas) dans un temps raisonnable.

2.3.4 Classes P et NP

Definition 2.3.1. *(Classe P) Un problème de décision est dans la classe P s'il existe un algorithme polynomial pour le résoudre. Les problèmes de la classe P sont dits faciles.*

Definition 2.3.2. *(Classe NP)*
Soit Q un problème de décision et $L(Q)$ l'ensemble de ses instances pour lesquelles la réponse est oui. Q appartient à la classe NP (Nondeterministic Polynomial) s'il existe un algorithme polynomial qui permet de vérifier que la réponse est oui pour toute instance I de $L(Q)$.

Exemple 2.3. *Étant donné une instance $I =< G, x, y, k >$ du problème de décision CHAÎNE et une séquence de sommets μ (certificat), on vérifie que μ est une chaîne de x à y et de longueur au plus k.*

2.3.5 Problèmes NP-complets et réductions polynomiales

Definition 2.3.3. *(Réduction polynomiale de problème) Un problème de décision D_1 se réduit polynomialement en un problème de décision D_2 s'il existe une fonction polynomiale f telle que, pour toute instance* I *de D_1, la réponse est oui si est seulement si la réponse de $f(I)$ pour D_2 est oui. Nous noterons alors $D_1 \propto D_2$.*

Definition 2.3.4. *(Classe NP-Complet) Un problème $\hat{D}$ est NP-Complet s'il satisfait les deux conditions suivantes :*

1. *$\hat{D}$ est dans la classe NP.*
2. *pour tout problème D de la classe NP, on a $D \propto \hat{D}$.*

Cook a été le premier à montrer la NP-complétude d'un problème, celui de la satisfaisabilité [16]. La classe NP-complet est donc composée des problèmes difficiles de la classe NP.

2.4 Éléments de la théorie des polyèdres

Dans cette section, nous allons introduire quelques définitions et propriétés de la théorie polyèdrale, qui seront utilisées dans ce document. Pour plus de détails, on peut se référer au livre de Schrijver [56].

2.4.1 Problème d'optimisation combinatoire

Un problème d'*Optimisation Combinatoire* peut être formulé de la manière suivante : Soit $E = \{e_1, ..., e_n\}$ un ensemble de n éléments appelé *ensemble de base* où chaque élément e_i possède un poids $c(e_i) \in \mathbb{Z}$. Soit $\mathfrak{F}$ une famille de sous-ensembles de E. $\mathfrak{F}$ est appelé l'*ensemble des solutions* du problème. Si $S \in \mathfrak{F}$, alors $c(S) = \sum_{e \in S} c(e)$ est le poids de S. Le problème consiste à déterminer un élément de $\mathfrak{F}$, ayant le plus petit (ou le plus grand) poids. On peut associé à tout problème d'optimisation combinatoire un problème de décision. Pour cela on rajoute une borne k aux données décrivant une instance du problème. Si un problème d'optimisation est facile alors le problème de décision associé est aussi facile. Il suffit de comparer la valeur optimale obtenue à partir de la solution à la borne du problème de décision. Tout problème d'optimisation combinatoire dont le problème de décision associé est NP-Complet est dit NP-*difficile*. On peut se référer à [56, 52] pour des compléments.

2.4.2 Polyèdres

Definition 2.4.1. *(Ensemble convexe) Un ensemble $C \subset \mathbb{R}^n$ est convexe si $\forall x, y \in C$, $\alpha x + (1-\alpha)y \in C$ pour tout $\alpha \in [0,1]$.*

Definition 2.4.2. *(Combinaison linéaire) Soit $x \in \mathbb{R}^n$. On dit que x est une combinaison linéaire des points $x_1, x_2, ..., x_k \in \mathbb{R}^n$ s'il existe k scalaires $\alpha_1, \alpha_2, ..., \alpha_k \in \mathbb{R}^n$ tels que $x = \alpha_1 x_1 + \alpha_2 x_2 + ...\alpha_k x_k$. lorsque $\sum_{i=1}^{k} \alpha_i = 1$ on dit que x est une combinaison affine de ces k points. De plus, si $\alpha_i \geq 0$, pour tout $i = [1;k]$ avec $\sum_{i=1}^{k} \alpha_i = 1$, x est une combinaison linéaire convexe de ces points.*

Definition 2.4.3. *(Indépendance linéaire) Des points $x_1, x_2, ..., x_k \in \mathbb{R}^n$ sont dits linéairement indépendants (resp. affinement indépendants) si le système :*

$$\sum_{i=1}^{k} \alpha_i x_i = 0$$

$$(resp. \sum_{i=1}^{k} \alpha_i x_i = 0 \; et \sum_{i=1}^{k} \alpha_i = 0)$$

admet une solution unique, $\alpha_i = 0, \forall i = [1;k]$.

Definition 2.4.4. *(Enveloppe convexe) Soit S un ensemble non vide de points de $\mathbb{R}^n$. L'enveloppe convexe des points de S, notée $conv(S)$, est le plus petit l'ensemble convexe qui contient S.*

Definition 2.4.5. *(Polyèdre) Un polyèdre P est un ensemble de points de $\mathbb{R}^n$ engendré par l'intersection d'un nombre fini de demi-espaces de $\mathbb{R}^n$. D'une manière équivalente, P est l'ensemble des solutions d'un système d'inégalités linéaires, c'est-à-dire $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, où A est une matrice à m lignes et n colonnes, et b un vecteur à m composantes. On dit que le système $Ax \leq b$ caractérise le polyèdre P.*

Definition 2.4.6. *(Polytope) Un polytope est un polyèdre borné. Ainsi, un polyèdre $P \subseteq \mathbb{R}^n$ est un polytope si et seulement s'il existe l, $u \in \mathbb{R}^n$ tels que $l \leq x \leq u$, pour tout $x \in P$.*

Exemple 2.4. *Par exemple, l'enveloppe convexe d'un nombre fini de points est un polytope.*

Definition 2.4.7. *(Dimension d'un polyèdre) Un polyèdre P de $\mathbb{R}^n$ est de dimension d, notée $dim(P)$ si le nombre maximum de points de P* affinement indépendants *est égal à $d+1$.*

Definition 2.4.8. *(Polyèdre de pleine dimension) Un polyèdre $P \in \mathbb{R}^n$ est de pleine dimension si $dim(P) = n$.*

Definition 2.4.9. *(Point extrême) Un point $x_0 \in \mathbb{R}^n$ est un point extrême d'un polyèdre P s'il n'existe pas $x_1, x_2 \in P$, $x_1 \neq x_2$ et $\alpha \in]0,1[$ tel que $x_0 = \alpha x_1 + (1-\alpha)x_2$.*

Definition 2.4.10. *(Inégalité valide) Une inégalité (ou contrainte) $a^t x \leq \alpha$ est dite valide pour un polyèdre P si elle est vérifiée par tous les points de P, soit $P \subseteq \{x \in \mathbb{R}^n : a^t x \leq \alpha\}$.*

Definition 2.4.11. *(Face d'un polyèdre) Soit $a^t x \leq \alpha$ une contrainte valide pour le polyèdre P. Alors le sous-ensemble $F \subseteq P$ défini par $F = \{x \in P : a^t x = \alpha\}$ est une face de P. Si de plus, $F \neq \emptyset$ et $F \neq P$ alors F est une face propre.*

Definition 2.4.12. *(Facette d'un polyèdre) Une face propre F est dite* facette *de P si $dim(F) = dim(P) - 1$. Une* facette *est une* face maximale.

Definition 2.4.13. *(Contrainte serrée, lâche) Soit $x^* \in \mathbb{R}^n$. L'inégalité $a^t x \leq \alpha$, valide pour un polyèdre P, est serrée (resp. lâche) pour $x^* \in P$ si $a^t x^* = \alpha$ (resp.$a^t x^* < \alpha$).*

Definition 2.4.14. *(Vecteur d'incidence) Soit $x(e)$ une variable associée à chaque élément e d'un ensemble fini E. Pour un sous-ensemble d'éléments $S \subset E$, le vecteur $x^S \in \mathbb{R}^E$ où $x^S(e) = 1$ si $e \in S$, $x^S(e) = 0$ sinon, est appelé le* vecteur d'incidence *de S.*

Definition 2.4.15. *(Programme linéaire) Un programme linéaire est un programme d'optimisation sous la forme*

$$\begin{aligned} \min c^T x & \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

où $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$.

2.4.3 Méthode de coupes et branchement

Definition 2.4.16. *(Problème de séparation) Considérons un polyèdre $P \subset \{x : Ax \leq b\}$. Étant donné un vecteur $y \in \mathbb{R}^n$, le problème de séparation associé à l'ensemble des contraintes $Ax \leq b$, consiste soit à vérifier que y est solution du système $Ax \leq b$, soit à trouver une contrainte valide $a^t x \leq \alpha$ pour P violée par y.*

Supposons que, pour un problème d'optimisation combinatoire, nous disposons d'une formulation du problème par un programme linéaire en nombres entiers (PLNE) sous la forme

$$\begin{aligned} \min c^T x & & (2.1)\\ Ax &\leq b, & (2.2)\\ x &\in \{0,1\}^n & (2.3) \end{aligned}$$

Soit $\mathbb{P}$ le polyèdre dont les sommets sont les solutions de (2.2)-(2.3). En général, le système $Ax \leq b$ contient un nombre important d'inégalités valides non redondantes. Pour les problèmes $NP - durs$ on ne connaît pas en général la caractérisation complète des polyèdres associés. L'algorithme de coupes et branchement associé au problème (2.1)-(2.3) consiste à sélectionner un sous-système de contraintes valides $A_0 x \leq b_0$ de taille raisonnable de type (2.2) et de résoudre le programme linéaire :

$$\min c^T x \tag{2.4}$$
$$A_0 x \leq b_0, \tag{2.5}$$
$$0 \leq x \leq 1 \tag{2.6}$$

Admettons de plus qu'il existe un algorithme de séparation polynomial associé au système $Ax \leq b$. Soit x^* la solution de (2.4)-(2.5).
L'algorithme de coupes et branchement résout le problème de séparation pour x^*. Si nous avons déterminé une inégalité $a^t x \leq \alpha$ parmi $Ax \leq b$, tel que $a^t x^* > \alpha$, alors nous ajoutons $a^t x \leq \alpha$ au système $A_0 x \leq b_0$ et nous résolvons à nouveau (2.4)-(2.5). Nous répétons cette procédure jusqu'à ce que $Ax^* \leq b$. Si la solution courante x^* est à composantes en 0-1, alors nous avons trouvé une solution optimale de (2.1)-(2.3). Sinon, l'algorithme fait un branchement sur une composante fractionnaire x_i^* en construisant deux nouveaux programmes obtenus à partir du programme courant, en ajoutant respectivement les égalités $x_i = 1$ ou $x_i = 0$. On itère le processus ci-dessus pour les polyèdres $\mathbb{P} \cap \{x : x_i = 0\}$ et $\mathbb{P} \cap \{x : x_i = 1\}$.

Chapitre 3

Propriétés et Complexité pour l'indépendant faiblement connexe

Sommaire

3.1 Introduction

Dans ce chapitre, nous donnons quelques propriétés de l' *indépendant faiblement connexe* ou *wcis* dans un graphe connexe. A notre connaissance, ce travail est l'une des premières études des propriétés particulières de ces ensembles. Le chapitre est organisé comme suit. Dans la section 3.2 nous présentons les propriétés de base d'un *wcis*. La section 3.3 est dédiée aux résultats de complexité et d'approximation pour le problème du *wcis* de cardinalité minimum. Dans la section 3.4 nous étudions ce problème dans certaines classes de graphes tels que les graphes bipartis, les split graphes et les graphes de comparabilité.

3.2 Propriétés de base des indépendants faiblement connexes

Soit $G = (V, E)$ un graphe non orienté et connexe, avec $|V| \geq 2$. Soit $\mathcal{W}(G)$ l'ensemble des indépendants faiblement connexes de G. Les points du lemme suivant sont faciles à prouver.

Lemme 3.2.1. *Si $W \in \mathcal{W}(G)$ alors*
i) W est un stable maximal,
ii) $G_W = (V, [W, V \backslash W])$ est un graphe biparti connexe,
iii) Il existe une partition $(V_1, \dots V_p)$ de V telle que $V_i \cap W = \{w_i\}$, $V_i \subseteq N[w_i]$ pour $i = 1, \dots p$ et $d_G(w_i, \cup_{j=1}^{i-1}\{w_j\}) = 2$, pour $2 \leq i \leq p$.

Preuve.
i) Considérons un wcis W de G. Supposons que W n'est pas un stable maximal de G. Alors, il existe au moins un sommet u_0 de V tel que $N(u_0) \cap W = \emptyset$. Par conséquent, le

graphe $G_W = (V, [W, V \backslash W])$ n'est pas connexe. Contradiction, puisque W est un wcis de G.

ii) Les arêtes de G_W sont celles de la coupe engendrée par W. G_W est donc un graphe partiel de G biparti et connexe par définition.

ii) Considérons un wcis W de G, $|W| \geq 2$. Soit $w_1 \in W$. Comme G_W est connexe, il existe $w \in W \cap N^2(w_1)$; $V_1 = N[w_1]$. On pose $w_2 = w$ et $V_2 = N[w_2] \backslash V_1$. Par connexité, il existe $w_3 \in W$ appartenant à $N(V_1 \cup V_2)$. On obtient $V_3 = N[w_3] \backslash (V_1 \cup V_2)$. En itérant le processus, on numérote les sommets de W et on détermine la partition recherchée.

□

Le prochain lemme se trouve dans [2].

Lemme 3.2.2. *Soit W un indépendant maximal de G. W est un wcis de G si et seulement si, pour tout sous-ensemble $A \subset W$, il existe un sommet $u \in A$ et un sommet $v \in W \setminus A$ tel que $d_G(u, v) = 2$.*

Notons par MWCIS(G) un indépendant faiblement connexe de cardinalité minimum de G. Le lemme suivant donne une description des bornes pour la cardinalité du MWCIS(G).

Lemme 3.2.3. *Si $W \in \mathcal{W}(G)$, alors*

i) $\frac{|V|-1}{\Delta(G)} \leq |W|$,

ii) $|W| \leq (\Delta(G) - 1)|MWCIS(G)| + 1$,

iii) $|MWCIS(G)| \leq |V| - \Delta(G)$,

iv) Il existe pas de réel β, $0 < \beta < 1$, et un entier N_β tels que $|MWCIS(G)| \leq \beta \times |V|$, pour tout graphe connexe $G = (V, E)$ avec $|V| \geq N_\beta$.

Preuve. Soit $|\text{MWCIS}(G)| = \bar{p}$. Premièrement, supposons que $\bar{p} = 1$ et MWCIS(G) $= \{\bar{w}_1\}$. Alors $\bar{w}_1$ est adjacent à tout sommet de V, et $V = N[\bar{w}_1]$. Donc $|V| - 1 = \Delta(G)$. Donc $i)$ est vérifiée pour tout $W \in \mathcal{W}(G)$. En outre, comme $W \subset N(\bar{w}_1)$, pour tout $W \in \mathcal{W}(G)$ ne contenant pas $\bar{w}_1$, nous obtenons $ii)$.
Supposons maintenant que $\bar{p} \geq 2$. D'après le Lemme 3.2.1 iii), notons par $(\bar{V}_1, \dots \bar{V}_{\bar{p}})$ une partition induite par MWCIS(G). Comme $\bar{V}_i \subseteq N[\bar{w}_i]$ pour tout $1 \leq i \leq \bar{p}$, et $d_G(\bar{w}_i, \cup_{j=1}^{i-1}\{\bar{w}_j\}) = 2$, pour tout $2 \leq i \leq \bar{p}$, nous avons

$$|\bar{V}_1| \leq \Delta(G) + 1 \text{ et } |\bar{V}_i| \leq \Delta(G) \text{ pour } i \geq 2. \tag{3.1}$$

i) Comme $V = \cup_{i=1}^{\bar{p}} \bar{V}_i$, nous obtenons

$$|V| \leq \Delta(G) + 1 + \Delta(G)(\bar{p} - 1).$$

Ce qui implique

$$\frac{|V| - 1}{\Delta(G)} \leq \bar{p}. \tag{3.2}$$

De plus $|W| \geq \bar{p}$, pour tout $W \in \mathcal{W}(G)$. L'inégalité (3.2) induit alors $i)$.

ii) Soit W dans $\mathcal{W}(G)$. Nous donnons une borne supérieure de $|\bar{V}_i \cap W|$, pour tout i. Si $\bar{w}_{i_0} \in W$, pour un certain $i_0 \in \{1, \dots \bar{p}\}$, alors $|\bar{V}_{i_0} \cap W| = 1$.

Ainsi, $|\bar{V}_i \cap W| \leq |\bar{V}_i| - 1$ pour tout $i \in \{1, \dots \bar{p}\}$. Par (3.1) on a $|\bar{V}_1 \cap W| \leq \Delta(G)$ et $|\bar{V}_i \cap W| \leq \Delta(G) - 1$ pour tout $i \geq 2$. Comme $W = \bigcup_{i=1}^{\bar{p}} (\bar{V}_i \cap W)$, nous obtenons

$$|W| \leq \Delta(G) + (\Delta(G) - 1)(\bar{p} - 1),$$

et *ii*) est vérifiée.

iii) Il suffit de prendre un *wcis* contenant un sommet de degré $\Delta(G)$.

iv) Supposons au contraire qu'il existe un réel β, $0 < \beta < 1$, et un entier N_β tel que $|\text{MWCIS}(G)| \leq \beta \times |V|$, pour tout graphe connexe $G = (V, E)$ avec $|V| \geq N_\beta$.
Soit $p_\beta = \left\lceil \frac{\beta}{1-\beta} \right\rceil$ et $K_N = (\{u_1, \dots u_N\}, E(K_N))$, une clique d'ordre $N \geq N_\beta$. Définissons le graphe $H = (V(H), E(H))$ par
- $V(H) = \{u_1, \dots u_N\} \cup \{v_i^j : 1 \leq j \leq p_\beta, 1 \leq i \leq N\}$,
- $E(H) = E(K_N) \cup \{(u_i, v_i^j) : 1 \leq j \leq p_\beta, 1 \leq i \leq N\}$.

Notons que l'ensemble de sommets $\bigcup_{i=1}^{N} \{v_i^1, \dots v_i^{p_\beta}\}$ n'est pas un *wcis* de H. Comme K_N est complet, nous pouvons voir que $|W \cap \{u_1, \dots u_N\}| = 1$, pour tout $W \in \mathcal{W}(H)$. Sans perte de généralité $\text{MWCIS}(H) = \{u_1\} \cup (\bigcup_{i=2}^{N} \{v_i^1, \dots v_i^{p_\beta}\})$. Alors

$$\frac{|\text{MWCIS}(H)|}{|V(H)|} = \frac{1 + p_\beta(N-1)}{N + p_\beta N} = \frac{p_\beta}{p_\beta + 1} + \frac{1 - p_\beta}{N(1 + p_\beta)}.$$

Et ce rapport dépasse β pour un N assez grand.

□

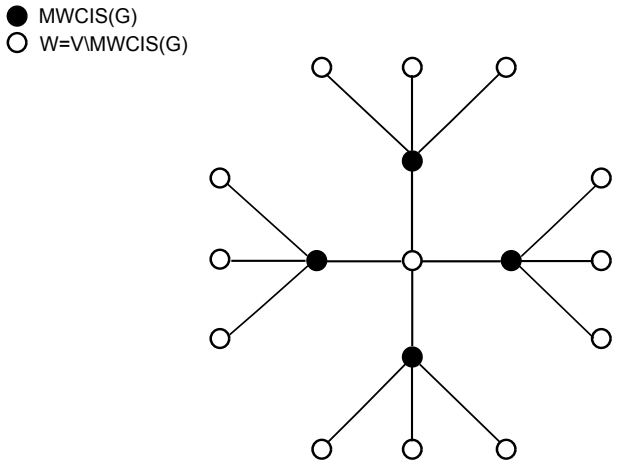


FIGURE 3.1 – Exemple où les bornes i) et ii) du Lemme 3.2.3 sont atteintes.

3.3 Résultats généraux de complexité

Soient $G = (V, E)$ un graphe non orienté connexe et k un entier positif. Le problème de l'indépendant faiblement connexe est de savoir s'il existe un $W \in \mathcal{W}(G)$ de taille inférieure ou égale à k dans G.

Théorème 3.3.1. *Le problème de l'indépendant faiblement connexe est NP-Complet.*

Le théorème 3.3.1 est une conséquence du théorème 3.4.3 du paragraphe 3.4. Signalons que ce résultat se trouve aussi dans l'article [21] dont le thème est le dominant faiblement connexe.

Le résultat négatif d'approximation pour le problème de l'indépendant maximal minimum prouvé dans [35] reste valable pour le problème de l'indépendant faiblement connexe minimum.

Théorème 3.3.2. *Il n'existe pas d'algorithme d'approximation polynomial pour le problème de l'indépendant faiblement connexe minimum de facteur $|V|^{1-\varepsilon}$, pour tout $\varepsilon > 0$, sauf si $P = NP$.*

Preuve. Supposons qu'il existe un algorithme polynomial A_ε, pour un graphe connexe $G = (V, E)$, construisons le *wcis* $A_\varepsilon(G)$ tel que

$$|A_\varepsilon(G)| \leq |V|^{1-\varepsilon}|\text{MWCIS}(G)| \tag{3.3}$$

avec $\varepsilon \in]0, 1[$.

On sait que le problème de l'indépendant maximal minimum est difficile à approximer [35]. Mais, l'algorithme A_ε peut être transformé en un algorithme d'approximation polynomial B pour ce dernier problème. Définissons le graphe $G'' = (V'', E'')$ par

- $V'' = V \cup Z$ avec $Z = \{z_u : u \in V\}$,
- $E'' = E \cup \{(u, z_u) : u \in V\} \cup \{(z_u, z_v) : u, v \in V, u \neq v\}$.

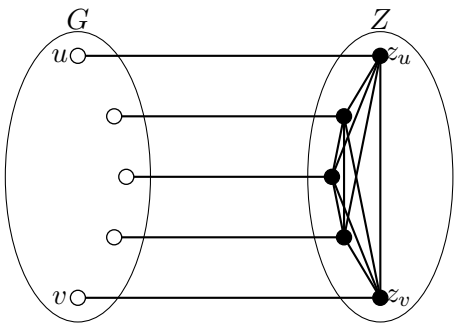


FIGURE 3.2 – G et G'' pour la preuve du Théorème 3.3.2.

Remarquons que Z est une clique de taille $|V|$. Notons par $\text{MMIS}(G)$ un indépendant maximal minimum de G.

Proposition 1.

$$|MMIS(G)| \leq |MWCIS(G'')| \leq |MMIS(G)| + 1.$$

Preuve.

Comme $\text{MMIS}(G) \neq V$, il existe $u_0 \in V \setminus \text{MMIS}(G)$. Soit $S_1 = \text{MMIS}(G) \cup \{z_{u_0}\}$. Il est facile de voir que S_1 est un *wcis* de G''. Alors

$$|\text{MWCIS}(G'')| \leq |S_1| = |\text{MMIS}(G)| + 1. \tag{3.4}$$

Comme Z est une clique et qu'il existe au moins une arête dans E, nous avons $\text{MWCIS}(G'') \cap Z = \{z_{u_1}\}$, pour certain $u_1 \in V$. Soit $T = \text{MWCIS}(G'') \setminus \{z_{u_1}\}$. T est un indépendant de G puisque $\text{MWCIS}(G'')$ est un stable de $G^{''}$. En outre, comme $\text{MWCIS}(G'')$ est un dominant de G'', tout sommet $v \in V \setminus T$, différent de u_1, a un voisin dans T. Soit S tel que

$$S = \begin{cases} T \cup \{u_1\}, & \text{si } T \cup \{u_1\} \text{ est un stable de } G; \\ T, & \text{sinon.} \end{cases}$$

Il est facile de voir que S est un indépendant maximal de G, et

$$|S| \quad \leq \quad |T| + 1 = |\text{MWCIS}(G^{''})|.$$

nous obtenons alors

$$|\text{MMIS}(G)| \leq |S| \leq |\text{MWCIS}(G^{''})|. \tag{3.5}$$

Les inégalités (3.4) et (3.5) donnent

$$|\text{MMIS}(G)| \leq |\text{MWCIS}(G^{''})| \leq |\text{MMIS}(G)| + 1,$$

d'où le résultat de la Proposition 1 (cf. figure 3.3 (a) et (b)).

♦

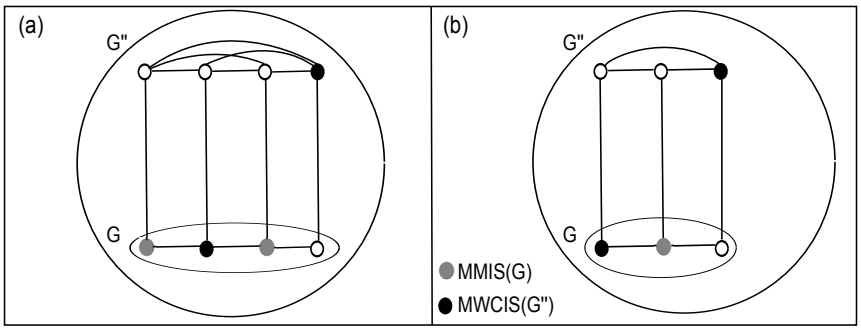


FIGURE 3.3 – (a)$|\text{MWCIS}(G^{''})|$=$|\text{MMIS(G)}|$.(b)$|\text{MWCIS}(G^{''})|$=$|\text{MMIS(G)}|$+1.

Considérons maintenant le *wcis* $A_\varepsilon(G'')$ obtenu par l'application de l'algorithme A_ε sur $G^{''}$. Nécessairement nous avons $A_\varepsilon(G'') = W_1 \cup \{z_a\}$ avec $W_1 \subset V$ et $a \in V$. Comme dans la Proposition 1, considérons l'indépendant dominant $W_2(G)$ donné par

$$W_2(G) = \begin{cases} W_1 \cup \{a\}, & \text{si } W_1 \cup \{a\} \text{ est un stable de } G; \\ W_1, & \text{sinon.} \end{cases}$$

De (3.3), nous déduisons que

$$|W_2(G)| \leq |A_\varepsilon(G'')| \leq |V''|^{1-\varepsilon} |\text{MWCIS}(G^{''})|.$$

Alors, la Proposition 1 implique que

$$|W_2(G)| \leq (2 \times |V|)^{1-\varepsilon}(|\text{MMIS}(G)| + 1).$$

On obtient alors

$$|W_2(G)| \leq 2^{2-\varepsilon} \times |V|^{1-\varepsilon}|\text{MMIS}(G)|.$$

Notons que

$$2^{2-\varepsilon}n^{1-\varepsilon} \leq n^{1-\frac{\varepsilon}{2}},$$

lorsque $n \geq n_0$, pour un certain $n_0 \in \mathbb{N}$. D'où,

$$|W_2(G)| \leq |V|^{1-\frac{\varepsilon}{2}}|\text{MMIS}(G)|, \tag{3.6}$$

si $|V| \geq n_0$.

Ainsi, nous pouvons esquisser un algorithme polynomial B qui produit un Indépendant Dominant $B(G)$ pour le graphe $G = (V, E)$ de sorte que

$$B(G) = \begin{cases} W_2(G), & \text{si } |V| \geq n_0; \\ \text{MMIS}(G), & \text{sinon (obtenu par énumération).} \end{cases}$$

L'inégalité (3.6) indique que B est un algorithme d'approximation polynomial pour le problème de l'indépendent dominant de cardinalité minimum, ce qui contredit le résultat de [35].

□

3.4 Complexité dans des classes particulières de graphes

Definition 3.4.1. *(graphe biparti) Un graphe $G = (V, E)$ est biparti si ses sommets peuvent être répartis en deux ensembles A et B, de sorte que toutes les arêtes de E relient un sommet de A à un sommet de B.*

Notons par $\mathcal{B}$ la classe des graphes bipartis connexes. Les preuves des deux premiers résultats ci-dessous sont faciles.

Théorème 3.4.1. *Le graphe $G = (A \cup B, E)$ dans $\mathcal{B}$ a exactement deux indépendants faiblement connexes, qui sont A et B.*

Preuve. Dans un graphe biparti connexe $G = (A \cup B, E)$, la longueur d'une chaîne entre deux sommets u et v est paire si et seulement si u et v appartiennent tous deux à A ou à B. Considérons un *wcis* W de G. Comme le graphe $G_W = (V, \delta(W))$ est connexe et biparti, une chaîne reliant deux sommets u et v de W a une longueur paire dans G. Donc u et v appartiennent à la même partie. □

Definition 3.4.2. *(split-graphe) On dit que $G = (V, E)$ est un split-graphe s'il existe une partition de V en une clique K et un stable I.*

Un split-graphe $(K \cup I, E)$ est connexe si $[\{v\}, K] \neq \emptyset, \forall v \in I$.

Théorème 3.4.2. *Dans un split-graphe connexe $G = (K \cup I, E)$, il existe au plus $(|K| + 1)$ indépendants faiblement connexes.*

Preuve. Soit W un *wcis* d'un split-graphe $G = (K \cup I, E)$ où K est une clique et I un indépendant. On a $|W \cap K| \leq 1$ car K est une clique de G.

1. Si $W \cap K = \emptyset$ alors I est inclus dans W puisque W est un dominant..
2. Si $W \cap K = \{u_0\}$ alors $W = \{u_0\} \cup I \backslash \{N(u_0)\}$.

Il existe au plus $(|K| + 1)$ $WCIS$ dans un split-graphe connexe. □

Étant donné un split-graphe connexe $G = (K \cup I, E(G))$, nous déduisons facilement du Théorème 3.4.2, que $|\text{MWCIS}(G)| = 1 + |I| - \max\{|[\{u\}, I]|; u \in K\}$.

Definition 3.4.3. *(graphe de comparabilité) Un graphe connexe $G = (V, E)$ est un graphe de comparabilité si on peut orienter ses arêtes de façon transitive.*

Théorème 3.4.3. *Le problème de l'indépendant faiblement connexe minimum est NP-difficile dans les graphes de comparabilité.*

Preuve.

Étant donné un graphe de comparabilité $G = (V, E)$, soit le graphe $G' = (V', E')$ tel que (cf. Figure 3.4)

i) $V' = V \cup \{x_1, x_2\} \cup Z$ avec $Z = \{z_u : u \in V\}$,

ii) $E' = E \cup \{(u, x_1) : u \in V\} \cup \{(x_1, x_2)\} \cup \{(x_2, z_u) : u \in V\}$.

Notons que Z est un indépendant d'ordre $|V|$.

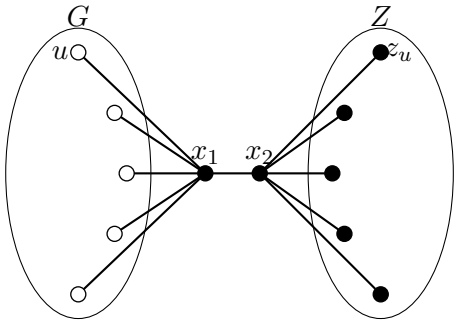


FIGURE 3.4 – G et G' dans la preuve du Théorème 3.4.3

G' est un graphe de comparabilité. En effet, il est facile de déduire une orientation transitive de G' à partir d'une orientation transitive de G (cf Figure 3.5).

Nous en déduisons que, pour tout indépendant maximal S de G, l'ensemble $S' = S \cup \{x_2\}$ est un *wcis* de G'. Comme $Z \cup \{x_1\}$ est le seul *wcis* de G' qui ne contient pas x_2, l'indépendant maximal minimum de G peut être associé à un indépendant faiblement connexe minimum de G'. Par conséquent, le problème MMIS et le problème MWCIS ont la même complexité dans la classe des graphes de comparabilité. □

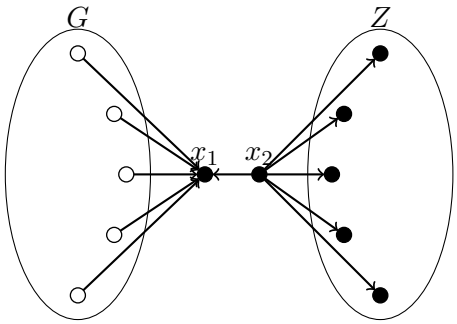


FIGURE 3.5 – Orientation du graphe $G^{'}$.

Considérons maintenant la classe de graphes $\mathcal{B}_1$ qui est plus large que $\mathcal{B}$. Un graphe connexe $G = (V, E)$ appartient à $\mathcal{B}_1$ si G est un graphe biparti connexe, ou s'il existe un sommet $u_0 \in V$ tel que $G \setminus \{u_0\}$ est un graphe biparti connexe.

Théorème 3.4.4. *Le problème de l'indépendant faiblement connexe est NP-complet dans* $\mathcal{B}_1$.

Preuve. Le problème consistant à déterminer si un graphe biparti connexe $G = (A \cup B, E)$ a un ensemble indépendant maximal de taille inférieure à k est NP-complet [39].

Nous transformons le graphe biparti connexe G en un graphe $G_1 = (V_1, E_1)$ de $\mathcal{B}_1$ comme suit (cf. Figure 3.6).

i) $V_1 = A \cup B \cup \{s, s_1, s_2\} \cup Z$, avec $Z = \{z_u : u \in A \cup B\}$,

ii) $E_1 = E \cup \{(s, s_1), (s, s_2)\} \cup \{(s_1, v) : v \in B\} \cup \{(s_2, u) : u \in A\} \cup \{(s, z_u) : u \in A \cup B\}$.

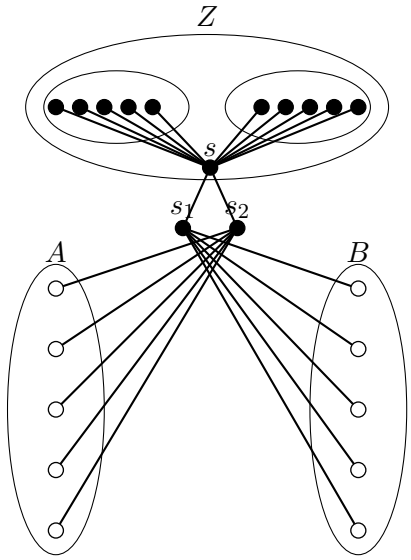


FIGURE 3.6 – Graphe G_1 obtenu à partir de G

Notons que $G_1 \setminus \{s_1\}$ et $G_1 \setminus \{s_2\}$ appartiennent à $\mathcal{B}$. Tout indépendant maximal M de G correspond à un indépendant faiblement connexe $W = M \cup \{s\}$ de G_1. En outre, tout *wcis* de G_1 qui est inclus dans $V_1 \setminus \{s\}$, contient l'ensemble Z et s_1 ou s_2. Et sa cardinalité

est plus grande que $|A| + |B| + 1$. Ensuite, il est facile de vérifier que G a un indépendant maximal M tel que $|M| \leq k$ si et seulement si G_1 a un indépendant faiblement connexe W de taille $k + 1$ (au plus). □

Comme tout *wcis* d'un graphe G est un *mis* de G, il peut être intéressant d'étudier la propriété suivante.

Definition 3.4.4. *(*wcis-propriété*) Un graphe connexe G a la* wcis-propriété *si tout indépendant maximal de G est un indépendant faiblement connexe.*

Notons que le cycle C_5 a la *wcis-propriété* alors que P_4 ne l'a pas. Nous n'avons pas caractérisé ces graphes, mais nous avons le résultat ci-dessous.

Lemme 3.4.1. *Soit $G = (V, E)$ un graphe non orienté et connexe. G et* tous *ses sous-graphes connexes vérifient la* wcis-propriété *si et seulement si G est sans P_4.*

Preuve. Soit G un graphe connexe sans P_4. Supposons qu'il existe un *mis* S non *wcis* de G. D'après le lemme 3.2.2, il existe un sous-ensemble non vide A de S tel que $l^* = \min\{d_G(u, v); u \in A, v \in S \setminus A\} \geq 3$ (cf. Figure 3.7). Comme S est un dominant, nous avons $l^* = 3$. Désormais, la plus courte chaîne entre A et $S \setminus A$ induit un P_4 de G, ce qui contredit l'hypothèse que G est sans P_4.

Maintenant, si G ainsi que tous ses sous-graphes connexes vérifient la *wcis-propriété*, alors G ne peut contenir un P_4 induit puisque P_4 ne vérifie pas la *wcis-propriété*.

□

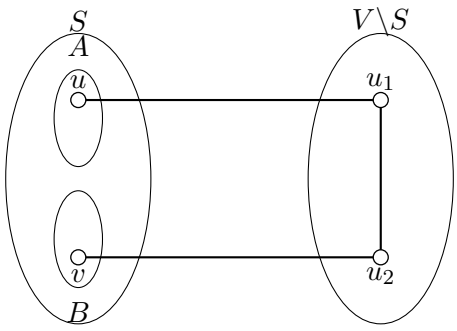


FIGURE 3.7 – Une partition (A,B) de S tel que $d_G(A, B) = 3$.

Évidemment, dans un graphe sans P_4, le problème de la détermination d'un *mis* minimum et d'un *wcis* minimum ont la même complexité polynomiale [27].

Chapitre 4

Algorithmes pour le problème du wcis minimum

Sommaire

4.1 Introduction

Dans des problèmes d'indépendants, des algorithmes triviaux qui énumèrent simplement les sous-ensembles de sommets et recherchent les solutions possibles peuvent être appliqués. Ainsi, toutes les solutions peuvent être obtenues en $O^*(2^n)$. Mais, il est possible de concevoir des algorithmes qui sont nettement plus rapides que la recherche exhaustive, bien que non polynomiaux ([9],[31]).
Dans[46], Moon et Moser ont montré que le nombre d'indépendants dominant d'un graphe de n sommets est borné par 1.443^n. Johnson et alii. [41] ont donné un algorithme pour générer tous les ensembles indépendants maximaux d'un graphe dans l'ordre lexicographique, à délai polynomial (c'est-à-dire l'intervalle de temps entre les générations de deux indépendants maximaux successifs est polynomial) . Dans [31], Gaspers et Liedloff présentent un algorithme en $O^*(1.357^n)$ pour résoudre le problème de l'indépendant dominant minimum. Récemment,

Bourgeois et alii. [9] ont amélioré ce résultat avec un algorithme de branchement qui calcule l'indépendant dominant minimum avec une complexité temps de $O^*(1.335^n)$.
La section 4.2 décrit un algorithme d'énumération implicite pour le problème du *wcis* minimum et analyse sa performance théorique. Le pseudo code de cette procédure se trouve dans la section 4.3. Dans 4.4 on donne une heuristique. Enfin, les résultats expérimentaux sont dans la section 4.5.

4.2 Algorithme d'énumération implicite

Nous présentons un algorithme exact en $O^*(1.4655^n)$ pour résoudre le problème de l'indépendant faiblement connexe de cardinalité minimum.

Pour un graphe non orienté et connexe $G = (V, E)$, soit $n = |V|$ et $m = |E|$. Notons par $T(n)$ le temps au pire des cas pour un algorithme pour résoudre une instance de taille au plus n sommets. Lorsque le calcul d'une solution sur une instance de n sommets se fait en un temps d'exécution qui est au plus le temps pour l'exécution d'une séquence de k instances de tailles respectives $n - \alpha_1, \ldots, n - \alpha_k$, on peut alors écrire

$$T(n) \leq \sum_{i=1}^{k} T(n - \alpha_i) + p(n)$$

où $p(n)$ est un polynôme. Par la suite, le temps d'exécution $T(n)$ est borné par $O^*(c^n)$ où le facteur de branchement c est la racine maximum de l'équation $\sum_{i=1}^{k} \frac{1}{x^{\alpha_i}} = 1$.

Notre algorithme d'énumération est basé sur un arbre binaire de recherche implicite [33]. Un indépendant W est dit *wcis partiel* de G si le sous-graphe $G_W = (W \cup N(W), [W, N(W)])$ est connexe. Évidemment, si $W \cup N(W) = V$, alors W est un *wcis* de G. Une *complétion* d'un *wcis* partiel W est un sous-ensemble de sommets C de $V \setminus W$ pour lequel $W \cup C$ est un *wcis* de G. Nous avons le lemme suivant.

Lemme 4.2.1. *Tout wcis partiel d'un graphe connexe peut être complété.*

Au cours de la procédure d'énumération, chaque nœud de l'arbre est caractérisé par une *solution partielle*. Une solution partielle L est une liste ordonnée de sommets de V, candidats à être dans le *wcis* partiel W_L, ou interdits dans n'importe quelle complétion de W_L. Un sommet interdit u de L est noté $\overline{u}$.

Notons par V_L l'ensemble $W_L \cup N(W_L)$ et par F_L l'ensemble de sommets interdits provenant de L. Un nœud appartenant à $V \setminus (V_L \cup F_L)$ est un nœud *libre*. Un nœud libre v est *accessible* si $v \in N^2(u)$ pour $u \in W_L$. Soit A_L l'ensemble des sommets libres accessibles de W_L. Notons que $W_L \cup \{v\}$ est un *wcis* partiel pour tout $v \in A_L$. Ainsi, à un nœud de l'arbre, la décision est d'ajouter ou pas un sommet $v_0 \in A_L$ dans un *wcis* partiel. Le sous-arbre droit d'un nœud de l'arbre d'énumération est alors formé par tous les *wcis* contenant v_0. tandis que le sous-arbre gauche est constitué par tous les *wcis* ne contenant pas v_0.

Une complétion L' d'une solution partielle L est une liste de sommets tel que L est le préfixe de L' et $A_{L'} = \emptyset$. Ainsi, une solution partielle L détermine au plus $2^{n-|V_L \cup F_L|}$

différentes complétions. Une complétion L' est dite *réalisable* si $W_{L'} \setminus W_L$ est une complétion de W_L, i.e. $W_{L'}$ est un *wcis*.

Par exemple, soit G un graphe avec V=$\{u_1, u_2, u_3, u_4, u_5\}$, et E=$\{(u_1, u_2), (u_1, u_3), (u_2, u_3), (u_3, u_4), (u_3, u_5), (u_4, u_5)\}$. Pour $L = \{\overline{u}_1, u_2\}$, nous avons $W_L = \{u_2\}$, $F_L = \{u_1\}$, $A_L = \{u_4, u_5\}$ et $V_L = \{u_1, u_2, u_3\}$ (voir Figure 4.1(a)). Ainsi, u_1 ne peut appartenir à aucune complétion de W_L et le sous-graphe $G_{W_L} = (\{u_2\} \cup \{u_1, u_3\}, \{(u_1, u_2), (u_2, u_3)\})$ est connexe (Voir Figure 4.1(b)). Pour cet exemple, il y a trois complétions, la dernière n'est pas réalisable :

$$\{\overline{u}_1, u_2, u_4\}, \{\overline{u}_1, u_2, \bar{u}_4, u_5\}, \{\overline{u}_1, u_2, \bar{u}_4, \bar{u}_5\}.$$

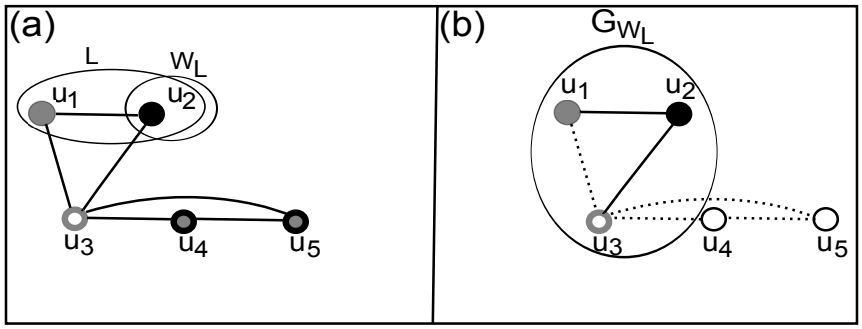


FIGURE 4.1 – (a) $G = (V, E)$. (b) G_{W_L}.

Notre algorithme d'énumération implicite consiste à générer une séquence de solutions partielles. Comme dans les calculs précédents, des complétions réalisables sont découvertes et la meilleure trouvée sera conservée. A chaque étape de l'algorithme, caractérisé par une solution partielle L, nous essayons d'ajouter un sommet accessible v_0 à W_L, sinon on stérilise le nœud L. Ensuite, nous faisons un retour en arrière à chaque stérilisation.

Introduisons quelques notations. Pour un sous-ensemble $S \subseteq V$ et un nœud $v \in V$, nous définissons $N_S(v) = N(v) \cap S$ et $d_S(v) = |N_S(v)|$, le S-degré du nœud v.

4.2.1 Initialisation

Nous choisissons un sommet de degré minimum w_0. Soit $N(w_0) = \{w_1, w_2, \ldots w_{\delta(G)}\}$. Tout *wcis* de G doit contenir w_0 ou un voisin de celui-ci. En effet, $\mathcal{W}(G)$ peut être partitionné en $\delta(G) + 1$ ensembles. Chacun d'entre eux est identifié par une solution partielle *initiale* de la forme :

$$L_0 = \{w_0\} \text{ ou } L_0^k = \{\overline{w_0}, \overline{w_1}, \ldots \overline{w_{k-1}}, w_k\}, \text{ pour } 1 \leq k \leq \delta(G).$$

Notre algorithme utilise successivement ces $\delta(G) + 1$ solutions partielles comme liste initiale.

4.2.2 Itération

Notons L la solution partielle courante. L peut être stérilisée dans l'un des cas suivants :

- condition $(F1)$ $V_L = V$;
- condition $(F2)$ $A_L = \emptyset$, et $V_L \neq V$;
- condition $(F3)$ $\exists u \in F_L, N(u) \subset (N(W_L) \cup F_L)$.

En effet, la condition $(F1)$ indique que W_L est un *wcis*, qui remplace éventuellement la meilleure solution connue si elle est plus petite. Avec la condition $(F2)$, L est une complétion non réalisable pour la liste courante initiale. Un sommet interdit vérifiant la condition $(F3)$ ne peut pas être dominé dans n'importe quelle complétion L' de L, car $W_L \subset W_{L'}$ et $F_L \subset F_{L'}$. Nous pouvons donc revenir en arrière dans l'arbre d'énumération.

Supposons maintenant qu'aucune des conditions ci-dessus n'est satisfaite. Nous sélectionnons un nœud accessible v_0 satisfaisant :

- Règle $(R1)$ $d_{V\setminus(V_L \cup F_L)}(v_0) = 0$,
- Règle $(R2)$ $d_{V\setminus(V_L \cup F_L)}(v_0) = 1$,
- Règle $(R3)$ $d_{V\setminus(V_L \cup F_L)}(v_0) \geq 2$.

4.2.2.1 Branchement sur un sommet selon la règle $(R1)$

Lemme 4.2.2. *Supposons qu'un nœud accessible v_0 n'est pas adjacent à un sommet libre. Alors nous pouvons ajouter v_0 sans branchement.*

Preuve. Comme v_0 est un nœud accessible dont ces voisins sont inclus dans $N(W_L) \cup F_L$, tous les prolongements en *wcis* de W_L doivent contenir ce sommet (voir Figure 4.2). Donc la solution partielle $L' = L \cup \{\overline{v_0}\}$ n'admet pas de solution réalisable. □

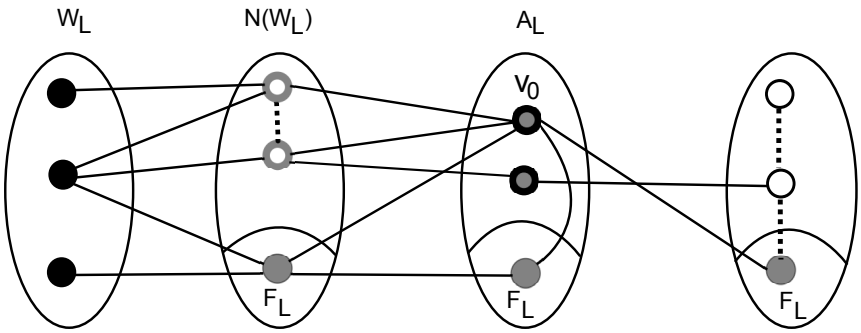


FIGURE 4.2 – Branchement sur un sommet selon la règle $(R1)$

4.2.2.2 Branchement sur un sommet selon la règle $(R2)$

Lemme 4.2.3. *Supposons qu'un nœud accessible v_0 est adjacent à exactement un sommet libre. Nous pouvons éliminer au moins deux sommets et $T(p) \leq 2T(p-2)$. Ainsi on obtient un facteur de branchement $\lambda \leq \sqrt{2} = 1.4142$.*

Preuve. Étant donné une solution partielle L, soit v_0 un sommet accessible de telle sorte que $N(v_0) \setminus (N(W_L) \cup F_L) = \{x_0\}$. Ainsi, avec $L' = L \cup \{v_0\}$ nous avons $T(p) \leq T(p-2)$. Supposons maintenant que v_0 est interdit. Considérons une solution partielle L'' qui admet

$L \cup \{\overline{v_0}\}$ comme préfixe. Supposons que $W_{L''}$ ne peut pas contenir x_0, i.e $x_0 \in F_{L''}$ ou $x_0 \in N(W_{L''})$. Comme $N(v_0) \subset (N(W_L) \cup F_L \cup \{x_0\}) \subset (N(W_{L''}) \cup F_{L''})$, v_0 satisfait la condition (F3) pour L''. Ceci implique que $T(p) \leq T(p-2)$. Enfin, le branchement donne $T(p) \leq 2T(p-2)$ (voir Figure 4.3). □

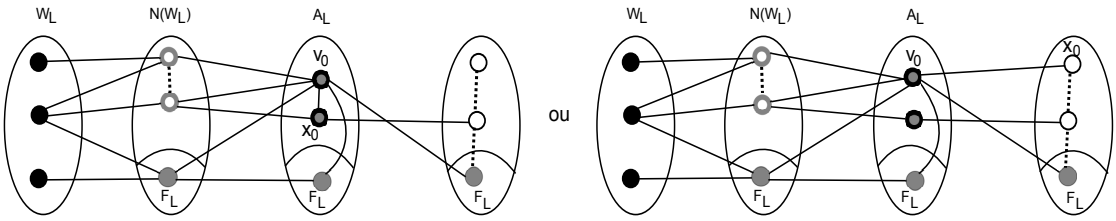


FIGURE 4.3 – Branchement sur un sommet selon la règle ($R2$)

4.2.2.3 Branchement sur un sommet selon la règle ($R3$)

Considérons un nœud v_0 satisfaisant la règle ($R3$). Quand nous prenons v_0 dans L, nous devons éliminer au moins deux sommets libres (voir Figure (4.4)). Donc, nous obtenons $T(p) \leq T(p-3) + T(p-1)$. Ici, le facteur de branchement λ est inférieur à 1.4655. Par conséquent, nous obtenons comme conséquence immédiate des lemmes 4.2.2 et 4.2.3 le théorème suivant.

Théorème 4.2.1. *L'algorithme d'énumération implicite résout le problème de l'indépendant faiblement connexe minimum en $O(n^2)$ en espace mémoire et en $O^*(1.4655^n)$ en temps d'exécution.*

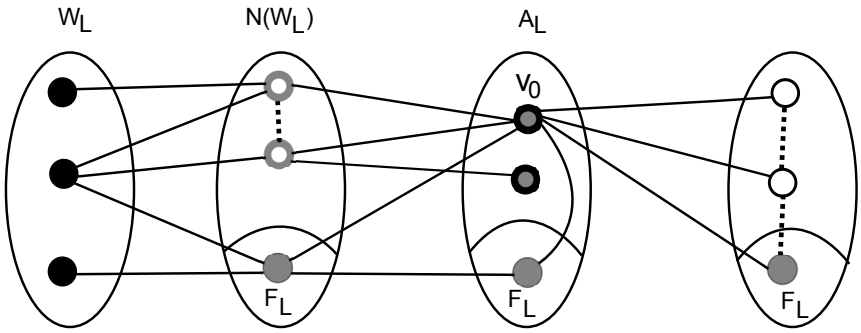


FIGURE 4.4 – Branchement sur un sommet selon la règle ($R3$)

4.3 Pseudo code de l'algorithme d'énumération implicite

On considère un graphe $G = (V, E)$ non orienté et connexe. L'algorithme que l'on va décrire permet de déterminer tous les ensembles indépendants faiblement connexes de G de

cardinalité minimum. On note $MWCIS(G)$ l'indépendant faiblement connexe de cardinalité minimum dans le graphe $G = (V, E)$. L'algorithme utilise pour cette énumération, un arbre binaire dans lequel le branchement se fait sur la décision de prendre ou pas un sommet dans la solution courante. On dira dans ce dernier cas que le sommet est *interdit*. Tout au long de l'exploration en profondeur de cet arbre, on assure :

- la propriété d'indépendance en éliminant l'ensemble des voisins d'un sommet dès qu'il est pris dans le *wcis*.
- la faible connexité en désignant comme nouveaux candidats à entrer dans le *wcis*, les voisins des voisins d'un sommet pris dans le *wcis*.

Un parcours en profondeur entamé aboutit à une des situations suivantes :

- Le parcours est un succès car il a permis de détecter un meilleur ou un aussi bon *wcis* de G que ceux obtenus jusque là,
- La cardinalité du *wcis* courant dépasse celle du plus petit obtenu jusque là.
- le parcours est un échec car la propriété de dominance n'est plus assurée dans la suite (procédure EchecDominance(v_0)),
- l'ensemble des sommets comprend ceux du *wcis* courant et ceux qu'il recouvre mais il reste des sommets non recouvert et interdits.

L'algorithme choisit un sommet initial v_0 de degré minimum $\delta(G)$ dans G pour démarrer l'exploration. Soit $N(v_0) = \{u_1, u_2, \ldots, u_{|\delta(G)|}\}$. On va successivement explorer les ensembles suivants :

Tous les *wcis* contenant v_0,

Tous les *wcis* ne contenant pas v_0 mais contenant u_1,

Tous les *wcis* ne contenant ni v_0 ni u_1 mais contenant u_2,

...

Tous les *wcis* ne contenant ni v_0, ni u_1, ni u_2 , ..., ni $u_{|\delta(G)|-1}$ mais contenant $u_{|\delta(G)|}$,

On utilisera aussi les tableaux :

- Le tableau M des *Maîtres*, sommets du *wcis*, $S \subset V$, courant tel que $M(u) = 1$ si $u \in S$ et $M(u) = 0$ sinon,
- Le tableau E des *Esclaves*, sommets voisins des sommets du *wcis* courant tel que $E(u) = 1$ si $u \in N(S)$ et $E(u) = 0$ sinon,
- Le tableau W des *Interdits*, sommets interdits de faire partie du *wcis* $S \subset V$ courant tel que $W(u) = 1$ si u est interdit et $W(u) = 0$ sinon,
- Le tableau A des *Accessibles*, sommets voisins des voisins du *wcis* $S \subset V$ courant tel que $A(u) = 1$ si u est accessible $A(u) = 0$ s'il peut devenir accessible et $A(u) = -1$ s'il ne peut plus être accessible à partir du *wcis* courant.
- Le tableau $Pere_E$ qui indique pour un sommet u esclave, le sommet v qui le rend esclave (son prédécesseur direct dans l'arbre) tel que $Pere_E(u) = v$ et $Pere_E(u) = -1$ si le sommet u n'a pas de prédécesseur.
- Le tableau $Pere_A$ qui indique pour un sommet u accessible, le sommet v qui le rend accessible tel que $Pere_A(u) = v$ et $Pere_A(u) = -1$ si le sommet u n'est pas accessible.

On note $\bar{N}(v) = N(v) \setminus (M \cup E)$ et $\bar{N}^2(v) = N^2(v) \setminus (M \cup E)$.

Algorithm 1 Algorithme d'énumération implicite : Programme principal

Require : $G = (V, E)$ un graphe non orienté et connexe.
Ensure : $\mathcal{M}^* = \{M^* \subset V : M^* \text{ est un } MWCIS(G)\}$.

```
1 : wcis_min ← ∞ ;                                   ▷ valeur minimum d'un WCIS
2 : SomInit(v_min, δ(G)) ;                           ▷ renvoie le sommet de plus petit degré
3 : for iv_0 = 0 to δ(G) do
4 :     Init ;                                       ▷ Initialisation des structures
5 :     CreerSituationInitiale(v_min, iv_0, v_0 ) ; ▷ v_0, voisin de v_min est la racine de l'arbre
6 :     stop ← false ;                               ▷ Booléen pour l'arrêt du parcours de l'arbre
7 :     while not stop do
8 :         Empiler(v_0) ;                           ▷ Exploration branche v_0
9 :         AjoutMaitre(v_0) ;                       ▷ Ajout maitre v_0 + esclaves + accessibles
10 :        if EchecDominance(v_0) then
11 :            InterdireMaitre(v_0) ;               ▷ branche v_0 stérile
12 :        else
13 :            if |M| + |E| = |V| then
14 :                if |M| < wcis_min then
15 :                    wcis_min ← |M| ;
16 :                    EnregistrerSolution ;        ▷ Solution Acceptée
17 :                end if
18 :                InterdireMaitre(v_0) ;           ▷ branche v_0 explorée
19 :            else
20 :                if |M| ≥ wcis_min then
21 :                    InterdireMaitre(v_0) ;       ▷ branche v_0 stérile
22 :                end if
23 :            end if
24 :        end if
25 :        gparcours ← true ;
26 :        while (gparcours ) do
27 :            RecherchNouvoMaitre(Nouv,v_0) ;
28 :            if Nouv then
29 :                gparcours ← false ;              ▷ v_0 est un nouveau maitre
30 :            else
31 :                LiberationSommetPile(maitre, v_0) ;
32 :                if maitre then
33 :                    InterdireMaitre(v_0) ;       ▷ branche v_0 stérile
34 :                else
35 :                    gparcours ← false ;
36 :                    stop ← true                  ▷ fin de parcours de l'arbre
37 :                end if
38 :            end if
39 :        end while
40 :    end while
41 : end for
```

Algorithm 2 Procédure recherche degré min

Ensure : Sommet $v_{\min}$ de degré minimum dans G ;

```
1 : procedure SomInit(v_min, δ(G))
2 :     δ(G) ← ∞ ;
3 :     for (v ∈ V) do
4 :         if |N(v)| < δ(G) then
5 :             δ(G) ← |N(v)| ; v_min ← v ;
6 :         end if
7 :     end for
8 : end procedure
```

Algorithm 3 Procédure d'initialisation

Ensure : Initialisation des tableaux.

```
1 : procedure Init
2 :     |M| ← 0 ; |E| ← 0 ; |W| ← 0 ; |A| ← 0 ;
3 :     for (v ∈ V) do
4 :         M(v) ← 0 ; E(v) ← 0 ; A(v) ← 0 ; W(v) ← 0 ;
5 :         Pere_E(v) ← −1 ; Pere_A(v) ← −1 ;
6 :     end for
7 : end procedure
```

Algorithm 4 Procedure d'initialisation à partir d'un sommet v_0

Require : sommet $v_{\min}$ et l'indice iv_0

Ensure : la creation du premier maitre v_0.

```
1 : procedure CreerSituationInitiale(v_min, iv_0, v_0))
2 :     if iv_0 = 0 then v_0 ← v_min ;
3 :     else
4 :         InterdireSommet(v_min) ; Empiler(v_min) ;
5 :         for i = 1 to iv_0 − 1 do
6 :             v = N(v_min)(i) ;
7 :             InterdireSommet(v) ; Empiler(v) ;
8 :         end for
9 :         v_0 ← N(v_min)(iv_0) ;
10 :    end if
11 :    A(v_0) ← 1 ; |A| ← 1 ;
12 : end procedure
```

Algorithm 5 Procédure Ajouter un maître

Require : sommet v_0 à ajouter dans WCIS

```
1 : procedure AjoutMaitre(v_0)
2 :     RendreMaitre(v_0) ;
3 :     RendreEsclaveVoisin(v_0) ;
4 :     RendreAccessible2voisin(v_0) ;
5 : end procedure
```

Algorithm 6 Procédure RendreMaitre(v_0)

Require : sommet v_0 à ajouter dans WCIS

```
1 : procedure RendreMaitre(v_0)
2 :     M(v_0) ← 1 ; |M| ← |M| + 1 ;
3 :     A(v_0) ← −1 ; |A| ← |A| − 1 ;                    ▷ v_0 était accessible
4 : end procedure
```

Algorithm 7 Procédure RendreEsclaveVoisin(v_0)

Require : sommet v_0 à ajouter dans WCIS

```
1 : procedure RendreEsclaveVoisin(v_0)
2 :     for (v ∈ N̄(v_0)) do
3 :         E(v) ← 1 ; Pere_E(v) ← v_0 ;
4 :         |E| ← |E| + 1 ;
5 :         if A(v) ≠ −1 then
6 :             if A(v) = 1 then
7 :                 |A| ← |A| − 1 ;
8 :             end if
9 :             A(v) ← −1 ;
10 :        end if
11 :    end for
12 : end procedure
```

Algorithm 8 Procédure RendreAccessible2voisin(v_0)

Require : sommet v_0 à ajouter dans WCIS

```
1 : procedure RendreAccessible2voisin(v_0)
2 :     for (v ∈ N̄²(v_0)) do
3 :         if W(v) = 0 then
4 :             if A(v) = 0 then
5 :                 A(v) ← 1 ; Pere_A(v) ← v_0 ; |A| ← |A| + 1 ;
6 :             end if
7 :         end if
8 :     end for
9 : end procedure
```

Algorithm 9 Fonction Echec de domination

Require : le sommet v_0, nouveau maitre, peut induire un ensemble non dominant.

Ensure : fonction renvoie vrai si echec de dominance

```
1 : function EchecDominance(v_0)          ▷ on peut remplacer N̄(v_0) par N(v_0)
2 :     echec ← (∃v ∈ N̄²(v_0) ∩ W, N(v) ⊂ E ∪ W ∪ N̄(v_0)) ;
3 :     return(echec) ;
4 : end function
```

Algorithm 10 Procédure Interdire un maître

Ensure : interdit le sommet maitre v_0 car echec Dominance ou backtrack

```
1 : procedure InterdireMaitre(v_0)
2 :     InterdireSommet(v_0) ;
3 :     M(v_0) ← 0 ; |M| ← |M| − 1 ;
4 :     LibererEsclaveVois(v_0) ;
5 :     LibererAccessible2vois(v_0) ;
6 : end procedure
```

Algorithm 11 Procédure Interdire un sommet

Ensure : interdit le sommet v_0 qui n'était pas obligatoirement inaccessible

```
1 : procedure InterdireSommet(v_0)
2 :     W(v_0) ← 1 ; |W| ← |W| + 1 ;
3 :     if A(v_0) = 1 then|A| ← |A| − 1 ;
4 :     end if
5 :     A(v_0) ← −1 ;
6 : end procedure
```

Algorithm 12 Procédure libérer les voisins d'un sommet

Ensure : libère les voisins dominés par le sommet v_0

```
1 : procedure LibererEsclaveVois(v_0)
2 :     for v ∈ N(v_0) do
3 :         if Pere_E(v) = v_0 then
4 :             Pere_E(v) ← −1 ;
5 :             E(v) ← 0 ;
6 :             |E| ← |E| − 1 ;
7 :             if W(v) = 0 then
8 :                 if (Pere_A(v) ≠ −1) and (A(v) = −1) then
9 :                     A(v) ← 1 ; |A| ← |A| + 1 ;
10 :                else
11 :                    A(v) ← 0 ;
12 :                end if
13 :            end if
14 :        end if
15 :    end for
16 : end procedure
```

Algorithm 13 Procédure libérer les 2voisins accessibles depuis un sommet

Ensure : libère les 2voisins accessibles du sommet v_0

```
1 : procedure LibererAccessible2vois(v_0)
2 :     for (v ∈ N^2(v_0)) do
3 :         if (Pere_A(v) = v_0 ) and (A(v) = 1) then
4 :             Pere_A(v) ← −1 ;
5 :             A(v) ← 0 ; |A| ← |A| − 1 ;
6 :         end if
7 :     end for
8 : end procedure
```

Algorithm 14 Procédure Rechercher nouveau maître, à partir d'un maître v_0

Ensure : Sommet v_0 nouveau maître et sinon Nouv à faux ;

```
1 : procedure RecherchNouvoMaitre(Nouv,v_0)
2 :     if W(v_0) = 1 then
3 :         if N(v_0) ⊆ E ∪ W then
4 :             Nouv ← false ; v_0 ← −1 ;                    ▷ branche v̄_0 stérile
5 :         else
6 :             if N(v_0) \ (E ∪ W) = {x_0} then              ▷ x_0 ∉ M
7 :                 if A(x_0) = 1 then
8 :                     Nouv ← true ; v_0 ← x_0 ;             ▷ branche v̄_0 x_0
9 :                 else
10 :                    if d_G(x_0) = 1 then
11 :                        Nouv ← false ; v_0 ← −1 ;         ▷ branche v̄_0 stérile
12 :                    else
13 :                        for t ∈ N(x_0) do
14 :                            if W(t) = 0 then
15 :                                InterdireSommet(t) ;      ▷ branche t stérile
16 :                                Empiler(t) ;              ▷ t ∉ M ∪ E ∪ W
17 :                            end if
18 :                        end for
19 :                        RecherchLibreNouvoMaitre(Nouv,v_0) ;
20 :                    end if
21 :                end if
22 :            else
23 :                RecherchLibreNouvoMaitre(Nouv,v_0) ;
24 :            end if
25 :        end if
26 :    else
27 :        RecherchLibreNouvoMaitre(Nouv,v_0) ;
28 :    end if
29 : end procedure
```

Algorithm 15 Procédure Rechercher librement un nouveau maître

Ensure : Sommet v_0 nouveau maître et sinon Nouv à faux ;

1 : **procedure** RECHERCHLIBRENOUVOMAITRE(Nouv,v_0)
2 : **if** $|A| > 0$ **then** ▷ on va rechercher les sommets de degré min et max parmi les accessibles
3 : Nouv $\leftarrow$ true ;
4 : $\overline{d}_{max} \leftarrow -1$; $\underline{d}_{min} \leftarrow +\infty$;
5 : **for** $v \in V$ **do**
6 : **if** $A(v) = 1$ **then**
7 : $\overline{d}_v \leftarrow |N(v) \setminus E|$;
8 : $\underline{d}_v \leftarrow |N(v) \setminus (E \cup W)|$;
9 : **if** $\overline{d}_v > \bar{d}_{max}$ **then**
10 : $v_{max} \leftarrow v$; $\overline{d}_{max} \leftarrow \overline{d}_v$;
11 : **end if**
12 : **if** $\underline{d}_v < \underline{d}_{min}$ **then**
13 : $v_{min} \leftarrow v$; $\underline{d}_{min} \leftarrow \underline{d}_v$;
14 : **end if**
15 : **end if**
16 : **end for**
17 : **if** $\underline{d}_{min} \leq 1$ **then**
18 : $v_0 \leftarrow v_{min}$
19 : **else**
20 : $v_0 \leftarrow v_{max}$
21 : **end if**
22 : **else**
23 : Nouv $\leftarrow$ false ; $v_0 \leftarrow -1$; ▷ Pas de nouveau maitre
24 : **end if**
25 : **end procedure**

Algorithm 16 Procédure Libérer Sommets dans pile

Ensure : Indique si l'on a trouvé un maître dans la pile et le donne ;

```
1 : procedure LiberationSommetPile(maitre, v_0)
2 :     maitre ← false ; v_0 ← −1 ;
3 :     while (not pilevide) and (not maitre) do
4 :         v ← sommet.pile ;
5 :         if W(v) = 1 then
6 :             depiler ;                                  ▷ Branche v̄ stérile
7 :             W(v) ← 0 ; |W| ← |W| − 1 ;
8 :             if Pere_A(v) ≠ −1 then                     ▷ v n'a pas pu être un esclave
9 :                 A(v) ← 1 ; |A| ← |A| + 1 ;
10 :            else
11 :                A(v) ← 0 ;
12 :            end if
13 :        else
14 :            maitre ← true ;
15 :            v_0 ← v ;
16 :        end if
17 :    end while
18 : end procedure
```

4.4 Algorithme glouton

Nous présentons ici l'heuristique décrite dans [54] où nous avons modifié le choix des candidats parmi les sommets accessibles (ligne 9). L'algorithme met initialement le sommet s de plus petit degré de G dans le *wcis* partiel W (ligne 1). L'ensemble $V^{'}$ correspond à $N(W) \cup W$. L'algorithme sélectionne un nouveau candidat de plus haut degré dans $G(V \backslash V^{'})$ (ligne 9) à partir de l'ensemble de candidats construit (lignes 17-19) tant que $|V^{'}| \neq |V|$. La Figure 4.5 illustre la règle de sélection d'un nouveau candidat.

Algorithm 17 Algorithme glouton

Require : $G(V, E)$ graphe non orienté et connexe.
Ensure : W l'indépendant faiblement connexe de G.
1 : choisir un sommet $s \in V$ de degré minimum dans G ;
2 : $W \leftarrow \{s\}$, $S \leftarrow N(s)$, $V^{'} \leftarrow W \cup S$;
3 : $E^{'} \leftarrow \{(s, u) : u \in N(s)\}$;
4 : $candidats \leftarrow \{\}$;
5 : **for** $((u, v) \in E)$ tel que $(u \notin V^{'})$ et $(v \in V^{'})$ **do**
6 : $candidats \leftarrow candidats \cup \{u\}$;
7 : **end for**
8 : **while** $(|V^{'}| \neq |V|)$ **do**
9 : $m \in \{u : d_{G(V \backslash V^{'})}(u) = \max_{v \in candidats} d_{G(V \backslash V^{'})}(v)\}$;
10 : $W \leftarrow W \cup \{m\}$;
11 : $S \leftarrow S \cup N(m)$;
12 : $V^{'} \leftarrow W \cup S$;
13 : **for** $(v \in N(m))$ **do**
14 : $E^{'} \leftarrow E^{'} \cup (m, v)$;
15 : **end for**
16 : $candidats \leftarrow \{\}$;
17 : **for** $((u, v) \in E)$ tel que $(u \notin V^{'})$ et $(v \in V^{'})$ **do**
18 : $candidats \leftarrow candidats \cup \{u\}$;
19 : **end for**
20 : **end while**

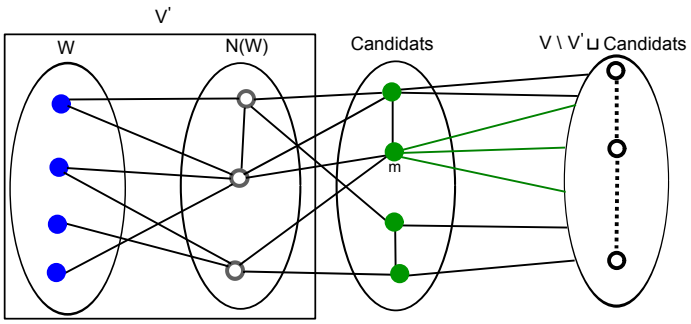


FIGURE 4.5 – Règle de sélection des candidats.

4.5 Résultats expérimentaux

Dans cette section, nous exposons nos instances de graphes et les résultats expérimentaux. Les algorithmes sont implémentés en C. Tous les essais sont effectués sur une Machine HP 8 CPU 2.7 Ghz, AMD Opteron QuadCore, avec 256 Go de RAM dans CentOS 5.5. Nous avons fixé le temps CPU maximum à 6 heures.

4.5.1 Description des instances

Nous utilisons trois classes de graphes pour nos tests : graphes issus de la bibliothèque TSPLIB [1] [53], graphes aléatoires et s-grilles.
En ce qui concerne la première classe, nous avons utilisé la *node-coord-section* proposée par la bibliothèque TSPLIB. Pour tout nœud nous fixons une portée de transmission r. Une arête entre deux nœuds u et v est générée si la distance euclidienne entre u et v est inférieur à r.

Pour les graphes aléatoires, les points sont répartis uniformément dans un carré unité et les liens sont créés en fonction d'un seuil de transmission. Le nombre de nœuds varie entre 50 à 120 et la formule de la densité D est donnée par $D = \frac{2*|E|}{|V|*(|V|-1)}$, est 10%.

Une s-grille $G_{m\times n} = (V_{m\times n}, E_{m\times n})$ est définie comme suit :

$$\begin{aligned} V_{m\times n} &= \{(i,j)|1 \le i \le m, 1 \le j \le n\} \cup \{s\}, \\ E_{m\times n} &= \{((i,j),(i,j+1)); 1 \le i \le m, 1 \le j \le n-1\} \\ &\cup \{\{(i,j),(i+1,j)\}; 1 \le i \le m-1, 1 \le j \le n\} \\ &\cup \{(s,(1,j)); 1 \le j \le n\}. \end{aligned}$$

Ces graphes ne sont pas bipartis et appartiennent à la classe $\mathcal{B}_1$. Ils peuvent modéliser un réseau de capteurs où les capteurs dispersés sur des terres cultivables. Ces dispositifs sont généralement agencés sous la forme d'une grille régulière et ils communiquent avec une station de base (i.e. le nœud s) en dehors du champ.

Chaque instance de graphe est donnée par son nom, suivi par une extension représentant le nombre de nœuds du graphe.

4.5.2 Tests numériques et analyses

Nous donnons également les résultats de l'heuristique, nommée H_{120s} présentée dans la section 4.4. H_{120s} donne la meilleure solution obtenue après plusieurs exécutions de cette heuristique durant deux minutes. Les sommets accessibles sont ajoutés successivement *au hasard* dans le *wcis* courant partiel. Comme la première méthode de recherche construit une solution réalisable très rapidement, nous gardons aussi le meilleur *wcis* trouvé après deux minutes de temps d'exécution de l'algorithme d'énumération. Cette solution est notée $A_{120s}(G)$.
Pour une instance de graphe $G = (V, E)$, notons Opt le nombre de nœuds dans MWCIS(G) construit par l'algorithme exact et par $\sharp Opt$ le nombre total de solutions optimales. Les autres entrées des différents tableaux sont :

1. www2.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

D	:	densité du graphe ($D = \frac{2*\|E\|}{\|V\|*(\|V\|-1)}$) ;
CPU	:	temps d'exécution en h :min :sec ;
TNET	:	nombre total de nœuds dans l'arbre d'énumération (en millions) ;
NFOS	:	nombre de nœuds de l'arbre d'énumération pour trouver la première solution optimale ;
$\frac{NFOS}{TNET}$	:	indique la part de l'arbre d'énumération pour trouver la première solution optimale ;
Gap	:	l'erreur relative entre la solution optimale (lorsque le problème a été résolu à l'optimalité) et la meilleure solution heuristique, donnée par $Gap = \frac{\min(\|A_{120s}(G)\|,\|H_{120s}(G)\|)-Opt}{Opt}$.

Les tables 4.1- 4.3 résument les résultats pour les trois classes de graphes. La table 4.1 montre que l'algorithme d'énumération peut résoudre rapidement les instances dont le nombre de nœuds est inférieur à 70 pour n'importe quelle densité. Pour des instances de tailles plus grandes, l'indépendant faiblement connexe minimum devient plus facile lorsque le nombre d'arêtes dans le graphe augmente, ce qui est illustré par la Figure 4.8 et Figure 4.9. Autour d'une densité de $6\% - 8\%$, les instances de plus de 100 sommets sont très difficiles à résoudre tel qu'il apparaît dans le tableau 4.1 et la Figure 4.9. Les instances indiqués par "*" dans le tableau 4.1 sont ceux dont le temps CPU dépasse 6 heures. Avec une densité de 10%, notre algorithme exact peut traiter des graphes de taille jusqu'à cent sommets dans un délai raisonnable.

La taille moyenne du *wcis* minimum dans les graphes avec une densité fixe $D \geq 10\%$ est relativement constante (Figure 4.11). Le temps CPU grandit de façon exponentielle en fonction de nombre de nœuds (Figure 4.10).
Pour le tableau 4.2, nous générons dix occurrences pour chaque graphe et les résolvons à l'optimalité avec l'algorithme d'énumération.

Nous pouvons voir que le *wcis* obtenu après deux minutes d'exécution de l'algorithme d'énumération est assez bon. Il surpasse la solution donnée par H_{120s} dans presque tous les tests. L'écart entre la solution optimale et le meilleur résultat de l'heuristique est à 14% au pire pour les graphes avec moins de 150 nœuds. Ainsi, une solution tout à fait satisfaisante peut être obtenue très rapidement comme dans de nombreux problèmes combinatoires. Pour les s-grilles, l'algorithme d'énumération trouve une solution optimale au début de l'arbre, mais il fait face à des difficultés lorsque le nombre de nœuds augmente. Quand la taille de graphe dépasse les 100 sommets, ces graphes sont des exemples difficiles pour l'algorithme (pour des densités inférieures à 4 %). En résumé, pour 50% des exemples de la TSPLIB, plus de 60% de l'arbre d'énumération a été nécessaire pour trouver la première solution optimale. La situation pour les graphes aléatoires est un peu médiane.

La Figure 4.7 nous montre l'organisation des sommets du graphe $G_W = (V, \delta(W))$ obtenue grâce à la solution optimale W (sommets noirs). Les nœuds de la Figure 4.7 sont partitionnés en trois catégories : les esclaves (sommets blancs), les maîtres (sommets de W) et les sommets intermédiaires. Les communications dans le graphe G_W se font de la manière suivante : les esclaves échangent seulement avec leur maître. Les maîtres à leur tour communiquent entre eux grâce aux sommets intermédiaires (connexité de G_W).

Instances	D	Opt	♯Opt	CPU	$\frac{NFOS}{TNET}$	A_{120s}	Gap	H_{120s}
eil51	8%	13	2	0 :00 :20	16%	13	0%	13
eil76	11%	12	834	0 :00 :20	0%	12	0%	13
pr76	19%	8	3230	0 :00 :02	7%	8	0%	8
kroA100	5%	20	39672	1 :03 :38	68%	21	5%	21
kroB100	7%	15	52	0 :41 :48	74%	16	7%	17
kroC100	5%	25	1248	0 :03 :14	0%	25	0%	26
kroD100	6%	18	1328	0 :33 :14	71%	19	6%	20
kroE100	7%	16	264	0 :56 :29	95%	17	6%	17
kroA100	10%	11	76596	0 :12 :28	16%	12	9%	12
kroB100	10%	11	1954	0 :10 :24	37%	12	9%	12
kroC100	10%	10	60	0 :15 :34	87%	11	10%	12
kroD100	10%	11	21074	0 :13 :58	2%	11	0%	12
kroE100	10%	11	14070	0 :17 :20	91%	12	9%	12
eil101	10%	12	8	0 :31 :11	65%	13	8%	15
lin105	16%	9	12824	0 :00 :58	0%	9	0%	9
ch130*	8%	-	—	6 :00 :00	—	18	7%	18
ch130	10%	12	154670	5 :59 :50	85%	13	8%	14
pr136	20%	6	376	0 :00 :16	25%	6	0%	7
pr144	13%	10	1644624	0 :08 :39	60%	11	10%	11
pr144	15%	7	787143	0 :11 :54	38%	8	14%	8
ch150*	4%	-	—	6 :00 :00	—	29	4%	29
ch150*	10%	11	8268	6 :00 :00	12%	12	9%	13
ch150	15%	8	47937	1 :19 :45	46%	9	12%	9
kroA150*	4%	-	—	6 :00 :00	—	35	0%	34
kroA150*	10%	-	85608	6 :00 :00	13%	12	9%	13
kroA150	15%	7	1440	0 :20 :52	11%	7	0%	8
kroB150*	5%	-	—	6 :00 :00	—	31	0%	30
kroB150*	10%	-	—	6 :00 :00	—	12	9%	13
kroB150	15%	7	6	0 :54 :04	90%	8	14%	9
pr152	30%	4	924	0 :00 :01	8%	4	0%	4
pr226	15%	8	11075899	1 :00 :50	63%	9	12%	9
pr226	20%	5	842	0 :14 :04	83%	7	20%	6

TABLE 4.1 – Algorithme exact et heuristique pour les instances de la TSPLIB.

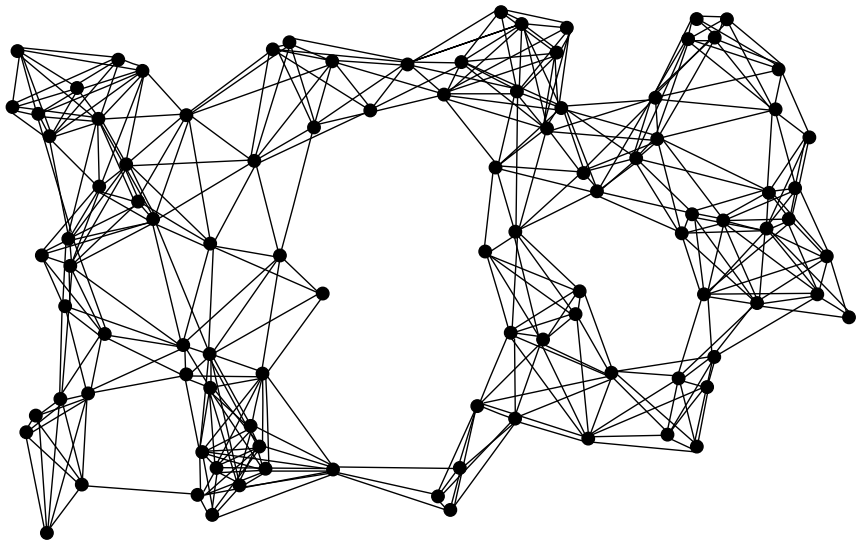


FIGURE 4.6 – Instance kroC100 avec une densité de 10%.

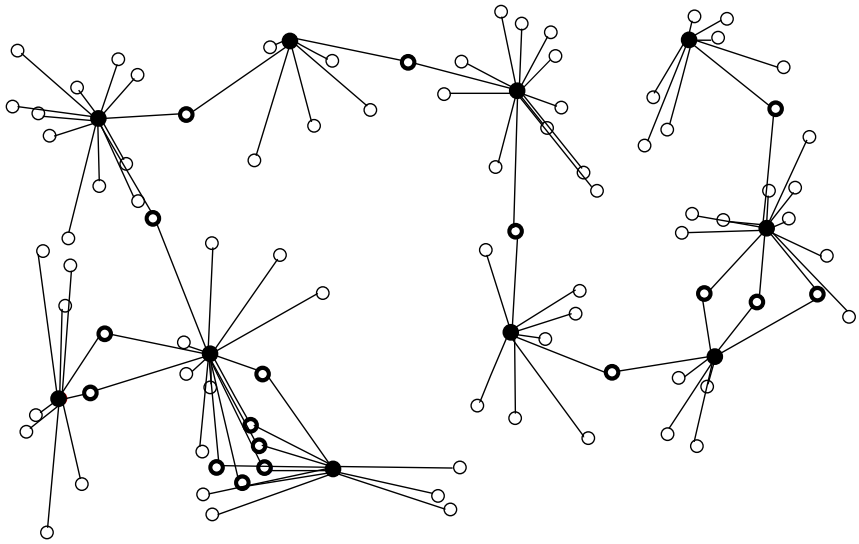


FIGURE 4.7 – Solution optimale de l'instance kroC100.

Graphes	D	Opt	♯Opt	CPU	$\frac{NFOS}{TNET}$	A_{120s}	Gap	H_{120s}
Random50	10%	11.00	378	0 :00 :00	62%	—	—	—
Random60	10%	11.40	534	0 :00 :02	41%	—	—	—
Random70	10%	10.75	7399	0 :00 :05	32%	—	—	—
Random80	10%	10.66	4584	0 :00 :39	25%	—	—	—
Random90	10%	10.50	1819	0 :02 :20	40%	10.75	2%	11.75
Random100	10%	11.33	34499	0 :14 :44	30%	12.10	7%	12.50
Random110	10%	10.88	16925	0 :36 :51	35%	12.10	11%	12.44
Random120	10%	10.90	7985	1 :58 :34	42%	12.10	11%	12.70

TABLE 4.2 – Algorithme exact et heuristique pour les graphes aléatoires.

s-Grids	$\|V\|$	Opt	♯Opt	CPU	$\frac{NFOS}{TNET}$	A_{120s}	Gap	H_{120s}
s-Grid$_{6\times12}$	73	21	802	0 :00 :23	2%	—	—	—
s-Grid$_{12\times6}$	73	24	140	0 :00 :12	5%	—	—	—
s-Grid$_{8\times9}$	73	21	4	0 :00 :21	56%	—	—	—
s-Grid$_{9\times8}$	73	22	196	0 :00 :18	8%	—	—	—
s-Grid$_{5\times16}$	81	22	396	0 :02 :50	0%	22	0%	24
s-Grid$_{16\times5}$	81	25	4	0 :00 :32	0%	25	0%	35
s-Grid$_{8\times10}$	81	24	574	0 :02 :04	3%	24	0%	29
s-Grid$_{10\times8}$	81	25	494	0 :01 :38	4%	25	0%	31
s-Grid$_{8\times11}$	89	25	12	0 :11 :00	23%	26	4%	31
s-Grid$_{11\times8}$	89	27	576	0 :08 :21	26%	28	4%	35
s-Grid$_{6\times16}$	97	27	5372	1 :27 :48	2%	28	4%	31
s-Grid$_{16\times6}$	97	32	936	0 :21 :29	5%	32	0%	41
s-Grid$_{5\times20}$	101	27	1592	4 :55 :38	21%	36	15%	31
s-Grid$_{20\times5}$	101	31	4	0 :21 :45	59%	32	3%	45
s-Grid$_{10\times10}$	101	30	1520	2 :19 :18	4%	32	3%	37

TABLE 4.3 – Algorithme exact et heuristique pour les s-grilles.

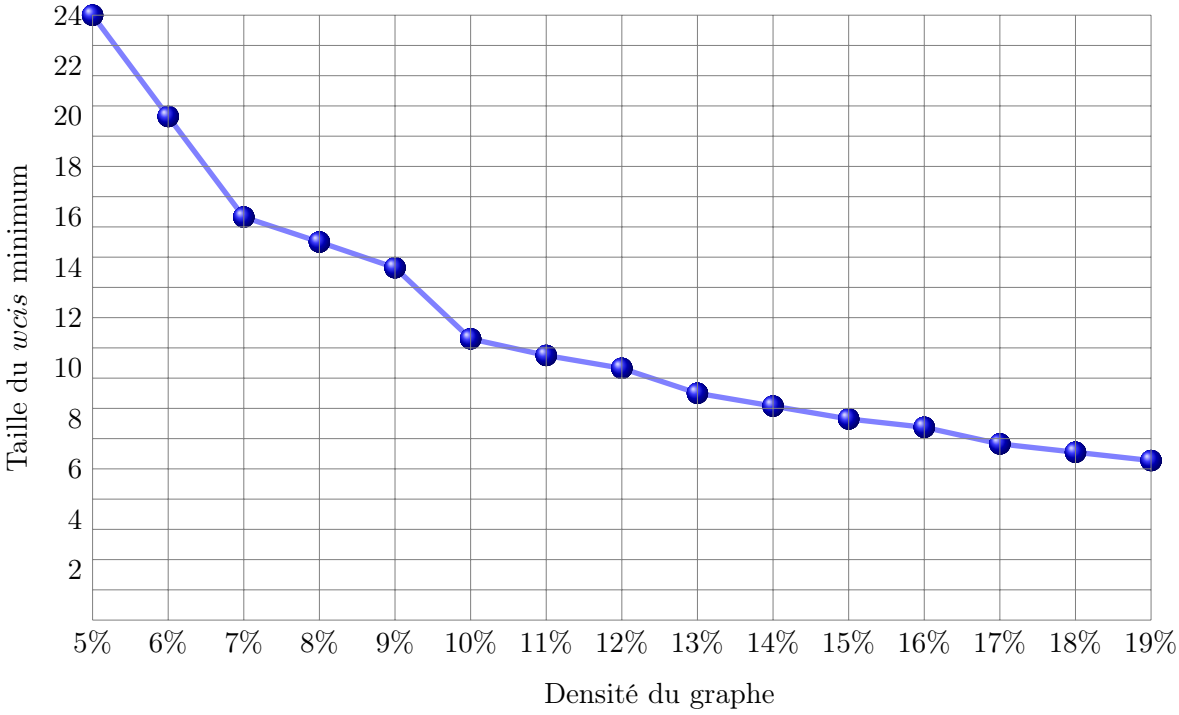


FIGURE 4.8 – Taille moyenne du *wcis* minimum pour $|V| = 120$ et $D_{min} \leq D \leq 19\%$.

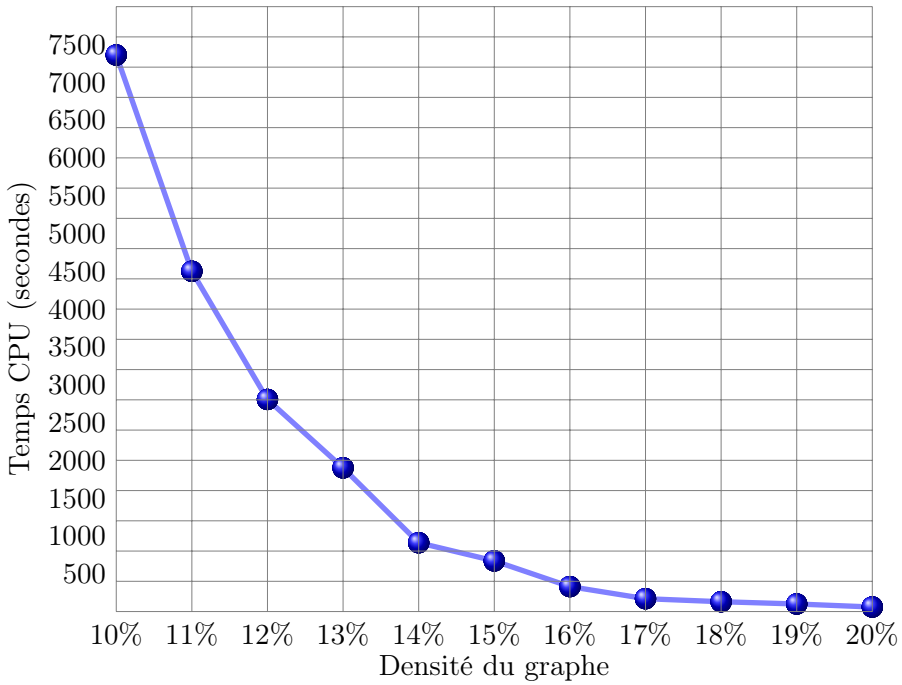


FIGURE 4.9 – Temps moyen de l'algorithme exact pour $|V| = 120$ et $10\% \leq D \leq 20\%$.

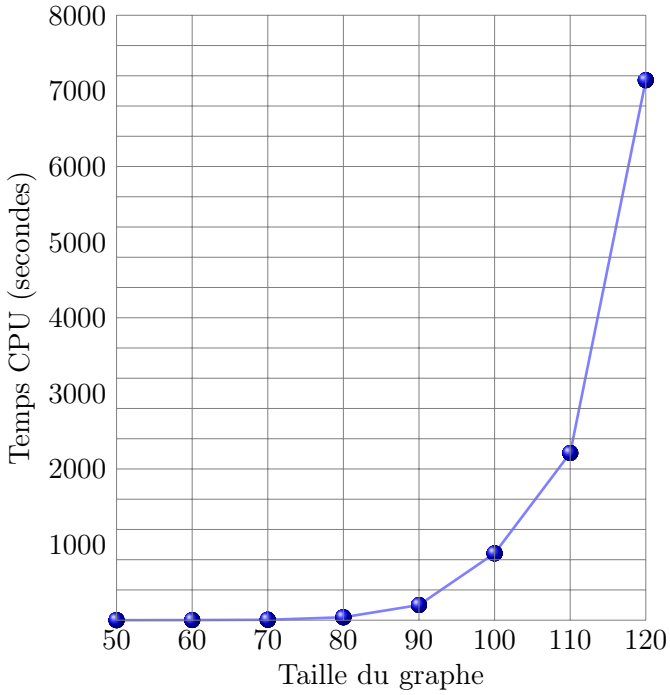


FIGURE 4.10 – Temps moyen de l'algorithme exact pour $D = 10\%$.

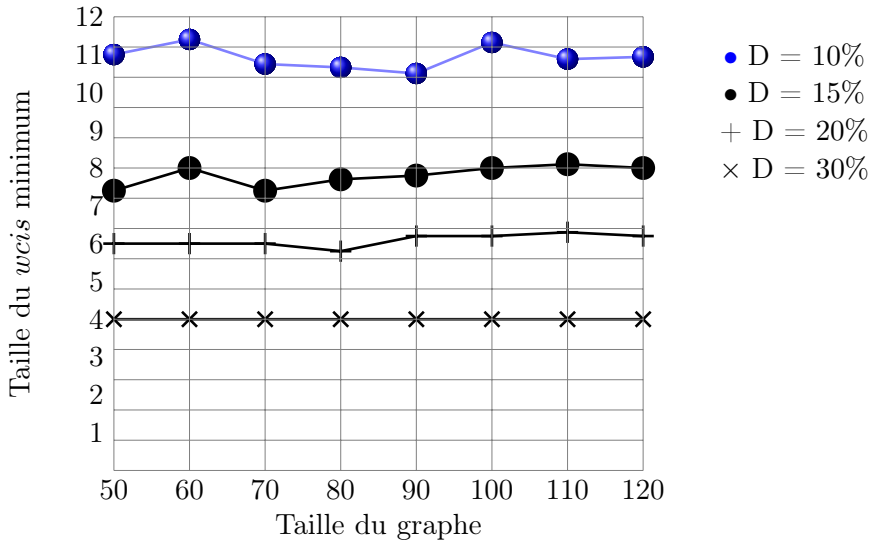


FIGURE 4.11 – *wcis* minimum moyen pour $D \in \{10\%, 15\%, 20\%, 30\%\}$.

4.5.3 Comparaison avec des approches indirectes

Comme un indépendant faiblement connexe est un indépendant maximal, tout algorithme d'énumération pour l'indépendant maximal, combiné avec le test de connexité appliqué pour chaque *mis* détecté, peut être utilisé comme moyen de recherche d'un *wcis* minimum. Nous présentons dans cette partie une comparaison de notre algorithme d'énumération avec l'algorithme de Laforest et Phan [42] et les tests donnés par [44] et [51] présentés dans [42].

Notons que pour cette sous-section, notre programme s'exécute sur une machine Intel(R) Core (TM)2 Duo CPU en 3.00 GHz avec 3.25 GB RAM.

La table 4.4 résume les résultats des temps de calculs en secondes sur les grilles de taille allant de 5×5 à 8×8.

$\|V\|$	5×5	6×6	7×7	8×8
Alg.énumération	0	0	0.02	0.88
Laforest [42]	0	0	9.90	630
IEA [51]	1	254	141242	–
Liu[44]	11	39225	–	–

TABLE 4.4 – Comparaisons de l'efficacité des quatre algorithmes sur les grilles.

La table 4.5 est la comparaison avec l'algorithme de Laforest et Phan [2]. Ces tables montrent

$\|V\|$	80	90	100	110
Alg.énumération	9.95	118.47	284.48	563.89
Laforest [42]	740.20	8049.50	38460.00	126985.00

TABLE 4.5 – Comparaisons des deux algorithmes sur les graphes aléatoires.

que notre algorithme d'énumération implicite pour le problème d'indépendant faiblement connexe est expérimentalement plus efficace que l'approche indirecte basée sur l'implémentation de l'algorithme d'énumération pour le problème d'indépendant maximal [42].

4.6 Perspectives d'amélioration par décomposition de graphe

Nous considérons un graphe non orienté connexe $G = (V, E)$. Notons par $mwcis(G) = \arg\min\{|W|; W \in \mathcal{W}(G)\}$ le *wcis* de G de cardinalité minimum.

4.6.1 Point d'articulation

Pour un $s \in V$ donné, l'ensemble de tous les indépendants faiblement connexes de G qui contient s est noté $\mathcal{W}_s(G)$. Notons par $s\text{-}mwcis(G)$ le *wcis* de $\mathcal{W}_s(G)$ de cardinalité minimum.

2. Nous remercions Raksmey Phan d'avoir mis ses exemples à notre disposition.

Notons par $\overline{s}$-$mwcis(G)$ le $wcis$ de G qui ne contient pas s de cardinalité minimum. Supposons que G admet un point d'articulation s. Soient $V_1, ..., V_k$ les k composantes connexes de $G(V\backslash\{s\})$ et $G_i = (V_i \cup \{s\}, E(V_i) \cup [s, V_i])$ $1 \leq i \leq k$.

Lemme 4.6.1. *On a :*
i) $|s\text{-}mwcis(G)| = 1 - k + \sum_{i=1}^{k} |s\text{-}mwcis(G_i)|$,
ii) $|\overline{s}\text{-}mwcis(G)| = \sum_{i=1}^{k} |\overline{s}\text{-}mwcis(G_i)|$,
iii) $|mwcis(G)|$=min$\{|\ s\text{-}mwcis(G)|, |\overline{s}\text{-}mwcis(G)|\}$.

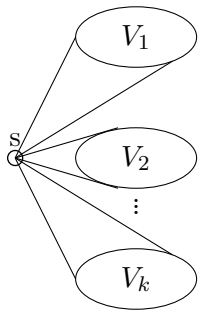


FIGURE 4.12 – $G\backslash\{s\}$ a k composantes connexes.

Preuve.

i) Considérons un $wcis$ W_s de G qui contient s de cardinalité minimum. Posons $W_i = (W_s \cap V_i)$. Il n'est pas difficile de constater que $W_i \cup \{s\}$ est un $wcis$ de G_i, $1 \leq i \leq k$, qui contient s. D'où

$$\begin{aligned} |W_i \cup \{s\}| &= 1 + |W_i|,\ 1 \leq i \leq k \\ &= 1 + |W_s \cap V_i|,\ 1 \leq i \leq k \\ &\geq |s\text{-}mwcis(G_i)|,\ 1 \leq i \leq k \end{aligned}$$

par conséquent

$$\begin{aligned} \sum_{i=1}^{k}(1 + |W_s \cap V_i|) &\geq \sum_{i=1}^{k} |s\text{-}mwcis(G_i)| \\ k + |W_s| - 1 &\geq \sum_{i=1}^{k} |s\text{-}mwcis(G_i)| \end{aligned}$$

finalement

$$|s\text{-}mwcis(G)| \geq \sum_{i=1}^{k} |s\text{-}mwcis(G_i)| - k + 1. \tag{4.1}$$

Soit W_s^i un $wcis$ de G_i, contenant s, de cardinalité minimum. $W_s = \bigcup_{i=1}^{k} W_s^i$ est un

wcis de G qui contient s. D'où

$$\begin{aligned} |W_s| &= |\bigcup_{i=1}^{k} W_s^i| \\ &= 1 + \sum_{i=1}^{k} (|W_s^i| - 1) \\ &= 1 - k + \sum_{i=1}^{k} |s\text{-}mwcis(G_i)| \end{aligned}$$

finalement

$$1 - k + \sum_{i=1}^{k} |s\text{-}mwcis(G_i)| \geq |s\text{-}mwcis(G)|. \tag{4.2}$$

De (4.1) et (4.2), on obtient i).

ii) Considérons un *wcis* de cardinalité minimum $W_{\overline{s}}^i$ de G_i qui ne contient pas s. Alors il existe au moins un sommet $y_i \in N(s)$ qui est dans le $\overline{s}$-$mwcis(G_i)$. Soit $W' = \bigcup_{i=1}^{k} W_{\overline{s}}^i$. Comme $d_G(y_i, y_j) = 2$, $i \neq j$, $1 \leq i$, $j \leq k$, W' est un *wcis* de G, ne contenant pas s. D'où

$$\sum_{i=1}^{k} |W_{\overline{s}}^i| \geq |\overline{s}\text{-}mwcis(G)|. \tag{4.3}$$

Si $W_{\overline{s}}$ est un *wcis* de G ne contenant pas s, alors $W_{\overline{s}} \cap V_i$ est un *wcis* de G_i ne contenant pas s, $1 \leq i \leq k$. Nous déduisons que

$$|W_{\overline{s}}| \geq \sum_{i=1}^{k} |\overline{s}\text{-}mwcis(G_i)|. \tag{4.4}$$

(4.3) et (4.4) impliques ii).

□

4.6.2 G a des sommets jumeaux

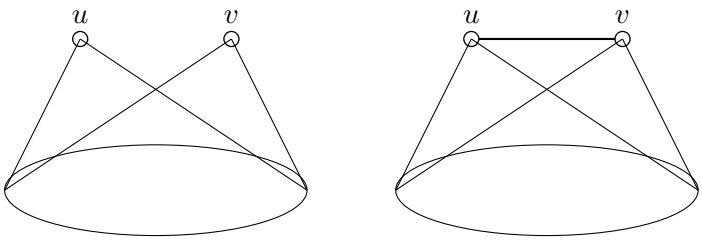
Lemme 4.6.2. *Soit $G = (V, E)$ un graphe non orienté connexe. Supposons qu'il existe deux jumeaux u et v dans G, alors :*

i) $mwcis(G) = mwcis(G \backslash v)$ si u et v sont des vrais jumeaux.

ii) $|mwcis(G)| = \min\{|u\text{-}mwcis(G \backslash v) \cup \{v\}|, |\overline{u}\text{-}mwcis(G \backslash v)|\}$ si u et v sont des faux jumeaux.

Preuve.

i) Ce résultat est dû au fait qu'on peut échanger les deux sommets u et v dans le $mwcis(G)$.



Faux jumeaux Vrais jumeaux

ii) Considérons un $wcis(G)$ de cardinalité minimum, on a deux cas possibles : soit $u \in mwcis(G)$ alors on est obligé de prendre v dans $mwcis(G)$ puisque $(u, v) \notin E$ et $N(u) = N(v)$. Dans ce cas, $mwcis(G) = u\text{-}mwcis(G\backslash v) \cup \{v\}$. Si $u \notin mwcis(G)$ alors il existe au moins un sommet $x \in N(u) \cap mwcis(G)$ et par conséquent le sommet v sera dominé par x puisque $N(u) = N(v)$, dans ce cas $mwcis(G) = \overline{u}\text{-}mwcis(G\backslash v)$.

□

4.6.3 G a un module M

Notons $mmis(G)$ l'indépendant maximum de cardinalité minimale d'un graphe G. Soit $G^{'}$ le graphe déduit de G par contraction de M en un sommet w_M. On a

$V(G^{'}) = (V\backslash M) \cup \{w_M\}$;

$E(G^{'}) = E(V\backslash M) \cup \{(w_M, u) : u \in N(M)\}$.

On remarque que les ensembles $((w_M\text{-}mwcis(G^{'}))\backslash\{w_M\}) \cup mids(M)$ et $\overline{w_M}\text{-}mwcis(G^{'})$ sont des $wcis$ de G. D'autre part, si $mwcis(G)$ contient un sommet m de M, $mwcis(G) \cap M$ est un indépendant maximal de M, ce qui implique $(mwcis(G)\backslash M) \cup \{w_M\}$ est un $wcis$ de $G^{'}$. Sinon, $mwcis(G)$ peut être considéré aussi comme un $wcis$ de $G^{'}$. On obtient ainsi :

Lemme 4.6.3.

$$|mwcis(G)| = \min\{|mids(M)| + |w_M\text{-}mwcis(G^{'})| - 1, |\overline{w_M}\text{-}mwcis(G^{'})|\}.$$

4.6.4 Exemple d'application de la décomposition de graphe

Après l'exécution de l'algorithme exact sur les deux graphes G_1, G_2 nous obtenons le $wcis$ minimum dans chacun des graphes (voire Figure 4.16).

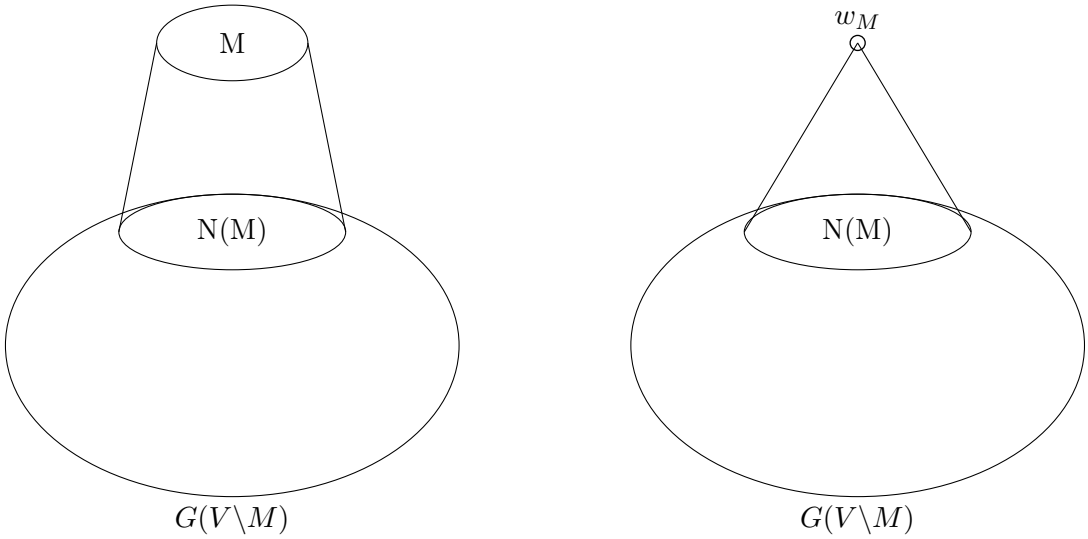


FIGURE 4.13 – Le graphe G avec son module M et le graphe $G^{'}$.

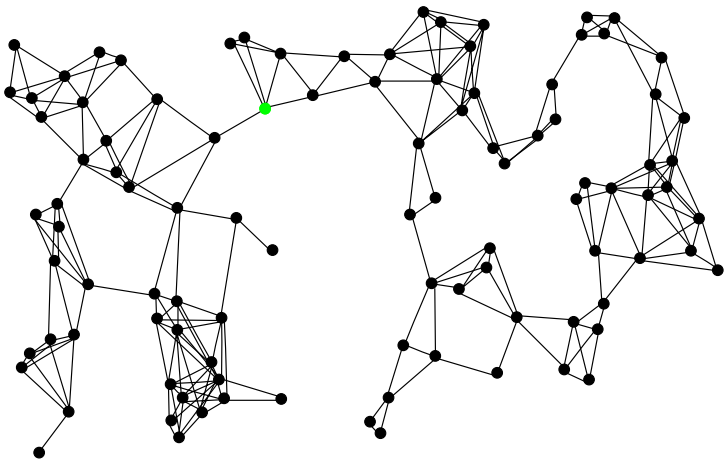


FIGURE 4.14 – Le graphe G avec le point d'articulation en gris.

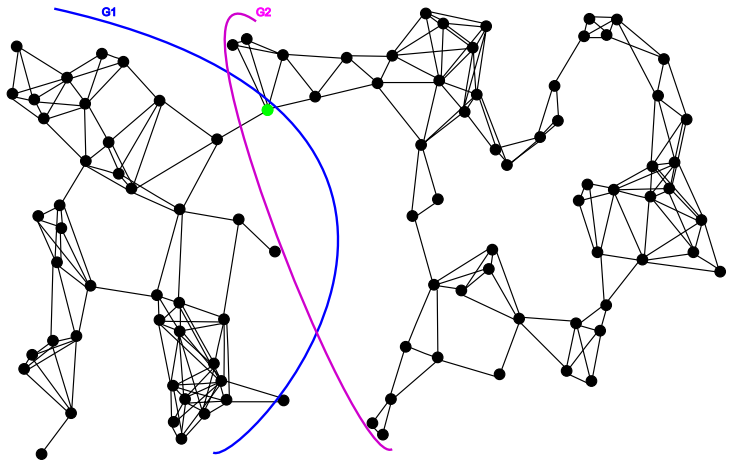


FIGURE 4.15 – Décomposition en deux sous-graphes G_1, G_2.

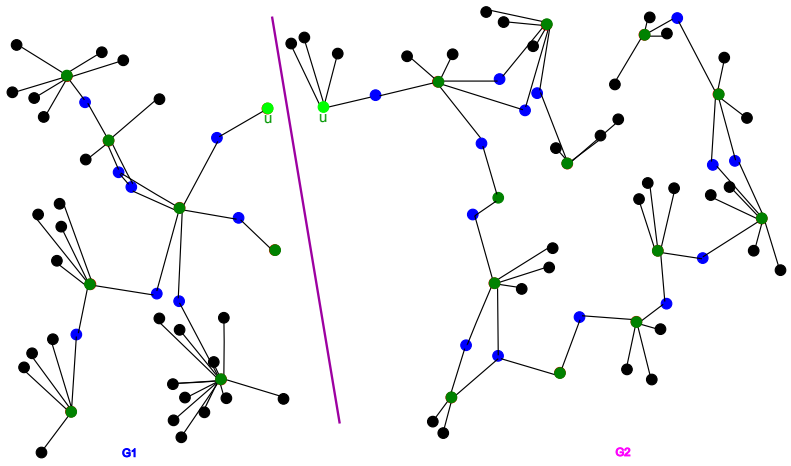


FIGURE 4.16 – Le *wcis* minimum dans les deux graphes G_1, G_2.

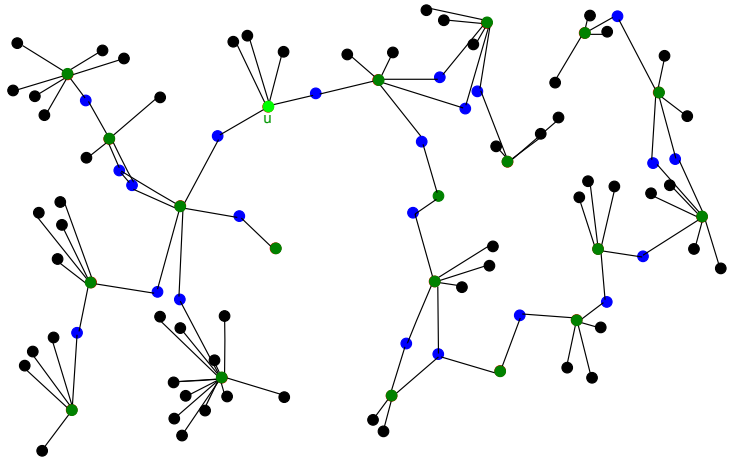


FIGURE 4.17 – Le *wcis* minimum dans le graphe G

Chapitre 5

Approche polyèdrale

Sommaire

5.1 Introduction

Farber [25] propose un algorithme linéaire en $O(|V| + |E|)$ pour la recherche de l'indépendant dominant minimum dans un graphe triangulé si les poids sur les sommets du graphe sont entiers. Dans [8] les auteurs montrent que les inégalités de cycles impairs définissent des facettes du polytope des indépendants dans un graphe série-parallèle. Des formulations en nombres entiers ont été proposées dans [32, 58] pour le problème de dominant connexe de cardinalité minimum dans un graphe. Dans ce chapitre, nous étudions le polyèdre associé au problème de l'indépendant faiblement connexe. Dans la section 5.2 nous présentons une formulation en nombres entiers pour ce problème. Une description linéaire complète est obtenue lorsque le graphe initial est un cycle impair. Les conséquences polyédrales d'opérations de composition de graphes sont aussi analysées. Nous présentons ensuite un algorithme de coupes et branchement ainsi que des heuristiques de séparation des contraintes dites de bord et multi-bords.

5.2 Formulation en nombres entiers du *mwcis*

Soit $G = (V, E)$ un graphe non orienté et connexe. Pour un sous-ensemble W de V, le vecteur $x^W \in \mathbb{R}^{|V|}$ avec

$$\chi^W(u) = \begin{cases} 1, & \text{si } u \text{ est dans } W\,; \\ 0, & \text{sinon.} \end{cases}$$

est appelé le vecteur d'incidence de W. L'enveloppe convexe des vecteurs d'incidence de tous les indépendants faiblement connexes de G est appelé le polytope des indépendants faiblement connexes de G et il est noté par $P_{wcis}(G)$.

$$P_{wcis}(G) = conv\{\chi^W \in \mathbb{R}^{|V|} : W \subset V \text{ est un indépendant faiblement connexe de } G\}.$$

Le $MWCIS$ est équivalent au programme linéaire

$$min\{\sum_{u \in V} x(u) \ : \ x \in P_{wcis}(G)\}.$$

Comme le $MWCIS$ est NP-difficile [24], nous ne pouvons pas espérer donner une description complète de $P_{wcis}(G)$ pour tous les graphes. De manière évidente, le vecteur d'incidence d'un indépendant faiblement connexe satisfait les inégalités :

$$\begin{aligned} x(u) + x(v) &\leq 1, \ \forall (u, v) \in E && (5.1) \\ x(N[u]) &\geq 1, \ \forall u \in V && (5.2) \\ x(u) &\geq 0, \ \forall u \in V && (5.3) \\ x(u) &\in \{0, 1\}, \ \forall u \in V. && (5.4) \end{aligned}$$

Les contraintes triviales (5.3) et (5.4) sont immédiatement vérifiées puisque x^W est un vecteur en 0-1. Comme W est un indépendant maximal de G, alors, pour toute arête $e = (u, v)$ de E, W contient au plus un sommet de e. Par conséquent, $x^W(u) + x^W(v) \leq 1$, et la contrainte (5.1) est satisfaite. La contrainte (5.2) traduit le fait que W est dominant. La formulation doit également inclure un ensemble de contraintes qui implique que le sous-graphe $G_W = (V, \delta(W))$ est connexe. Remarquons que, si W est un indépendant faiblement connexe alors le vecteur d'incidence de $\delta(W)$, noté par $\chi^{\delta(W)}$ est donné par :

$$\chi^{\delta(W)}(e) = \chi^W(u) + \chi^W(v), \text{pout tout } e = \{u, v\} \in E.$$

Définissons le bord d'un sous-ensemble U, qui sera utilisé pour exprimer la contrainte de connexité.

Definition 5.2.1. *(Bord d'un ensemble de sommets)*

Soit un sous-ensemble de sommets $U \subsetneq V$. Le bord de l'ensemble U, noté par $\sigma(U)$ est l'ensemble de sommets incidents à $\delta(U)$.

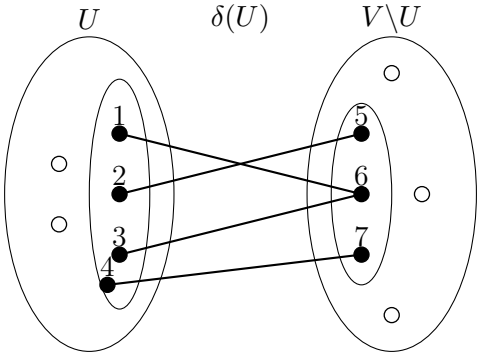


FIGURE 5.1 – $\sigma(U) = \{1, 2, 3, 4, 5, 6, 7\}$.

$$\sigma(U) = \bigcup_{e=(u,v)\in\delta(W)} \{u, v\}.$$

Les inégalités de bord sont définies par :

$$\sum_{u\in\sigma(U)} x(u) \quad \geq \quad 1, \ \forall U \subsetneq V, \ |U| \geq 2. \tag{5.5}$$

Théorème 5.2.1. *Le MWCIS est équivalent au programme linéaire en nombres entiers*

$$min\{\sum_{u\in V} x(u) \ : \ x \ satisfait \ (5.1) - (5.5)\}.$$

Preuve.

1) Étant donné un graphe non orienté connexe $G = (V, E)$, un *wcis* W est un indépendant maximal de G. Ainsi χ^W satisfait les inégalités (5.1)-(5.4). Considérons un sous-ensemble non vide U de V. Le graphe partiel $G_W = (V, \delta(W))$ est connexe donc il existe une arête $e = (u, v)$ appartenant à $\delta(U) \cap \delta(W)$. Cela implique que u et v sont dans $\sigma(U)$ et χ^W satisfait (5.5).

2) Un vecteur $x^{'}$ solution de (5.1)-(5.5) détermine un sous-ensemble $W^{'}$ de V. Montrons que $W^{'}$ est un *wcis* de G. Par les inégalités (5.1) et (5.2), $W^{'}$ est un *mis* de G. De plus, pour tout U, $U \subset V$, $U \neq \emptyset$,

$$|\delta(W^{'}) \cap \delta(U)| = \sum_{e\in\delta(W^{'})\cap\delta(U)} \chi^{\delta(W^{'})}(e) = \sum_{\{u,v\}\in\delta(U)} (x^{'}(u) + x^{'}(v)) \geq \sum_{u\in\sigma(U)} x^{'}(u) \geq 1.$$

D'où, le graphe partiel $G_{W'} = (V, \delta(W^{'}))$ est connexe. Par conséquent, $W^{'}$ est un *wcis* de G.

□

Lemme 5.2.1. *Si l'inégalité $x(\sigma(U)) \geq 1$ est facette pour $P_{wcis}(G)$ pour un sous-ensemble de sommets U de V, alors le graphe $G(U) = (U, E(U))$ est connexe.*

Preuve. Supposons, au contraire, que l'inégalité $x(\sigma(U)) \geq 1$ est facette de $P_{wcis}(G)$ pour un ensemble $U \subset V$ mais que le graphe $G(U) = (U, E(U))$ n'est pas connexe. Alors, il existe au moins deux sous-ensembles $U_1 \neq \emptyset$, $U_2 \neq \emptyset$ tels que $U = U_1 \cup U_2$ et $[U_1, U_2] = \emptyset$. Alors, $\sigma(U) = \sigma(U_1) \cup \sigma(U_2)$. Par conséquent, l'inégalité $x(\sigma(U)) \geq 1$ est dominée par les contraintes $x(\sigma(U_1)) \geq 1$ et $x(\sigma(U_2)) \geq 1$. Cela contredit le fait que l'inégalité $x(\sigma(U)) \geq 1$ est facette de $P_{wcis}(G)$. ♦

5.3 Polytope de l'indépendant faiblement connexe dans un cycle impair

Comme un indépendant faiblement connexe est un stable maximal, on peut s'attendre à des difficultés pour décrire le polytope $P_{wcis}(G)$. Toutefois, lorsque le graphe G est de faible densité, la connexité peut simplifier la description.

5.3.1 Cycle impair

Théorème 5.3.1. *Soit C un cycle impair de longueur $2k+1$. $P_{wcis}(C_{2k+1})$ est déterminé par le système :*

$$\sum_{i=0}^{2k} x(i) = k, \tag{5.6}$$

$$x(i) + x(i+1) \leq 1, \qquad \forall\, i = 0, ..., 2k. \tag{5.7}$$

Preuve.

Notons par P le polytope donné par les inégalités (5.6) et (5.7) (les indices sont modulo 2k+1).

1) $P_{wcis}(C_{2k+1}) \subset P$.
Pour tout wcis W de C_{2k+1}, χ^W satisfait (5.7). Par sommation de toutes ces contraintes, on obtient $\sum_{i=0}^{2k} \chi^W(i) \leq k + \frac{1}{2}$. Ainsi le vecteur en $0-1$ χ^W vérifies $\sum_{i=0}^{2k} x(i) \leq k$. Par ailleurs, comme le graphe partiel G_W est connexe, nous avons $|\delta(W)| \geq 2k$. Comme $|\delta(W)| = \sum_{e \in E(C_{2k+1})} \chi^{\delta(W)}(e) = 2 \sum_{i \in V(C_{2k+1})} \chi^W(i)$, nous obtenons que $\sum_{i=0}^{2k} \chi^W(i) \geq k$. Finalement, χ^W satisfait (5.6).
2) $P \subset P_{wcis}(C_{2k+1})$.
Il est facile de voir que le système (5.6)-(5.7) est équivalent au système suivant :

$$\sum_{i=0}^{2k} x(i) \geq k, \tag{5.8}$$

$$-\sum_{i=0}^{2k} x(i) \geq -k, \tag{5.9}$$

$$\sum_{i=0}^{2k-1} x(i) \geq k-1, \tag{5.10}$$

$$x(i) + x(i+1) \leq 1, \qquad \forall i = 0, \ldots, 2k-1. \tag{5.11}$$

Notons ce système par Π. La matrice A des contraintes associée à Π

$$\mathrm{A}= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & . & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & . & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & -1 & -1 & . & -1 & -1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & . & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & . & 0 & 0 & 0 \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & . & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & 1 & 1 \end{pmatrix}$$

est une matrice d'intervalle (en multipliant certaines lignes par -1) [1] ; par conséquent elle est totalement unimodulaire. Comme le vecteur

$$\mathrm{b}= \begin{pmatrix} k \\ -k \\ 1-k \\ 1 \\ 1 \\ . \\ . \\ . \\ 1 \\ 1 \end{pmatrix}$$

est entier, les points extrêmes x' de P sont entiers.
En outre, à partir de (5.6) et en utilisant les contraintes de types (5.7) découlant des k arêtes $(i+1, i+2), (i+3, i+4), \ldots, (i-2, i-1)$, on en déduit

$$x'(i) \geq 0 \qquad \text{pour tout } i = 0, \ldots, 2k.$$

Alors, comme $x'(i) + x'(i+1) \leq 1$, nous avons aussi

$$x'(i) \leq 1 \qquad \text{pour tout } i = 0, \ldots, 2k.$$

1. Une matrice $A \in \{0,1\}^{m \times n}$ est dite intervalle si pour chaque ligne i, si $a_{ij} = a_{ik} = 1$ pour $k > j+1$, alors $a_{il} = 1$ pour tout $j < l < k$.

Donc x' est le vecteur d'incidence de sous-ensemble W' de $\{0, 1, \ldots, 2k\}$ avec k éléments. En outre, par (5.11), si $i \in W'$, alors ni $i-1$ ni $i+1$ appartient à W'. Ainsi, W' est un indépendant de C_{2k+1}. Comme $2\sum_{i \in W'} x^{'}(i) = \sum_{e \in \delta(W')} \chi^{\delta(W')}(e)$, $\delta(W')$ contient $2k$ arêtes. Donc W' est un wcis de C_{2k+1}.

□

5.3.2 Composition de cycles impairs

Definition 5.3.1. *Considérons τ nombres entiers $p_1, p_2, \ldots, p_\tau$. Le graphe $H_{p_1,\ldots,p_\tau}$ est tel que*

i) $V(H_{p_1,\ldots,p_\tau}) = \bigcup_{t=1}^{\tau} \{u_2^t, u_3^t, \ldots, u_{2p_t}^t\} \cup \{u_0, u_1\}$,

ii) $H_{p_1,\ldots,p_\tau}(\{u_0, u_1, u_2^t, u_3^t, \ldots, u_{2p_t}^t\})$ est le cycle impair contenant $2p_t + 1$ nœuds, pour tout $t \in \{1, \ldots, \tau\}$ (Voir figure 5.2).

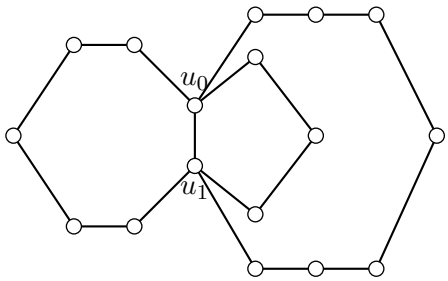


FIGURE 5.2 – $H_{2,3,4}$

Le graphe $H_{p_1,\ldots,p_\tau}$ peut être vu comme une union de τ cycles impairs partageant la même arête $\{u_0, u_1\}$.
Nous avons le résultat de type Chvátal [13] suivant.

Théorème 5.3.2. *L'union de τ systèmes linéaires*

$$(\Sigma_t)_{t\in\{1,\ldots,\tau\}} \begin{cases} x(u_0) + x(u_1) + \sum_{i=2}^{2p_t} x(u_i^t) & = p_t, \\ x(u_i^t) + x(u_{i+1}^t) & \leq 1, \quad 2 \leq i \leq 2p_t - 1, \\ x(u_0) + x(u_{2p_t}^t) & \leq 1, \\ x(u_1) + x(u_2^t) & \leq 1, \\ x(u_0) + x(u_1) & \leq 1, \end{cases}$$

définit $P_{wcis}(H_{p_1,\ldots,p_\tau})$.

Preuve. Soit $\mathbb{P}$ le polytope généré par l'union des τ systèmes linéaires (Σ_t). Notons par C_{2p_t+1} le cycle impair $\{u_0, u_1, u_2^t, \ldots, u_{2p_t}^t\}$, avec $t \in \{1, ..., \tau\}$.

1) $P_{wcis}(H_{p_1,\ldots,p_\tau}) \subset \mathbb{P}$.
Soit W un wcis de $H_{p_1,\ldots,p_\tau}$. Posons $W_t = W \cap C_{2p_t+1}$, pour $1 \leq t \leq \tau$. Comme W est un indépendant de $H_{p_1,\ldots,p_\tau}$, W_t est un stable de C_{2p_t+1}, pour tout $t \in \{1, \ldots, \tau\}$.

Montrons que W_t est un wcis de C_{2p_t+1}. Sans perte de généralité, nous prenons $t = 1$ et nous omettons l'exposant 1.

Cas 1. *Il existe un indice* $i \in \{2, \ldots, 2p_1 - 1\}$*, tel que* u_i *et* u_{i+1} *n'appartiennent pas à* W_1.

Alors, $\{u_i, u_{i+1}\} \notin \delta(W)$. Comme le graphe partiel G_W est connexe, les arêtes des chemins $\{u_{i+1}, u_{i+2}, \ldots, u_0\}$ et $\{u_i, u_{i-1}, \ldots, u_1\}$ doivent être dans $\delta(W)$. Si i est pair (resp. impair), alors u_1 (resp. u_0) est dans W. Ainsi, l'ensemble des arêtes $C_{2p_1+1} \setminus \{\{u_i, u_{i+1}\}\}$ sont inclues dans $\delta(W)$. Comme $\delta_{C_{2p_1+1}}(W_1) = \delta(W) \cap C_{2p_1+1}$, W_1 est un wcis de C_{2p_1+1}.

Cas 2. W_1 *ne contient ni* u_0 *ni* u_{2p_1}.

La connexité de G_W implique que les arêtes du chemin $\{u_{2p_1}, u_{2p_1-1}, \ldots, u_1\}$ sont dans $\delta(W)$. Comme u_{2p_1-1} doit être dans W_1, nous avons $W_1 = \{u_{2p_1-1}, u_{2p_1-3}, \ldots, u_1\}$. D'où, W_1 est un wcis de C_{2p_1+1}. Le même raisonnement s'applique lorsque ni u_1 ni u_2 n'appartiennent à W_1.

Cas 3. $\delta(W)$ *contient* $C_{2p_1+1} \setminus \{u_0, u_1\}$.

Ainsi, $W_1 = \{u_2, u_4, \ldots, u_{2p_1}\}$ et W_1 est un wcis de C_{2p_1+1}.

2) $\mathbb{P} \subset P_{wcis}(H_{p_1,\ldots,p_\tau})$.

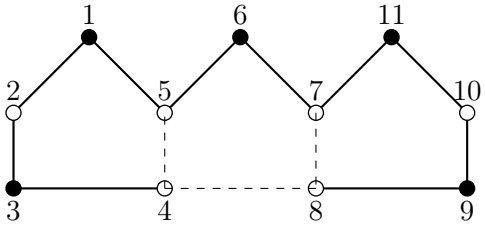
Considérons un point extrême $\hat{x}$ dans $\mathbb{P}$. Pour tout $t \in \{1, \ldots, \tau\}$, notons par $\hat{x}^t$ sa restriction sur C_{2p_t+1}. Comme $\hat{x}^t$ satisfait (Σ_t), il appartient à $P_{wcis}(C_{2p_t+1})$. Donc, $\hat{x}^t$ peut s'écrire comme une combinaison convexe des points extrêmes de $P_{wcis}(C_{2p_t+1})$, $x^{W_1^t}, x^{W_2^t}, \ldots, x^{W_{i_t}^t}$, tel que toute contrainte de (Σ_t) serrée par $\widehat{x}^t$ est aussi serrée par $x^{W_j^t}$, $1 \leq j \leq i_t$.

Premièrement, supposons que $\hat{x}(u_0) = \hat{x}(u_1) = 0$. Donc, $\hat{x}^t(u_0) = \hat{x}^t(u_1) = 0$, $t \in \{1, \ldots, \tau\}$. D'où, W_1^t ne contient ni u_0 ni u_1. Noter que u_2^t et $u_{2p_t}^t$ sont dans W_1^t. Soit $W' = \cup_{t=1}^{\tau} W_1^t$. Il est facile de voir que W' est un wcis de $H_{p_1,\ldots,p_\tau}$, dont le vecteur d'incidence $\chi^{W'}$ satisfait à égalité les mêmes contraintes que $\hat{x}$. On en déduit que $\hat{x} = \chi^{W'}$.

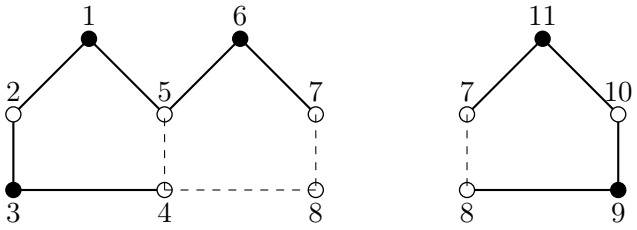
Sans perte de généralité, supposons maintenant que $\hat{x}(u_0) > 0$. Nous pouvons choisir un wcis $W_{q_t}^t$, $q_t \in \{1, ..., i_t\}$ contenant u_0, et dont le vecteur d'incidence satisfait à égalité les mêmes contraintes que $\hat{x}^t$ pour tout $t \in \{1, \ldots, \tau\}$. Comme au-dessus, avec $W'' = \cup_{t=1}^{\tau} W_{q_t}^t$, nous constatons que $\hat{x} = \chi^{W''}$.

□

Remarque 5.3.1. *Le résultat de Théorème 5.3.2 n'est pas vérifié pour la 2-somme générale. Par exemple, considérons le graphe de la figure 5.3(a) où l'ensemble des sommets est* $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ *et* $\{1, 2, 3, 4, 5\}$, $\{4, 5, 6, 7, 8\}$ *et* $\{7, 8, 9, 10, 11\}$ *forment trois cycles impairs de longueur* 5*. Le graphe de la figure 5.3(a) est le résultat de la 2-somme des trois cycles sur l'arête* $\{4, 5\}$ *et* $\{7, 8\}$ *respectivement. Le wcis* $\{1, 3, 6, 9, 11\}$ *n'induit pas de wcis dans le sous-graphe* $\{1, 2, 3, 4, 5, 6, 7, 8\}$ *comme le montre la figure 5.3(b).*



(a) Un wcis W = $\{1, 3, 6, 9, 11\}$



(*b*)$\{1, 3, 6\}$ n'est pas un wcis (*c*)$\{9, 11\}$ est un wcis du sous-graphe $\{7, 8, 9, 10, 11\}$ dans le sous-graphe induit par $\{1, ..., 8\}$

FIGURE 5.3 – Le théorème 5.3.2 n'est pas vrai pour une 2-somme quelconque

5.4 Opérations sur les graphes et le polytope $P_{wcis}(G)$

Maintenant, nous présentons quelques opérations classiques sur les graphes et décrivons leurs conséquences sur le polytope sur $P_{wcis}(G)$.

5.4.1 La 1-somme

Soit $G_i = (V_i, E_i), i = 1, 2$ deux graphes non orientés et connexes.

Definition 5.4.1. *Le graphe $G = (V, E)$ est la 1-somme de G_1 et G_2 si G est obtenu par identification d'un sommet $u_1 \in V_1$ de G_1 et d'un sommet $u_2 \in V_2$ de G_2 (figure 5.4).*

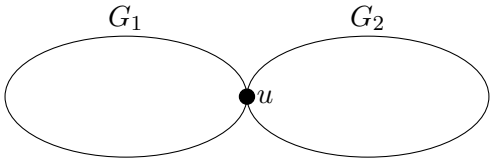


FIGURE 5.4 – La 1-somme de G_1 et G_2

Soit u le sommet résultant de l'identification de u_1 et u_2. Si on note par $G'_i = (V'_i, E'_i)$ le graphe avec $V'_i = V_i \setminus \{u_i\}$ et $E'_i = E_i \setminus \{(u_i, v) : v \in V'_i\}$ pour $i = 1, 2$. Nous avons

i) $V = \bigcup_{i=1,2} (V'_i) \cup \{u\}$,

ii) $E = \bigcup_{i=1,2} E'_i \cup \{(u, v) : (u_i, v) \in E_i, v \in V'_i, i = 1, 2\}$.

Soit S_i une description complète de $P_{wcis}(G_i)$ de la forme

$$\sum_{v\in V_i'} a_j^i(v)x(v) + a_j^i(u_i)x(u_i) \leq \alpha_j^i, j \in J_i \tag{5.12}$$

$$x(v) \geq 0, \forall v \in V_i \tag{5.13}$$

pour $i = 1, 2$. Nous obtenons de nouveau un théorème de type Chvátal [13].

Théorème 5.4.1. *Si $G = (V, E)$ est la 1-somme de G_1 et G_2, obtenue par identification des deux sommets u_1 et u_2, alors $P_{wcis}(G)$ est donné par*

$$\sum_{v\in V_i'} a_j^i(v)x(v) + a_j^i(u_i)x(u) \leq \alpha_j^i,\ j \in J_i, i = 1, 2 \tag{5.14}$$

$$x(v) \geq 0, \forall v \in V_i^{'} \cup V_2^{'} \cup \{u\} \tag{5.15}$$

Preuve. Notons par $\mathbb{P}$ le polytope décrit par le système suivant :

$$\sum_{v\in V_i'} a_j^i(v)x(v) + a_j^i(u_i)x(u) \leq \alpha_j^i,\ j \in J_i, i = 1, 2 \tag{5.16}$$

$$x(v) \geq 0, \forall v \in V_i^{'} \cup V_2^{'} \cup \{u\} \tag{5.17}$$

1) $P_{wcis}(G) \subset \mathbb{P}$.
Considérons un wcis W de G. Soit
$W_1 = \begin{cases} W \cap V_1^{'} \cup \{u_1\}, & \text{si } u \in W\,; \\ W \cap V_1^{'}, & \text{sinon.} \end{cases}$
Comme W est un indépendant de G, W_1 est un stable de G_1. De plus, toute arête $\{u, v_1\}$, pour $v_1 \in N_{G_1}(u_1)$, peut être identifiée à une arête $\{u_1, v_1\}$ de E_1. D'où, comme le graphe $G_W = (V, \delta(W))$ est connexe, tout chemin dans G reliant deux sommets v_1 et w_1 de $V_1^{'}$ peut être associé à un chemin reliant v_1 et w_1 dans G_1 et le chemin reliant v_1 et u correspond au chemin reliant v_1 et u_1 dans G_1. Donc, le graphe partiel $G_{W_1} = (V_1, \delta_{G_1}(W_1))$ est connexe dans G_1 et W_1 est un wcis de G_1. Par le même raisonnement, nous pouvons montrer que W_2 est un wcis de G_2. Donc, $\chi^{W_1} \in P_{wcis}(G_1)$ et $\chi^{W_2} \in P_{wcis}(G_2)$. Ce qui implique $\chi^W \in \mathbb{P}$.

2) $\mathbb{P} \subset P_{wcis}(G)$.
Soit $\widehat{x}$ une solution de $\mathbb{P}$. Pour $i = 1, 2$, Nous définissons $x^i \in \mathbb{R}^{|V_i|}$ par
$x^i(v) = \begin{cases} \widehat{x}(v), & \text{si } v \in V_i^{'}\,; \\ \widehat{x}(u), & \text{si } v = u_i. \end{cases}$
Comme x^i appartient à $P_{wcis}(G_i)$, x^i est une combinaison convexe des q_i points extrêmes $\chi^{W_1^i}, \chi^{W_2^i}, ..., \chi^{W_{q_i}^i}$ de $P_{wcis}(G)$.
Si $\widehat{x}(u) > 0$, alors nous pouvons choisir un indice j_i tel que $u_i \in W_{j_i}^i, i = 1, 2$. Noter

que $\chi^{W^i_{j_i}}$ satisfait à égalité les mêmes contraintes de (S_i) que $x^i, i = 1, 2$. Considérons maintenant l'ensemble $W^{'} = (W^1_{j_1} \backslash \{u_1\}) \cup (W^2_{j_2} \backslash \{u_2\}) \cup \{u\}$. $W^{'}$ est un stable de G. Par conséquent, un chemin reliant $v_1 \in V^{'}_1$ et $v_2 \in V^{'}_2$ peut être construit à partir des chemins reliant v_1 et u_1 et v_2 et u_2 par le changement des deux arêtes où leurs extrémités sont respectivement u_1 et u_2 par les arêtes adjacentes au nouveau sommet u. Donc, le sous-graphe $G_{W'} = (V, \delta_G(W^{'}))$ est connexe et $W^{'}$ est un wcis de G. Les contraintes vérifiées à égalités par $\widehat{x}$ sont les mêmes à celles vérifiées par $\chi^{W^{'}}$, d'où $\chi^{W^{'}} = \widehat{x}$.

Si $\widehat{x}(u) = 0$, alors soit $W^{''} = W^1_1 \cup W^2_1$. Comme pour tout chemin dans G_1 (resp. dans G_2) reliant v_1 à u_1 (resp. u_2 à v_2) peut être associé à un chemin reliant v_1 à u dans G (resp. u à v_2 dans G), pour tout $v_1 \in V^{'}_1$, $v_2 \in V^{'}_2$, il est facile de voir que le sous-graphe $G_{W''} = (V, \delta_G(W^{''}))$ est connexe. Comme $W^{''}$ est un stable, $W^{''}$ est un wcis de G et comme ci-dessus, nous avons $\chi^{W^{''}} = \widehat{x}$.

□

5.4.2 Ajout d'un sommet universel

A partir d'un graphe $G = (V, E)$ un graphe non orienté connexe, on construit le graphe $G_{u_0} = (V(G_{u_0}), E(G_{u_0}))$ tel que $V(G_{u_0}) = V \cup \{u_0\}$ et $E(G_{u_0}) = E \cup \{(u_0, v) : v \in V\}$ (Figure 5.5).

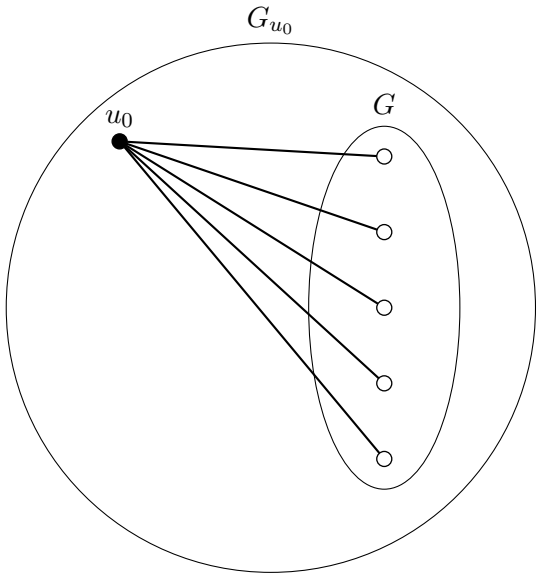


FIGURE 5.5 – Le graphe G_{u_0}.

Supposons que $P_{mis}(G)$ est donné par les inégalités

$$\sum_{u\in V} a_i(u)x(u) \leq \alpha_i, i \in I. \tag{5.18}$$

Nous avons le résultat suivant.

Lemme 5.4.1. *Le polytope $P_{wcis}(G_{u_0})$ est caractérisé par*

$$\sum_{u\in V} a_i(u)x(u) + \alpha_i x(u_0) \leq \alpha_i, \qquad i \in I, \tag{5.19}$$

$$x(u_0) \geq 0. \tag{5.20}$$

Le résultat ci-dessus nous permet de déduire la description de l'enveloppe convexe des indépendants faiblement connexes de G_{u_0} à partir de l'enveloppe convexe des indépendants maximaux de G.

Preuve.

Notons par $\mathbb{P}$ le polytope induit par les inégalités (5.19)-(5.20).

1) $P_{wcis}(G_{u_0}) \subset \mathbb{P}$.

Considérons un indépendant faiblement connexe W_0 de G_{u_0}. Si W_0 contient u_0 alors $W_0 = \{u_0\}$ et $\chi^{\{u_0\}}$ satisfait (5.19)-(5.20). Supposons maintenant que $W_0 \subset V$. Comme $\delta_{G_{u_0}}(W_0) = \delta_G(W_0) \cup [W_0, \{u_0\}]$ et que $G_{u_0}(W_0)$ est connexe, tout $u \in (V \backslash W_0)$ est relié à W_0. D'où W_0 est un stable maximal de G. La restriction de χ^{W_0} sur V appartient alors à $P_{mis}(G)$ et $\chi^{W_0} \in \mathbb{P}$. Par conséquent, $P_{wcis}(G_{u_0}) \subset \mathbb{P}$.

2) $\mathbb{P} \subset P_{wcis}(G_{u_0})$.

Supposons qu'il existe une facette $\mathbb{F}_1$ de $P_{wcis}(G_{u_0})$ induite par les inégalités $\sum_{u\in V} b(u)x(u) + b(u_0)x(u_0) \leq \beta$, différentes de (5.19) et (5.20). Grâce à u_0, tout *mis* de G devient un *wcis* de G_{u_0}. Cela implique que $\sum_{u\in V} b(u)x(u) \leq \beta$ doit être une inégalité valide pour $P_{mis}(G)$. Par conséquent l'ensemble des wcis de $\mathbb{F}_1$ inclus dans V est une face de $P_{mis}(G)$ qui est contenue dans une facette $\mathbb{F}$ définie par l'inégalité $\sum_{u\in V} a_{i_1}(u)x(u) \leq \alpha_{i_1}$, pour un certain $i_1 \in I$. Il s'ensuit que chaque wcis dans $\mathbb{F}_1$ satisfait l'inégalité valide $\sum_{u\in V} a_{i_1}(u)x(u) + \alpha_{i_1}x(u_0) \leq \alpha_{i_1}$ à égalité, nous aboutissons à une contradiction.

♦

Dans[45], les auteurs ont donné une description complète du polytope $P_{mis}(C_n)$ avec C_n est un cycle sans corde à n nœuds. En conséquence du Lemme 5.4.1, le polytope P_{wcis} de la roue W_{n+1} obtenue à partir de cycle C_n en ajoutant un sommet universel, a aussi une description complète.

On observe aussi que le polytope des *wcis* d'une clique K_n est simplement donné par

$$\{x \in \mathbb{R}^n : \sum_{i=1}^{n} x(i) = 1\ ,\ x(i) >= 0, i = 1, ..., n\}.$$

5.4.3 La 2-arête-somme

Dans cette section, nous étudions un autre type de composition de graphe.

Definition 5.4.2. *Considérons deux graphes connexes $G_i = (V_i, E_i), i = 1, 2$ avec $u_i, v_i \in V_i$ et $(u_i, v_i) \in E_i, i = 1, 2$. Le graphe $H = (V, E)$ est dit 2-arête-somme de G_1, G_2 si $V(H) = V_1 \cup V_2$ et $E(H) = E_1 \cup E_2 \cup \{(u_1, v_2), (v_1, u_2)\}$ (Figure 5.6).*

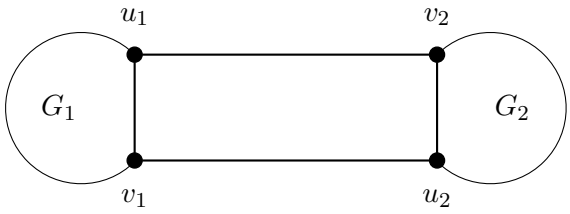


FIGURE 5.6 – La 2-arête-somme de G_1 et G_2.

Nous allons montrer que le polytope des indépendants faiblement connexes du graphe H est obtenu à partir des polytopes $P_{wcis}(G_i)$ décrits par les systèmes (S_i)

$$(S_i) \begin{cases} a^i_j x^i & \leq & \alpha^i_j, \ \forall j \in J_i, \\ x^i(u) & \geq & 0, \ \forall u \in V_i, \end{cases}$$

où J_i est un ensemble indexant les contraintes de $P_wcis(G_i)$, pour $i = 1, 2$.
Un vecteur $x \in \mathbb{R}^{|V_1|+|V_2|}$ peut être décomposé en deux vecteurs $z^1 \in \mathbb{R}^{|V_1|}$ et $z^2 \in \mathbb{R}^{|V_2|}$ quand $x = (z^1, z^2)$. Soit $\mathbb{P}$ le polytope décrit par le système

$$\begin{aligned}
a^1_j z^1 &\leq \alpha^1_j, \ \forall j \in J_1 && (5.21)\\
a^2_j z^2 &\leq \alpha^2_j, \ \forall j \in J_2 && (5.22)\\
z^1(u_1) + z^1(v_1) + z^2(u_2) + z^2(v_2) &\geq 1 && (5.23)\\
z^1(v_1) + z^2(u_2) &\leq 1 && (5.24)\\
z^1(u_1) + z^2(v_2) &\leq 1 && (5.25)\\
z^i(u) &\geq 0, \ \forall u \in V_i, \ i = 1, 2. && (5.26)
\end{aligned}$$

Théorème 5.4.2. *$P_{wcis}(H) = \mathbb{P}$.*

Preuve.

1) $P_{wcis}(H) \subset \mathbb{P}$.
Soit W un wcis de H. Comme H_W est connexe, (u_1, v_2) ou (v_1, u_2) appartiennent à $\delta(W)$. Sans perte de généralité, supposons que $(u_1, v_2) \in \delta(W)$ avec $u_1 \in W$. Ni v_1 ni v_2 ne le sont dans W. Soit $W_1 = W \cap V_1$ (resp. $W_2 = W \cap V_2$). W_1 (resp. W_2) est un stable de G_1 (resp. G_2).
– Supposons que $u_2 \in W$. Donc, $(v_1, u_2) \in \delta(W)$. Comme tout chemin de longueur minimum entre $u \in V_1$ (resp. $u \in V_2$) et $u_1 \in V_1$ (resp. $u_2 \in V_2$) est inclus dans V_1 (resp. V_2), puisque $(u_1, v_1) \in \delta_{G_1}(W_1)$ (resp. $(u_2, v_2) \in \delta_{G_2}(W_2)$). D'où le graphe partiel $(V_1, \delta_{G_1}(W_1))$ (resp $(V_2, \delta_{G_2}(W_2))$) est connexe. W_1 (resp. W_2) est un wcis

de G_1 (resp. G_2). Donc, χ^{W_1} (resp. χ'^{W_2}) satisfait (5.21) (resp. (5.22)) et $x^W = (\chi^{W_1}, \chi'^{W_2}) \in \mathbb{P}$.

– Si $u_2 \notin W$, alors $(v_1, u_2) \notin \delta(W)$ et $(v_2, u_2) \notin \delta(W)$. Tout chemin élémentaire de G_W reliant deux nœuds dans le même ensemble $V_i, i = 1, 2$ est contenu dans $V_i, i = 1, 2$. D'où $(V_1, \delta_{G_1}(W_1))$ et $(V_2, \delta_{G_2}(W_2))$ sont connexes. Ainsi, W_1 (resp W_2) est un wcis de G_1 (resp. G_2). Par conséquent $x^W = (\chi^{W_1}, \chi'^{W_2}) \in \mathbb{P}$.

2) $\mathbb{P} \subset P_{wcis}(H)$.

Considérons une solution $\widehat{x} = (\widehat{z}, \widehat{z}')$ de $\mathbb{P}$. $\widehat{z}$ (resp. $\widehat{z}'$) appartiennent à $P_{wcis}(G_1)$ (resp. $P_{wcis}(G_2)$). Il doit exister q_1 points extrêmes de $P_{wcis(G_1)}$ correspondant à q_1 wcis de G_1, $W_1^1, W_2^1, ..., W_{q_1}^1$ tel que

$$\widehat{z} = \sum_{j=1}^{q_1} \lambda_j \chi^{W_j^1}, \ \sum_{j=1}^{q_1} \lambda_j = 1, \ \lambda_j > 0, \ 1 \leq j \leq q_1,$$

où $\chi^{W_j^1}$ est le vecteur d'incidence de l'ensemble $W_j^1, 1 \leq j \leq q_1$. De la même manière, nous avons

$$\widehat{z}' = \sum_{k=1}^{q_2} \mu_k \chi^{W_k^2}, \ \sum_{k=1}^{q_2} \mu_k = 1, \ \mu_k > 0, \ 1 \leq k \leq q_2,$$

où $\chi^{W_k^2}$ est le vecteur d'incidence de l'ensemble $W_k^2, 1 \leq k \leq q_2$.

Noter que $\chi^{W_j^1}, 1 \leq j \leq q_1$ (resp. $\chi^{W_k^2}, 1 \leq k \leq q_2$), satisfait à égalité les mêmes inégalités de (5.21) (resp. (5.22)) que $\widehat{x}$. Pour $i = 1, 2$, soit

$$Q_0^i = \{j \in \{1, ..., q_i\} \ : \ u_i, v_i \notin W_j^i\},$$

$$Q_{u_i} = \{j \in \{1, ..., q_i\} \ : \ u_i \in W_j^i\},$$

et

$$Q_{v_i} = \{j \in \{1, ..., q_i\} \ : \ v_i \in W_j^i\}.$$

Les ensembles Q_0^1, Q_{u_1} et Q_{v_1} forment une partition de $\{1, ..., q_1\}$. De plus,

$$1 = \sum_{j=1}^{q_1} \lambda_j = \sum_{j \in Q_0^1} \lambda_j + \sum_{j \in Q_{u_1}} \lambda_j + \sum_{j \in Q_{v_1}} \lambda_j = \sum_{j \in Q_0^1} \lambda_j + \widehat{z}(u_1) + \widehat{z}(v_1).$$

De la même manière, nous avons

$$1 = \sum_{k=1}^{q_2} \mu_k = \sum_{k \in Q_0^2} \mu_k + \sum_{k \in Q_{u_2}} \mu_k + \sum_{k \in Q_{v_2}} \mu_k = \sum_{k \in Q_0^2} \mu_k + \widehat{z}'(u_2) + \widehat{z}'(v_2).$$

Noter que les inégalités (5.23), (5.24) et (5.25) ne sont pas toutes serrées pour le même point extrême.

Proposition 1. *Toute solution entière de $\mathbb{P}$ est un wcis de H.*

Preuve.

Soit $\widehat{x} = (\widehat{z}, \widehat{z}')$ une solution entière de $\mathbb{P}$. $\widehat{z}$ (resp. $\widehat{z}'$) est une solution entière pour (S_1) (resp. (S_2)). Comme (S_1) (resp. (S_2)) décrit le polytope en 0-1, $\widehat{z}$ (resp. $\widehat{z}'$) est le vecteur

d'incidence d'un wcis W_1 de G_1 (resp. W_2 de G_2). Soit $\widehat{W} = W_1 \cup W_2$. La contrainte (5.23) implique qu'au moins un sommet de $\{u_1, v_1, u_2, v_2\}$ est dans $\widehat{W}$. Ainsi l'arête (u_1, v_2) ou (v_1, u_2) est dans $\delta_H(\widehat{W})$, cela entraîne que $(V(H), \delta_H(\widehat{W}))$ est connexe. Avec les contraintes (5.24) et (5.25), $\widehat{W}$ est un stable de H, nous déduisons que $\widehat{W}$ est un wcis de H. ♦

De la Proposition 1, nous pouvons supposer que $\widehat{x}$ a des composantes fractionnaires.

Cas 1. *Une des variables $\widehat{z}(u_1)$, $\widehat{z}(v_1)$, $\widehat{z}'(u_2)$ et $\widehat{z}'(v_2)$ est égale à 1.*

Sans perte de généralité, supposons que $\widehat{z}(u_1) = 1$. Ainsi par (5.24) et les contraintes d'arêtes $x(u_1) + x(v_1) \leq 1$, nous avons $\widehat{z}(v_1) = \widehat{z}'(v_2) = 0$. Si (5.23) est serrée par $\widehat{x}$, alors $\widehat{z}'(u_2) = 0$. Autrement, $u_1 \in W_1^1$ et W_1^2 ne contient ni u_2 ni v_2.

Soit $\widetilde{W}_0 = W_1^1 \cup W_1^2$. Il est facile de voir que $\widetilde{W}_0$ est un wcis de H et que $\chi^{\widetilde{W}_0}$ vérifie à égalité les contraintes (5.23), (5.24) et les mêmes contraintes dans (5.21) et (5.22) que $\widehat{x}$. On obtient $\widehat{x} = \chi^{\widetilde{W}_0}$. D'autre part, $\widehat{z}'(u_2) \geq 0$. Ainsi, il existe un wcis $W_{k_1}^2$ contenant u_2, $1 \leq k_1 \leq q_2$. Soit $\widetilde{W}_1 = W_1^1 \cup W_{k_1}^2$, alors les contraintes (5.21) et (5.23) sont satisfaites à égalité par $\chi^{\widetilde{W}_1}$. Comme précédemment nous avons $\widehat{x} = \chi^{\widetilde{W}_1}$.

Désormais, nous supposons que $\widehat{x}(u) < 1$ pour $u \in \{u_1, v_1\}$ et $\widehat{z}'(u) < 1$ pour $u \in \{u_2, v_2\}$.

Cas 2. *Aucune contrainte (5.23)-(5.25) n'est serrée par $\widehat{x}$.*

Comme $\widehat{x}$ est un point extrême de $\mathbb{P}$, les contraintes satisfaites à égalité par $\widehat{x}$ proviennent de (5.21) et (5.22). Donc $\widehat{z}$ (resp. $\widehat{z}'$) est un point extrême de $P_{wcis}(G_1)$ (resp. $P_{wcis}(G_2)$). Ainsi, $\widehat{z}$ (resp. $\widehat{z}'$) sont des vecteurs en 0-1 et $x^0 = (\widehat{z}, \widehat{z}')$ est une solution entière de $\mathbb{P}$. Avec la Proposition 1, x^0 est le vecteur caractéristique d'un wcis de H.

Cas 3. *Les contraintes (5.24) et (5.25) sont serrées par $\widehat{x}$.*

Les variables $\widehat{z}(u_1)$, $\widehat{z}(v_1)$, $\widehat{z}'(u_2)$ et $\widehat{z}'(v_2)$ sont strictement positives. Donc, il existe $j_1 \in \{1, ..., q_1\}$ et $k_1 \in \{1, ..., q_2\}$ tel que $u_1 \in W_{j_1}^1$ et $u_2 \in W_{k_1}^2$. En considérant $\widetilde{W}_2 = W_j^1 \cup W_{k_1}^2$, nous obtenons que $\widetilde{W}_2$ est un wcis de H et $\widehat{x} = \chi^{\widetilde{W}_2}$.

Cas 4. *La contrainte (5.23) est serrée par $\widehat{x}$.*

Comme $\widehat{z}(u_1) + \widehat{z}(v_1) + \widehat{z}'(u_2) + \widehat{z}'(v_2) = 1$, on obtient

$$1 = \sum_{j \in Q_0^1} \lambda_j + \sum_{k \in Q_0^2} \mu_k, \tag{5.27}$$

sans perte de généralité, considérons que $\sum_{j \in Q_0^1} \lambda_j \geq \sum_{k \in Q_0^2} \mu_k$, et $\widehat{z}(u_1) \geq \widehat{z}(v_1)$. Par (5.27), $Q_0^1 \neq \emptyset$ et $\sum_{k \in Q_0^2} \mu_k \leq \frac{1}{2}$. Donc

$$\widehat{z}'(u_2) + \widehat{z}'(v_2) \quad \geq \quad \frac{1}{2}, \tag{5.28}$$

Si (5.25) est serrée par $\widehat{x}$, alors, comme $\widehat{z}'(u_2) < 1$, $\widehat{z}(u_1) > 0$. Ainsi $\widehat{z}(u_1) > 0$. Cela implique que (5.23) n'est pas serrée par $\widehat{x}$, contradiction. Si (5.24) est serrée par $\widehat{\widehat{x}}$, alors, $\widehat{z}(u_1) < 1$ et $\widehat{z}'(v_2) > 0$. D'où, $Q_{v_2} \neq \emptyset$. Soit $\widetilde{W}_3 = W_{j_3}^1 \cup W_{k_3}^2$ avec $j_3 \in Q_0^1$ et $k_3 \in Q_{v_2}$, alors $\widetilde{W}_3$ est un wcis de H vérifiant à égalité les mêmes contraintes que $\widehat{x}$. Par conséquent, $\widehat{x} = \chi^{\widetilde{W}_3}$. Autrement, par (5.28) $Q_{u_2} \cup Q_{v_2} \neq \emptyset$. Soit $\widetilde{W}_4 = W_{j_3}^1 \cup W_{k_4}^2$ avec $k_4 \in Q_{u_2} \cup Q_{v_2}$. $\widetilde{W}_4$ est un wcis de H et nous pouvons conclure facilement que $\widehat{x} = \chi^{\widetilde{W}_4}$.

□

5.5 Décomposition modulaire

Considérons un graphe non orienté connexe $G = (V, E)$. Supposons qu'il existe un module M de G tel que le polytope $P_{mis}(G(M))$ est donné par

$$(S_M) \begin{cases} \sum_{v \in M} b_i(u)x(u) & \leq & \beta_i, \ \forall i \in I \\ x(u) & \geq & 0, \ \forall u \in M. \end{cases}$$

A partir de G nous définissons deux graphes connexes, $G_0 = (V_0, E_0)$ par

i) $V_0 = (V \backslash M) \cup \{t_0\}$,

ii $E_0 = E(V \backslash M) \cup \{(t_0, w) : \ w \in N_G(M)\}$

et $G_0^{'} = (V_0^{'}, E_0^{'})$ par

i) $V_0^{'} = M \cup \{t_1\}$,

ii) $E_0 = E(M) \cup \{(t_1, m) : \ m \in M\}$. (Figure 5.7).

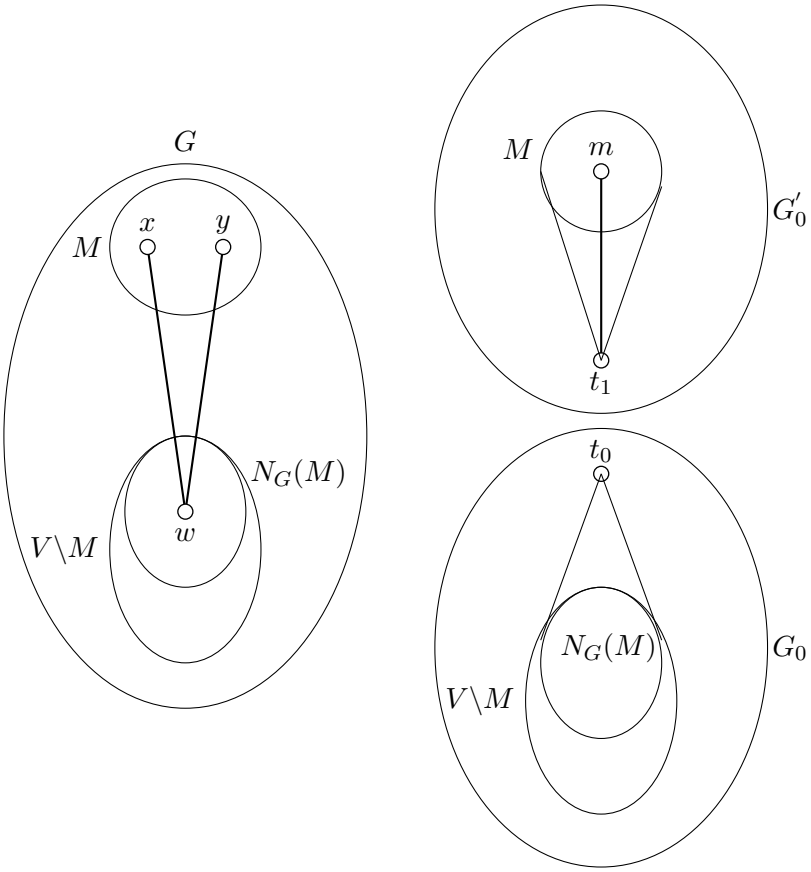


FIGURE 5.7 – Les graphes G, G_0 et $G_0^{'}$.

Supposons que $P_{wcis}(G_0)$ soit décrit par

$$(S_0)\begin{cases} \sum\limits_{u\in V\setminus M} a_j(u)y(u) + a_j(t_0)z(t_0) \leq \alpha_j,\ \forall j \in J \\ z(t_0)\geq 0, y(u)\geq 0, \qquad\qquad \forall u \in (V\setminus M) \end{cases}$$

Nous construisons le graphe auxiliaire $G_1 = (V_1, E_1)$ tel que
i) $V_1 = V_0 \cup V_0^{'}$
ii) $E_1 = E_0 \cup E_0^{'} \cup \{(t_0, t_1)\}$.

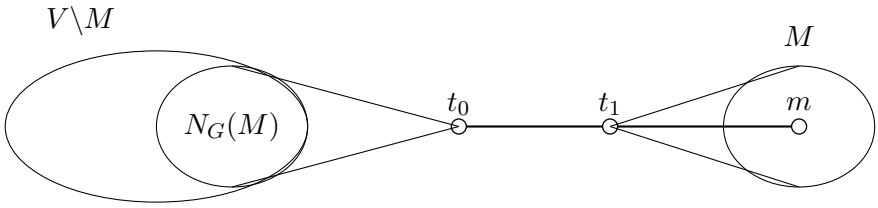


FIGURE 5.8 – Le graphe G_1.

Lemme 5.5.1. *Le système (S_1) décrivant $P_{wcis}(G_1)$ est donné par*

$$(S_1)\begin{cases} \sum\limits_{v\in M} b_i(u)x(u) + \beta_i z(t_1) \leq \beta_i,\ \forall i \in I \\ z(t_1) + z(t_0) = 1 \\ \sum\limits_{u\in V\setminus M} a_j(u)x(u) + a_j(t_0)z(t_0) \leq \alpha_j,\ \forall j \in J \\ x(u)\geq 0, \qquad\qquad \forall u \in V \\ z(t_0)\geq 0, z(t_1)\geq 0. \end{cases}$$

Preuve. Le graphe $G_0^{'}$ est obtenu à partir de $G(M)$ en lui ajoutant le sommet universel t_1. Quand à G_1, il est le résultat de deux opérations de 1-somme successivement entre G_0 et K_2 puis $G_0^{'}$, en utilisant les sommets t_0 et t_1 (Figure 5.8). Du Lemme (5.4.1), nous construisons une description complète de $P_{wcis}(G_0^{'})$. (S_1) est le résultat de deux applications du Théorème (5.4.1). ♦

5.5.1 Description du $P_{wcis}(G)$ en utilisant des projections

Un exposé complet se trouve dans Balas [4]sur les projections dans la programmation linéaire en nombres entiers et l'optimisation combinatoire. Une description du polytope $P_{wcis}(G)$ est obtenue par la projection du polytope $P_{wcis}(G_1)$ sur l'espace des x. Soit

$$\mathcal{P} = Proj_x(P_{wcis}(G_1)) = \{x \in \mathbb{R}^{|V|} :\ \exists(\tau_0, \tau_1) \in \mathbb{R}^2 :\ (x, \tau_0, \tau_1) \in P_{wcis}(G_1)\}$$

Théorème 5.5.2. $\mathcal{P} = P_{wcis}(G)$.

Preuve.

Un point extrême de $\mathcal{P}$ est une projection d'un point entier $(\widehat{x}, \widehat{t_0}, \widehat{t_1})$ de $P_{wcis}(G_1)$. Ainsi, le vecteur entier $\widehat{x}$ est le vecteur caractéristique de l'ensemble W inclus dans V. Rappelons que l'ensemble $\widehat{W} = W \cup \{t_0\}$ si $\widehat{t_0} = 1$ et $W \cup \{t_1\}$ sinon, est un wcis de G_1 (puisque un

wcis de G_1 contient soit t_0 soit t_1). Nous avons $\delta_{G_1}(\widehat{W}) \cap E(V\backslash M) = \delta_G(\widehat{W}) \cap E(V\backslash M)$ et $\delta_{G_1}(\widehat{W}) \cap E(M) = \delta_G(W) \cap E(M)$.

Cas 1. $t_1 \in \widehat{W}$.

Comme le graphe partiel $(V_1, \delta_{G_1}(\widehat{W}))$ est connexe, la chaîne contenant $u \in V\backslash M$ et t_1 contient un nœud w_0 de $N_G(M) \cap \widehat{W}$. De plus, toute chaîne reliant un nœud $u \in V\backslash M$ et w_0 est soit inclue dans $V\backslash M$ soit elle est de la forme $uw_1, ..., w_q t_0 w_0$ avec $w_q \in N_G(M) \cap \widehat{W}$. Donc dans G, pour tout $u \in V\backslash M$ et w_0, nous pouvons obtenir des chaînes qui sont totalement inclues dans $V\backslash M$ ou une chaîne utilisant les arêtes $\{w_q, m\}$ et $\{m, w_0\}$ pour un certain $m \in M$. On constate aussi que toute arête incidente à un nœud de $N_G(M) \cap \widehat{W}$ et un nœud de M appartient à $\delta_G(W)$. Cela implique que le graphe partiel $(V, \delta_G(W))$ est connexe. W est un stable de G puisque $\widehat{W}$ l'est aussi. Cela indique que W est un wcis de G.

Cas 2. $t_0 \in \widehat{W}$.

Comme $\widehat{W}$ est un mis de G_1, l'ensemble non vide $W \cap M$ doit être un mis de $G(M)$ et l'ensemble $W \cap (V\backslash M)$ est un stable de $G(V\backslash M)$. Soit $m_0 \in (W \cap M)$. Toute chaîne minimum entre $u \in V\backslash M$ et m_0 noté par μ_{um_0} est de la forme $uw_1...w_q t_0 t_1 m_0$ pour un certain sommet $w_q \in N_G(M), q \in \mathbb{N}$. De μ_{um_0} on peut déduire une chaîne $uw_1...w_q m_0$ dans G. Considérons maintenant un élément m de M. Si $m \in W \cap M$, alors les arêtes $\{m_0, w\}$ et $\{w, m\}$ appartiennent à $\delta_G(W)$ pour tout $w \in N_G(M)$. Sinon, il existe un sommet $m' \in W \cap M$ tel que $\{m, m'\} \in \delta_{G_1}(\widehat{W})$. Évidement, $\{m, m'\} \in \delta_G(W)$ ce qui implique que le graphe partiel $(V, \delta_G(W))$ est connexe. Finalement, W est un wcis de G.
Considérons maintenant une wcis de G, W. Soit $S = W \cap M$, alors S et $W\backslash S$ sont des stables.

Proposition 1. *Si S n'est pas vide, alors S est un mis de $G(M)$.*

Preuve. Soit $s \in S$. Nous avons $W \cap N_G(M) = \emptyset$ puisque s est adjacent à tous les sommets de $N_G(M)$. Comme W est un dominant de G, S domine M. Par conséquent, S est un stable maximal de $G(M)$ et χ^S le vecteur caractéristique de S satisfait le système (S_M). ♦
Soit $z_0 = \chi^S$ si $S \neq \emptyset$ et 0 sinon. Définissons l'ensemble W_0 par

$$W_0 = \begin{cases} W \cap (V\backslash M) \cup \{t_0\}, & \text{si } S \neq \emptyset\,; \\ W \cap (V\backslash M), & \text{sinon.} \end{cases}$$

Il est facile de voir que $\delta_G(W) \cap E(V\backslash M) = \delta_{G_0}(W_0) \cap E(V\backslash M)$.

Proposition 2. *W_0 est un wcis de G_0.*

Preuve. D'abord, si $S = \emptyset$, $W_0 = W \subset (V\backslash M)$. Donc, W_0 est un indépendant maximal de G_0. Comme pour tout sommet de M est dominé par un nœud de W, il existe au moins un sommet u_1 appartient à $W \cap N_G(M)$. Alors, comme $\{t_0, u_1\} \in \delta_{G_0}(W_0)$, le sous-graphe $(V_0, \delta_{G_0}(W_0))$ est connexe. Cela entraîne que W_0 est un wcis de G_0. Maintenant, supposons que $S \neq \emptyset$. Ainsi, $W \cap N_G(M) = \emptyset$. Toute chaîne minimum P connectant deux sommets u et v de $V\backslash M$ dans $G(V, \delta(W))$ contient au plus un sommet de M. On en déduit une chaîne P_0 entre u et v dans G_0, utilisant éventuellement t_0. D'où, le sous-graphe $(V_0, \delta_{G_0}(W_0))$ est connexe et le vecteur χ^{W_0} est solution du système (S_0). ♦
Soit

$$x(u) = \begin{cases} \chi^{W_0}(u), & \text{si } u \in V_0\,; \\ z_0(u), & \text{si } u \in M\,; \\ 1, & \text{si } u = t_1 \text{ et } S = \emptyset\,; \\ 0, & \text{si } u = t_1 \text{ et } S \neq \emptyset. \end{cases}$$

x vérifie le système (S) et sa projection sur $\mathcal{P}$ correspond au vecteur caractéristique du W.

□

5.5.2 Cas d'un sommet jumeau

Soit $G = (V, E)$ un graphe non orienté connexe tel que $|V| \geq 2$ et $u \in V$. Supposons que le polytope $P_{wcis}(G)$ est décrit par le système (S)

$$\sum_{v \in V \setminus \{u\}} a_j(v)x(v) + a_j(u)x(u) \leq \alpha_j, \forall j \in J \tag{5.29}$$

$$x(v) \geq 0, \forall v \in V \tag{5.30}$$

Considérons le graphe connexe $G^{'}$ déduit de G en ajoutant un vrai jumeau $u^{'}$ de u.

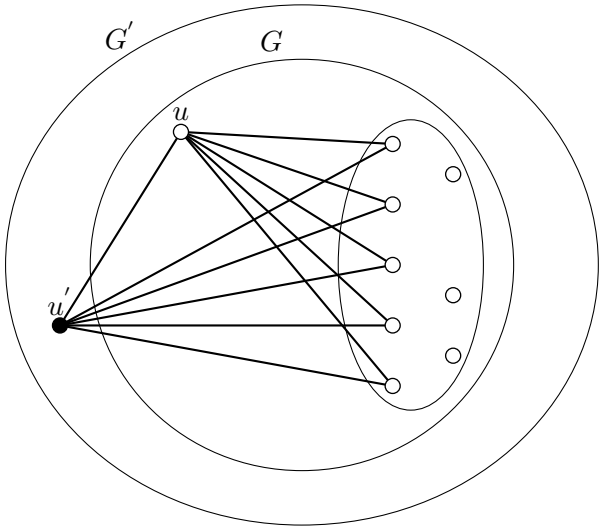


FIGURE 5.9 – $\{u, u^{'}\}$ est un module de $G^{'}$.

Corollaire 5.5.1. *Si u et $u^{'}$ sont des vrai jumeaux, alors $P_{wcis}(G^{'})$ est décrit par le système des inégalités $(S^{'})$*

$$\sum_{v \in V \setminus \{u\}} a_j(v)x(v) + a_j(u)(x(u) + x(u^{'})) \leq \alpha_j, \forall j \in J \tag{5.31}$$

$$x(v) \geq 0, \forall v \in V^{'} \tag{5.32}$$

Preuve.

On remarque dans $G^{'}$, $\{u, u^{'}\}$ forme un module. Soit $G_1^{'}$ le graphe associé à $G^{'}$. D'après le Lemme 5.5.1, le système

$$\begin{cases} x(u) + x(u^{'}) + z(t_1) & = & 1 \\ z(t_1) + z(t_0) & = & 1 \\ a_j(u)z(t_0) + \sum\limits_{v \in V \setminus \{u\}} a_j(v)x(v) & \leq & \alpha_j,\ \forall j \in J \\ x(v) \geq 0, & & \forall v \in V \cup \{u^{'}\} \\ z(t_0) \geq 0, z(t_1) \geq 0. & & \end{cases}$$

décrit $P_{wcis}(G_1^{'})$. Notons par e_j le vecteur ligne où la j^{eme} composante est égale à 1 et 0 sinon et e_0 le vecteur nul de taille $|J|$. Grâce au Théorème [4] l'ensemble des générateurs de la projection associée à $\mathcal{P}$ est

$$\{(a_j(u), -a_j(u), e_j) :\ j \in J \text{ et } a_j(u) \neq 0\} \cup \{(0, 0, e_j) :\ j \in J \text{ et } a_j(u) = 0\}$$

Par l'application de Théorème de Balas [4], on obtient $(S^{'})$. □

Corollaire 5.5.2. *Si u et $u^{'}$ sont deux faux jumeaux, alors $P_{wcis}(G^{'})$ est obtenu en prenant l'union du système (S) avec l'équation $\{x(u) = x(u^{'})\}$*

Preuve. Comme ci-dessus, $\{u, u^{'}\}$ est un module. Le système

$$\begin{cases} x(u) + z(t_1) & = & 1 \\ x(u^{'}) + z(t_1) & = & 1 \\ z(t_1) + z(t_0) & = & 1 \\ a_j(u)z(t_0) + \sum\limits_{v \in V \setminus \{u\}} a_j(v)x(v) & \leq & \alpha_j,\ \forall j \in J \\ x(v) \geq 0, & & \forall v \in V \cup \{u^{'}\} \\ z(t_0) \geq 0, z(t_1) \geq 0. & & \end{cases}$$

détermine $P_{wcis}(G^{'})$. L'ensemble des générateurs de la projection associée à $\mathcal{P}$ est

$$\begin{array}{ll} & \{(a_j(u), 0, -a_j(u), e_j) :\ j \in J \textit{ et } a_j(u) \neq 0\} \\ \cup & \{(0, 0, 0, e_j) :\ j \in J \textit{ et } a_j(u) = 0\} \\ \cup & \{(1, -1, 0, e_0)\} \\ \cup & \{(-1, 1, 0, e_0)\} \end{array}$$

Par l'application de Théorème de Balas [4], on obtient $(S^{'})$. □

5.6 Inégalités valides pour $P_{wcis}(G)$

5.6.1 Contraintes issues du polytope du stable

Toutes les contraintes issues du polytope du stable sont valides pour $P_{wcis}(G)$. En particulier les contraintes de clique :

$$\sum_{u \in K} x(u) \leq 1$$

pour toute clique K de G. Et les contraintes de cycle impair :

$$\sum_{u \in C} x(u) \leq \frac{|C| - 1}{2}$$

pour tout cycle impair C de G.

5.6.2 Contraintes de 2-voisinage

Lemme 5.6.1. *Soient $G = (V, E)$ un graphe non orienté connexe et $N^2(u)$ l'ensemble des sommets à distance deux de $u \in V$. Alors l'inégalité :*

$$x(u) - x(N^2(u)) \quad \leq \quad 0, \forall u \in V \tag{5.33}$$

est valide pour $P_{wcis}(G)$.

Preuve. Considérons un indépendant faiblement connexe W tel que $|W| \geq 2$. Soit $u \in V$. Si $u \in W$ alors, il faut que W contient au moins un sommet de $N^2(u)$ pour connecter dans G_W u à un sommet quelconque de $W \backslash \{u\}$. D'où $\chi^W(N^2(u)) \geq 1 = \chi^W(u)$. Si $u \notin W$ alors, $\chi^W(N^2(u)) \geq 0$. ♦

Lemme 5.6.2. *Soit u et v deux sommets de V tel que $v \in N^2(u)$ et $N(v) \subseteq N(u)$. L'inégalité*

$$x(u) - x(v) \quad \leq \quad 0, \tag{5.34}$$

est valide pour $P_{wcis}(G)$.

Preuve.

Soit W un wcis de G. Supposons que $u \in W$. Comme $N(v) \subseteq N(u)$, le sommet v ne peut pas être dominé par un de ces voisins (Figure 5.10). Nécessairement, $v \in W$. La contrainte (5.34) est par conséquent valide pour $P_{wcis}(G)$. ♦

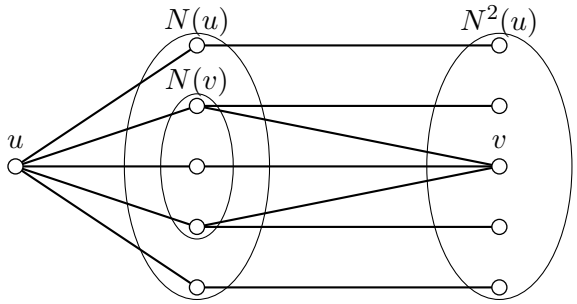


FIGURE 5.10 – u, $N(u)$, $N^2(u)$ et $N(v)$.

5.6.3 Contraintes multi-bord

Soient $G = (V, E)$ un graphe non orienté connexe et $\Pi = (V_1, ..., V_p)$ une partition de V. Soit $\delta(V_1, ..., V_p)$ l'ensemble des arêtes ayant leurs extrémités dans des ensembles différents de la partition ($\delta(V_1, ..., V_p) = \bigcup_{i=1}^{p} \delta(V_i)$). Considérons l'ensemble de sommets "bord de la partition" qu'on notera $\mu\sigma(\Pi)$ défini par :

$$\mu\sigma(\Pi) = \{u \in V : \exists v \in V, (u, v) \in \delta(V_1, ..., V_p)\}.$$

Soient $u \in V_i$ pour tout $i \in \{1, ..., p\}$, notons par $d_\Pi(u)$ le coefficient du sommet u dans la partition Π donné par :

$$d_\Pi(u) = \begin{cases} |\{j \in \{1, ..., p\} : u \in \sigma(V_j)\}| - 1, & \text{si } u \in \mu\sigma(\Pi)\,; \\ 0, & \text{sinon.} \end{cases}$$

Considérons l'inégalité

$$\sum_{u \in \mu\sigma(\Pi)} d_\Pi(u)x(u) \geq p - 1, \text{pour toute partition } \Pi = (V_1, ..., V_p) \text{ de } V. \tag{5.35}$$

Théorème 5.6.1. *L'inégalité (5.35) est valide pour $P_{wcis}(G)$.*

Preuve. Soit W un *wcis* de G. Comme le graphe G_W est connexe, $W \cap \mu\sigma(\Pi) \neq \emptyset$ et $|E[W] \cap \delta(V_1, ..., V_p)| \geq p - 1$. Chaque sommet u de $W \cap \mu\sigma(\Pi)$ engendre au moins $d_\Pi(u)$ arêtes de $\delta(V_1, ..., V_p)$. Si $E[u] \cap \delta(V_1, ..., V_p) = \emptyset$ alors $d_\Pi(u) = 0$. Sinon $d_\Pi(u)$ est le nombre de sous-ensembles de la partition Π que permet de connecter u s'il est dans le *wcis* W. Comme les sommets de $W \cap \mu\sigma(\Pi)$ permettent de connecter tous les sous-ensembles de la partition Π, alors $\sum_{u \in \mu\sigma(\Pi)} d_\Pi(u)\chi^W(u) \geq p - 1$. □

Soit P_{wcis}^{mbord} la formulation du problème $MWCIS$ en remplaçant l'inégalité (5.3) par l'inégalité (5.35)

$$\begin{aligned}
\min \sum_{u \in V} x(u) & & & \\
x(u) + x(v) &\leq 1, \ \forall (u, v) \in E & & (5.36)\\
x(N[u]) &\geq 1, \ \forall u \in V & & (5.37)\\
\sum_{u \in \mu\sigma(\Pi)} d_\Pi(u)x(u) &\geq p - 1, \text{pour toute partition } \Pi = (V_1, ..., V_p) \text{ de } V & & (5.38)\\
x(u) &\geq 0, \ \forall u \in V & & (5.39)\\
x(u) &\in \{0, 1\}, \ \forall u \in V & & (5.40)
\end{aligned}$$

Notons par R_{wcis}^{bord} la relaxation linéaire du problème P_{wcis}^{bord} et R_{wcis}^{mbord} celle du P_{wcis}^{mbord}.

Théorème 5.6.2. $R_{wcis}^{mbord} \subseteq R_{wcis}^{bord}$.

Preuve.

i) Montrons d'abord que $R_{wcis}^{mbord} \subseteq R_{wcis}^{bord}$. Si on remplace p par 2 dans la contrainte (5.38) de la relaxation R_{wcis}^{mbord} nous obtenons exactement R_{wcis}^{bord}.

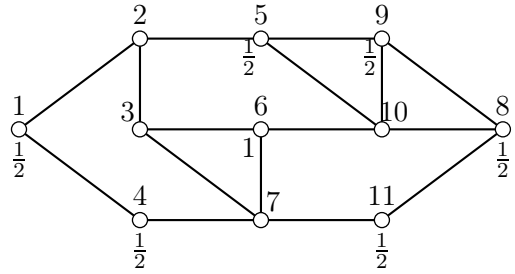


FIGURE 5.11 – Point extrême fractionnaire pour P^{bord}_{wcis}.

ii) Montrons maintenant que R^{mbord}_{wcis} peut être meilleure que la relaxation linéaire R^{bord}_{wcis}. Considérons un point extrême fractionnaire $\widehat{x} = (\frac{1}{2}, 0, 0, \frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2})$ de la relaxation linéaire R^{bord}_{wcis} pour le graphe de la Figure 5.11. Soit la partition de sommets $\Pi = (V_1, V_2, V_3)$ avec $V_1 = \{1, 4\}$, $V_2 = \{2\}$ et $V_3 = \{3, 5, 6, 7, 8, 9, 10, 11\}$. $\mu\sigma(\Pi) = \{1, 2, 3, 4, 5, 7\}$ et $d_\Pi(1) = 1$, $d_\Pi(2) = 2$, $d_\Pi(3) = 1$, $d_\Pi(4) = 1$, $d_\Pi(5) = 1$, $d_\Pi(7) = 1$. L'inégalité (5.38) devient

$$x_1 + 2x_2 + x_3 + x_4 + x_5 + x_7 \geq 2.$$

Le point extrême fractionnaire $\widehat{x}$ ne satisfait pas cette inégalité : $(\frac{1}{2}+0+0+\frac{1}{2}+\frac{1}{2}+0 = \frac{3}{2} \not\geq 2)$.

□

Chapitre 6

Algorithme de coupes et branchement

Sommaire

6.1 Algorithme de coupes et branchement

Nous présentons dans cette section un algorithme de coupes et branchement (branch-and-cut algorithm) pour résoudre le programme linéaire en nombres entiers ($\mathbb{P}^I$) suivant :

$$\min \sum_{u \in V} x(u)$$

$$\begin{aligned}
x(u) + x(v) &\leq 1,\ \forall (u,v) \in E && (6.1)\\
x(N[u]) &\geq 1,\ \forall u \in V && (6.2)\\
x(u) - x(N^2(u)) &\leq 0,\ \forall u \in V && (6.3)\\
\sum_{u \in V} d(u)x(u) &\geq |V| - 1 && (6.4)\\
\sum_{u \in \sigma(W)} x(u) &\geq 1,\ \forall W \subsetneq V,\ |W| \geq 2 && (6.5)\\
\sum_{u \in \mu\sigma(\Pi)} d_\pi(u)x(u) &\geq p - 1 \text{ pour toute partition } \Pi = (V_1, ..., V_p) \text{ de } V && (6.6)\\
x(u) &\geq 0,\ \forall u \in V && (6.7)\\
x(u) &\in \{0,1\},\ \forall u \in V && (6.8)
\end{aligned}$$

6.1.1 Description de l'algorithme

Cet algorithme est basé sur les résultats polyédraux vus dans le chapitre 5. L'algorithme commence par résoudre le programme linéaire suivant :

$$\begin{aligned} \min \sum_{u \in V} x(u) & \\ x(u) + x(v) &\leq 1,\ \forall (u,v) \in E, \\ x(N[u]) &\geq 1,\ \forall u \in V, \\ x(u) - x(N^2(u)) &\leq 0,\ \forall u \in V, \\ \sum_{u \in V} d(u)x(u) &\geq |V| - 1, \\ x(u) &\geq 0,\ \forall u \in V. \end{aligned}$$

où $d(u) = |N(u)|$, $u \in V$ et la contrainte multibord $\sum_{u \in V} d(u)x(u) \geq |V| - 1$ est induite par la partition $\{\{u\}\, u \in V\}$. Nous ajoutons toutes les contraintes possibles de type 5.34 pour toutes les instances. La solution optimale $\widehat{x} \in \mathbb{R}^{|V|}$ de cette relaxation linéaire est réalisable pour le problème MWCIS si $\widehat{x}$ est entier et satisfait toutes les contraintes de bord. En général, la solution courante $\widehat{x}$ n'est pas réalisable pour le problème MWCIS, et pour chaque itération de l'algorithme de coupes et branchement, il est nécessaire de générer d'autres inégalités valides pour le problème MWCIS mais violées par $\widehat{x}$. L'algorithme de coupes et branchement utilise deux types de contraintes pour couper la solution courante $\widehat{x}$:

1. contraintes de bord,
2. contraintes multibord.

Nous présentons dans la section suivante la procédure de séparation de contraintes de bord et multibord.

6.1.2 Séparation des inégalités de bord et multibord

Actuellement, nous ne connaissons pas la complexité du problème de séparation des contraintes de bord et multibord. C'est pourquoi, nous présentons dans cette section une heuristique en $O(n^2)$ pour séparer les inégalités de bord et multibord. Considérons une solution optimale courante $\widehat{x}$ dans l'arbre de l'algorithme de coupes et branchement. Définissons $\widehat{y} \in [0,1]^E$ par $\widehat{y} = \widehat{x}(u) + \widehat{x}(v)$, pour toute arête $(u,v) \in E$. Soit $E(\widehat{x}, \theta_1) = \{\{u,v\} \in E : \widehat{y}\{u,v\} \geq \theta_1\}$ pour un certain paramètre $\theta_1 \leq 1$. $E(\widehat{x}, \theta_1)$ est l'ensemble des arêtes de E obtenu en supprimant toutes les arêtes $\{u,v\}$ de E qui ont une valeur $\widehat{y}\{u,v\} < \theta_1$. Premièrement, nous cherchons les composantes connexes $(C_1, C_2, ..., C_q)$, $q \geq 1$ dans le graphe $G(\widehat{x}, \theta_1) = (V, E(\widehat{x}, \theta_1))$.
Si $q = 1$, alors l'algorithme de coupes n'ajoute pas de coupe dans le nœud courant, et il génère deux branches dans l'arbre binaire. Sinon, nous sélectionnons une composante connexe C_{i_0} tel que :

$$|\sigma(C_{i_0})| = \min\{|\sigma(C_i)|\ 1 \leq i \leq k\}.$$

Noter que C_{i_0} satisfait les conditions du Lemme 5.2.1. Nous introduisons la contrainte de bord induite par C_{i_0} si elle est violée par $\widehat{x}$. Sinon, si $q \geq 3$, alors la contrainte multibord associée à la partition $\Pi = (C_1, ..., C_q)$ est testée et elle est ajoutée si elle est violée par $\widehat{x}$.

Si la solution optimale courante $\widehat{x}$ est entière. Alors $\widehat{x}$ est le vecteur caractéristique d'un *mis* W de G. W induit un graphe $G_W = (V, [W, V \backslash W])$. Nous appliquons une procédure de marquage sur le graphe G_W pour déterminer le nombre de ses composantes connexes. Rappelons que si G_W est connexe, alors W est un *wcis*. Supposons que l'on ait trouvé q composantes connexes $(C_1, C_2, ..., C_q)$ dans G_W. Nous créerons une de bord de type (6.5) pour la composante connexe $C_{\widehat{i}}$ tel que :

$$|\sigma(C_{i_0})| = \min\{|\sigma(C_i)| \; 1 \leq i \leq k\}.$$

6.2 Protocole expérimental

L'algorithme de coupes et branchement est programmé en C en utilisant CPLEX12.4.0.0 sous la machine Intel(R) Xeon(R) CPU E-2600@2.60GHz, 64 GO ram. Le temps maximal d'exécution est fixé à 2 heures. Les paramètres fixés de CPLEX sont :

- contraintes de cliques (K) activées ;
- contraintes de cycles (C) (Zero-half cuts) activées ;
- contraintes de gomory (GM) (Gomory fractional cuts) activées ;
- algorithmes d'optimisation : le primal au nœud 0 et dual dans les autres nœuds de l'arbre binaire.

Nous présentons les résultats numériques pour la résolution du programme linéaire en nombres entiers ($\mathbb{P}^I$). Nous commençons par résoudre le programme linéaire ($\mathbb{P}_0^R$) :

$$(\mathbb{P}_0^R) \begin{cases} \min \sum_{u \in V} x(u) & & \\ x(u) + x(v) & \leq & 1, \; \forall (u,v) \in E \\ x(N[u]) & \geq & 1, \; \forall u \in V \\ x(N^2(u)) & \geq & x(u), \; \forall u \in V \\ \sum_{u \in V} d(u)x(u) & \geq & |V| - 1 \\ x(u) & \geq & 0, \; \forall u \in V \end{cases}$$

6.2.1 Instances

Dans le graphe des communications associé à un réseau de capteurs, le nœud règle sa puissance d'émission sans dépasser un certain seuil. La puissance nécessaire pour une transmission d'un nœud i à un nœud j est donnée par la formule suivante :

$$C(i,j) = a \times d(i,j)^{\alpha} + b$$

Où $d(i,j)$ est la distance entre i et j ; $a > 0$; $b \geq 0$; $\alpha \in [2,5]$. Cette puissance détermine la portée de communication (ou connectivité). La transmission de messages se fait entre les capteurs selon la portée (le rayon de communication). Nous générons une arête $\{i,j\}$ dans le graphe des communications si $C(i,j) \leq \min\{p(i), p(j)\}$. Dans la suite de nos tests, nous supposons que le rayon de communication est uniforme et constant pour tous les nœuds. Décrivons la génération des deux types de graphes :

- Les s-grilles sont caractérisées par leur taille $(m \times n)$.
- En ce qui concerne les graphes aléatoires, nous générons des points dans un carré de 10×10. Nous fixons ensuite un rayon de transmission unique pour générer les arêtes.

6.2.2 Descriptif des entrées des tableaux

Nous présentons dans cette section les résultats numériques pour pouvoir mesurer la performance de notre algorithme appliqué aux deux classes présentées précédemment.
La première colonne de la table contient la liste des instances traitées, le nombre donné indiqué à la fin du nom de chaque problème correspond à la taille du graphe. Les entrées des autres colonnes de gauche à droite sont :

Bin	:	nombre de vecteurs binaires générés dans la phase de branchement
binB	:	nombre de vecteurs binaires violés dans la phase de branchement
C	:	nombre d'inégalités de cycles impairs utilisées par CPLEX
Nfrac	:	nombre de vecteurs fractionnaires générés dans la phase de branchement
fraB	:	nombre d'inégalités de bord utilisées par l'algorithme de coupes pour les solutions fractionnaires
fraMB	:	nombre d'inégalités de multi-bords utilisées par l'algorithme de coupes pour les solutions fractionnaires
Opt	:	valeur de la solution optimale
Gap1	:	l'erreur relative entre Opt^0 et Opt. Gap1 = $(\frac{Opt-Opt^0}{Opt}) \times 100$ où Opt^0 est la valeur de la solution optimale du programme linéaire initial (P_0^R)
Gap2	:	l'erreur relative entre Opt^1 et Opt. Gap2 = $(\frac{Opt-Opt^1}{Opt}) \times 100$ où Opt^1 est la valeur de la solution optimale à la racine de l'arbre de branchement après l'ajout des inégalités de bords et multi-bords
Narbre	:	nombre de sommets de l'arbre générés dans la phase de branchement (en milliers)
CPU	:	temps total en h :m :s de l'algorithme de coupes et branchement.

6.2.3 Versions algorithmiques

Nous comparons deux versions de l'algorithme de coupes et branchement. L'algorithme I appelle la procédure de séparation uniquement si la solution courante est entière. L'algorithme II appelle la procédure de séparation pour toutes les solutions courantes entières et une partie des solutions fractionnaires.
Pour l'algorithme II, nous fixons un entier positif θ_2 et nous utilisons les procédures de séparations si et seulement si $Narbre = 0 \bmod \theta_2$.

6.2.4 Résultats numériques

Instances	Bin	binB	C	Opt	Gap_1	Gap_2	Narbre	CPU
10×10	4021	4016	839	30	17.7	15.0	153.3	00 :17 :12
11×11	3959	3951	809	34	12.1	9.6	437.5	01 :24 :26
$^*12 \times 12$	6244	6238	939	–	*17.4*	*15.1*	207.8	02 :00 :00
8×10	1330	1326	674	24	18.5	15.3	25.3	00 :01 :10
8×11	609	604	645	25	14.5	12.0	12.8	00 :00 :34
8×12	2632	2627	695	28	17.4	14.3	67.4	00 :06 :04
9×10	831	827	644	26	14.7	12.5	18.2	00 :01 :02
10×8	2070	2067	544	25	19.5	16.0	35.5	00 :02 :00
10×9	1026	1021	634	26	13.9	10.6	25.2	00 :02 :39
11×8	1869	1866	669	27	17.6	14.8	52.5	00 :04 :41
12×8	5633	5628	625	30	18.8	15.8	298.5	00 :37 :36
6×16	1746	1742	769	27	17.8	15.7	62.9	00 :03 :25
7×15	444	440	916	29	15.1	12.1	39.9	00 :02 :41
15×7	2067	2062	657	31	12.7	9.7	123.7	00 :12 :32
16×6	9674	9669	580	32	21.8	18.8	607.7	01 :53 :37

TABLE 6.1 – Algorithme I sur les s-*grilles*.

Comme le nombre d'inégalités de clique utilisées par l'algorithme de coupes et branchement est très faible, nous supprimons la colonne dans la table 6.2.

Grilles	Bin	binB	C	Nfra	fraB	fraMB	Opt	Gap_1	Gap_2	Narbre	CPU
10×10	4135	4131	766	209435	868	19	30	17.7	15.0	193.4	00 :27 :32
11×11	2801	2795	830	185394	1182	42	34	12.1	9.6	155.9	00 :33 :14
$^*12 \times 12$	5749	5743	909	293876	2767	67	–	*15.5*	*13.1*	177.1	02 :00 :00
8×10	1276	1274	635	23028	127	8	24	18.5	14.6	21.3	00 :01 :15
8×11	463	460	691	15358	129	11	25	14.5	12.0	13.5	00 :00 :37
8×12	2569	2566	759	72892	508	9	28	17.4	14.3	66.1	00 :06 :54
9×10	910	906	646	23807	162	6	26	14.7	12.5	19.6	00 :01 :20
10×8	1980	1977	482	36511	354	7	25	19.5	16.0	33.6	00 :02 :52
10×9	1301	1297	687	40336	228	4	26	13.9	10.6	30.2	00 :02 :09
11×8	1854	1848	588	52409	624	9	27	17.6	14.8	47.2	00 :04 :35
12×8	4877	4874	636	269436	2407	24	30	18.8	15.8	248.3	00 :41 :55
6×16	1733	1729	740	70171	200	10	27	17.8	15.7	65.1	00 :04 :30
7×15	641	636	855	53892	340	6	29	15.1	12.1	46.6	00 :03 :49
15×7	2723	2719	657	78528	962	10	31	12.7	9.7	60.2	00 :10 :38
16×6	9545	9542	635	420429	3270	66	32	21.8	18.8	385.7	01 :44 :14

TABLE 6.2 – Algorithme II sur les s-*grilles* ($\theta_2 = 100$)).

Toutes les grilles carrées de tailles inférieures à 81 sont traitées très rapidement (moins de 20 secondes) par les deux versions de l'algorithme de coupes et branchement. Sur les six instances ($grille_{10\times10}$, $grille_{11\times11}$, $grille_{8\times12}$, $grille_{12\times8}$, $grille_{15\times7}$ et $grille_{16\times6}$), l'algorithme II ($\theta = 100$) est meilleur (en temps d'exécution) trois fois et il est à égalité une fois avec

l'algorithme I. On constate une amélioration significative pour l'instance $grille_{11\times11}$ qui a été résolue en 33 minutes contre 84 pour l'algorithme I. La $grille_{12\times12}$ n'a pas été résolue par les deux versions après deux heures d'exécutions. Les autres grilles ont été résolues presque avec les mêmes performances. Nous nous continuerons les expérimentations sur les graphes aléatoires pour mieux conclure sur l'efficacité des deux versions.

$\|V\|$	Bin	binB	C	Opt	Gap_1	Gap_2	Narbre	CPU
300	13.1	10.7	23.0	29.3	3.9	1.1	< 0.1	00 :00 :02
350	54.6	50.8	143.9	42.8	5.0	1.7	0.6	00 :00 :06
400	91.1	87.3	345.1	52.8	5.9	2.5	2.4	00 :00 :35
450	81.8	76.6	276.4	52.3	5.5	2.8	1.0	00 :00 :13
500	44.0	39.9	104.3	52.6	4.4	2.2	0.2	00 :00 :07
550	31.0	26.9	163.8	52.6	5.0	2.7	0.3	00 :00 :12
600	112.5	107.8	640.3	67.2	4.9	2.8	1.6	00 :00 :42
650	150.3	144.5	1132.7	76.8	5.5	2.8	4.6	00 :02 :41
700	150.3	145.3	910.0	75.7	4.5	2.1	4.2	00 :03 :05
750	205.8	200.3	3297.5	79.3	6.1	3.7	19.9	00 :33 :53
800	133.2	127.6	2374.2	73.6	5.3	3.5	7.2	00 :17 :49
850	364.7	356.0	3649.0	102.7	5.3	2.5	38.7	00 :17 :57
900	20.8	16.0	2003.3	59.3	5.9	4.9	4.7	00 :06 :24
950	308.3	302.0	3960.0	98.0	4.9	2.8	13.8	00 :19 :56
1000	516.0	508.5	5281.0	111.0	4.9	2.4	58.1	00 :49 :09

TABLE 6.3 – Algorithme I sur les graphes aléatoires.

$\|V\|$	Bin	binB	C	Nfra	fraB	fraMB	Opt	Gap_1	Gap_2	Narbre	CPU
300	11.3	8.9	17.3	22.2	1.6	0.1	29.3	3.9	1.0	<0.1	00 :00 :01
350	49.6	46.1	172.0	1297.6	32.8	0.9	42.8	5.0	1.8	0.8	00 :00 :08
400	72.4	68.1	271.8	2879.1	69.8	2.1	52.8	5.9	2.2	1.7	00 :00 :26
450	72.4	68.1	257.6	1398.3	32.4	1.5	52.3	5.5	2.6	0.8	00 :00 :13
500	45.0	40.9	95.1	410.6	15.8	0.8	52.6	4.4	2.2	0.2	00 :00 :07
550	30.1	25.8	175.4	522.3	10.9	0.5	52.6	5.0	2.6	0.3	00 :00 :13
600	119.3	114.7	634.3	2936.2	92.7	6.7	67.2	4.9	2.9	1.7	00 :00 :55
650	178.0	171.8	1230.0	8497.2	236.0	5.0	76.8	5.5	2.7	5.1	00 :06 :42
700	82.7	78.2	477.8	2865.5	113.8	4.2	75.7	4.5	2.0	1.7	00 :01 :07
750	180.8	174.8	3339.3	49667.0	700.5	51.0	79.3	6.1	3.6	29.4	00 :35 :18
800	60.2	55.4	1393.6	6129.0	59.2	4.2	73.6	5.3	3.5	3.6	00 :06 :44
850	100.3	95.3	2461.0	14738.0	234.0	25.3	83.7	4.7	2.4	8.4	00 :09 :47
900	19.3	14.0	1040.8	3347.0	2.0	1.0	60.0	5.6	4.1	2.2	00 :03 :19
950	61.5	55.8	1246.0	5240.5	80.8	3.5	86.5	4.2	2.5	2.9	00 :02 :33
1000	270.0	264.0	2850.5	40184.5	1660.0	87.0	112.0	5.1	2.4	23.0	00 :12 :56

TABLE 6.4 – Algorithme II sur les graphes aléatoires ($(\theta_2 = 100)$).

Au cours de l'expérimentation nous avons constaté que les instances de faibles densités sont les plus difficiles à résoudre. C'est pour cette raison que nous avons choisi de tester seulement les instances de faibles densités. Les densités des graphes sont comme suit : les instances de

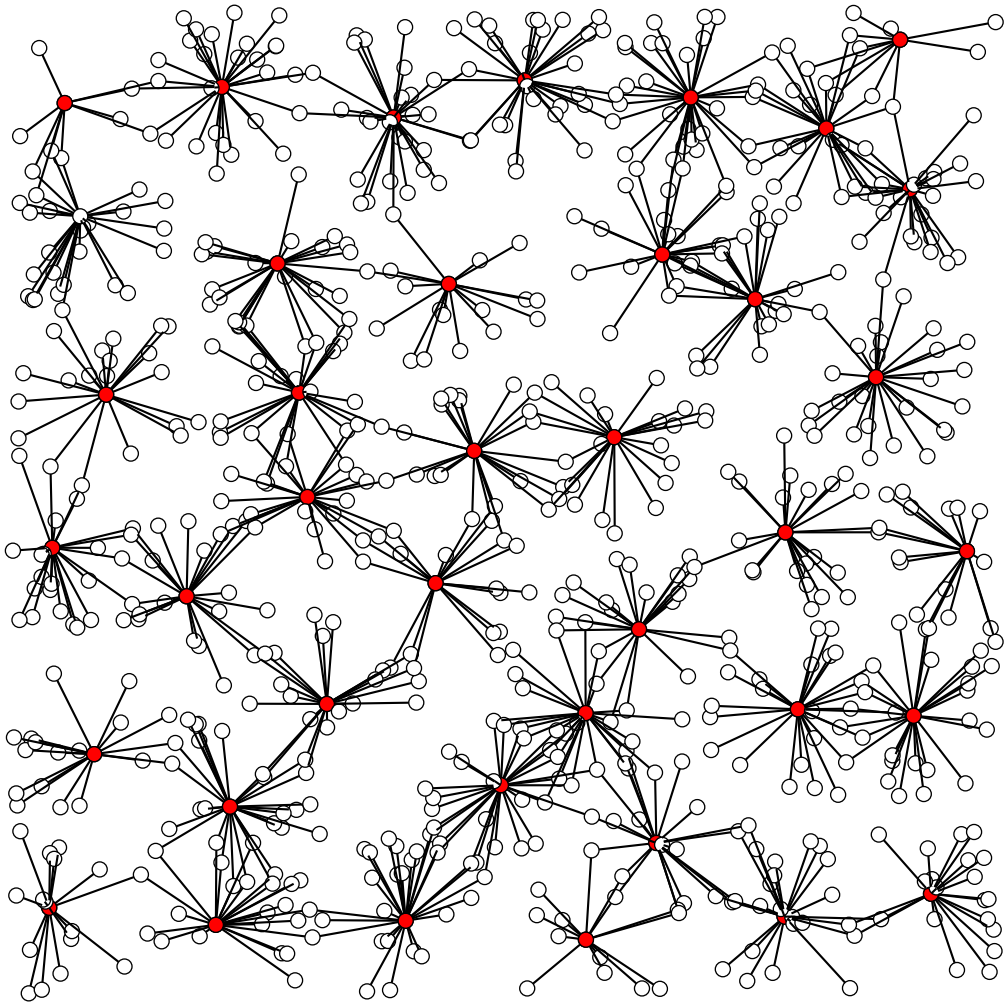


FIGURE 6.1 – Solution optimale pour un graphe de 800 sommets

tailles (100,150, 200,250, 300 et 350) sont respectivement à $(9\%, 5\%, 4.5\%, 4.5\%, 4\% et 2.5\%)$ les instances 400 à 550 ont une densité de 2%, les instances 600 à 900 sont à 1.5% et les instances 950 et 1000 sont à 1%. Après avoir testé plusieurs versions de l'algorithme de coupes et branchement, nous avons constaté que lorsque la solution trouvée sur un nœud donné de l'arbre est binaire l'ajout de la contrainte de bord est plus efficace que l'ajout d'une contrainte multibord pour couper le point. C'est pour ça que nous avons décidé de couper les solutions binaires par les inégalités de bord pour les deux versions de l'algorithme de coupes. Pour les solutions fractionnaires, nous avons constaté que l'ajout des contraintes multibord accélère dans certains cas l'algorithme de coupes et branchement. Les tables 6.1, 6.2, 6.3 et 6.4 montrent que le nombre de solutions fractionnaires généré par l'arbre de branchement est très grand. Pour réduire la taille des programmes linéaires sur chaque nœuds fractionnaires de l'arbre nous avons choisi une période de 100 de telle sorte à ne pas tester tous les nœuds fractionnaires. L'expérimentation a montré qu'à plus de 70% des solutions fractionnaires violées par les inégalités de bord et/ou multibord sont trouvées avec un seuil de 0.2.

Conclusion

Dans ce travail, nous considérons le problème de l'indépendant faiblement connexe de cardinalité minimum (MWCIS) dans un graphe non orienté connexe. Ce problème est NP-difficile en général. Une étude de sa complexité a été présentée pour certaines classes de graphes (bipartis, split-graphes, comparabilité). Nous énonçons aussi quelques propriétés structurelles des indépendants faiblement connexes dans un graphe. Nous avons traité le problème MWCIS sous deux angles, énumératif et polyédral.

Nous proposons un algorithme exact conçu *spécifiquement* pour le problème MWCIS dont les complexités en temps et en espace sont respectivement $O^*(1.4655^n)$ et $O(n^2)$.
Les résultats expérimentaux montrent que notre méthode d'énumération implicite peut gérer de manière satisfaisante des graphes de taille allant jusqu'à 120 nœuds mais une difficulté apparaît avec des graphes de faible densité à partir de 100 nœuds. Nous avons comparé notre algorithme avec l'approche indirecte qui consiste à combiner une énumération des indépendants maximaux du graphe avec un test de connexité. Pratiquement, notre approche semble plus efficace.

Nous examinons dans un second temps le problème MWCIS sous l'aspect polyédral. Une formulation en nombres entiers, basée sur les sommets, est décrite. Quand le graphe est un cycle impair, on obtient facilement la caractérisation linéaire complète du polytope. Nous avons alors examiné le polytope associé à ce problème lorsque l'on applique des opérations de composition de graphes (1-somme, ajout d'un sommet universel, ajout d'un sommet jumeau). Nous avons exhibé une famille d'inégalités valides pour le polytope $P_{wcis}(G)$ s'appuyant sur une partition de l'ensemble des sommets du graphe. Une méthode de coupes et branchement a été implémentée pour ce problème. Des heuristiques permettent de séparer les contraintes dites de bord et multibord. Les tests numériques indiquent que l'on peut traiter des graphes aléatoires jusqu'à 1000 sommets, et des s-grilles de taille 11×11 au maximum. Cette dernière classe de graphes fournit des instances particulièrement ardues pour le problème MWCIS.

Les expérimentations informatiques que nous avons effectuées induisent des questions théoriques. La première porte sur la nature du problème MWCIS dans les s-grilles, qui constituent un sous-ensemble de la classe des graphes presque bipartis, pour lesquels ce problème est NP-complet. Un autre point consiste à améliorer la complexité de l'algorithme d'énumération directe pour viser les complexités nettement meilleures des procédures énumérant les indépendants maximaux.

De nouvelles contraintes qui traduiraient plus efficacement la propriété globale de connexité sont à notre avis nécessaires pour le polytope des indépendants faiblement connexes. Plus pragmatiquement, notre séparation des contraintes de bord et multibord est heuristique. Une séparation exacte de ces contraintes améliorerait aussi l'algorithme de coupes et branchement.

Enfin, la structure de l'indépendant faiblement connexe doit être mise à l'épreuve dans les

réseaux de capteurs, en mesurant notamment sa capacité à fournir un support "topologique" aux protocoles de communications.

Voici la liste des soumissions et publications liées à la thèse.

Publications dans des revues internationales (avec comité de lecture)

- Fatiha Bendali, Jean Mailfert, Djelloul Mameri. On minimum weakly-connected independent sets for wireless sensor networks properties and enumerations algorithm. à paraître dans RAIRO Operations Research.

Travail soumis

- Fatiha Bendali, Jean Mailfert, Djelloul Mameri. The minimum Weakly Connected Independent Set Problem : Polyhedral results and Branch-and-cut. Soumis à Special issue of Discrete Optimization on Combinatorial Optimization (ISCO 2014).

Conférences nationales et internationales (avec comité de lecture)

- Fatiha Bendali, Jean Mailfert, Djelloul Mameri. Approche polyédrale pour le problème d'indépendant faiblement connexe. Présentation à ROADEF2014, Bordeaux (France).
- Fatiha Bendali, Jean Mailfert and Djelloul Mameri. Algorithme exact pour le problème d'indépendant faiblement connexe de cardinalité minimum. AlgoTel2012, Montpellier (France).
- Fatiha Bendali, Jean Mailfert, Djelloul Mameri. Indépendants maximaux et réseau de capteurs. Présentation à JPOC7. 2011 Valenciennes (France).
- Fatiha Bendali, Jean Mailfert, Djelloul Mameri. Un tour d'horizon sur les ensembles dominants dans les réseaux sans fil. Présentation à ROADEF2011, Saint-Etienne (France).

Rapports de recherche

- Fatiha Bendali, Jean Mailfert and Djelloul Mameri. On minimum weakly-connected independent sets for wireless sensor networks : properties and enumerations algorithm. Rapport de recherche RR-13-01, LIMOS, Clermont-Ferrand, France (2012).

Bibliographie

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. *Wireless sensor networks : A survey. Computing Networks*, 38 :393–422, 2002. (Cité en page 1.)

[2] K. M. Alzoubi, P. J. Wan, and O. Frieder. *Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. ACM Mobihoc*, 2002. (Cité en pages 2, 3 et 18.)

[3] K. M. Alzoubi, P. J. Wan, and O. Frieder. *Weakly Connected Dominating Sets and Spanners in Wireless Ad Hoc Networks. Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003. (Cité en page 3.)

[4] E. Balas. *Projection, Lifting and Extended Formulation in Integer and Combinatorial Optimization. Annals of Operation Research*, 140 :125–161, 2005. (Cité en pages 70 et 73.)

[5] C. Berge. *Graph theory and computing.* 1972. (Cité en page 9.)

[6] M. Bouchakour, T.M. Contenza, C.W. Lee, and A. R. Mahjoub. *On the dominating set polytope. European Journal of Combinatorics*, 29(3) :652–661, 2008. (Cité en page 3.)

[7] M. Bouchakour and A. R. Mahjoub. *One-node cutsets and the dominating set polytope. Discrete Mathematics*, (165-166) :101–123, 1997. (Cité en page 3.)

[8] M. Boulala and J.P. Uhry. *Polytope des indépendants d'un graphe serie-parallele. Discrete Mathematics*, 27 :225–243, 1979. (Cité en page 55.)

[9] N. Bourgeois, F. Della Croce, B. Escoffier, and V.Th. Paschos. *Fast algorithms for min independent dominating set. Discrete Applied Mathematics*, 161(4-5) :558–572, 2012. (Cité en pages 27 et 28.)

[10] M. S. Chang. *Efficient algorithms for the domination problems on interval and circular-arc graphs. SIAM Journal on computing*, 27 :1671–1694, 1998. (Cité en page 3.)

[11] Y. P. Chen and A. L. Liestman. *Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks. The 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002. (Cité en page 3.)

[12] X. Cheng, X. Cheng, D. Z. Du, and M. Cardei. *Connected Domination in multihop Ad Hoc Wireless Networks. Proceedings of the 6th International Conference on Computer Science and Informatics*, 2002. (Cité en pages 2 et 3.)

[13] V. Chvátal. *On certain polytopes associated with graphs. Journal of Combinatorial Theory*, (18) :138–154, 1975. (Cité en pages 3, 60 et 63.)

[14] V. Chvátal. *A greedy heuristic for the set-covering problem. Mathematics of Operations Research*, 4(3) :233–235, 1979. (Cité en page 2.)

[15] B. N. Clark, C. J. Colbourn, and D. S. Johnson. *Unit Disk Graphs. Discrete mathematics*, 86 :165–177, 1990. (Cité en page 2.)

[16] S. A. Cook. *The complexity of theorem-proving procedures. In proceedings 3rd Annual ACM Sypmosium on Theory of Computing, New York*, pages 151–158, 1971. (Cité en pages 10 et 13.)

[17] D. G. Corneil and Y. Perl. *Clustering and domination in perfect graphs. Discrete Applied Mathematics*, 9 :27–39, 1984. (Cité en page 3.)

[18] B. Das and V. Bharghavan. *Routing in ad Hoc Networks Using Minimum Connected Dominating Sets. IEEE International Conference*, 1 :376–380, 1997. (Cité en page 2.)

[19] M. Davis, G. Logemann, and D. Loveland. *A machin program for theorem-proving. Communications of the ASM*, (5) :394–397, 1962. (Cité en page 12.)

[20] M. Davis and H. Putnam. *A computing procedure for qualification theory. Journal of the ASM*, (7) :201–205, 1960. (Cité en page 12.)

[21] J. E. Dunbar, J. W. Grossman, J. H. Hattingh, S. T. Hedetniemi, and A. A. McRae. *On Weakly connected Domination in graphs. Discrete Mathematics*, 167-168 :261–269, 1997. (Cité en pages 3 et 20.)

[22] J. Edmonds. *Covers and packings in a family of sets. Bull. American Mathematical Society 68*, pages 474–499, 1962. (Cité en page 10.)

[23] F. Eisenbrand, G. Oriolo, G. Stauffer, and P. Ventura. *The Stable Set Polytope of Quasi-Line Graphs. Eleventh Conference on Integer Programming and Combinatorial Optimization, IPCO XI*, 2005. (Cité en page 3.)

[24] J. Mailfert F. Bendali and D. Mameri. *On minimum weakly-connected independent sets for wirless sensor networks : properties ans enumeration algorithm.* Technical report, Rapport limos 2013, RR-13-01. (Cité en page 56.)

[25] M. Farber. *Independent domination in chordal graphs. Operation Research Letters*, 1(4) :134–138, 1982. (Cité en pages 3 et 55.)

[26] M. Farber. *Domination, independent domination, and duality in strongly chordal graphs. Discrete Applied Mathematics*, (7) :115–130, 1984. (Cité en page 3.)

[27] M. Farber and J. M. Keil. *Domination in permutation graphs. Journal of algorithms*, 6(3) :309–321, 1985. (Cité en page 25.)

[28] E. Fleury and D. Simplot-Ryl. *Réseaux de capteurs, théorie et modélisation.* Lavoisier, 2009. (Cité en pages 1 et 2.)

[29] M. R. Garey and D. S. Johnson. *Computers and intractability : A guide to the theory of NP-completeness. W. H. Freeman et Co., New York*, 1979. (Cité en pages 2 et 3.)

[30] Michael R. Garey and David S. Johnson. *Computers and Intractability. A guide to the Theory of NP-Completeness.* 1980. (Cité en page 10.)

[31] S. Gaspers and M. Liedloff. *A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs. Fomin, LNCS*, 4271 :78–89, 2006. (Cité en page 27.)

[32] B. Gendron, A. Lucena, A. S. da Cunha, and L. Simonetti. *Benders Decomposition, Branch-and-Cut and Hybrid Algorithms for the Minimum Connected Dominating Set Problem. INFORMS Journal on Computing*, 2014. (Cité en pages 3 et 55.)

[33] A.M. Geoffrion. *Integer programming by implicit enumeration and Balas' Method. Siam Review*, 9(2) :178–190, 1967. (Cité en page 28.)

[34] S. Guha and S. Khuller. *Approximation algorithms for connected dominating sets. Algorithmica*, 20(4) :374–387, 1998. (Cité en page 2.)

[35] M. M. Halldórsson. *Approximating the minimum maximal independence number. Information Processing Letters, Elsevier*, 46(4) :169–172, 1993. (Cité en pages 20 et 22.)

[36] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in graphs : Advanced Topics. Marcel Dekker, Inc.*, 1998. (Cité en page 2.)

[37] M. Held and R.M. Karp. *A dynamic programming approach to sequencing problems. Journal of SIAM*, pages 196–210, 1962. (Cité en page 12.)

[38] E. Horowitz and S. Sahni. *Computing partitions with applications to the knapsack problem. Journal of the ASM*, (21) :277–292, 1974. (Cité en page 12.)

[39] Robert W. Irving. *On approximating the minimum independent dominating set. Information Processing Letters*, 37 :197–200, 1991. North-Holland. (Cité en page 24.)

[40] B. Jeremy, D. Min, T. Andrew, and C. Xiuzhen. *Connected Dominating set in Sensor networks and manets. Handbook of Combinatorial optimization. Springer.*, 2004. (Cité en page 2.)

[41] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. *On generating all maximal independent sets. Information Processing Letters*, pages 119–123, 1988. (Cité en page 27.)

[42] C. Laforest and R. Phan. *Experimentations on an exact algorithm for the minimum independent dominating set problem in graphs using clique partition. RAIRO*, 47(3) :199–221, 2013. (Cité en page 48.)

[43] Y. S. Li, M. T. Thai, F. Wang, C. W. Yi, P. J. Wan, and D. Z. Du. *On Greedy Construction of Connected Dominating Sets in Wireless Networks. Journal on Wireless Communications and Mobile Computing*, 5(8) :927–932, 2005. (Cité en pages 2 et 3.)

[44] C. Liu and Y. Song. *Exact algorithms for finding the minimum independent dominating set in graphs. ISAAC, LNCS*, 4288 :439–448, 2006. (Cité en page 48.)

[45] A.R. Mahjoub and J. Mailfert. *On the independent dominating set polytope. European Journal of Combinatorics*, (27) :601–616, 2005. (Cité en pages 3 et 65.)

[46] J.M. Moon and L. Moser. *On cliques in graphs. Israel J. Math.*, 3 :23–28, 1965. (Cité en page 27.)

[47] G.L. Nemhauser and L.E. Trotter. *Properties of vertex packing and independence system polyhedra. Mathematics Programming*, (6) :48–61, 1974. (Cité en page 3.)

[48] M.W. Padberg. *On note on zero-one programming. Operation Research*, (23) :833–837, 1975. (Cité en page 3.)

[49] M.W. Padberg. *On the complexity of set packing polyhedra. Annals of Discrete Mathematics*, (1) :421–434, 1977. (Cité en page 3.)

[50] A. Pêcher and A. K. Wagler. *On facets of stable set polytopes of claw-free graphs with stability number three. Disctete Mathematics*, (310) :493–498, 2010. (Cité en page 3.)

[51] A. Potluri and A. Negi. *Some observations on algorithms for computing minimum independent dominating set. IC3, Communications in Computer and Information Science*, 168 :57–68, 2011. (Cité en page 48.)

[52] W. R. Pulleyblank. *Polyhedral combinatorics. Handbooks in Operations research and Management science*, pages 371–446, 1989. (Cité en page 13.)

[53] G. Reinelt. *TSPLIB-A Traveling Salesman Problem Library. INFORMS Journal on Computing*, 3(4) :376–384, 1991. (Cité en page 41.)

[54] A. C. Santos, F. Bendali, J. Mailfert, C. Duhamel, and K. M. Hou. *Heuristices for designing Energy-efficient Wireless Sensor Network Topologies. Journal of networks*, 4(6) :436–444, 2009. (Cité en pages 3 et 40.)

[55] A. Saxena. *Polyhedral studies in Domination Graph Theory.* 2003. (Cité en page 3.)

[56] Alexander Schrijver. *Algorithms and Combinatorics*, volume A. 2000. (Cité en page 13.)

[57] L. Simonetti, A. S. da Cunha, and A. Lucena. *The Minimum Connected Dominating Set Problem : Formulation, Valid Inequalities and a Branch-and-Cut Algorithm. J. Pahl, T. Reiners and S. VoB, eds. 5th Internat. Network Optim. Conf., Hamburg, Lecture Notes in Computer Science (Springer, Berlin).*, 6701 :162–169, 2011. (Cité en page 3.)

[58] L. Simonetti, A. Salles da Cunha, and A. Lucena. *The Minimum Connected Dominating Set Problem :Formulation, Valid Inequalities and a Branch-and-Cut Algorithm. Discrete Mathematics*, pages 162–169, 2011. (Cité en page 55.)

[59] I. Stojmenovic, M. Seddigh, and J. Zunic. *Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. IEEE Transactions on parallel and distributed systems*, 13(1) :14–25, 2002. (Cité en page 2.)

[60] R. E. Tarjan and A. E. Trojanowski. *Finding a maximum independent set.* STAN-CS-76-550, 1976. (Cité en page 12.)

[61] L.E. Trotter. *A class of facet producing graphs for vertex packing polyhedra. Disctete Mathematics*, (12) :373–388, 1975. (Cité en page 3.)

[62] P. J. Wan, K. M. Alzoubi, and O. Frieder. *Distributed construction of connected dominating set in wireless ad hoc networks. Mobile Networks and Applications*, 9 :141–149, 2004. (Cité en pages 2 et 3.)

[63] P. J. Wan, L. Wang, and F. Yao. *Two-Phased Approximation Algorithms for Minimum CDS in Wireless Ad Hoc Networks. IEEE ICDCS*, pages 337–344, 2008. (Cité en page 3.)

[64] G. J. Woeginger. *Exact algorithms for NP-hard problems : A survey. Operation Research Lettres - Eureka, You Shrink !, LNCS 2570*, pages 185–207, 2003. (Cité en page 11.)

[65] W. Wu, H. Du, X. Jia, Y. Li, and S. C. H. Huang. *Minimum connected dominating sets and maximal independent sets in unit disk graphs. Theoretical Computer Science*, 352 :1–7, 2006. (Cité en page 2.)