d-extensibles, d-bloqueurs et d-transversaux de problèmes d'optimisation combinatoire

Grégoire Cotté

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

le cnam

École Doctorale d'Informatique Télécommunications et Électronique

Laboratoire CEDRIC, équipe Optimisation Combinatoire

THÈSE DE DOCTORAT

présentée par : **Grégoire COTTÉ**

soutenue le : **9 juin 2016**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Spécialité : **Informatique**

d-extensibles, d-bloqueurs et d-transversaux de problèmes d'optimisation combinatoire

THÈSE dirigée par

Mme COSTA Marie-Christine	*PR, ENSTA Paristech*
M. PICOULEAU Christophe	*PR, CNAM*

RAPPORTEURS

Mme BAZGAN Cristina	*PR, Université Paris Dauphine*
M. MEUNIER Frédéric	*PR, Ecole Nationale des Ponts et Chaussées*

EXAMINATEURS

M. CORNAZ Denis	*MCF, Université Paris Dauphine*
M. HUDRY Olivier	*PR, Télécom Paristech*
M. RIES Bernard	*MCF, Université de Fribourg*

For what do we live, but to make sport for our neighbors, and laugh at them in our turn ?

Jane Austen, Pride and Prejudice

Quelle merveilleuse harmonie règne dans l'univers. Bien que pris dans le détail ça fasse un fameux merdier.

René Barjavel

Remerciements

Je tiens tout d'abord à remercier mes encadrants Christophe Picouleau et Marie-Christine Costa d'avoir accepté d'encadrer ma thèse bien que je ne boive pas de café.

Je remercie Cristina Bazgan et Frédéric Meunier de m'avoir fait l'honneur d'être rapporteurs de cette thèse. Je remercie également Olivier Hudry, Bernard Ries et Denis Cornaz d'avoir accepté de faire partie de mon jury.

Je souhaite remercier l'ensemble de l'équipe Optimisation Combinatoire du laboratoire CEDRIC. Merci tout particulièrement à Sourour Elloumi qui a encadré mon stage de deuxième année d'école d'ingénieur et m'a donné goût à la recherche, ainsi qu'à Amélie Lambert, Agnès Plateau, Alain Billonnet, Alain Faye, Cédric Bentz, Daniel Porumbel et Eric Soutil. Je remercie également les autres occupants des bureaux du 55 rue Turbigo. Merci donc à Stéphane Rovedakis, pour avoir accepté d'être dans mon jury de mi-parcours, ainsi qu'à Viviane Gal, Anne Wei, Tifanie Bouchara, Sami Taktak, Daniel Enselme, Christophe Pitrey, Françoise Sailhan et Henri Pugliese. Je remercie aussi les membres de l'UMA à l'Ensta pour leur accueil quand j'y passais, particulièrement Christophe Mathulik pour le bon fonctionnement de l'informatique et Maurice Diamantini pour les leçons de football de table.

Je tiens à remercier tous ceux avec qui j'ai eu le plaisir de travailler à Paris 6 dans le cadre de mon monitorat, Karine Heydemann, Pierre Fouilhoux, Thibaut Lust, Mourad Gouicem, Sylvain Lobry, Lucien Goubet et Alexandre Wallet.

Je remercie mes collègues, Dan Gugenheim, Pierrick Vallat et Nicolas Wolters, pour leur soutien pendant ces derniers mois.

Je souhaite aussi remercier mes collègues doctorants que j'ai eu le plaisir de côtoyer au

cours de ces dernières années. Merci à Pierre-Louis pour son aide sur le codage de l'échelle télescopique Marigot. Merci Sabine pour ta contribution au nombre de (bons ?) jeux de mots à la minute lors de tes passages au bureau. Merci à Dimitri pour les discussions sur les aventures de Flacuf. Merci à Anthony et Nicolas pour leur accueil quand je passais à l'Ensta. Merci Lilia pour les nombreux échanges sur des sujets divers (et d'été). Merci à Thibaut avec qui j'ai eu le plaisir de parcourir la même route depuis le Master.

Merci également à Lucile pour l'aide apportée pendant son stage.

Je souhaite remercier tous ceux qui, sans être impliqués directement dans le déroulement de la thèse, m'ont soutenu ou aidé à me changer les idées.

Merci tout d'abord à Gen et David pour nos week-ends de team-building interdisciplinaires, la bonne humeur qui y régnait et les sujets de discussions variés, de la construction des statues de poulets à l'optimisation dans les jeux de sociétés. Merci Rémy pour tous les liens que tu m'as envoysé et nos nombreux échanges. Merci à Samantha, Charlotte, Marion, Pierre et Yohann pour les univers que nous avons parcourus. Enfin, merci également à Florian, Tiphaine, Haïtem, Remi, Charles-Edouard, Charles, Delphine, Jocelyn, Cyrille, Claire et à tous ceux que j'ai oublié de citer.

Enfin, je tiens à remercier ma famille pour son soutien pendant ces trois ans. Merci à Fabienne et Lukas pour leur accueil à Marseille. Merci à Marie-Claude et Jean-Pierre pour les nombreux excellents repas à Evry. Merci également à mon frère pour nos discussions sur les derniers livres que nous avions lu et merci à mes grand-parents. Enfin, je tiens à remercier tout particulièrement mes parents qui m'ont toujours soutenu tout au long de mon parcours, dans les moments difficiles de la thèse et sans qui je n'aurais probablement pas réussi à arriver au bout.

Résumé

Dans cette thèse, nous étudions trois catégories de problèmes : les d-extensibles, les d-bloqueurs et les d-transversaux.

Les d-extensibles de stables optimaux sont des ensembles de sommets d'un graphe G tels que tout stable de cardinal d du sous-graphe induit par un d-extensible peut être étendu à un stable optimal de G à l'aide de sommets qui n'appartiennent pas au d-extensible. Nous étudions les d-extensibles de cardinal maximal de stables dans les graphes bipartis. Nous démontrons quelques propriétés structurelles puis nous déterminons une borne inférieure du cardinal maximal d'un d-extensible. Nous étudions quelques classes de graphes dans lesquelles déterminer un d-extensible optimal de stables est un problème polynomial. Nous nous intéressons ensuite aux d-extensibles de stables dans les arbres. Nous prouvons plusieurs propriétés structurelles, déterminons une autre borne inférieure du cardinal maximal d'un d-extensible et étudions quelques classes d'arbres dans lesquelles déterminer un d-extensible optimal de stables est un problème polynomial.

Les d-bloqueurs de stables sont des ensembles de sommets d'un graphe G tels que, si on retire les sommets d'un d-bloqueur, le cardinal maximal d'un stable du graphe induit par les sommets restants est inférieur d'au moins d au cardinal maximal d'un stable du graphe initial. Nous nous intéressons ici aux d-bloqueurs de coût minimal de stables dans les arbres. Après avoir prouvé une caractérisation des d-bloqueurs de stables dans les arbres, nous démontrons que déterminer un d-bloqueur de coût minimal de stable est un problème polynomial dans une classe d'arbres particulière.

Soit Π un problème d'optimisation sur un ensemble d'éléments fini. Un d-transversal de Π est un ensembles d'éléments tel que l'intersection entre le d-transversal et toute

solution optimale au problème Π est de cardinal supérieur égal à d. Nous proposons ici une approche de génération de contraintes pour déterminer des d-transversaux de cardinal maximal de problèmes modélisés par des programmes mathématiques en variables binaires. Nous étudions deux variantes de cette approche que nous testons sur des instances de graphes générés aléatoirement pour déterminer des d-transversaux de stables optimaux et des d-transversaux de couplages optimaux

Mots clés : d-extensibles, d-bloqueurs, d-transversaux, graphe, biparti, stable maximum

Abstract

In this thesis, we study three types of problems : the d-extensibles sets, the d-blockers and the d-transversals.

In a graph G, a d-extensible set of maximum independent sets is a subset of vertices of G such that every stable set of cardinality d in the subgraph restricted to the d-extensible set can be extented to a maximum stable set of G using only vertices that do not belong to the d-extensible set. We study d-extensible sets of mxaimum cardinality of stable sets in bipartite graphs. We show some structural properties and we determine a lower bound of the maximum cardinality of a d-extensible set. We consider some classes of graph where finding an optimum d-extensible set can be done in polynomial time. Then, we study the d-extensibles sets of stable sets in trees. We prove some properties on the structures of the d-extensibles sets and we determine another lower bound of the maximum cardinality of a d-extensible set. Finaly, we study somme classes of tree where a d-extensible sets of maximum cardinality can be done in polynomial time.

In a graph G, a d-blocker is a subset of vertices such that, if removed, a maximum stable set of the resulting subgraph is of cardinality at most the cardinality of a maximum stable set of G minus d. We study d-blocker of minimal cost of stable sets in tree.We prove a caracterisation of d-blockers in tree and we study a particular classe of trees where computing a d-blocker of minimal cost of stable sets can be done in polynomial time.

Let Π be an optimisation problem on a finite set of elements. A d-transversal of Π is a subset of elements such that the intersection between the d-transversal and every optimal solution of Π contains at lest d elements. We propose an approach to compute d-transversal of any optimisation problem modelised by mathematical program with binary variables.

We use a contraints generation approach. We compare two variations of this approach on randomly generated graph by computing d-transversals of stables sets and d-transversals of matching.

Keywords : d-extensibles, d-blockers, d-transversal, graph, bipartite, maximum stable set

Table des matières

Introduction

Pour faire fonctionner un système de façon optimale, on peut être amené à résoudre un problème d'optimisation combinatoire. Une fois le mode de fonctionnement choisi, on peut se demander si certains éléments du système sont vulnérables aux pannes. Deux questions peuvent alors se poser.

La première question est : comment protéger le système le plus efficacement possible ? On suppose que les éléments du système sont faillibles et qu'un comportement optimal du système ne peut durer indéfiniment. Il est donc nécessaire d'effectuer une maintenance sur les éléments ou de les protéger contre de potentielles attaques pour s'assurer que le système puisse fonctionner correctement.

La deuxième question est liée à la première, mais considère le point de vue inverse : comment attaquer le système le plus efficacement possible ? En identifiant des éléments vulnérables, un attaquant peut utiliser ses ressources pour faire le plus de dégâts possibles ; un défenseur peut déterminer quels éléments ont le plus besoin de protection. Par exemple, si un fonctionnement optimal du système correspond à résoudre un problème de plus court chemin sur un ensemble de routes, un attaquant peut choisir quelles routes couper et un défenseur quelles routes protéger.

Ainsi, utiliser un système nécessite de savoir comment prévenir les pannes et identifier des éléments vulnérables. Il s'agit de résoudre des problèmes d'attaques et de défenses sur le système

Dans cette thèse, nous supposons qu'un système est modélisé par un graphe et que son fonctionnement optimal correspond à un paramètre de ce graphe.

Dans cette thèse, nous supposons qu'un système est modélisé par un graphe, et qu'un

fonctionnement optimal correspond à une solution optimale d'un problème d'optimisation sur ce graphe. Par exemple, si les éléments du système correspondent aux sommets du graphe, les éléments en fonctionnement peuvent devoir former un stable. Un fonctionnement optimal du système correspondra donc à un stable optimal du graphe. Nous nous intéressons à trois catégories de problèmes : les d-extensibles, les d-bloqueurs et les d-transversaux. Ces trois problèmes considèrent une situation d'attaque et de défense d'un système selon des perspectives différentes.

La recherche de d-extensibles, se place du point de vue d'un défenseur. On considère pour ces problèmes que les éléments du système peuvent subir des pannes, qui peuvent être dues à l'usure ou à un sabotage. Le défenseur peut empêcher ces dysfonctionnements, par exemple avec une maintenance des éléments du système. Mais cette protection a un coût, le défenseur souhaite donc protéger un minimum d'éléments du système de façon à tolérer un certain seuil de défaillances des éléments qui ne sont pas protégés.

La recherche de d-bloqueurs, se place dans une perspective d'attaque. Il s'agit cette fois d'identifier et attaquer des éléments vulnérables d'un système de façon à en diminuer l'efficacité . On souhaite déterminer quels éléments du système il faut retirer pour garantir que le fonctionnement optimal du reste du système soit inférieur à un seuil fixé par rapport au fonctionnement optimal du système intact. Cela peut être utile pour un attaquant, ou un défenseur, qui souhaite déterminer les failles d'un système.

La recherche des d-transversaux, considère à nouveau le point de vue d'un attaquant. Contrairement au cas précédent, on suppose cette fois que le défenseur ne sait pas que certains éléments sont défaillants. Ainsi, il utilisera un fonctionnement optimal du système complet. Le but de l'attaquant est d'intercepter toutes les stratégies possibles du défenseur. Il doit donc choisir quels éléments attaquer en s'assurant que tout fonctionnement du système choisi par le défenseur sera dégradé.

Cette thèse est structurée de la façon suivante.

Le premier chapitre expose des notions de théorie des graphes, une caractérisation des stables dans les graphes bipartis, les définitions des problèmes étudiés et des généralités sur la programmation mathématique biniveau.

Le deuxième chapitre porte sur la programmation mathématique biniveau. Après avoir exposé les définitions générales de la programmation mathématique biniveau et les approches de résolution usuelles, nous nous intéressons à une classe de programmes mathématiques biniveaux en nombre entiers pour lesquels nous tentons d'étendre une méthode de résolution de la littérature.

Le troisième chapitre est consacré aux d-extensibles de cardinal maximal de stables optimaux dans les graphes bipartis. Nous donnons leur définition et quelques propriétés structurelles. Nous déterminons une borne inférieure de leur cardinalité maximale.Nous nous intéressons ensuite aux d-extensibles de stables lorsque le graphe est un arbre. Nous montrons que déterminer un d-extensible de cardinal maximal est un problème de complexité polynomiale dans certaines classes d'arbres.

Le quatrième chapitre porte sur les d-bloqueurs de coût minimal de stables optimaux lorsque le graphe est un arbre. Après avoir donné une caractérisation, nous montrons que déterminer un d-bloqueur de coût minimal de stables optimaux est un problème de complexité polynomiale pour une classe d'arbres particulière.

Dans le cinquième et dernier chapitre nous nous intéressons aux d-transversaux de problèmes modélisés par des programmes mathématiques en variables binaires. Après avoir détaillé quelques cas particuliers de d-transversaux pondérés de stables de cardinal maximal dans les graphes bipartis, nous proposons une approche de résolution par génération de contraintes pour déterminer des d-transversaux optimaux de problèmes d'optimisation en variables binaires. Nous étudions deux variantes de cette approche que nous comparons en les testant sur des graphes générés aléatoirement en déterminant des d-transversaux optimaux de stables optimaux et des d-transversaux optimaux de couplages optimaux.

Enfin, nous proposons quelques perspectives dans la conclusion.

Chapitre 1

Définitions et Généralités

Dans ce chapitre nous définissons les notions qui serviront dans les chapitres suivants. La première section introduit des notions classiques de théorie des graphes. La deuxième section donne une caractérisation des ensembles stables dans les graphes bipartis. Les trois sections suivantes donnent les définitions spécifiques aux chapitres 3, 4 et 5. Ces définitions seront rappelées dans leurs chapitres respectifs.

1.1 Notions générales de théorie des graphes

Cette section présente des notions classiques de théorie des graphes. Les notations proviennent de [42] (Picouleau, Faure, Lemaire) et de [71] (Schrijver).

Un graphe non orienté G est défini par un ensemble de sommets V et un ensemble d'arêtes E qui est un ensemble de paires non ordonnées d'éléments distincts de V. On notera $G = (V, E)$. Les arêtes $\{u, v\}$ où $u \in V$ et $v \in V$ pourront être notés uv ou vu. Si $uv \in E$ alors les sommets u et v sont dits adjacents ou voisins. Pour tout sommet $u \in V$ on note $\Gamma(u)$ l'ensemble de ses voisins et on note $\delta_G(u)$ le nombre de ses voisins qu'on nomme le degré du sommet. Pour tout sous-ensemble $\hat{V} \subset V$ on note $\Gamma(\hat{V})$ l'ensemble des voisins des sommets de $\hat{V}$ qui n'appartiennent pas à $\hat{V}$.

Une chaîne entre deux sommets u et v d'un graphe est un ensemble de sommets successifs et distincts $v_0 v_1 .. v_k$ tel que $v_0 = u$, $v_k = v$ et $\forall i \in [\![1, k]\!]$, $v_{i-1} v_i \in E$. La longueur d'une chaîne est le nombre d'arêtes de la chaîne. Un graphe est dit connexe s'il existe une chaîne entre tout couple de sommets. Un cycle est une chaîne dont le premier élément est

aussi le dernier donc si $v_k = v_0$.

Un graphe $\hat{G} = (\hat{V}, \hat{E})$ est appelé un sous-graphe de $G = (V, E)$ induit par $\hat{V}$ si $\hat{V} \subseteq V$ et $\hat{E} = \{uv \in E, u \in \hat{V}, v \in \hat{V}\}$. On note $G[\hat{V}]$ le sous-graphe de G induit par $\hat{V}$.

Un graphe $\hat{G} = (V, \hat{E})$ est appelé un graphe partiel de $G = (V, E)$ si $\hat{E} \subsetneq E$.

Un *ensemble stable* $S \subset V$ de G, aussi appelé stable ou ensemble indépendant, est un ensemble de sommets qui pris deux à deux sont non adjacents. On note $\alpha(G)$ le cardinal maximal d'un stable de G.

Considérons le problème STABLE suivant :

— Instance : Un graphe $G = (V, E)$, un entier positif $k \leqslant |V|$

— Question : Est-ce que G contient un stable S de taille k ou plus ?

Le problème STABLE est NP-complet [46].

On définit une partition de l'ensemble des sommets de G :

— Les sommets *forcés* sont les sommets qui appartiennent à tous les stables de cardinal maximal de G. L'ensemble des sommets forcés est noté $\Xi(G)$. Le nombre de sommets forcés est noté $\xi(G) = |\Xi(G)|$.

— Les sommets *exclus* sont les sommets qui n'appartiennent à aucun stable de cardinal maximal de G. L'ensemble des sommets exclus est noté $\Psi(G)$. Le nombre de sommets exclus est noté $\psi(G) = |\Psi(G)|$.

— Les sommets *libres* sont les sommets qui ne sont ni forcés ni exclus. L'ensemble des sommets libres est noté $\Phi(G)$. Le nombre de sommets libres est noté $\phi(G) = |\Phi(G)|$.

L'ensemble des sommets forcés est parfois noté *core(G)* et l'union des sommets forcés et des sommets libres est noté *corona(G)* [20].

Un *couplage* $C \subset E$ de G est un ensemble d'arêtes qui n'ont aucune extrémité commune. On note $\mu(G)$ le cardinal maximal d'un couplage de G. Pour tout couplage M, on note $V(M)$ l'ensemble des sommets du graphe incidents à une arête de M.

Considérons le problème COUPLAGE suivant :

— Instance : Un graphe $G = (V, E)$, un entier positif $k \leqslant |V|$

— Question : Est-ce que G contient un couplage M de taille k ou plus ?

Le problème COUPLAGE est polynomial [40].

On définit une partition de l'ensemble des arêtes de G :

— Les arêtes *forcées* sont les arêtes qui appartiennent à tous les couplages de cardinal maximal de G. L'ensemble des arêtes forcées est noté $\Xi'(G)$. Le nombre d'arêtes forcées est noté $\xi'(G) = |\Xi'(G)|$.

— Les arêtes *exclues* sont les arêtes qui n'appartiennent à aucun couplage de cardinal maximal de G. L'ensemble des arêtes exclues est noté $\Psi'(G)$. Le nombre d'arêtes exclues est noté $\psi'(G) = |\Psi'(G)|$.

— Les arêtes *libres* sont les arêtes qui ne sont ni forcées ni exclues. L'ensemble des arêtes libres est noté $\Phi'(G)$. Le nombre d'arêtes libres est noté $\phi'(G) = |\Phi'(G)|$.

Un graphe est dit biparti si V peut être séparé en deux ensemble disjoints B et R tels que chacun de ces ensembles forme un stable. Autrement dit, chaque arête de E a pour extrémités un sommet de B et un sommet de R. On peut aussi définir les graphes bipartis comme des graphes sans cycle de longueur impaire. On note $G = (B, R, E)$ un graphe biparti.

Un arbre est un graphe connexe et sans cycles. Notons qu'un arbre est un graphe biparti.

Le graphe complémentaire d'un graphe $G = (V, E)$ est le graphe $G = (V, \hat{E})$ tel que $\hat{E} = \{ij, ij \notin E\}$.

Un graphe $G = (V, E)$ est dit orienté si les éléments de E sont des couples ordonnés de sommets. On nomme alors arcs les éléments de E. Étant donnés deux sommets $u \in V$ et $v \in V$, on note (u, v) l'arc d'origine u et d'extrémité v.

Un chemin entre deux sommets u et v d'un graphe orienté est un ensemble de sommets successifs $v_0 v_1 .. v_k$ tel que $v_0 = u$, $v_k = v$ et $\forall i \in [\![1, k]\!]$, $v_{i-1} v_i \in E$. La longueur d'un chemin est le nombre d'arcs du chemin. Un circuit est un chemin dont le premier élément est aussi le dernier donc si $v_k = v_0$.

Une arborescence est obtenue en orientant un arbre de façon à ce qu'il existe un sommet

r tel que pour tout sommet $x \in V$ du graphe, il existe un chemin de r à x. Le sommet r est appelé la racine de l'arborescence.

1.2 Une caractérisation des stables dans les graphes bipartis

Nous présentons ici une caractérisation des stables de cardinal maximal issue de [17] qui sera utilisée dans les sections suivantes.

Soit $w : V \to \mathbb{N}$ une fonction de poids des sommets d'un graphe $G = (V, E)$. Soit $\hat{V} \subset V$ un sous-ensemble de sommets. On note $w(\hat{V})$ la somme des poids des éléments de $\hat{V}$. On considère le problème STABLEW de recherche de stables de poids maximal :

— Instance : Un graphe $G = (V, E)$, une fonction de poids $w : V \to \mathbb{N}$, un entier positif $k \leqslant w(V)$

— Question : Est-ce que G contient un stable S tel que $w(S) \geqslant k$?

Remarquons que le problème STABLE correspond au problème STABLEW dans le cas où $\forall x \in V$, $w(x) = 1$. On note $\alpha_w(G)$ le poids maximal d'un stable de G.

Propriété 1.2.1. *[17] Soit un graphe biparti $G = (B, R, E)$. Tous les sommets de G sont libres si et seulement si $\alpha_w(G) = w(B) = w(R)$.*

Corollaire 1.2.1. *[17] Si $\forall x \in V$, $w(x) = 1$ alors les sommets de G sont libres si et seulement si $\mu(G) = \alpha(G) = \frac{\phi(G)}{2} = \frac{|B \cup R|}{2}$. Donc G admet un couplage parfait.*

Propriété 1.2.2. *[17] Soit un graphe biparti $G = (B, R, E)$ dont tous les sommets sont libres. Il existe une partition maximale $\{V_1, ..., V_K\}$ des sommets de G telle que $w(B \cap V_i) = w(R \cap V_i)$ et pour tout stable de poids maximal S^* de G, soit $S^* \cap V_i = R \cap V_i$, soit $S^* \cap V_i = B \cap V_i$.*

On note $C_i = G[V_i]$ le sous-graphe de G induit par l'ensemble V_i. Le sous-graphe C_i est appelé une *composante* de G.

Propriété 1.2.3. *[17] Une composante C_i de G admet exactement deux stables de poids maximal : $V_i \cap B$ et $V_i \cap R$.*

Construisons le graphe orienté $G_c = (V_c, A_c)$ de la façon suivante : chaque sommet de V_c porte le nom d'une composante de G : V_c=$\{C_i\ \forall i = 1..K\}$ et $A_c = \{(C_i, C_j)\ \forall i \neq j$ il existe $xy \in E$ avec $x \in V_i \cap B$ et $y \in V_j \cap R\ \}$. Ce graphe est appelé *graphe des composantes* de G.

Propriété 1.2.4. *[17] G_c est un graphe sans circuits.*

Pour tout sommet x de G on note $C(x)$ la composante qui contient le sommet x. Soient deux composantes $C(x)$ et $C(y)$ de G_c. On note $C(x) \rightarrow C(y)$ le fait qu'il existe un chemin de $C(x)$ à $C(y)$ dans G_c, le chemin pouvant être de longueur nulle si $C(x) = C(y)$. On note $V(C_i)$ l'ensemble des sommets contenus dans la composante C_i.

Propriété 1.2.5. *[17] Soient S^* un stable de cardinal maximal de G et x et y deux sommets de S^* tels que $C(x) \rightarrow C(y)$. Si $x \in B$ alors $y \in B$. Si $y \in R$ alors $x \in R$.*

Corollaire 1.2.2. *[17] Soient deux composantes $C(x)$ et $C(y)$ de G. Si $C(x) \rightarrow C(y)$ alors tout stable optimal S^* tel que $S^* \cap V(C(x)) \subsetneq B$ vérifie $S^* \cap V(C(y)) \subsetneq B$.*

Corollaire 1.2.3. *[17] Soient S^* un stable de cardinal maximal de G et x et y deux sommets de G tels que $C(x) \rightarrow C(y)$. Si $x \in B \cap S^*$ et $y \in R$, alors $y \notin S^*$ et si $x \in R$ et $y \in B$ alors $S^* \cap \{x, y\} \neq \emptyset$.*

Dans la suite de cette section, on considérera le cas où les poids des sommets de G sont unitaires. Dans ce cas, la partition des sommets de G correspond aux composantes connexes du graphe partiel $G = (V, E - \Psi'(G))$. On peut définir l'ensemble des arcs du graphe des composantes ainsi : $A_c = \{(C_i, C_j)\ \forall i \neq j\ ,\ \exists xy \in \Psi'(G),\ x \in V(C_i) \cap B,\ y \in V(C_j) \cap R\}$. On a aussi la propriété suivante :

Propriété 1.2.6. *[17] Soit $g = (B, R, E)$ un graphe biparti dont tous les sommets sont libres. Toute composante de G admet un couplage parfait.*

Exemple 1.2.1. *Appliquons la caractérisation sur le graphe biparti $G = (B, R, E)$ de la Figure 1.1. Les sommets de B sont les losanges et les sommets de R sont les cercles.*

Tout d'abord, on commence par identifier et retirer les arêtes exclues. Elles sont en pointillé dans la Figure 1.2. Après le retrait, on obtient la Figure 1.3. Les composantes connexes du graphe de la Figure 1.3 correspondent aux composantes du graphe G.

On construit ensuite le graphe des composantes de G. Comme indiqué sur la Figure 1.3, le graphe contient quatre composantes.Les sommets du graphe des composantes correspondent aux composantes et sont indiqués dans la Figure 1.4.

Les arcs du graphe des composantes correspondent aux arêtes exclues de G. Il y a ici trois arêtes exclues. L'arête ah correspond à un arc entre la composante C_{af} et la composante C_{ch}. L'arête bj correspond à un arc entre la composante C_{bg} et la composante C_{deij}. L'arête ci correspond à un arc entre la composante C_{ch} et la composante C_{deij}. Le graphe des composantes est indiqué dans la Figure 1.5.

D'après la caractérisation, tout stable de cardinal maximal qui contient un sommet de B doit contenir les sommets de B des composantes successeurs. Ainsi, si un stable de cardinal maximal contient le sommet a, il doit aussi contenir les sommets de B des composantes successeurs donc il doit contenir les sommets c, d et e.

Dans la suite du document, on utilisera les conventions suivantes pour les figures des graphes bipartis. Les sommets de B seront des losanges, les sommets de R seront des cercles. Si deux sommets sont reliés par une arête, c'est qu'ils sont dans la même composante. Si deux sommets sont reliés par un arc, c'est qu'il existe un arc entre les deux composantes dans le graphe des composantes. De cette façon, on peut représenter en une figure un graphe et son graphe des composantes.

La Figure 1.6 applique cette représentation au graphe de l'exemple précédent et fait apparaître les composantes dans des cercles. Ainsi, le graphe de l'exemple sera représenté par le graphe de la Figure 1.7.

Le graphe des composantes est sans circuits, il peut donc être organisé en *niveaux* tel que le premier niveau contient les composantes sans prédécesseurs dans le graphe des composantes. On numérote les composantes par ordre croissant à partir du premier niveau. Par exemple, dans la figure 1.5, les composantes C_{AF} et C_{GB} sont de niveau 1, la composante C_{CH} est de niveau 2 et la composante C_{DEIJ} est de niveau 3.

Appliquons maintenant cette caractérisation dans les arbres.

Soit un arbre $G = (B, R, E)$.

Propriété 1.2.7. *Le graphe G_c est un arbre orienté.*

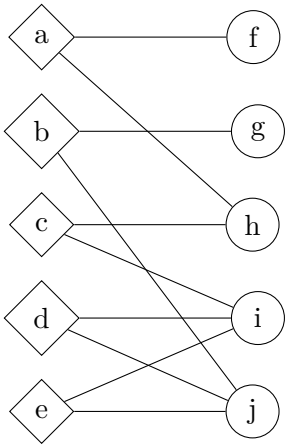


FIGURE 1.1 – Un graphe biparti

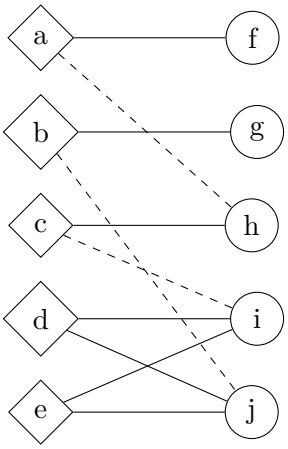


FIGURE 1.2 – Graphe biparti où les arêtes exclues sont en pointillés

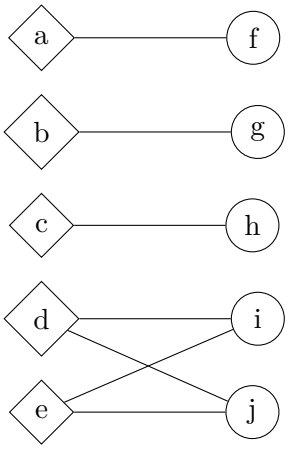


FIGURE 1.3 – Graphe biparti sans les arêtes exclues

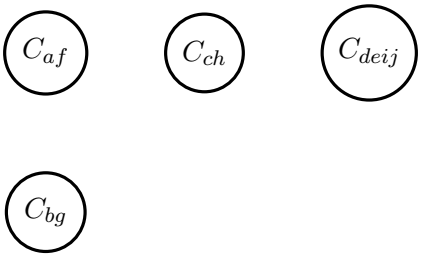


FIGURE 1.4 – Sommets du graphe des composantes

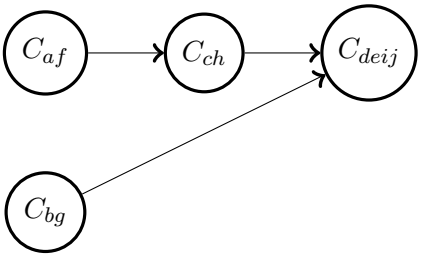


FIGURE 1.5 – Graphe des composantes

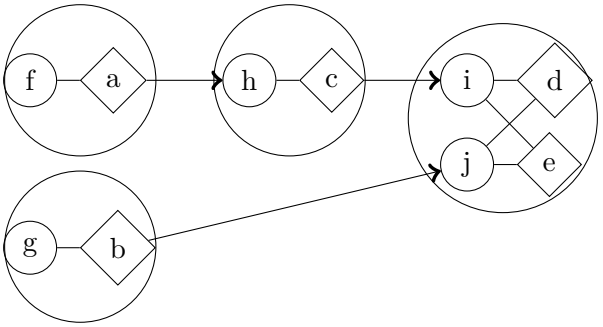


FIGURE 1.6 – Représentation du graphe avec les composantes visibles

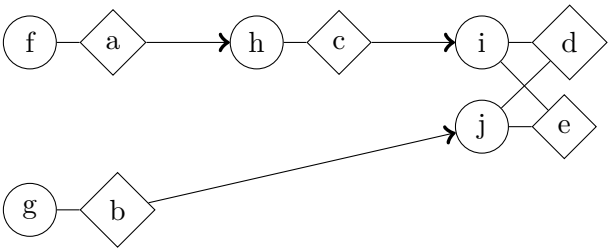


FIGURE 1.7 – Représentation du graphe et de son graphe des composantes

Propriété 1.2.8. *Soit $G = (B, R, E)$ un arbre dont tous les sommets sont libre. Les composantes C_i de G sont telles que $|V(C_i) \cap B| = 1$, et $|V(C_i) \cap B| = 1$. De plus, si on note u et v les sommets de C_i, alors $uv \in E$.*

Démonstration. G est un arbre dont tous les sommets sont libres. Il admet donc un unique couplage parfait. Chaque composante n'est donc constituée que de deux sommets voisins l'un de l'autre.

□

Propriété 1.2.9. *Soit un arbre $G = (B, R, E)$ tel que $\Phi(G) = B \cup R$. Soit $X = \{x \in V | \delta_G(x) > 2\}$. G_c est une arborescence (respectivement une anti-arborescence) si et seulement si $X \cap R = \varnothing$ (respectivement $X \cap B = \varnothing$).*

Démonstration. Il existe un arc entre deux composantes dans G_c si et seulement si il existe une arête dans G du sommet de B de la première composante au sommet de R de la deuxième. Donc le degré d'un sommet de R est le nombre d'arcs incidents intérieurement à la composante dans G_c auquel est additionné 1 pour l'arête entre les deux sommets de la composante.

Si G_c est une arborescence alors il existe exactement un arc incident intérieurement à chaque composante, sauf pour la composante racine. Donc les sommets de R sont de degré 1 ou 2 et $X \cap R = \varnothing$.

Supposons que G_c ne soit pas une arborescence mais que $X \cap R = \varnothing$. Comme G_c n'est

pas une arborescence, il existe au moins deux composantes sans prédécesseurs donc, comme G_c est connexe, il existe une composante avec au moins deux arcs incidents intérieurement. Donc il existe un sommet de R de degré au moins 3. Contradiction.

La preuve est identique pour le cas où G_c est une anti-arborescence.

□

1.3 d-extensibles de stables

Cette section donne les définitions utilisées dans le chapitre 3. Elles sont rappelées dans le chapitre.

Soit un graphe $G = (V, E)$. Soit $U \subset V$. Un stable $S \subset U$ de G est dit *extensible* par rapport à U si il existe $\widehat{S} \subset (V - U)$ tel que $S \cup \widehat{S}$ est un stable de cardinal maximal de G.

Soit un entier d tel que $1 \leqslant d < \alpha(G)$. Un ensemble $F \subset V$ est un d-extensible de G si tout stable S de $G[F]$ de cardinal d est extensible par rapport à F.

On s'intéresse au problème EXTENSIBLE suivant :

— Instance : Un graphe $G = (V, E)$, un entier positif $1 \leqslant d \leqslant \alpha(G)$, un entier positif $k \leqslant |V|$

— Question : Est-ce que G contient un d-extensible F de taille k ou plus ?

1.4 d-bloqueurs

Cette section donne les définitions du chapitre 4.

Soit Π un problème d'optimisation combinatoire défini sur un ensemble fini M=(m_1, m_2 ... m_n). Soit P une propriété sur cet ensemble telle que $X \subset M$ satisfait P si et seulement si X est une solution admissible de Π. Étant donné $L \subseteq M$, on définit C_L l'ensemble des sous-ensembles de L qui satisfont P. Ainsi, l'ensemble des solutions admissibles de M, c'est à dire l'ensemble des sous-ensembles de M qui satisfont P, est noté C_M. On suppose que P est héréditaire, c'est-à-dire que si $L \subseteq M$ satisfait P alors tout $K \subseteq L$ satisfait P.

Á chaque élément m de M sont associées deux valeurs positives : un coût $c(m)$ et un poids $w(m)$. Le poids (respectivement le coût) d'un sous-ensemble de M est la somme des

poids (respectivement des coûts) des éléments de ce sous-ensemble. On note $\nu_w(C_M)$ le poids maximum d'un élément de C_M et $\delta_w(C_M)$ le cardinal maximum d'un élément de C_M de poids maximum. Autrement dit, $\nu_w(C_M) = max\{w(C_j)|C_j \in C_M\}$ et $\delta_w(C_M) = max\{|C_j||C_j \in C_M, w(C_j) = \nu_w(C_M)\}$.

Soit un entier $1 \leqslant d \leqslant \nu_w(C_M)$. Un *d-bloqueur* de Π est un sous-ensemble $B \subset M$ tel que, si on retire les éléments de B de V, on obtient $\nu_w(C_{M-B}) \leqslant \nu_w(C_M) - d$. On s'intéresse au problème BLOCK_$\Pi(M, w, c, d)$:

— Instance : Un ensemble fini M=(m_1, m_2...m_n), une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M|$.

— Est-ce qu'il existe un d-bloqueur $B \subseteq M$ tel que $|B| \leqslant k$?

1.5 d-transversaux

Cette section donne les définitions du chapitre 5.

On reprend les notations de la section précédentes et note $\nu_w(C_M)$ le poids maximum d'un élément de C_M, c'est-à-dire que $\nu_w(C_M) = max\{w(C_j)|C_j \in C_M\}$. On note également $\delta_w(C_M)$ le cardinal minimum d'un élément de C_M de poids maximum, c'est-à-dire que $\delta_w(C_M) = min\{|C_j|\ \ |C_j \in C_M, w(C_j) = \nu_w(C_M)\}$.

Soit $d \geqslant 1$ un entier.

— Pour $d \leqslant \nu_w(C_M)$, on appelle *d-transversal pondéré* un sous-ensemble $T \subseteq M$ tel que pour tout $C \in C_M$ tel que $w(C) = \nu_w(C_M)$, on a $w(C \cap T) \geqslant d$.

— Pour $d \leqslant \delta_w(C_M)$, on appelle *d-transversal* un sous-ensemble $T \subseteq M$ tel que pour tout $C \in C_M$ tel que $w(C) = \nu_w(C_M)$, on a $|C \cap T| \geqslant d$.

On considère les problèmes suivants :

— $WTRANS_\Pi(w, c, d)$:

  — Instance : Un ensemble fini M=(m_1, m_2, ..., m_n), une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M|$.

  — Question : Est-ce qu'il existe un d-transversal pondéré $T \subseteq M$ du problème Π tel que $c(T) \leqslant k$?

— $TRANS_{\Pi}(w, c, d)$:

  — Instance : Un ensemble fini M=(m_1, m_2, ..., m_n), une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M|$.

  — Question : Est-ce qu'il existe un d-transversal $T \subseteq M$ du problème Π tel que $c(T) \leqslant k$?

Chapitre 2

La programmation mathématique biniveau en nombres entiers

Ce chapitre a pour but d'étudier une classe de programmes biniveaux en nombres entiers. Tout d'abord, la première section propose un bref état de l'art en donnant les définitions usuelles de la programmation biniveau et les approches de résolution de la littérature. Puis, après avoir constaté qu'il n'existe pas à notre connaissance de méthodes de résolution pour une classe de programmes biniveaux en nombres entiers, une deuxième section tente d'étendre les méthodes existantes et expose les difficultés de résolution.

Nous tentons ici de résoudre efficacement des programmes mathématiques biniveaux en nombres entiers pour, dans le chapitre 5 déterminer des d-transversaux en résolvant des programmes biniveaux.

2.1 Généralités sur la programmation mathématique biniveau

Cette section introduit la programmation mathématique biniveau, tout d'abord, en donnant les définitions générales puis en décrivant succinctement les différentes méthodes de résolution de la littérature.

Un programme biniveau modélise un problème d'optimisation dans lequel deux personnes prennent chacune des décisions de manière hiérarchisée. Le premier preneur de décision est appelé le *meneur* et le deuxième preneur de décision, qui réagit à la décision

du meneur, est appelé le *suiveur*. Le choix du suiveur peut avoir un impact sur le meneur, qui doit prendre en compte les choix possibles du suiveur.

Dans [23], la programmation biniveau est introduite par un problème de partage de tarte. Deux protagonistes, Alice et Bob doivent partager une tarte. Alice est le meneur, doit couper la tarte et sait que Bob choisira le plus gros des deux morceaux coupés. Alice souhaite aussi manger le plus gros morceau possible. Bob est le suiveur, il choisit le plus gros des deux morceaux et sa décision se répercute sur Alice, qui choisit le morceau restant. Ainsi, pour manger le plus gros morceau de tarte possible, Alice doit couper la tarte en deux parts égales. Ce problème peut être modélisé par le programme ci-après, présenté dans [19] :

$$\begin{cases} \max\limits_{x_1,x_2} & x_1 + x_2 - y & (2.1.1) \\ & x_1 + x_2 \leqslant P & (2.1.2) \\ & x \in \mathbb{R}^2 & (2.1.3) \\ & \begin{cases} \min\limits_{y} & y & (2.1.4) \\ & y \geqslant x_1 & (2.1.5) \\ & y \geqslant x_2 & (2.1.6) \\ & y \in \mathbb{R} & (2.1.7) \end{cases} \end{cases}$$

Les variables x_1 et x_2 sont celles gérées par le meneur. Elles représentent la taille des parts coupées par Alice. La contrainte (2.1.2) assure que la somme des tailles des parts ne dépasse pas la taille P de la tarte. La fonction économique (2.1.1) maximise la taille de la part qui n'est pas choisie par Bob. La variable y est gérée par le suiveur, donc par Bob. Elle modélise la taille de la part de tarte que Bob choisira. Le programme $((2.1.4)-(2.1.7))$ assure que Bob choisira la plus grande des parts coupées.

Dans [15], les programmes biniveaux sont introduits à l'aide d'un problème de tarification de péage. Étant donné un réseau, le meneur peut fixer les prix des péages d'un sous-ensemble d'arêtes. Il souhaite maximiser le gain qu'il tire des péages qu'il contrôle. Le suiveur représente les usagers du réseau qui eux, souhaitent voyager au moindre coût. Une fois que le meneur a fixé les tarifs des arêtes qu'il contrôle, les usagers vont chercher le chemin de moindre coût. Donc, si les prix des péages sont trop élevés, les usagers emprunteront au minimum les arêtes contrôlées par le meneur qui aura un gain plus réduit que s'il avait fixé des tarifs moins importants. Ce problème peut être modélisé par le programme suivant, dans lequel :

— A est l'ensemble des routes utilisables.

— $\overline{A} \subseteq A$ est l'ensemble des routes sur lesquelles le meneur peut placer un péage.

— Θ est l'ensemble des couples origine-destination (r, s) des usagers.

— $P_{r,s}$ est l'ensemble des chemins de r à s.

— c_a désigne le coût d'utilisation de la route a. Cette donnée contient par exemple le coût en essence.

— Les $\delta_{rs}^{a,p}$ sont des données qui valent 1 si le chemin $p \in P_{r,s}$ passe par la route a et 0 sinon.

— Les variables T_a représentent le tarif du péage sur la route a.

— Les variables x_a désignent le flux qui circule sur la route a.

— Les variables f_p^{rs} désignent le flux qui part de r, qui arrivent en s et qui passe par le chemin p.

Dans ce programme, on considère A l'ensemble des routes utilisables et $\overline{A}$ l'ensemble des routes sur lesquelles le meneur peut placer un péage. On note Θ l'ensemble des couples origine-destination (r, s). On note $P_{r,s}$ l'ensemble des chemins de r à s. Les variables T_a désignent le tarif du péage sur la route a. Les variables x_a désignent le flux qui circule sur la route a. Les variables f_p^{rs} désignent le flux qui part de r, qui arrivent en s et qui passe par le chemin p.

$$
\begin{cases}
\max\limits_{T} \quad \sum\limits_{a \in \overline{A}} T_a x_a & & (2.1.8) \\
\quad T_a \leqslant u_a & \forall a \in \overline{A} & (2.1.9) \\
\quad T_a \geqslant l_a & \forall a \in \overline{A} & (2.1.10) \\
\quad \begin{cases}
\min\limits_{x,f} \quad \sum\limits_{a \in \overline{A}} T_a x_a + \sum\limits_{a \in A} c_a x_a & & (2.1.11) \\
\qquad \sum\limits_{p \in P_{rs}} f_p^{rs} = d_{rs} & \forall (r,s) \in \Theta & (2.1.12) \\
\qquad x_a = \sum\limits_{(r,s) \in \Theta} \sum\limits_{p \in P_{rs}} \delta_{rs}^{a,p} f_p^{rs} & \forall a \in A & (2.1.13) \\
\qquad f_p^{rs} \geqslant 0 & \forall p \in P_{rs} & (2.1.14)
\end{cases}
\end{cases}
$$

La fonction économique (2.1.8) maximise le profit du meneur. Les contraintes (2.1.9) et (2.1.10) assurent que le tarif des péages est encadré par une borne inférieure et une borne supérieure. La fonction économique (2.1.11) minimise le coût du suiveur, qui est constitué

de la somme du coût de passer par chaque route a et le coût du péage éventuel des routes a. Les contraintes (2.1.12) assurent que pour chaque couple $(r,s) \in \Theta$, le flux est bien égal à la demande. Les contraintes (2.1.13) déterminent le flux sur chaque route.

2.1.1 Définitions générales

Le modèle général des programmes biniveaux est le suivant :

$$(PB)\begin{cases} \max\limits_{x} & F(x,y) & (2.1.1.1)\\ & H(x) \leqslant 0 & (2.1.1.2)\\ & G(x,y) \leqslant 0 & (2.1.1.3)\\ & \begin{cases} \max\limits_{y} & f(x,y) & (2.1.1.4)\\ & h(y) \leqslant 0 & (2.1.1.5)\\ & g(x,y) \leqslant 0 & (2.1.1.6) \end{cases} \end{cases}$$

Les variables x sont appelées *variables de premier niveau* et les variables y *variables de second niveau*. De même, la fonction F est appelée *fonction objectif de premier niveau* et la fonction f *fonction objectif de second niveau*.

On note $Sub(x)$ le problème suivant, qui correspond à la prise de décision du suiveur, une fois que le meneur a fixé ses variables.

$$(Sub(x))\begin{cases} \max\limits_{y} & f(x,y) & (2.1.1.4)\\ & h(y) \leqslant 0 & (2.1.1.5)\\ & g(x,y) \leqslant 0 & (2.1.1.6) \end{cases}$$

Le problème biniveau relâché associé à (PB) est :

$$(PBr)\begin{cases} \max\limits_{x} & F(x,y) & (2.1.1.1)\\ & H(x) \leqslant 0 & (2.1.1.2)\\ & G(x,y) \leqslant 0 & (2.1.1.3)\\ & h(y) \leqslant 0 & (2.1.1.5)\\ & g(x,y) \leqslant 0 & (2.1.1.6) \end{cases}$$

Dans (PBr), la fonction économique (2.1.1.4) a été retirée. La valeur optimale de (PBr) fournit ainsi une borne inférieure de la valeur optimale de (PB).

Dans le programme (PB), le nombre de variables de premier niveau est noté n^1 et le nombre de variables de deuxième niveau est noté n^2. Considérons les ensembles suivants :

— $\Omega = \{(x,y) \in \mathbb{R}^{n^1} \times \mathbb{R}^{n^2} : H(x) \leqslant 0, G(x,y) \leqslant 0, h(y) \leqslant 0, g(x,y) \leqslant 0\}$ est l'ensemble admissible relâché.

— Pour un vecteur $\hat{x}$ fixé, $\Omega(\hat{x}) = \{y \in \mathbb{R}^{n^2} : h(y) \leqslant 0,\ g(\hat{x},y) \leqslant 0\}$ est l'ensemble

admissible de second niveau

— Pour un vecteur $\hat{x}$ fixé, $R(\hat{x}) = \{y \in \mathbb{R}^{n^2} : y \in argmax\{f(\hat{x}, y) : y \in \Omega(\hat{x})\}\}$ est l'ensemble de réaction rationnelle. c'est-à-dire l'ensemble des y optimaux pour un $\hat{x}$ donné.

— $IR = \{(x, y) \in \mathbb{R}^{n^1} \times \mathbb{R}^{n^2} : H(x) \leqslant 0,\ G(x, y) \leqslant 0,\ y \in R(x)\}$ est l'ensemble des solutions admissibles du programme biniveau.

Comme indiqué dans [73], il est possible que $\Omega \neq \varnothing$ et que $IR = \varnothing$. Considérons l'exemple suivant :

$$\begin{cases} \min\limits_{x \geqslant 0} & x - 8y & (2.1.1.7) \\ & -5x + 2y \geqslant -33 & (2.1.1.8) \\ & x + 2y \geqslant 9 & (2.1.1.9) \\ & \begin{cases} \min\limits_{y \geqslant 0} & y & (2.1.1.10) \\ & -7x + 3y \leqslant 5 & (2.1.1.11) \\ & x + y \leqslant 15 & (2.1.1.12) \end{cases} \end{cases}$$

Il est évident que Ω est non vide. En effet, $(0, 0) \in \Omega$. Mais pour tout x, la solution optimale du sous-problème est $y = 0$. Ainsi, pour respecter la contrainte (2.1.9), il faut que $x \geqslant 9$ mais alors la contrainte (2.1.8) n'est plus respectée. Donc $IR = \varnothing$.

Il existe plusieurs variations du modèle général (PB) présenté plus haut. Nous en présentons ici quelques unes, qui sont reprises de [19].

Le modèle biniveau linaire (PBL) Un programme biniveau linéaire est un programme biniveau dont les contraintes et les fonctions économiques sont linéaires. Dans ce modèle x est le vecteur des variables de premier niveau et y le vecteur des variables de second niveau. $c^1 \in \mathbb{Q}^{n^1}$, $d^1 \in \mathbb{Q}^{n^2}$, $d^2 \in \mathbb{Q}^{n2}$. En notant m^1 le nombre de contraintes de premier niveau et m^2 le nombre de contraintes de deuxième niveau, $A_1 \in \mathbb{Q}^{m^1 \times n^1}$, $B_1 \in \mathbb{Q}^{m^1 \times n^2}$, $b^1 \in \mathbb{Q}^{m^1}$, $A_2 \in \mathbb{Q}^{m^2 \times n^1}$, $B_2 \in \mathbb{Q}^{m^2 \times n^2}$, $b^2 \in \mathbb{Q}^{m^2}$.

$$(PBL) \begin{cases} \max\limits_{x} & {c^1}^t x + {d^1}^t y & (2.1.1.1) \\ & A^1 x + B^1 y \leqslant b^1 & (2.1.1.2) \\ & x \in \mathbb{R}^{n^1} & (2.1.1.3) \\ & \begin{cases} \max\limits_{y} & {d^2}^t y & (2.1.1.4) \\ & A^2 x + B^2 y \leqslant b^2 & (2.1.1.5) \\ & y \in \mathbb{R}^{n^2} & (2.1.1.6) \end{cases} \end{cases}$$

La fonction économique de deuxième niveau (2.1.4) est parfois écrite $c^{2^t}x + d^{2^t}y$ mais lors de la résolution du problème de second niveau, x est un paramètre fixe dont constant. Il peut donc être retiré.

Le modèle biniveau simplifié (PBS) Ce modèle ne possède pas de contraintes dites de liaison [35], c'est-à-dire de contraintes de premier niveau avec des variables de deuxième niveau.

$$(PBS)\begin{cases} \max\limits_{x} & F(x,y) & (2.1.1.1)\\ & H(x) \leqslant 0 & (2.1.1.2)\\ & \begin{cases} \max\limits_{y} & f(x,y) & (2.1.1.4)\\ & h(y) \leqslant 0 & (2.1.1.5)\\ & g(x,y) \leqslant 0 & (2.1.1.6) \end{cases} \end{cases}$$

Autres modèles Il existe d'autres variantes de modèles, comme les programmes biniveau avec des variables mixtes au premier niveau (PBMP), au second niveau (PBMD) ou aux deux niveaux (PBM). Ils peuvent aussi avoir des variables entières au premier niveau (PBNEP), au deuxième niveau (PBNED) ou aux deux niveaux (PBNE). L'ajout de "L" après "PB" dans l'acronyme indiquera que le programme biniveau est linéaire et l'ajout du suffixe "S" indiquera que le programme biniveau est simplifié et ne contient donc pas de contraintes de liaison.

Complexité D'après [54], résoudre le problème biniveau linaire simplifié continu est NP-difficile. De plus, trouver une approximation de $(PBLS)$ est NP-difficile, c'est à dire que pour tout $\alpha > 0$, trouver une solution α-approchée de $(PBLS)$ est NP-difficile.

D'après [76] vérifier l'optimalité locale d'une solution d'un programme biniveau linéaire est NP-difficile.

2.1.2 Conditions d'optimalité

Dans le cas où l'ensemble des solutions rationnelles n'est pas réduit à un singleton, les solutions rationnelles y peuvent ne pas influencer de la même manière la fonction économique de premier niveau $F(x,y)$. Par exemple, dans le cas précédent de tarification des péages, un usager peut avoir le choix entre deux chemins de même coût dont un seul

passe par les arêtes contrôlées par le meneur. Ainsi, selon le trajet choisi, le meneur peut ne pas recevoir de gain. Il est donc nécessaire de pouvoir choisir une solution rationnelle dans le cas où plusieurs sont disponibles. On choisit généralement une des deux hypothèses suivantes ([63], [15]) :

— Hypothèse optimiste : on suppose que le suiveur fera le choix qui profitera le plus au meneur. Dans notre exemple, l'usager prendra le chemin qui empruntera le plus d'arêtes contrôlées par le meneur. Cela correspond à trouver une solution optimale de $\max_{x\in\Omega(y),y\in R(x)} F(x,y)$.

— Hypothèse pessimiste : on suppose que le suiveur fera le choix qui profitera le moins au meneur. Dans notre exemple, l'usager prendra le chemin qui empruntera le moins d'arêtes contrôlées par le meneur. Cela correspond à une solution optimale du problème $\max_{x\in\Omega(y)} \min_{y\in R(x)} F(x,y)$.

Si ces deux hypothèses sont les plus courantes, il existe un troisième choix dans lequel le meneur peut prédire pour chacun de ses choix un ensemble de choix possible du suiveur. Cette hypothèse est détaillée dans le premier chapitre de [37].

Dans certains cas, [72], plutôt que de rechercher une solution optimale, il peut être intéressant de rechercher un optimum local, toujours avec l'une des deux hypothèses. Un vecteur (x^0, y^0) est donc appelé solution optimiste locale si $\begin{cases} x^0 \in X \\ G(x^0,y^0) \leqslant 0 \\ y^0 \in R(x^0) \\ F(x^0,y^0) \leqslant F(x^0,y)\ \forall y \in R(x^0) \end{cases}$ et s'il existe un voisinage ouvert $V(x^0,\delta)$ de x^0 de rayon $\delta > 0$ tel que $\phi_o(x^0) \leqslant \phi_o(x)$ pour tout $x \in V(x^0,\delta) \cap X$ où $\phi_o(x) = \min_y\{F(x,y) : y \in R(x)\}$.

De même, un vecteur (x^0, y^0) est donc appelé solution pessimiste locale si $\begin{cases} x^0 \in X \\ G(x^0,y^0) \leqslant 0 \\ y^0 \in R(x^0) \\ F(x^0,y^0) \geqslant F(x^0,y) \forall y \in R(\end{cases}$ et s'il existe un voisinage ouvert $V(x^0,\delta)$ de x^0 tel que $\phi_p(x^0) \leqslant \phi_p(x)$ pour tout $x \in V(x^0,\delta) \cap X$ où $\phi_p(x) = \min_y\{F(x,y) : y \in R(x)\}$.

2.1.3 Exemples d'applications

Cette section présente quelques applications de la programmation biniveau. Elle est principalement inspirée de [29]. Pour plus de détails, le lecteur est invité à se référer à [35], [29] et [23].

Gestion des revenus

Le gestion des revenus est un problème rencontré par une entreprise lorsqu'elle souhaite maximiser son profit sur un secteur d'activité où elle peut agir, par exemple en investissant ou en fixant des tarifs. Le problème de tarification présenté plus haut en est un exemple. Des problèmes similaires ont été étudiés dans [58] et [22]. Une autre application dans le domaine de l'aviation a été étudiée dans [34].

Gestion d'un réseau routier

Dans ces problèmes, le deuxième niveau représente les usagers du réseau et le premier niveau le gestionnaire du réseau. Ces problèmes ont beaucoup d'applications comme le transport de matériaux dangereux [55], la gestion de la congestion dans les réseaux à l'aide de péages [50] ou de changements sur le réseau [65].

Problème Principal-Agent

Ce problème se présente lorsqu'un premier acteur, le principal, doit effectuer une action qui dépend d'un deuxième acteur, l'agent, sur lequel le principal n'est pas parfaitement informé. Ce problème est détaillé dans [35], chapitre 2.

2.1.4 Méthodes de résolution : cas continu

Cette section expose les principales méthodes de résolutions dans le cas des programmes biniveaux dont les variables sont continues. Elle reprend principalement le survey [29].

Méthodes de pénalité

Dans [74], [1], une méthode de pénalité est proposée pour résoudre les programmes biniveaux. Les contraintes de deuxième niveau sont introduites avec une pénalité dans la fonction objectif de deuxième niveau. Le modèle (PB) général devient ainsi :

$$(PB_{pen}) \begin{cases} \max\limits_{x} & F(x,y) \\ & H(x) \leqslant 0 \\ & G(x,y) \leqslant 0 \\ & \begin{cases} \max\limits_{y} & f(x,y) - r_1 p_1(h(x)) - r_2 p_2(g(x,y)) \end{cases} \end{cases}$$

où r_1 et r_2 sont des scalaires positifs et p_1 et p_2 les fonctions de pénalité. Un algorithme itératif est ensuite proposé pour résoudre le problème mais à chaque itération, le sous-problème n'est pas beaucoup plus simple à résoudre que le problème de départ.

Une variante de cette méthode est proposé dans [51] dans laquelle la fonction objectif de premier niveau est pénalisée et le deuxième niveau est remplacé par une condition d'optimalité du deuxième niveau pénalisé.

Dans [78], une autre approche est utilisée pour résoudre $(PBLS)$ dans laquelle le saut de dualité du deuxième niveau est ajouté dans le premier niveau. Mais les auteurs posent des hypothèses incompatibles entre elles. L'approche est modifiée et adaptée dans [25] où de nouvelles hypothèses plus applicables sont introduites.

Pivotage complémentaire

Cette méthode, introduite par [18], utilise les conditions KKT pour reformuler le programme biniveau en un programme à un niveau. L'approche cherche, à chaque itération, une solution au programme telle que la valeur de la fonction économique soit supérieure ou égale à un paramètre α qui est mis à jour à chaque itération. Mais, dans [16], il est montré que cette méthode ne retournait pas nécessairement un optimum global.

Énumération des points extrêmes

Cette méthode a été introduite par [26] pour résoudre PBL sans contraintes de premier niveau. Elle est basé sur le fait que si IR est non vide, alors il contient au moins un élément du polyèdre Ω. L'algorithme énumère des bases du sous-problème jusqu'à trouver

une solution optimale du programme biniveau.

Le méthode du k-ième meilleur de [18] parcourt les points extrêmes du problème biniveau relâché (PBr) par ordre décroissant des valeurs de la fonction objectif de premier niveau. L'algorithme s'arrête dès que la solution trouvée est rationnelle car c'est un optimum global.

Méthodes de descente

Dans les méthodes de descente, si la fonction objectif de premier niveau est dérivable, si pour tout x, $|R(x)| = 1$ et si les variables y peuvent être vues comme une fonction $y(x)$ de x alors étant donné une solution réalisable (x), on recherche une direction d qui donne une nouvelle solution $(x + \alpha d)$ réalisable et qui améliore la valeur de la fonction économique. Cette méthode a été appliqué dans [70] pour les cas des programmes biniveaux sans contraintes de premier niveau. Ce résultat est étendu par [76] aux programmes biniveaux dont les deux fonctions objectifs sont quadratiques. D'autres méthodes sont présentées dans [35].

Branch & Bound

Il est possible de remplacer le second niveau d'un programme biniveau par ses conditions d'optimalités pour obtenir un programme avec un seul niveau et où un seul décideur contrôle toutes les variables. Dans le cas où le second niveau peut être remplacé par les conditions KKT, on obtient le programme :

$$\begin{cases} \max\limits_{x,y,\lambda,\mu} & F(x,y) & \\ & H(x) \leqslant 0 & \\ & G(x,y) \leqslant 0 & \\ & \lambda_i \geqslant 0 & \forall i = 1..m_2 \\ & \mu_i \geqslant 0 & \forall i = 1..p_2 \\ & \lambda_i g_i(x,y) = 0 & \forall i = 1..m_2 \\ & \mu_i g_i(y) = 0 & \forall i = 1..p_2 \\ & \nabla_y f(x,y) + \sum\limits_{i=1}^{m_2} \nabla_y \lambda_i g_i(x,y) + \sum\limits_{i=1}^{p_2} \nabla_y \mu_i h_i(y) = 0 & \end{cases}$$

Ce programme reformulé est utilisé pour résoudre des problèmes biniveaux par un Branch & Bound dans [43], [9] et [3].

Dans [43], ce programme peut être transformé en un programme linéaire en variables mixtes et résolu comme dans [5] par un Branch& Cut .

Une autre approche, dans [49] et [53] pour le cas quadratique convexe, propose un Branch & Bound qui n'utilise pas cette reformulation mais est basé sur le problème biniveau relâché.

Méthodes de régions de confiance

Les méthodes de régions de confiance sont apparues pour la première fois dans [61]. Le principe de ces méthodes est rappelé dans les paragraphes suivants. Pour plus de détails, le lecteur est invité à se référer à [30]. Ces méthodes sont des algorithmes itératifs qui, étant donné un point x_k obtenu à l'itération k, considèrent un modèle qui est une approximation du problème pour le résoudre dans un voisinage de x_k. Plus précisément, étant donné un problème

$$\begin{cases} \min\limits_{x} & f(x) \end{cases}$$

Soit x_k la solution obtenue à l'itération k. L'algorithme considère le modèle m_k qui fournit une approximation de la fonction objectif sur une sphère de centre x_k et de rayon Δ_k. Cette sphère constitue la région de confiance. L'algorithme résout ensuite le programme suivant pour obtenir la solution s_k :

$$\begin{cases} \min\limits_{s} & m_k(x_k + s) \\ & s.t. \quad |s| \leqslant \Delta_k \end{cases}$$

Puis l'algorithme évalue la qualité du modèle en déterminant le ratio $\rho_k = \frac{f(x_k)-f(x_k+s_k)}{m_k(x_k)-m_k(x_k+s_k)}$. Selon la valeur de ρ_k, le rayon de la sphère qui sert de région de confiance peut varier et un nouveau centre de la sphère peut être choisi. Les algorithmes de région de confiance fournissent une solution approchée de la solution optimale.

Un algorithme de région de confiance a également été utilisé dans [62] dans le cas des problèmes biniveaux sans contraintes au premier niveau, dans [28] pour résoudre des programmes biniveau sans contraintes de premier niveau qui font intervenir des variables de deuxième niveau et dans [36] pour des programmes biniveaux bilinéaires.

2.1.5 Méthodes de résolution : La programmation biniveau en nombre entiers

Cette section détaille les méthodes de résolutions des programmes biniveaux qui contiennent des variables discrètes.

Branch & Bound

Des approches de Branche & Bound ont été proposées dans [66] pour résoudre des programmes biniveaux linéaires simplifiés en variables mixtes et dans [10] pour résoudre des programmes biniveaux linéaires simplifiés en nombres entiers. Nous présentons ici ce deuxième algorithme.

On suppose dans cette approche que Ω n'est pas vide et que pour toute décision x du meneur, $|R(x)| = 1$. On considère que toutes les variables sont binaires.

Cet algorithme parcourt les solutions du meneur et détermine, pour chacune des solutions x considérées, la réponse su suiveur.

On considère le programme $(\hat{P})$ dans lequel :

— Les variables x, y et les données A^2, B^2, b^2, c^1, d^1 proviennent du modèle de programme biniveau simplifié en nombre entiers.

— α est une borne inférieure de la fonction économique de premier niveau du programme biniveau. Elle est initialisée à $-\infty$.

— β est une borne inférieure du nombre de variables x non nulles. Elle est initialisée à 0

$$(\hat{P}) \begin{cases} \max\limits_{x,y} & {d^2}^t y & (2.1.5.1) \\ & A^2 x + B^2 y \leqslant b^2 & (2.1.5.2) \\ & {c^1}^t x + {d^1}^t y \geqslant \alpha & (2.1.5.3) \\ & \sum\limits_{i=1}^{n^1} x_i \geqslant \beta & (2.1.5.4) \\ & x \in X & (2.1.5.5) \\ & y \in Y & (2.1.5.6) \end{cases}$$

Ce programme est résolu en chaque nœud du Branch & Bound. Il sert à déterminer s'il peut exister dans le nœud courant ou les nœuds fils une solution pour laquelle la valeur de

la fonction économique de premier niveau est plus élevée que celle de la meilleure solution trouvée jusque là. Si le programme n'a pas de solution admissible, c'est qu'il n'existe pas de meilleure solution que la meilleure solution courante dans le nœud ou les nœuds fils. Toutefois, l'existence d'une solution admissible à ce programme n'implique pas qu'il existe une meilleure solution au programme biniveau que la meilleure solution courante.

On souhaite ici résoudre un programme biniveau linéaire simplifié donc les contraintes de premier niveau ne porte que sur les variables de premier niveau. On considère ici que les contraintes $x \in X$ contiennent les contraintes d'intégrité sur les variables x et les contraintes de premier niveau qui ne portent que sur x. On considère aussi que les contraintes $y \in Y$ contiennent les contraintes d'intégrité de y. Dans ce programme, on retrouve les contraintes de deuxième niveau et deux autres contraintes. La contrainte (2.1.5.3) assure que la valeur de la fonction économique de premier niveau de la solution est plus élevée que celle de la meilleure solution courante. La contrainte (2.1.5.4) fixe une borne inférieure au nombre de variables de premier niveau non nulles. Elle n'est pas indispensable mais d'après [10], elle améliore la vitesse d'exécution de l'algorithme et permet d'effectuer la phase de branchement plus facilement.

Soit k le numéro de l'itération courante. On définit les ensembles suivants :

— W_k ensemble des indices des variables fixées lors de l'itération k

— $S_k^1 = \{i : i \in W_k \text{ et } x_i = 1\}$ ensembles des indices des variables x_i fixées à 1 lors de l'itération k

— $S_k^0 = \{i : i \in W_k \text{ et } x_i = 0\}$ ensembles des indices des variables x_i fixées à 0 lors de l'itération k

— $S_k^* = \{i : i \in W_k\}$ ensemble des indices des variables qui ne sont pas fixées lors de l'itération k

L'algorithme de résolution est le suivant :

1. **Initialisation** : $k = 0$, $W_k = \varnothing$, $S_k^1 = \varnothing$, $S_k^0 = \varnothing$, $S_k^* = \{1..n^1\}$, $\alpha = -\infty$, $\beta = 0$, $\underline{F} = -\infty$
2. **Itération générale**
   — Fixer $x_i := 1$ pour tout $i \in S_k^1$ et $x_i := 0$ pour tout $i \in S_k^0$.

— Trouver une solution admissible à $(\hat{P})$. Si elle n'existe pas, aller à l'étape 5. Sinon soit (x_k, y_k) cette solution.

— Résoudre le programme $Sub(x_k)$ pour obtenir $\hat{y}_k \in R(x_k)$.

— $\underline{F} = max(\underline{F}, F(x_k, \hat{y}_k))$

— Soit $J = \{i \in S^*_{k-1} : x^k_i = 1\}$.

— Si $J = \varnothing$ alors $S^1_k = S^1_{k-1}$, $S^0_k = S^0_{k-1}$, $S^*_k = S^*_{k-1}$ et $P_k = P_{k-1}$ aller à l'étape 4

— Sinon, aller à l'étape 3

3. **Branchement** Créer $|J|$ nouveaux nœuds en ajoutant les indices de J à S^1_k.
4. $\alpha := \underline{F} + 1$ et $\beta := 1 + |S^1_k|$, revenir à l'étape 1.
5. **Retour arrière** S'il ne reste plus de nœuds à évaluer, aller à l'étape suivante. Sinon, remonter d'un niveau, trouver la variable correspondante x_j et faire un branchement dans la branche complémentaire $x_j = 0$. Mettre à jour S^1_k, S^0_k et S^*_k. $\beta = 0$. Revenir au début de l'étape 2.
6. **Fin** Si $\underline{F} = -\infty$ alors il n'existe pas de solution admissible au problème. Sinon, retourner la solution associée à $\underline{F}$

À l'initialisation de l'algorithme, aucune variable de premier niveau n'est fixée. À chaque itération, on cherche une solution admissible au programme $\hat{P}$. S'il n'en existe pas, c'est qu'il n'est pas possible de trouver une meilleure solution que la solution courante donc on arrête l'exploration de cette branche et on remonte. Sinon on note (x_k, y_k) cette solution.

On cherche ensuite à déterminer une réponse rationnelle du suiveur si le meneur fixe ses variables à x_k. Pour cela, on résout le problème $Sub(x_k)$ pour obtenir une réponse rationnelle du suiveur que l'on note $\hat{y}_k \in R(x_k)$. On compare la valeur de la fonction économique de premier niveau fournie par $(x_k, \hat{y}_k)$. Si elle est meilleure que celle de la meilleure solution courante, alors $(x_k, \hat{y}_k)$ devient la meilleure solution courante. Si des variables de S^*_k ne sont pas nulles, l'algorithme fixe définitivement dans cette branche ces variables à 1 et se place dans le nœud correspondant puis passe à l'itération suivante. Sinon, l'algorithme incrémente β et recommence une itération dans le nœud courant, la contrainte

(2.1.5.4) assurant d'atteindre un nouveau nœud à la prochaine itération. L'algorithme itère jusqu'à ce que $\hat{P}$ n'ait plus de solution admissible dans aucun des nœuds. La meilleure solution trouvée est alors la solution optimale du problème.

Pour adapter cet algorithme à l'hypothèse pessimiste dans le cas où, pour x fixé, $|R(x)| \neq 1$, comme indiqué dans [10], il suffit de d'ajouter dans la fonction économique de $\hat{P}$ le terme $-\epsilon F(x, y)$ lorsque que $\hat{P}$ est résolu sans les contraintes (2.1.5.3) et (2.1.5.4) pour trouver une réponse rationnelle du suiveur. En choisissant un $\epsilon > 0$ suffisamment petit, on détermine une solution rationnelle du suiveur qui minimise la fonction objectif du meneur.

Plans Coupants

Dans [35] est proposée une approche pour résoudre les programmes biniveaux simplifiés dont les variables de premier niveau sont continues et n'apparaissent pas dans les contraintes de second niveau, et les variables de second niveau sont discrètes. L'idée de l'algorithme est d'utiliser l'enveloppe convexe des solutions entières du deuxième niveau pour se ramener à un programme biniveau en variables continues en appliquant un algorithme de plans coupant sur le deuxième niveau. L'algorithme 1 détaille l'approche.

Algorithm 1 Algorithme permettant de résoudre un programme biniveau simplifié dont les variables de premier niveau sont continues et les variables de deuxième niveau discrète

1: Soit (PB_{rc}) la relaxation du problème biniveau que l'on souhaite résoudre en relâchant les contraintes d'intégrité du deuxième niveau
2: Résoudre (PB_{rc})
3: Soit y_{rc} les variables de second niveau de la solution de (PB_{rc})
4: **while** y_{rc} ne sont pas entières **do**
5: Générer une coupe de Gomory pour y_{rc}, l'ajouter à (PB_{rc})
6: Résoudre (PB_{rc})
7: **end while**
8: **return** Retourner la solution de (PB_{rc})

A chaque itération de cette méthode, il faut résoudre un programme biniveau (PB_{rc}) dont les variables sont continues par l'une des méthodes de la section précédente, ce qui peut être long à calculer.

Branch & Cut

Dans [39], une approche de Branch & Cut est présentée pour résoudre les problèmes biniveaux simplifiés linéaires en nombres entiers (PBLSNE). Elle s'inspire du Branch & Bound de [9].

Dans cette approche on note $\Omega^I = \{(x,y) \in \mathbb{Z}^{n^1} \times \mathbb{Z}^{n^2} : H(x) \leqslant 0, G(x,y) \leqslant 0, h(y) \leqslant 0, g(x,y) \leqslant 0\}$ l'ensemble admissible discret et $\Omega = \{(x,y) \in \mathbb{R}^{n^1} \times \mathbb{R}^{n^2} : H(x) \leqslant 0, G(x,y) \leqslant 0, h(y) \leqslant 0, g(x,y) \leqslant 0\}$ l'ensemble dans lequel on relâche les contraintes d'intégrité.

L'approche utilise sur la propriété suivante qui permet de générer des inégalités valides pour les programmes biniveaux.

Propriété 2.1.1. *([39]) Soit $(\hat{x},\hat{y}) \in \Omega^I$ une solution réalisable de (PBr). Soit K l'ensemble des contraintes saturées par $(\hat{x},\hat{y})$. La contrainte $x\sum_{i\in K} A_i + y\sum_{i\in K} B_i \leqslant \sum_{i\in K} b_i - 1$ est valide pour IR.*

Soit t un entier et Ω_t un ensemble de contraintes. On note (PBr^t) le programme $\max_{(x,y)\in\Omega_t} c^{1^t}x + d^{1^t}y$ qui maximise la fonction économique de premier niveau sous l'ensemble de contraintes Ω_t.

L'algorithme présenté dans [39] est le suivant. On note L la meilleure valeur optimale trouvée et (x_m, y_m) la meilleure solution optimale trouvée.

1. Initialisation : $t = 1$, $\Omega_t = \Omega$, $L = -\infty$.
2. Évaluation d'un nœud

   Résoudre (PBr^t), soit $(\hat{x},\hat{y})$ une de ses solutions optimales si elles existent et soit $\hat{z}$ la valeur de la fonction économique

   — Si (PBr^t) n'a pas de solution admissible ou si $\hat{z} < L$: élagage du nœud.

   — Génération d'inégalités valides :

   — Si $(\hat{x},\hat{y}) \in \Omega^I$, résoudre $Sub(\hat{x})$. Soit $\hat{z}_s$ la valeur d'une solution optimale de $Sub(\hat{x})$.

   — Si $\hat{z}_s = d^2y$ alors $L := f(\hat{x},\hat{y})$, $x_m := \hat{x}$, $y_m := \hat{y}$. Puis, élagage du nœud.

— Sinon soit K l'ensemble des contraintes de (LP_t) saturées par $(\hat{x}, \hat{y})$. $\Omega_{t+1} := \Omega_t \cup \{(x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} | \sum_{i \in K} a_i x_i + b_i y_i \leqslant \sum_{i \in K} b_i - 1\}$. Revenir à l'étape 1.

— Sinon soit générer une inégalité valide pour $\Omega_t \cup (\mathbb{N}^{n_1} \times \mathbb{N}^{n_2})$ ou aller à l'étape 3

3. Branchement

4. Retour arrière : s'il ne reste plus de nœuds, aller à l'étape suivante. Sinon, remonter d'un niveau.

5. Fin : Si $L = -\infty$ alors il n'existe pas de solution admissible au problème. Sinon, retourner (x_m, y_m)

Cet algorithme utilise la relaxation du problème biniveau et la résout en chaque nœud. Si elle n'a pas de solution ou si la valeur de la solution est inférieure à la meilleure valeur trouvée, alors le nœud est élagué. Sinon, si la solution de la relaxation est entière, l'algorithme cherche une réponse rationnelle. Si cette réponse rationnelle correspond à la solution de la relaxation alors elle devient la meilleure solution courante et le nœud est élagué. Sinon, l'algorithme ajoute une coupe à la relaxation et tente de la résoudre à nouveau. Dans le cas où la solution de la relaxation n'est pas entière, soit l'algorithme ajoute une inégalité valide pour $\Omega_t \cup (\mathbb{N}^{n_1} \times \mathbb{N}^{n_2})$, soit il effectue un branchement.

Reformulations

Nous avons vu plus haut qu'il était possible de résoudre un problème biniveau continu en le reformulant en un programme à un seul niveau à l'aide des conditions d'optimalité du deuxième niveau. Cette section présente une reformulation des problèmes biniveaux discrets pour retirer les contraintes d'intégrité.

Des reformulations de $(PBLNEP)$, $(PBLNED)$ et de $(PBLNE)$ sont proposées dans [75]. Nous détaillons ici celle de $(PBLNE)$.

Le modèle de $(PBLNE)$ présenté dans [75] est

$$\begin{cases} \max\limits_{x} & {c^1}^t x + {d^1}^t y \\ & A^1 x + B^1 y \leqslant b^1 \\ & x \in \{0,1\}^{n^1} \\ & \begin{cases} \max\limits_{y} & {d^2}^t y \\ & A^2 x + B^2 y \leqslant b^2 \\ & y \in \{0,1\}^{n^2} \end{cases} \end{cases}$$

Dans [75] il est montré qu'il existe deux réels M_x et M_y tels que le programme suivant a la même valeur optimale que le programme précédent. Dans le programme suivant, les vecteurs e_x et e_y sont des vecteurs de 1 de dimension n^1 et n^2, respectivement.

$$\begin{cases} \max\limits_{x} & {c^1}^t x + {d^1}^t y + M_x e_x^t u \\ & A^1 x + B^1 y \leqslant b^1 \\ & 0 \leqslant x \leqslant e_x \\ & \begin{cases} \max\limits_{y,u} & {d^2}^t y - {e_x}^t u + M_y {e_y}^t v \\ & A^2 x + B^2 y \leqslant b^2 \\ & u \leqslant x \\ & u \leqslant e_x - x \\ & 0 \leqslant y \leqslant e_y \\ & \begin{cases} \min\limits_{v} & -{e_y}^t v \\ & v \leqslant y \\ & v \leqslant e_y - y \end{cases} \end{cases} \end{cases}$$

Malheureusement, il n'est pas plus facile de résoudre ce programme que le précédent. En effet, les valeurs de M_x et M_y ne sont pas connues et, même si elle l'étaient, il n'existe, à notre connaissance, aucun algorithme efficace pour résoudre les programmes mathématiques à trois niveaux.

Une autre reformulation a été proposée dans [21] pour reformuler un problème de sac à dos biniveau en un programme à un seul niveau. Mais cette reformulation est propre au problème et ne peut être généralisée.

Cas particulier : nombre borné de variables de second niveau

Dans [57] est proposé un algorithme polynomial dans le cas où le programme biniveau est linéaire, en variables entières et où le nombre de variables de second niveau est borné. Cet algorithme détermine l'existence d'une solution optimale, à l'aide d'un résultat de l'algorithme de [60], et la trouve, si elle existe. Cet algorithme procède par une recherche de bornes supérieures en résolvant le problème de décision paramétré par une valeur α :

"Existe-t-il une solution admissible au programme biniveau telle que la valeur de la fonction économique soit supérieure ou égale à α". Ce problème de décision peut être résolu en temps polynomial si le nombre de variables du deuxième niveau est borné. Par une recherche dichotomique, il est ainsi facile de déterminer la valeur optimale du problème. Une fois la valeur optimale connue, une approche pour déterminer une solution optimale est proposée.

Cette approche repose sur un nombre borné de variables de second niveau. Mais dans le cas général, cette condition n'est pas remplie.

2.2 Considérations sur la programmation mathématique biniveau en nombres entiers

Comme nous pouvons le constater, les approches de résolution des programmes biniveaux en nombres entiers ne considèrent pas le cas des programmes biniveaux dont les contraintes de premier niveau font intervenir des variables de second niveau. Dans cette section, nous essayons de résoudre ces programmes. Le but est de pouvoir déterminer des d-transversaux en résolvant un programme mathématique biniveau. Cette application sera détaillée dans le chapitre 5.

La première sous-section expose les limites des conditions d'optimalité utilisées habituellement et en propose de nouvelles. La deuxième section étend l'algorithme proposé dans [10] aux programmes biniveaux en nombre entiers qui contiennent des contraintes de liaison et détaille les limites de cette approche.

2.2.1 Limites des conditions d'optimalité

Les hypothèses optimiste et pessimiste permettent de définir une solution optimale dans le cas d'un programme biniveau simplifié, mais elles ne sont pas toujours suffisantes lorsque il existe une contrainte de premier niveau avec des variables de second niveau. En effet, le suiveur peut avoir le choix entre plusieurs solutions optimistes ou pessimistes qui donnent une même valeur pour la fonction économique de premier niveau. Mais certaines de ces solutions peuvent ne pas respecter les contraintes de liaison. Considérons par exemple le programme suivant :

$$
(P_{ex})\begin{cases}
\max\limits_{x} & 3x_1 + 2x_2 + x_3 + 2y_1 + 3y_2 + y_3 + 2y_4 + 3y_5 & (2.2.1.1)\\
& x_1 + x_2 + x_3 \leqslant 2 & (2.2.1.2)\\
& x_1 + y_5 \leqslant 1 & (2.2.1.3)\\
& x_1 + y_4 \leqslant 1 & (2.2.1.4)\\
& x \in \{0,1\}^3 & (2.2.1.5)\\
& \begin{cases}
\max\limits_{y} & y_1 + y_2 + 3y_3 + y_4 + y_5 & (2.2.1.6)\\
& y_2 + y_5 \leqslant 1 & (2.2.1.7)\\
& y_1 + y_4 \leqslant 1 & (2.2.1.8)\\
& y_4 + y_5 + x_3 \leqslant 1 & (2.2.1.9)\\
& y_1 + y_2 + y_3 + y_4 + y_5 \leqslant 2 & (2.2.1.10)\\
& y \in \{0,1\}^5 & (2.2.1.11)
\end{cases}
\end{cases}
$$

Dans ce programme, supposons que le meneur choisisse la solution $(1, 1, 0)$. Dans ce cas, la valeur d'une solution optimale du suiveur est 4 en fixant à 1 la variable y_3 et une des autres variables. Dans l'hypothèse optimiste, le suiveur cherche à favoriser le meneur. Il a ici deux solutions optimistes possibles, $(0, 1, 1, 0, 0)$ et $(0, 0, 1, 0, 1)$, pour lesquelles la valeur optimale du premier niveau est 9. Or, si la première solution est bien admissible, la deuxième ne l'est pas car la contrainte (2.2.1.3) n'est pas respectée. Pourtant, les deux solutions respectent l'hypothèse optimiste et le suiveur cherche une solution qui favorise le meneur. De même dans l'hypothèse pessimiste le suiveur cherche à minimiser le gain du meneur et a deux solutions possibles : $(1, 0, 1, 0, 0)$ et $(0, 0, 1, 1, 0)$ qui donnent une valeur optimale de 8 pour la fonction objectif de premier niveau. À nouveau, la première solution est réalisable mais la deuxième ne l'est pas car elle ne respecte pas la contrainte (2.2.1.4).

Si le meneur souhaite que le choix du suiveur respecte toujours les contraintes de liaison, il doit choisir la solution $(1, 0, 1)$. Dans ce cas, la valeur d'une solution optimale du suiveur est toujours 4 mais il n'existe plus qu'une solution optimiste, $(0, 1, 1, 0, 0)$, et la valeur de la fonction économique du meneur est 8. Dans le cas pessimiste, il n'y a plus qu'une solution, $(1, 0, 1, 0, 0)$ qui fournit une valeur optimale de 7 pour le meneur. Donc en choisissant la solution $(1, 0, 1)$ plutôt que la solution $(1, 1, 0)$ le meneur s'assure que le choix du suiveur respecte les contraintes de liaison mais la fonction économique de premier niveau a une valeur inférieure à celle obtenue au paragraphe précédent.

Ainsi, même en choisissant une hypothèse optimiste ou pessimiste, pour une solution x du meneur, il peut exister dans les réponses rationnelles du suiveur $R(x)$ des solutions qui respectent les hypothèses pessimiste et optimiste mais qui ne sont pas réalisables car

elles ne respectent pas les contraintes de liaison. Pour choisir une solution optimale, nous considérerons donc qu'en plus de pouvoir choisir une solution qui favorise ou non la solution optimale du meneur, le suiveur peut choisir une solution qui ne respecte pas les contraintes de liaison . Nous proposons donc de nouvelles hypothèses d'optimalité.

L'hypothèse optimiste.

Dans cette hypothèse optimiste, le suiveur choisira si possible une solution qui sera réalisable pour le meneur et qui le favorisera. Ainsi, si possible, il choisira une solution qui respecte les contraintes $G(x,y) \leqslant 0$ et qui maximisera $F(x,y)$. Pour une solution x donnée, si on note γ la valeur optimale de la solution du problème $Sub(x)$, la solution optimiste est donnée par le programme suivant :

$$(SubOpt(x)) \begin{cases} \max\limits_{y} & F(x,y) \\ & h(y) \leqslant 0 \\ & g(x,y) \leqslant 0 \\ & f(x,y) = \gamma \\ & G(x,y) \leqslant 0 \\ & y \in \{0,1\}^{n^2} \end{cases}$$

Dans ce programme, la fonction économique maximise la fonction économique du meneur. On retrouve les contraintes de deuxième niveau. On trouve également une contrainte qui impose que la solution soit bien une solution optimale de $Sub(x)$ et les contraintes de liaison. S'il existe une solution admissible à ce programme, c'est qu'il existe une solution optimale optimiste pour la valeur de x fixée. Sinon, c'est qu'il n'existe pas de solution optimiste admissible pour la valeur de x fixée.

L'hypothèse pessimiste.

Dans cette hypothèse pessimiste, le suiveur choisira une solution qui défavorisera le plus le meneur. Ainsi, il choisira si possible une solution qui ne respecte pas une des contraintes de liaison $G(x,y) \leqslant 0$ et si ce n'est pas possible une solution qui minimisera $F(x,y)$.

Notons $G_i(x,y) \leqslant 0$ pour $i \in \{1..m\}$ chaque contrainte de $G(x,y) \leqslant 0$. Pour une solution x donnée, si on note γ la valeur optimale de la solution du problème $Sub(x)$, le suiveur pessimiste cherchera une solution qui ne respecte pas les contraintes $G(x,y) \leqslant 0$

cherchant une solution admissible au programme suivant pour tout $i \in \{1..m\}$:

$$(SubPesC(x,i)) \begin{cases} \min\limits_{y} & F(x,y) \\ & h(y) \leqslant 0 \\ & g(x,y) \leqslant 0 \\ & f(x,y) = \gamma \\ & G_i(x,y) > 0 \\ & y \in \{0,1\}^{n2} \end{cases}$$

On retrouve dans ce programme les contraintes de second niveau. Une autre contrainte assure que la solution trouvée est bien une solution optimale de $Sub(x)$. La dernière contrainte impose que la contrainte de liaison i n'est pas respectée. S'il existe une solution à ce programme, c'est qu'il existe une solution optimale de $Sub(x)$ qui ne respecte pas une des contraintes de liaison, donc qu'il existe une solution pessimiste pour x fixé. Il n'est donc pas nécessaire de trouver une solution optimale à ce programme, une solution admissible est suffisante.

Si le suiveur ne trouve pas de solution qui ne respecte pas les contraintes de liaison alors il cherchera une solution qui favorisera le moins le meneur, donc qui diminuera autant que possible la valeur de la fonction économique de premier niveau. Il résoudra donc le programme suivant :

$$(SubPes(x)) \begin{cases} \min\limits_{y} & F(x,y) \\ & h(y) \leqslant 0 \\ & g(x,y) \leqslant 0 \\ & f(x,y) = \gamma \\ & y \in \{0,1\}^{n2} \end{cases}$$

La fonction économique de ce programme est celle de premier niveau. On cherche ici à la minimiser. Les contraintes sont celles de deuxième niveau et une contrainte qui assure que la solution est bien une solution optimale de $Sub(x)$. Il est inutile d'ajouter les contraintes de liaison car on ne résout ce programme que si on n'a pas trouvé de solution de $Sub(x)$ qui ne respecte pas toutes les contraintes de liaison.

L'hypothèse trahison.

Dans l'hypothèse trahison, le suiveur cherche si possible une solution qui ne respecte pas les contraintes de liaison et si ce n'est pas possible, une solution qui favorise le plus le

meneur donc qui maximise $F(x, y)$.

Notons $G_i(x, y) \leqslant 0$ pour $i \in \{1..m\}$ chaque contrainte de $G(x, y) \leqslant 0$. Pour une solution x donnée, si on note γ la valeur optimale de la solution du problème $Sub(x)$, le suiveur trahison cherchera une solution qui ne respecte pas les contraintes $G(x, y) \leqslant 0$ en cherchant, comme pour l'hypothèse pessimiste, une solution admissible au programme suivant pour tout $i \in \{1..m\}$:

$$(SubTraC(x, i)) \begin{cases} \min\limits_{y} & F(x, y) \\ & h(y) \leqslant 0 \\ & g(x, y) \leqslant 0 \\ & f(x, y) = \gamma \\ & G_i(x, y) > 0 \\ & y \in \{0, 1\}^{n2} \end{cases}$$

Comme dans l'hypothèse pessimiste, on retrouve dans ce programme les contraintes de second niveau. Une autre contrainte assure que la solution trouvée est bien une solution optimale de $Sub(x)$. La dernière contrainte impose que la contrainte de liaison i n'est pas respectée. S'il existe une solution à ce programme, c'est qu'il existe une solution optimale de $Sub(x)$ qui ne respecte pas une des contraintes de liaison. Il n'est donc pas nécessaire de trouver une solution optimale à ce programme, une solution admissible est suffisante.

Si le suiveur ne trouve pas de solution qui ne respecte pas les contraintes $G(x, y) \leqslant 0$, il cherchera une réponse rationnelle qui favorisera le meneur, comme dans l'hypothèse optimiste. Pour cela, il résoudra le programme suivant :

$$(SubTra(x)) \begin{cases} \max\limits_{y} & F(x, y) \\ & h(y) \leqslant 0 \\ & g(x, y) \leqslant 0 \\ & f(x, y) = \gamma \\ & y \in \{0, 1\}^{n2} \end{cases}$$

Dans ce programme, la fonction économique maximise la fonction économique du meneur. On retrouve les contraintes de deuxième niveau. Une contrainte impose que la solution est bien une solution optimale de $Sub(x)$. Puisque toutes les solutions optimales de $Sub(x)$ respectent les contraintes de liaison, il est inutile de les inclure.

L'hypothèse concurrence

Dans l'hypothèse de concurrence, le suiveur cherche une solution qui respecte les contraintes du premier niveau mais qui défavorise le plus le meneur donc qui minimise $F(x, y)$. Ainsi, pour une solution x donnée, si on note γ la valeur optimale de la solution du problème $Sub(x)$, la solution de concurrence est donnée par le programme suivant :

$$(SubConc(x)) \begin{cases} \min\limits_{y} & F(x,y) \\ & h(y) \leqslant 0 \\ & g(x,y) \leqslant 0 \\ & f(x,y) = \gamma \\ & G(x,y) \leqslant 0 \\ & y \in \{0,1\}^{n2} \end{cases}$$

Dans ce programme, la fonction économique minimise la fonction économique du meneur. On retrouve les contraintes de deuxième niveau. On trouve également une contrainte qui impose que la solution est bien une solution optimale de $Sub(x)$ et les contraintes de liaison. S'il existe une solution admissible à ce programme, c'est qu'il existe une solution optimale de concurrence pour la valeur de x fixée. Sinon, c'est qu'il n'en existe pas.

Remarque

Il existe des cas dans lesquels plusieurs hypothèses d'optimalité, ou même toutes les hypothèses d'optimalité peuvent aboutir aux même solutions. Considérons par exemple le programme suivant ;

$$(Ptrans\_Stable\_Ex) \begin{cases} \min\limits_{x} & x_1 + x_2 + x_3 + x_4 + x_5 \\ & x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 + x_5y_5 \geqslant 2 \\ & x \in \{0,1\}^n \\ & \begin{cases} \min\limits_{y} & x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 + x_5y_5 \\ & y_1 + y_2 \leqslant 1 \\ & y_1 + y_3 \leqslant 1 \\ & y_2 + y_3 \leqslant 1 \\ & y_3 + y_4 \leqslant 1 \\ & y_3 + y_5 \leqslant 1 \\ & y_1 + y_2 + y_3 + y_4 + y_5 = 3 \\ & y \in \{0,1\}^n \end{cases} \end{cases}$$

Dans ce programme, les différentes hypothèses d'optimalité donnent les mêmes solutions. En effet, ce programme comporte une contrainte de liaison qui borne la fonction

économique du second niveau. Ainsi, pour une solution x du meneur donnée, si une réponse rationnelle du suiveur ne respecte pas la contrainte de liaison, c'est le cas pour toutes les réponses rationnelles car elles donnent la même valeur de fonction économique. De plus, la fonction économique de premier niveau ne contient pas de variables de second niveau, le choix d'une réponse rationnelle du suiveur n'a donc aucun impact sur la valeur de la fonction économique de premier niveau. Donc les hypothèses d'optimalités aboutissent ici aux mêmes solutions. Toute solution rationnelle correspond à une réponse selon toutes les hypothèses.

2.2.2 Extension d'un Branch & Bound à l'ensemble des programmes biniveaux en nombres entiers

Dans cette section, nous proposons une version modifiée du Branch & Bound présenté dans [10] et détaillé plus haut dans ce chapitre. Nous reprenons donc toutes les notations utilisées par cet algorithme.

Pour adapter l'algorithme présenté dans [10], nous allons modifier l'étape d'itération générale. Dans la version originale, un programme mathématique était résolu et l'absence de solutions à ce programme impliquait qu'il n'y avait pas de solutions, ou de meilleures solutions que celles déjà trouvées, dans le nœud et dans les nœuds fils. Dans la version modifiée, nous procédons de la même façon en incluant les contraintes de liaison dans le programme. Nous obtenons donc le programme $\hat{P}_2$ suivant :

$$(\hat{P}_2) \begin{cases} \max\limits_{x,y} & f(x,y) \\ & h(y) \leqslant 0 \\ & g(x,y) \leqslant 0 \\ & G(x,y) \leqslant 0 \\ & H(x) \leqslant 0 \\ & F(x,y) \geqslant \alpha \\ & \sum\limits_{i=1}^{n^1} x_i \geqslant \beta \\ & x \in X \\ & y \in Y \end{cases}$$

Dans ce programme, on retrouve toutes les contraintes du modèle biniveau, une contrainte qui force la valeur de $F(x,y)$ a être supérieure à la valeur de la meilleure solution trouvée, et une contrainte qui impose qu'un certain nombre de variables x_i soient fixées à 1 et qui

sert, comme dans l'algorithme d'origine, pour l'étape de branchement. S'il n'existe pas, dans le nœud et ses fils, de solution admissible à ce problème, c'est qu'il n'existe pas de $(x, y) \in \Omega$ admissibles qui améliore la valeur de la meilleure solution trouvée. S'il existe une solution admissible, on fixe les variables x et on cherche une réponse y selon l'hypothèse choisie. Les traitements spécifiques aux hypothèses sont détaillés plus haut dans les section correspondantes.

La variante de l'algorithme pour résoudre des programmes biniveaux en nombres entiers avec des contraintes de liaison donne ainsi :

1. **Initialisation** : $k = 0$, $W_k = \varnothing$, $\alpha = -\infty$, $\beta = 0$, $\underline{F} = -\infty$
2. **Itération générale**
   - Fixer $x_i := 1$ pour tout $i \in S_k^1$ et $x_i := 0$ pour tout $i \in S_k^0$.
   - Trouver une solution admissible à $(\hat{P}2)$. Si elle n'existe pas, aller à l'étape 5. Sinon soit (x_k, y_k) cette solution.
   - Résoudre le programme $Sub(x)$ en fixant les variables x à x_k. Soit γ la valeur de la solution trouvée.
   - Déterminer une réponse y_k selon l'hypothèse choisie (optimiste, pessimiste, trahison, concurrence)
   - Si $G(x_k, y_k) \leqslant 0$, $\underline{F} = max(\underline{F}, F(x_k, \hat{y}_k))$
   - Soit $J = \{i \in S_{k-1}^* : x_i^k = 1\}$.
     - Si $J = \varnothing$ alors $S_k^1 = S_{k-1}^1$, $S_k^0 = S_{k-1}^0$, $S_k^* = S_{k-1}^*$ et $P_k = P_{k-1}$ aller à l'étape 4
     - Sinon, aller à l'étape 3
3. **Branchement :** Créer $|J|$ nouveaux nœuds en ajoutant les indices de J à S_k^1.
4. $\alpha := \underline{F} + 1$ et $\beta := 1 + |S_k^+|$, revenir à l'étape 1.
5. **Retour arrière :** S'il ne reste plus de nœuds, aller à l'étape suivante. Sinon, remonter d'un niveau, trouver la variable correspondante X_j et faire un branchement dans la branche complémentaire $x_j = 0$. Mettre à jour S_k^1, S_k^0 et S_k^*. $\beta = 0$. Revenir au début de l'étape 2.

6. **Fin** Si $\underline{F} = -\infty$ alors il n'existe pas de solution admissible au problème. Sinon, retourner la solution associée à $\underline{F}$

Puisque cet algorithme peut visiter tous les vecteurs x, il aboutit à une solution optimale. Mais il n'est pas pour autant efficace et souffre de plusieurs défauts. Considérons à nouveau le programme :

$$(Ptrans\_Stable\_Ex) \begin{cases} \min\limits_{x} & x_1 + x_2 + x_3 + x_4 + x_5 \\ & \sum\limits_{i=1}^{n} x_i y_i \geqslant 3 \\ & x \in \{0,1\}^n \\ & \begin{cases} \min\limits_{y} & x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 + x_5 y_5 \\ & y_1 + y_2 \leqslant 1 \\ & y_1 + y_3 \leqslant 1 \\ & y_2 + y_3 \leqslant 1 \\ & y_3 + y_4 \leqslant 1 \\ & y_3 + y_5 \leqslant 1 \\ & y_1 + y_2 + y_3 + y_4 + y_5 = 3 \\ & y \in \{0,1\}^n \end{cases} \end{cases}$$

Comme indiqué dans la section précédente, pour ce programme, les différentes hypothèses d'optimalité donnent les mêmes solutions.

Un premier problème de l'algorithme est que, contrairement à celui décrit dans [10], on ne détermine pas de solution $(x, y) \in IR$ (définition de IR donnée dans le chapitre 1) à chaque nœud. Autrement dit, une fois que les variables x ont été fixées, la réponse du suiveur peut ne pas respecter les contraintes de liaison et donc l'exploration d'un nœud ne donne pas de nouvelle solution. Mais cette absence de solution ne permet pas pour autant d'élaguer le nœud. Par exemple, supposons que dans un nœud, la variable x_1 soit fixée à 1. Une solution admissible de $(Ptrans\_Stable\_Ex)$ est $x_1 = 1$, $x_4 = 1$, $x_5 = 1$, $y_4 = 1$, $y_5 = 1$, $y_1 = 1$. Mais cette réponse du suiveur ne respecte pas la contrainte de liaison. Or, il peut exister dans un nœud fils la solution dans laquelle les variables x_1, x_2, x_4 et x_5 valent 1. Dans ce cas, toutes les réponses rationnelles du suiveur respectent la contrainte de liaison. C'est aussi la solution optimale du problème. Ainsi, on peut être amené à explorer tous les nœuds d'une branche même si on ne trouve pas, dans un nœud, de solution admissible du programme biniveau.

Le deuxième problème de l'algorithme est que la résolution de $(Ptrans\_Stable\_Ex)$

peut aussi être inefficace pour guider rapidement l'algorithme vers une solution optimale. Par exemple, supposons que la variable x_3 ne soit pas fixée et que $\beta = 4$. Une solution admissible de $(Ptrans\_Stable\_Ex)$ est $x_1 = 1$, $x_4 = 1$, $x_5 = 1$, $x_3 = 1$, $y_4 = 1$, $y_5 = 1$, $y_1 = 1$. Dans ce cas, l'algorithme effectue un branchement, sur une variable x non nulle qui n'est pas déjà fixée, par exemple x_3. Or le sommet correspondant n'appartient à aucun stable optimal du graphe. L'exploration de la branche où $x_3 = 1$ n'aboutit à aucune solution optimale. Ainsi, la règle de branchement ne guide pas nécessairement l'exploration des branches vers de bonnes solutions. Trouver une solution optimale de $(Ptrans\_Stable\_Ex)$ ne permet pas d'améliorer l'efficacité de l'algorithme. En effet, dans l'exemple, $x_1 = 1, x_4 = 1, x_5 = 1, x_3 = 1, y_4 = 1, y_5 = 1, y_1 = 1$ était une solution optimale de $(Ptrans\_Stable\_Ex)$.

2.3 Perspectives

Si l'algorithme de la section précédente permet effectivement de résoudre les programmes biniveaux en variables binaires avec des contraintes de liaison, il n'est pas pour autant efficace. La règle de branchement ne guide pas nécessairement l'algorithme vers une solution optimale et il peut être nécessaire d'explorer des branches complètement car l'absence de solution admissible du programme biniveau dans un nœud n'implique pas qu'une solution puisse être trouvée dans un des nœuds fils. Cet algorithme ne peut donc pas être utilisé. C'est pourquoi, dans le chapitre 5, nous déterminerons des d-transversaux sans passer par la résolution de problèmes biniveaux.

Chapitre 3

d-extensibles de stables dans les graphes bipartis

3.1 Introduction

Considérons un système dans lequel des composants peuvent tomber en panne. Il est possible de maintenir des éléments pour éviter les pannes mais cela a un coût. Pour que le bon fonctionnement ne soit pas affecté en cas de défaillance, on souhaite maintenir une partie aussi petite que possible du système pour qu'en cas de panne dans l'autre partie, le système puisse toujours opérer dans de bonnes conditions. Autrement dit, on veut sélectionner une partie aussi grande que possible du système qu'il ne sera pas nécessaire de maintenir à tout prix et ainsi protéger un minimum d'éléments sans perturber le bon fonctionnement du système.

Nous représentons ici le système par un graphe biparti $G = (B, R, E)$ dont les sommets sont les éléments du système. Ces éléments ne peuvent pas fonctionner si un de leurs voisins fonctionne. Ainsi, pour que le système fonctionne, ses éléments doivent former un stable. Le système opère donc de façon optimale lorsque les éléments forment un stable de cardinal maximal du graphe G.

Soit $d < \alpha(G)$ un entier positif. Dans notre problème, on souhaite déterminer le plus grand sous-ensemble F de sommets de G tel que tout stable de F de cardinal égal à d peut être complété en un stable de cardinal maximal de G en n'utilisant que des sommets qui ne sont pas dans F. L'ensemble F est appelé un *d-extensible*. On peut remarquer que

le problème admet toujours une solution. En effet, tout sous-ensemble de d sommets d'un stable optimal est un d-extensible.

Le chapitre est organisé de la façon suivante. Tout d'abord, la première section donne les définitions d'un d-extensible et illustre cette notion à travers un exemple. Une deuxième section donne une modélisation du problème par la programmation mathématique en nombres entiers. Une troisième section traite des graphes bipartis qui ne contiennent que des sommets libres et fournit une caractérisation des d-extensibles, une borne inférieure du cardinal maximal d'un d-extensible et s'intéresse à des classes de graphes particulières. Dans la quatrième section nous déterminons un d-extensible optimal si le graphe contient des sommets forcés si d est supérieur à une valeur. La cinquième section s'intéresse aux arbres qui ne contiennent que des sommets libres, améliore la borne donnée dans la troisième section et s'intéresse à des classes d'arbres particulières.

3.2 Définitions

Soit un graphe $G = (V, E)$. Soit $U \subset V$. Un stable $S \subset U$ de G est dit *extensible* par rapport à U si il existe $\widehat{S} \subset (V - U)$ tel que $S \cup \widehat{S}$ est un stable de cardinal maximal de G.

Soit un entier d tel que $1 \leqslant d < \alpha(G)$. Un ensemble $F \subset V$ est un d-extensible de G si tout stable S de $G[F]$ de cardinal d est extensible par rapport à F.

Autrement dit, F est un d-extensible si tout stable de cardinal égal à d de $G[F]$ peut être complété en un stable de cardinal maximal de G en n'utilisant que des sommets qui ne sont pas dans F.

Par exemple, considérons le graphe de la figure 3.1. Le cardinal maximal d'un stable de ce graphe est 3. Il a deux stables de cardinal maximal : $\{a, d, e\}$ et $\{b, d, e\}$. Considérons l'ensemble de sommets $F = \{a, b, c, d\}$, en gris dans la figure 3.2. Le stable $\{a, d\}$ est extensible par rapport à F. En effet, il est inclus dans F et il existe un stable optimal du graphe, $\{a, d, e\}$ dont l'intersection avec F est exactement $\{a, d\}$ (voir figure 3.3). De même, le stable $\{b, d\}$ est extensible par rapport à F (voir figure 3.4). Tous les stables inclus dans F de cardinal 2 sont extensibles par rapport à F, c'est donc un 2-extensible.

On s'intéresse au problème EXTENSIBLE suivant :

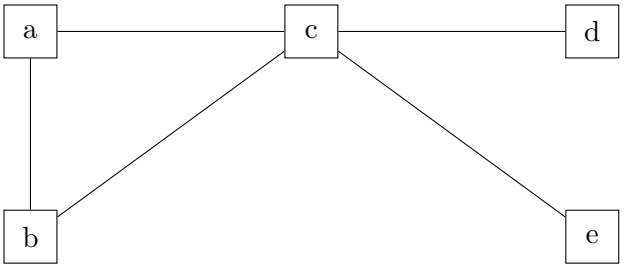


FIGURE 3.1 – Graphe

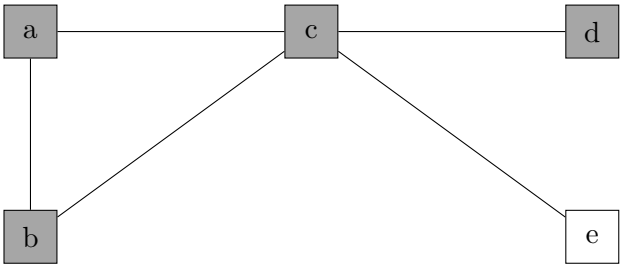


FIGURE 3.2 – Graphe avec 4 sommets sélectionnés dans un 2-extensible F

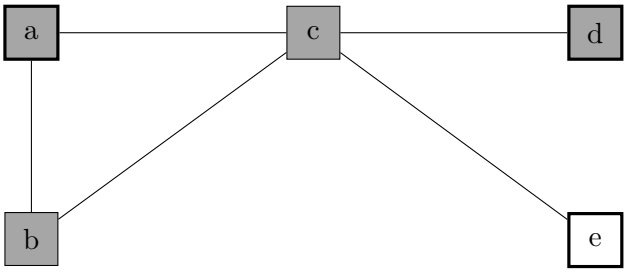


FIGURE 3.3 – Le stable $\{a, d\}$ est extensible par rapport à F

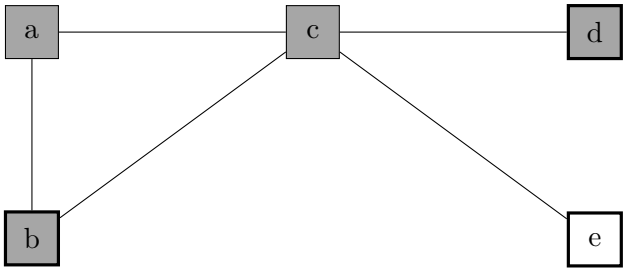


FIGURE 3.4 – Le stable $\{b, d\}$ est extensible par rapport à F

— Instance : Un graphe $G = (V, E)$, un entier positif $1 \leqslant d < \alpha(G)$, un entier positif $k \leqslant |V|$

— Question : Est-ce que G contient un d-extensible F de taille k ou plus ?

Dans la suite du chapitre, on recherche un d-extensible de plus grand cardinal possible. Un d-extensible est dit optimal si son cardinal est maximal.

Ce problème est une variation des problèmes d'extension de couplage ([4], [67], [27], [33]) dans lesquels on recherche pour deux entiers d et n un graphe $G = (V, E)$ de n sommets avec un nombre minimum d'arêtes tel que à tout couplage $\hat{M}$ de taille d dans le graphe complémentaire, on peut associer un couplage M de $G[V - V(\hat{M})]$ de taille $\lfloor \frac{n}{2} \rfloor - d$ tel que $M \cup \hat{M}$ est un couplage parfait (ou presque parfait) du graphe G.

Dans les graphes qui contiennent des sommets forcés, on a la propriété suivante :

Propriété 3.2.1. *Soient G un graphe et F un d-extensible de G. Pour tout stable S de $G[F]$ tel que $|S| = d$, on a $\Xi(G) \cap F \subset S$*

Démonstration. Soit S un stable de $G[F]$ tel que $|S| = d$ et soit $x \in (\Xi(G) \cap F)$. On sait que F est un d-extensible donc $\exists\ S^*$, stable optimal de G tel que $S^* \cap F = S$. Par ailleurs, $x \in \Xi(G)$ donc $x \in S^*$. Ainsi, $x \in S$. □

Les sommets exclus n'appartiennent à aucun stable de cardinal maximal. Ils ne peuvent donc pas être utilisés pour étendre un stable et ne peuvent pas être contenus dans un stable de cardinal d d'un d-extensible. On peut donc les retirer du graphe. Dans les sections suivantes, les graphes ne contiennent plus de sommets exclus. Remarquons qu'alors les sommets forcés sont des sommets isolés.

3.3 Modélisation par la programmation linéaire multiniveau en nombres entiers

Le problème de recherche d'un d-extensible de cardinal maximal peut être modélisé par un programme mathématique linéaire en variables $0 - 1$ à trois niveaux. Les programmes mathématiques à trois niveaux sont semblables aux modèles à deux niveaux présentés dans le chapitre précédent avec une prise de décision en plus. Le premier niveau prend une

décision, le deuxième niveau réagit à la décision du premier et le troisième niveau réagit à la décision du deuxième. Les décisions de chaque niveau influencent le résultat des niveaux supérieurs.

Nous allons décrire les trois niveaux dans le cas des d-extensibles. Le premier niveau cherche à déterminer un d-extensible de plus grand cardinal possible. Le deuxième niveau cherche s'il existe un stable de d sommets du graphe induit par l'ensemble déterminé au premier niveau qui ne soit pas extensible par rapport à cet ensemble ; c'est-à-dire qu'il cherche si l'ensemble déterminé au premier niveau est bien un d-extensible. Le troisième niveau vérifie si le stable sélectionné au deuxième niveau est extensible. Pour cela il cherche à étendre le stable de d sommets du deuxième niveau en un stable de plus grand cardinal possible du graphe à l'aide de sommets qui ne sont pas dans le d-extensible.

Dans le programme suivant, on considère un graphe $G = (V, E)$ qui contient n sommets. Le premier niveau contrôle les variables $x_i \in \{0, 1\}$. Les variables x_i valent 1 si le sommet i appartient au d-extensible et 0 sinon. Donc si on note F le d-extensible déterminé par le programme mathématique, $F = \{i | x_i = 1\}$. Le deuxième niveau contrôle les variables $y_i \in \{0, 1\}$. Les variables y_i valent 1 si le sommet i appartient au stable de cardinal d inclus dans le d-extensible et 0 sinon. Le troisième niveau contrôle les variables $z_i \in \{0, 1\}$. Les variables z_i valent 1 si le sommet i n'appartient pas à F et a été ajouté au stable déterminé par le deuxième niveau et 0 sinon.

$$
\begin{cases}
\max\limits_{x} & \sum\limits_{i=1}^{n} x_i & & \\
& \sum\limits_{i=1}^{n} z_i = \alpha(G) - d & & (1) \\
& x_i \in \{0,1\} & \forall i \in \{1,..,n\} & \\
& \begin{cases}
\min\limits_{y} & \sum\limits_{i=1}^{n} z_i & & \\
& y_i + y_j \leqslant 1 & \forall (ij) \in E & (2) \\
& y_i \leqslant x_i & \forall i \in \{1,..,n\} & (3) \\
& \sum\limits_{i=1}^{n} y_i = d & & (4) \\
& y \in \{0,1\} & \forall i \in \{1,..,n\} & \\
& \begin{cases}
\max\limits_{z} & \sum\limits_{i=1}^{n} z_i & & \\
& z_i + z_j \leqslant 1 & \forall (ij) \in E & (5) \\
& z_i + x_i \leqslant 1 & \forall i = \{1,..,n\} & (6) \\
& y_i + z_j \leqslant 1 & \forall (ij) \in E & (7) \\
& z \in \{0,1\} & \forall i \in \{1,..,n\} &
\end{cases}
\end{cases}
\end{cases}
$$

Le premier niveau maximise le nombre de sommets contenus dans le d-extensible. La contrainte (1) assure que le nombre de variables de troisième niveau, donc le nombre de sommets ajoutés au stable de cardinal d pour l'étendre en un stable optimal du graphe, est bien de $\alpha(G) - d$, c'est-à-dire que le stable est extensible par rapport à l'ensemble des sommets sélectionnés par les variables x_i.

Le deuxième niveau cherche un stable de cardinal d du graphe induit par l'ensemble de sommets déterminé au premier niveau, qui ne soit pas extensible. Pour cela, le deuxième niveau cherche à minimiser la somme des variables z_i, c'est-à-dire le plus grand nombre de sommets qu'on peut ajouter au stable de cardinal d pour l'étendre en un stable maximal du graphe. Si la somme des variables z_i est inférieure à $\alpha(G) - d$, c'est que le stable n'est pas extensible. La contrainte (2) indique que l'ensemble de sommets sélectionnés au deuxième niveau est bien un stable. La contrainte (3) permet de ne sélectionner des sommets que dans le graphe induit par le d-extensible. La contrainte (4) assure de sélectionner d sommets.

Le troisième niveau cherche à étendre le plus possible le stable déterminé au deuxième niveau à l'aide de sommets qui ne sont pas dans le d-extensible. Pour cela, il considère le graphe induit par les sommets qui n'appartiennent pas au d-extensible et qui ne sont

pas voisins de sommets du stable déterminé au deuxième niveau. Puis il cherche un stable de cardinal maximal de ce graphe. Ainsi, il étend le plus possible le stable déterminé au deuxième niveau. Si l'ensemble déterminé au premier niveau est bien un d-extensible, le stable déterminé au troisième niveau est de cardinal $\alpha(G) - d$. La contrainte (5) assure que l'ensemble déterminé est bien un stable. La contrainte (6) empêche de sélectionner des sommets dans le d-extensible. La contrainte (7) garantit que l'ensemble déterminé au troisième niveau forme bien un stable avec l'ensemble déterminé au deuxième niveau.

En résumé :

— Les variables x_i désignent les sommets de G qui appartiennent au d-extensible F

— Les variables y_i désignent les sommets d'un stable S de cardinal d de $G[F]$ tel que le cardinal du plus grand stable de G dont l'intersection avec F est S est le plus petit possible

— Les variables z_i désignent les sommets du plus grand stable $\hat{S}$ de G tel que $S \cup \hat{S}$ est un stable de G.

— La contrainte (1) assure que tout stable de cardinal d de $G[F]$ est extensible

— La contrainte (2) assure que S est bien un stable

— La contrainte (3) assure que S ne contient que des sommets de F

— La contrainte (4) assure que S contient exactement d sommets

— La contrainte (5) assure que $\hat{S}$ est bien un stable

— La contrainte (6) assure que $\hat{S}$ ne contient pas de sommets de F

— La contrainte (7) assure que $\hat{S} \cup S$ est bien un stable de G

A notre connaissance, il n'existe pas dans la littérature d'algorithme efficace pour ce type de programmes mathématiques. En effet, l'approche de [8] et de [77], ne prend pas en compte les contraintes de niveau i qui font intervenir des variables de niveau inférieur. Un article récent [48] prend en compte ces contraintes mais il n'a pour l'instant été utilisé que sur de petites instances.

La résolution de ce programme semblant difficile, nous nous intéressons dans la suite du chapitre au cas des d-extensibles de stables dans les graphes bipartis. Nous présentons une

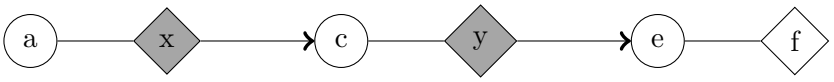


FIGURE 3.5 – Illustration du point 1 de la Propriété 3.4.1. Dans ce graphe $\alpha(G) = 3$

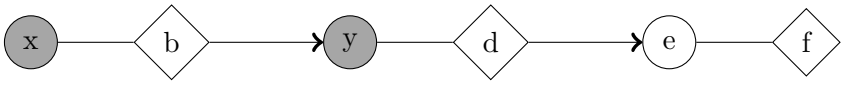


FIGURE 3.6 – Illustration du point 2 de la Propriété 3.4.1

caractérisation des d-extensibles, nous proposons des bornes inférieures de la cardinalité maximale d'un d-extensible et nous résolvons le problème dans des cas particuliers.

3.4 Cas des graphes bipartis qui ne contiennent que des sommets libres

Dans cette section, on considère un graphe biparti $G = (B, R, E)$ qui ne contient que des sommets libres. Donc $\Phi(G) = B \cup R$, $\alpha(G) = |B| = |R| = \frac{\phi(G)}{2}$ et $1 \leqslant d \leqslant \frac{\phi(G)}{2}$.

Cette section présente une caractérisation des stables extensibles, une caractérisation des d-extensibles, une borne inférieure du cardinal maximal d'un d-extensible, des propriétés sur les d-extensibles et des cas particuliers dans lesquels déterminer un d-extensible de cardinal maximal se fait en temps polynomial.

Propriété 3.4.1. *Soit $F \subset (B \cup R)$. Un stable $S \subset F$ est extensible par rapport à F si et seulement si pour tout couple de sommets (x, y) de G tels que $C(x) \to C(y)$, on a*

1. *si $x \in S \cap B$ et $y \in F \cap B$ alors $y \in S$*
2. *si $x \in F \cap R$ et $y \in S \cap R$ alors $x \in S$*
3. *si $x \in F \cap R$ et $y \in F \cap B$ alors $S \cap \{x, y\} \neq \varnothing$*
4. *si $x \in F \cap B$ et $y \in F \cap R$ alors $|S \cap \{x, y\}| \leqslant 1$*

Le premier cas de la Propriété est illustré dans le Figure 3.5, le deuxième cas dans la Figure 3.6, le troisième cas dans la figure 3.7 et le quatrième cas dans la Figure 3.8.

Démonstration. Soit S un stable extensible par rapport à F. Comme S est extensible, $\exists$ S^* stable optimal de G tel que $S^* \cap F = S$. Soit un couple de sommets (x, y) de G tels que $C(x) \to C(y)$.

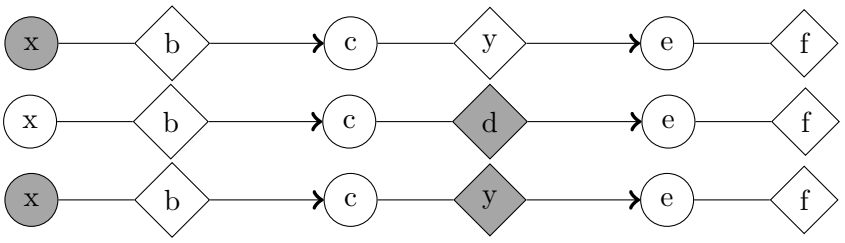


FIGURE 3.7 – Illustration du point 3 de la Propriété 3.4.1

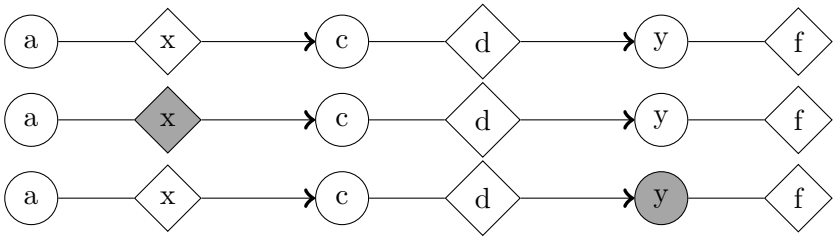


FIGURE 3.8 – Illustration du point 4 de la Propriété 3.4.1

Supposons que $x \in S \cap B$ et $y \in F \cap B$. D'après le Corollaire 1.2.3, si $x \in S^*$ alors $y \in S^*$. Or $S^* \cap F = S$ donc $y \in S$. Le premier point est donc vérifié.

De la même façon, le deuxième point est vérifié en considérant $x \in R$ et $y \in R$.

Supposons que $x \in F \cap R$ et $y \in F \cap B$. Si $x \notin S$ alors $x \notin S^*$. Donc $V(C(x)) \cap S^* = V(C(x)) \cap B$. Or $y \in B$ donc d'après la Propriété 1.2.5, $y \in S^*$. De plus $S^* \cap F = S$ donc $y \in S$. Le troisième point est donc vérifié.

Supposons que $x \in F \cap B$ et $y \in F \cap R$. Si $x \in S^*$ alors $x \in S$ et d'après la Propriété 1.2.5, $V(C(y)) \cap S^* = V(C(y)) \cap B$. Comme $y \in R$, $y \notin S^*$ donc $y \notin S$. Le quatrième point est donc vérifié.

Considérons maintenant un stable S de $G[F]$ qui respecte les quatre points de la propriété et montrons qu'il est extensible par rapport à F. Pour cela, construisons un stable S^* optimal de G tel que $S^* \cap F = S$.

Construisons S^* selon les étapes suivantes et montrons que c'est bien un stable optimal de G tel que $S^* \cap F = S$. Lors de l'initialisation, $S^* = \varnothing$

1. Pour tout sommet $x \in R \cap F$ tel que $x \notin S$: pour toute composante $C(y)$ telle que $C(x) \to C(y)$, $S^* \leftarrow S^* \cup (V(C(y)) \cap B)$

2. Pour tout sommet $x \in S \cap B$ tel que $V(C(x)) \cap S^* = \varnothing$: pour toute composante $C(y)$ telle que $C(x) \to C(y)$, $S^* \leftarrow S^* \cup (V(C(y)) \cap B)$
3. Pour tout sommet $y \in B \cap F$ et $y \notin S$: pour toute composante $C(x)$ telle que $C(x) \to C(y)$, $S^* \leftarrow S^* \cup (V(C(x)) \cap R)$
4. Pour tout sommet $y \in S \cap R$ tel que $V(C(y)) \cap S^* = \varnothing$: pour toute composante $C(x)$ telle que $C(x) \to C(y)$, $S^* \leftarrow S^* \cup (V(C(x)) \cap R)$
5. Pour toutes les composantes C_i telle que $V(C_i) \cap S^* = \varnothing$, $S^* \leftarrow S^* \cup (V(C_i) \cap R)$

L'étape 5 implique que l'ensemble S^* contient des sommets de toutes les composantes. Montrons que chaque composante C_i, soit $V(C_i) \cap S^* = V(C_i) \cap R$, soit $V(C_i) \cap S^* = V(C_i) \cap B$.

Pour les composantes sur lesquelles s'applique l'étape 5, il est évident que $V(C_i) \cap S^* = V(C_i) \cap R$. Montrons que si des sommets d'une composante sont sélectionnés lors de l'étape 1 ou de l'étape 2, alors aucun sommet de cette composante n'est sélectionné lors de l'étape 3 ou de l'étape 4.

Soit $C(z)$ une composante dont des sommets ont été sélectionnés lors de l'étape 1. Dans ce cas, $\exists\ x \in R \cap F$ tel que $x \notin S$ et $C(x) \to C(z)$. Si des sommets de $C(z)$ étaient sélectionnés lors de l'étape 3, alors il existerait $y \in B \cap F$ tel que $y \notin S$ et $C(z) \to C(y)$. Or, dans ce cas, $C(x) \to C(y)$ et d'après le troisième point de la propriété $S \cap \{x, y\} \neq \varnothing$. Contradiction.

Soit $C(z)$ une composante dont des sommets ont été sélectionnés lors de l'étape 1. Dans ce cas, $\exists\ x \in R \cap F$ tel que $x \notin S$ et $C(x) \to C(z)$. Si des sommets de $C(z)$ étaient sélectionnés lors de l'étape 4 alors il existerait $y \in R \cap S$ tel que $C(z) \to C(y)$. Donc $C(x) \to C(y)$. Donc d'après le deuxième point de la propriété, $x \in S$. Contradiction.

Soit $C(z)$ une composante dont des sommets ont été sélectionnés lors de l'étape 2. Dans ce cas, $\exists\ x \in B \cap S$ tel que $C(x) \to C(z)$. Si des sommets de $C(z)$ étaient sélectionnés lors de l'étape 3 alors il existerait $y \in B \cap F$ tel que $y \notin S$ et $C(z) \to C(y)$. Donc d'après le premier point de la propriété, $y \in S$. Contradiction.

Soit $C(z)$ une composante dont des sommets ont été sélectionnés lors de l'étape 2. Dans ce cas, $\exists\ x \in B \cap S$ tel que $C(x) \to C(z)$. Si des sommets de $C(z)$ étaient sélectionnés lors

de l'étape 4 alors il existerait $y \in R \cap S$ tel que $C(z) \to C(y)$. Donc $C(x) \to C(y)$. Donc d'après le quatrième point de la propriété, $y \notin S$. Contradiction.

Donc pour chaque composante, ses sommets ne sont sélectionnés que lors d'une seule étape. Donc pour toute composante C_i, $|V(C_i) \cap S^*| = \frac{|V(C_i)|}{2}$. Ainsi, S^* a bien le cardinal d'un stable optimal de G. Démontrons maintenant que S^* est un stable.

Pour chaque composante C_i, soit $V(C_i) \cap S^* = V(C_i) \cap B$, soit $V(C_i) \cap S^* = V(C_i) \cap R$. Donc si S^* n'est pas un stable, il existe deux composantes $C(x)$ et $C(y)$ telle que $V(C(x)) \cap S^* = V(C(x)) \cap B$, $V(C(y)) \cap S^* = V(C(y)) \cap R$, $C(x) \to C(y)$ et $C(x) \neq C(y)$. Cela implique que des sommets de $C(x)$ sont sélectionnés lors des étapes 1 ou 2 et des sommets de $C(y)$ lors des étapes 3 ou 4. Or, $C(x) \to C(y)$ donc des sommets de $C(y)$ doivent être sélectionnés lors des étapes 1 ou 2. Contradiction.

Donc S^* est bien un stable optimal de G. Il reste à démontrer que $S^* \cap F = S$.

Soit $x \in B \cap F$ et $x \notin S$. La composante $C(x)$ a ses sommets sélectionnés uniquement lors de l'étape 3 donc $x \notin S^*$. De même, si $x \in R \cap F$ et $x \notin S$, la composante $C(x)$ a ses sommets sélectionnés uniquement lors de l'étape 1 donc $x \notin S^*$. Donc aucun sommet de F n'est dans S^* s'il n'appartient pas à S. Il reste à démontrer que $S \subset S^*$.

Soit $x \in S$. La composante $C(x)$ a des sommets sélectionnés soit lors de la deuxième étape, soit de la quatrième et $x \in S^*$. Donc $S \subset S^*$.

Donc comme S^* est un stable optimal de G tel que $S^* \cap F = S$, S est bien extensible par rapport à F.

□

A l'aide de cette caractérisation, nous prouvons la propriété suivante qui permet, pour $d \geqslant 2$, de déduire une borne inférieure du cardinal maximal d'un $(d+1)$-extensible à partir d'un d-extensible.

Propriété 3.4.2. *Pour $d \geqslant 2$, si $F \subset (B \cup R)$ est un d-extensible de G alors il est aussi un k-extensible de G pour $k > d$.*

Démonstration. Supposons que F est un d-extensible mais pas un k-extensible pour $k > d$. Soit S un stable de $G[F]$ tel que $|S| = k$.

Si S n'est pas extensible par rapport à F, un des points de la Propriété 3.4.1 page 64 n'est pas respecté.

Si c'est le premier point, $\exists\ x, y \in F \cap B$ tels que $C(x) \rightarrow C(y)$, $x \in S$ et $y \notin S$. Or, dans ce cas, on peut construire un stable de cardinal d qui n'est pas extensible en retirant $k - d$ sommets à S tout en conservant x. Donc F n'est pas un d-extensible.

De même, si c'est le deuxième point, on peut construire un stable de cardinal d qui n'est pas extensible.

Si c'est le troisième point, $\exists\ x \in F \cap R$ et $y \in F \cap B$ tels que $C(x) \rightarrow C(y)$, $x \notin S$ et $y \notin S$. De nouveau, en retirant $k-d$ sommets à S, on peut construire un stable de cardinal d qui ne contient ni x ni y donc qui n'est pas extensible.

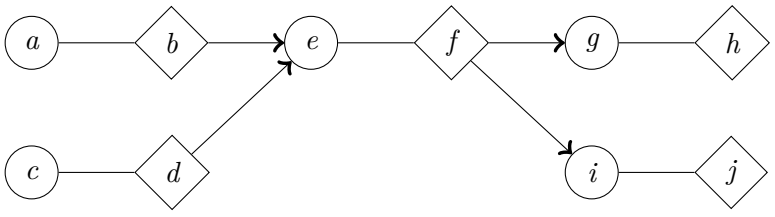
Si c'est le quatrième point, $\exists\ x \in F \cap B$ et $y \in R \cap F$ tels que $C(x) \rightarrow C(y)$, $x \in S$ et $y \in S$. Comme $d \geqslant 2$, on peut construire un stable de cardinal d qui contient x et y donc qui n'est pas extensible.

Donc, si $d \geqslant 2$, si F est un d-extensible alors c'est aussi un k-extensible pour $k \geqslant d$.

□

Remarque 3.4.1. *La Propriété 3.4.2 n'est pas toujours vérifiée pour $d = 1$.*

Exemple :



L'ensemble $\{b, d, g, i\}$ est un 1-extensible de cardinal maximal mais ce n'est pas un 2-extensible : le stable $\{b, g\}$ n'est pas extensible par rapport à $\{b, d, g, i\}$ d'après la Propriété 3.4.1.

La caractérisation des stables extensibles permet de caractériser l'ensemble des d-extensibles, comme indiqué dans la propriété suivante.

Propriété 3.4.3. *$F \subset (B \cup R)$ est un d-extensible de G si et seulement si pour tout couple (x, y) de sommets de F tels que $C(x) \to C(y)$*

— *Si $C(x) \neq C(y)$ et $(xy) \notin E$:*

1. *Si $x \in R$ et $y \in B$ alors $\alpha(G[F - \{x, y\}]) < d$*
2. *Si $x \in B$ et $y \in B$ alors $\alpha(G[F - (\{y\} \cup \Gamma(x))]) < d$*
3. *Si $x \in R$ et $y \in R$ alors $\alpha(G[F - (\{x\} \cup \Gamma(y))]) < d$*
4. *Si $x \in B$ et $y \in R$ alors $\alpha(G[F - (\Gamma(x) \cup \Gamma(y))]) < d$*

— *Si $C(x) = C(y)$:*

5. *Si $x \in B$ et $y \in B$ alors $\alpha(G[F-(\{y\}\cup\Gamma(x))]) < d$ et $\alpha(G[F-(\{x\}\cup\Gamma(y))]) < d$*
6. *Si $x \in R$ et $y \in R$ alors $\alpha(G[F-(\{y\}\cup\Gamma(x))]) < d$ et $\alpha(G[F-(\{x\}\cup\Gamma(y))]) < d$*
7. *Si $x \in B$ et $y \in R$ ou $x \in R$ et $y \in B$ alors $\alpha(G[F - \{x, y\}]) < d$ et, si $xy \notin E$ alors $\alpha(G[F - (\Gamma(x) \cup \Gamma(y)]) < d$*

Démonstration. Soit F un d-extensible de G. Montrons que F respecte bien la propriété.

Soit $x \in F$, $y \in F$ tels que $C(x) \to C(y)$, $C(x) \neq C(y)$ et $(xy) \notin E$

1. $x \in R$ et $y \in B$. Si $\alpha(G[F - \{x, y\}]) \geqslant d$, $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \notin S$ et $y \notin S$. D'après le point 3 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.
2. $x \in B$ et $y \in B$. Si $\alpha(G[F - (\{y\} \cup \Gamma(x))]) \geqslant d$, $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \in S$ et $y \notin S$. D'après le point 1 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.
3. $x \in R$ et $y \in R$. Si $\alpha(G[F - (\{x\} \cup \Gamma(y))]) \geqslant d$, $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \notin S$ et $y \in S$. D'après le point 2 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.
4. $x \in B$ et $y \in R$. Si $\alpha(G[F - (\Gamma(x) \cup \Gamma(y))]) \geqslant d$, $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \in S$ et $y \in S$. D'après le point 4 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.

Soient $x \in F$ et $y \in F$ tels que $C(x) = C(y)$

5. $x \in B$ et $y \in B$. Si $\alpha(G[F - (\{y\} \cup \Gamma(x))]) \geqslant d$ ou $\alpha(G[F - (\{x\} \cup \Gamma(y))]) \geqslant d$ alors $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \in S$ et $y \notin S$ ou $x \notin S$ et $y \in S$. D'après le point 1 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.
6. $x \in R$ et $y \in R$. Si $\alpha(G[F - (\{y\} \cup \Gamma(x))]) \geqslant d$ ou $\alpha(G[F - (\{x\} \cup \Gamma(y))]) \geqslant d$ alors $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \in S$ et $y \notin S$ ou $x \notin S$ et $y \in S$. D'après le point 2 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.
7. $x \in R$ et $y \in B$ ou $x \in B$ et $y \in R$. Si $\alpha(G[F - \{x, y\}]) \geqslant d$, $\exists$ S stable de $G[F]$ tel que $|S| = d$, $x \in S$ et $y \in S$. D'après le point 4 de la Propriété 3.4.1 page 64, S n'est pas extensible par rapport à F. Contradiction.

Considérons maintenant $F \subset (B \cup R)$ tel que F respecte la propriété. Montrons que F est bien un d-extensible.

Soit S un stable de $G[F]$ tel que $|S| \geqslant d$. Montrons que S est extensible, donc qu'il respecte tous les points de la Propriété 3.4.1.

Soient $x \in B \cap F$ et $y \in B \cap F$. D'après le point 2, comme $|S| = d$, si $C(x) \to C(y)$ et $x \in S$ alors $y \in S$. De même, d'après le point 5, si $C(x) = C(y)$ et $x \in S$ alors $y \in S$. Donc S respecte le premier point de la Propriété 3.4.1.

Soient $x \in R \cap F$ et $y \in R \cap F$. D'après le point 3, comme $|S| = d$, si $C(x) \to C(y)$ et $x \in S$ alors $y \in S$. De même, d'après le point 6, si $C(x) = C(y)$ et $x \in S$ alors $y \in S$. Donc S respecte le deuxième point de la Propriété 3.4.1.

Soient $x \in R \cap F$, $y \in B \cap F$ et $C(x) \to C(y)$. D'après le point 1, comme $|S| \geqslant d$, $S \cap \{x, y\} \neq \varnothing$. Donc S respecte le troisième point de la Propriété 3.4.1.

Soient $x \in B \cap F$, $y \in R \cap F$. D'après le point 4, comme $|S| \geqslant d$, si $C(x) \to C(y)$ et $C(x) \neq C(y)$, si $x \in S$ alors $y \notin S$ et si $y \in S$ alors $x \notin S$. De même d'après le point 7 si $C(x) = C(y)$. Donc S respecte le quatrième point de la Propriété 3.4.1.

Donc S est extensible par rapport à F et F est bien un d-extensible de G.

□

Propriété 3.4.4. *Soit F un d-extensible de G. Si F contient deux sommets a et b tels*

que $C(a) = C(b)$ *et* $(ab) \in E$ *alors* $|F \cap B| \leqslant d$ *et* $|F \cap R| \leqslant d$.

Démonstration. D'après la Propriété 3.4.1 page 64, pour être extensible par rapport à F, un stable de cardinal d de $G[F]$ doit contenir soit a, soit b.

Si $|F \cap B| > d$ alors il existe un stable de d sommets de $G[F]$ qui contient d sommets de $F \cap B$ mais qui ne contient ni a, ni b donc qui n'est pas extensible par rapport à F. Donc F n'est pas un d-extensible. Contradiction.

Si $|F \cap R| > d$ on aboutit à la même contradiction. □

Propriété 3.4.5. *Soit* F *un* d*-extensible de* G*. Si* $|V_c| \geqslant 2$ *et s'il existe deux sommets* x *et* y *de* F *tels que* $\delta_G(x) \leqslant 2$, $\delta_G(y) \leqslant 2$, $C(x) \to C(y)$ *et* $(xy) \notin E$ *alors* $|F| \leqslant 2d$.

Démonstration. $\delta_G(x) \leqslant 2$, $\delta_G(y) \leqslant 2$ donc chaque sommet a au plus un voisin dans une autre composante que la sienne car chaque composante est connexe et contient un nombre pair de sommets.

D'après la Propriété 3.4.4, si F contient deux sommets a et b tels que $C(a) = C(b)$ et $(ab) \in E$ alors $|F| \leqslant 2d$

On peut donc considérer que F ne contient pas deux sommets a et b tels que $C(a) = C(b)$ et $(ab) \in E$. Donc $|\Gamma(x) \cap F| \leqslant 1$ et $|\Gamma(y) \cap F| \leqslant 1$.

D'après la Propriété 3.4.3 (1 à 4) page 69 :

1. Si $x \in R$ et $y \in B$ alors $\alpha(G[F - \{x, y\}]) < d$. Donc $|(F - \{x, y\}) \cap B| \leqslant d - 1$ et $|(F - \{x, y\}) \cap R| \leqslant d - 1$. Donc $|F| \leqslant 2d$.
2. Si $x \in B$ et $y \in B$ alors $\alpha(G[F - (\{y\} \cup \Gamma(x))]) < d$. Donc $|(F - (\{y\} \cup \Gamma(x)) \cap B| \leqslant d - 1$ et $|(F - (\{y\} \cup \Gamma(x)) \cap R| \leqslant d - 1$. Donc $|F| \leqslant 2d$.
3. Si $x \in R$ et $y \in R$ alors $\alpha(G[F - (\{x\} \cup \Gamma(y))]) < d$. Donc $|(F - (\{x\} \cup \Gamma(y)) \cap B| \leqslant d - 1$ et $|(F - (\{x\} \cup \Gamma(y)) \cap R| \leqslant d - 1$. Donc $|F| \leqslant 2d$.
4. Si $x \in B$ et $y \in R$ alors $\alpha(G[F - (\Gamma(x) \cup \Gamma(y))]) < d$. Donc $|(F - (\Gamma(x) \cup \Gamma(y))) \cap B| \leqslant d - 1$ et $|(F - (\Gamma(x) \cup \Gamma(y)) \cap R| \leqslant d - 1$. Donc $|F| \leqslant 2d$.

□

La propriété suivante permet de déterminer un d-extensible de cardinal $2d-1$ dans le cas où le graphe n'est constitué que d'une seule composante. L'algorithme présenté servira ensuite pour déterminer un d-extensible de cardinal $2d-1$ pour tout graphe biparti dont les sommets sont libres.

Propriété 3.4.6. *Soit G=(B,R,E) un graphe biparti tel que G_c vérifie $|V_c| = 1$. Il existe un d-extensible F vérifiant $|F| = 2d-1$. Ce d-extensible peut être déterminé en temps polynomial par l'algorithme 2.*

Algorithm 2 Algorithme permettant de déterminer un d-extensible de cardinal $2d-1$ dans un graphe qui ne contient qu'une composante

1: Déterminer un couplage parfait M de G
2: Soit b un sommet de B
3: $F := \{b\}$
4: **while** $|F \cap B| < d$ **do**
5: Soit $x \in \Gamma(F) \cap R$ tel que $\exists\ y$ tel que $(xy) \in M$ et $y \notin F$
6: $F := F \cup \{x, y\}$
7: **end while**
8: **return** F

Démonstration. Soit F l'ensemble retourné par l'algorithme 2 page 72.

D'après la Propriété 1.2.6 page 21, G admet un couplage parfait M. Montrons que si $|F \cap B| < d$ alors il existe $x \in \Gamma(F) \cap R$ tel que $(xy) \in M$ et $y \notin F$.

Soit L l'ensemble des sommets $a \in R$ tels que $(ab) \in M$ et $b \notin F$. Si $L \cap \Gamma(F \cap B) = \varnothing$ alors $L \cup (F \cap B)$ est un stable. De plus $|L| + |(F \cap B)| = |B|$ car il reste $|L|$ sommets de B qui ne sont pas dans F. Contradiction d'après la propriété 1.2.3 page 20.

Donc il existe toujours un sommet x de $R \cap \Gamma(F)$ tel que $(xy) \in M$ et $y \notin F$.

Montrons maintenant que F est bien un d-extensible.

Montrons que F admet un unique stable de cardinal d : $F \cap B$. Soit $L \subset F \cap R$ tel que $L \neq \varnothing$. Par construction, $|\Gamma(L) \cap F| > |L|$. Donc $|((B - \Gamma(L)) \cap F) \cup L| < |(B - \Gamma(L)) \cup (\Gamma(L) \cap F)|$. Donc $|((B - \Gamma(L)) \cap F) \cup L| < d$. Donc tout stable de F qui contient L a un cardinal strictement inférieur à d. Or $F \cap B$ est un stable de F de cardinal d, c'est donc l'unique stable de cardinal d de F.

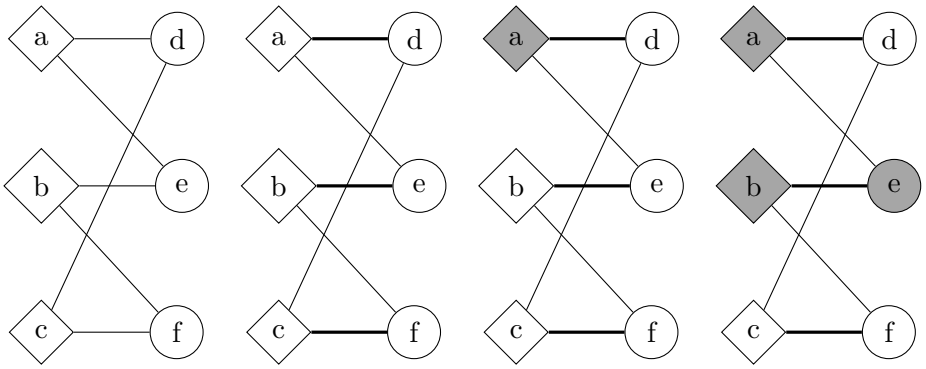


FIGURE 3.9 – Exécution de l'Algorithme 2

Le stable $F \cap B$ est extensible et c'est l'unique stable de cardinal d de F donc F est un d-extensible.

□

Exemple 3.4.1. *Appliquons l'Algorithme 2 sur le graphe de la Figure 3.9 pour construire un* 2*-extensible.*

L'algorithme commence par déterminer un couplage parfait du graphe. On considère ici $\{ad, be, cf\}$*, en gras dans la figure.*

L'algorithme sélectionne ensuite un sommet quelconque de B*, ici le sommet* a*.*

Puis l'algorithme sélectionne deux sommets $x \in R$ *et* $y \in B$ *tel que* $xy \in M$*. Dans notre exemple, cela correspond aux sommets* b *et* e*.*

L'algorithme a déterminé un ensemble de taille 3 *ce qui correspond à* $2d-1$ *pour* $d = 2$*. Il retourne donc l'ensemble des sommets sélectionnés.*

Remarque 3.4.2. *La solution obtenue par l'algorithme 2 page 72 peut être optimale mais ce n'est pas toujours le cas, comme on peut le voir dans le graphe de la Figure 3.9.*

- *Pour* $d = 1$*, un* d*-extensible optimal contient deux sommets et l'algorithme retourne un* d*-extensible constitué d'un seul sommet.*
- *Pour* $d = 2$*, un* d*-extensible optimal contient trois sommets et l'algorithme retourne un* d*-extensible de 3 sommets.*

Généralisons la Propriété 3.4.6 à l'ensemble des graphes bipartis dont tous les sommets sont libres.

Propriété 3.4.7. *Il existe un d-extensible F de G tel que $|F| = 2d - 1$. De plus, F peut être déterminé par l'algorithme 3.*

Algorithm 3 Algorithme permettant de déterminer un d-extensible de cardinal $2d - 1$ dans un graphe biparti dont tous les sommets sont libres

```
1: On considère que les composantes, notées C_i, sont classées par niveau croissant
2: F := ∅
3: i := 1
4: while |F| < 2d − 1 do
5:    if |F| + |V(C_i)| < 2d − 1 then
6:       F := F ∪ V(C_i)
7:    else
8:       Soit L l'ensemble de sommets retourné par l'application de l'algorithme 2 à C_i.
         pour obtenir un (d − |F|/2)-extensible de C_i
9:       F := F ∪ L
10:   end if
11:   i :=i+1
12: end while
13: return F
```

Démonstration. Soit F un ensemble de sommets retourné par l'Algorithme 3 page 74.

L'ensemble F contient d sommets de B et $d - 1$ sommets de R. Soit S un stable de $G[F]$ tel que $|S| = d$. L'algorithme considère les composantes une par une et sélectionne des sommets tant que c'est possible avant de passer à la composante suivante. Il existe donc une unique composante C_i telle que $V(C_i) \cap F \neq \emptyset$ et $V(C_i)$ n'est pas inclus dans F. Dans la composante C_i, $S \cap V(C_i) = B \cap V(C_i)$. Pour les autres composantes C_j, soit $V(C_j) \cap S = V(C_j) \cap B$, soit $V(C_j) \cap S = V(C_j) \cap R$. De plus, pour tout couple de composantes (C_i, C_j) tel que $C_i \to C_j$ dans G_c, $C_i \to C_j$ dans $G[F]_c$. Donc S est extensible par rapport à F car il peut être étendu en un stable optimal du graphe en ajoutant tous les sommets de B qui n'appartiennent pas à F. Donc F est bien un d-extensible.

□

Dans la suite de la section, des cas particuliers dans lesquels déterminer un d-extensible optimal peut se faire en temps polynomial sont étudiés.

Propriété 3.4.8. *Soit $G = (B, R, E)$ un graphe dont tous les sommets sont libres. Si G_c est un arbre, alors trouver un 1-extensible optimal de G est un problème polynomial.*

Démonstration. D'après la Propriété 3.4.3, un ensemble de deux sommets voisins forme un 1-extensible. On cherche donc à déterminer s'il existe un 1-extensible de cardinal strictement supérieur à 2.

Soit F un 1-extensible tel que $|F| > 2$. Soient x et y deux sommets de F tels que $C(x) \to C(y)$.

— Si $x \in R$ and $y \in B$ alors d'après la Propriété 3.4.3 page 69, $|F| \leqslant 2$. Contradiction.

— Si $x \in B$ et $y \in B$ alors d'après la Propriété 3.4.1 page 64, le stable $\{x\}$ de cardinal 1 n'est pas extensible par rapport à F. Contradiction.

— Si $x \in R$ et $y \in R$ alors d'après la Propriété 3.4.1, le stable $\{y\}$ de cardinal 1 n'est pas extensible par rapport à F. Contradiction.

— Si $x \in B$, $y \in R$ et $C(x) = C(y)$ alors d'après la Propriété 3.4.3 $|F| = 2$. Contradiction.

Ainsi, pour chaque chemin du graphe G, un 1-extensible de cardinal strictement supérieur à 2 ne peut contenir plus d'un sommet de B et d'un sommet de R. Pour déterminer un 1-extensible optimal, on cherche à sélectionner le maximum de sommets tel que, sur chaque chemin de G_c, le 1-extensible contienne au plus 2 sommets. On peut donc modéliser la recherche d'un 1-extensible optimal par le programme linéaire suivant :

$$\begin{cases} \max\limits_{x} & \sum\limits_{i=1}^{n} x_i \\ & \sum\limits_{i \in P} x_i \leqslant 2 \quad \forall P \text{ chemin maximal de } G_c \\ & x \in \{0,1\}^n \end{cases}$$

— Les variables x_i valent 1 si le 1-extensible contient un sommet de la composante i et 0 sinon

— On cherche à sélectionner le plus de sommets possible

— Sur chaque chemin maximal (au sens de l'inclusion) de G_c, on ne peut pas sélectionner plus de deux sommets

— Si $x_i = 1$ et il n'existe pas de j tels que $x_j = 1$ et $C_j \rightarrow C_i$ alors le 1-extensible contient un sommet de B de la composante C_i. Sinon, il contient un sommet de R de la composante C_i.

D'après la Propriété 3.4.3 page 69, l'ensemble obtenu est bien un 1-extensible. S'il est de cardinal supérieur à 2, alors il est optimal. En effet, tout ensemble qui contiendrait plus de sommets ne serait pas un 1-extensible car il contiendrait aussi trois sommets dont les composantes seraient sur un même chemin dans G_c.

La matrice des chemins dans un arbre orienté est totalement unimodulaire et déterminer les chemins maximaux d'un arbre est polynomial. Résoudre le programme mathématique ci-dessus est donc polynomial. Déterminer un 1-extensible d'un graphe biparti dont le graphe des composantes est un arbre se fait donc en temps polynomial.

□

Nous allons maintenant démontrer que déterminer un d-extensible optimal dans une grille est polynomial. Une grille est un graphe défini par deux entiers m et n. L'entier m est le nombre de colonnes de la grille et l'entier n le nombre de lignes. On note v_{ij} le sommet de la ligne i et de la colonne j. Une grille $G_{m,n} = (V, E)$ est un graphe tel que $V = \{v_{ij}, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$ et $E = \{v_{ij}v_{(i+1)j}, 1 \leqslant i \leqslant m-1, 1 \leqslant j \leqslant n\} \cup \{v_{ij}v_{i(j+1)}, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n-1\}$. Par exemple, la Figure 3.10 représente la grille $G_{4,3}$. Une grille est un graphe biparti.

Si les nombres de colonnes et de lignes d'une grille sont impairs, alors le nombre de sommets le la grille est impair. Il n'existe donc pas de couplage parfait donc d'après la propriété 1.2.6, tous les sommets de la grille ne sont pas libres. Si le nombre de colonnes ou le nombre de lignes d'une grille G est pair, alors d'après [68], toutes les arêtes de G appartiennent à au moins un couplage parfait. Donc le graphe des composantes de G ne contient qu'un sommet et un stable optimal de G est de cardinal $|B| = |R|$. Les ensembles B et R sont des stables optimaux donc tous les sommets de G sont libres.

Propriété 3.4.9. *Soit $G = (B, R, E)$ une grille dont tous les sommets sont libres. Déterminer un d-extensible optimal d'une grille est un problème polynomial.*

Démonstration. Le graphe des composantes de G n'est constitué que d'un sommet car toutes les arêtes de G appartiennent à au moins un couplage parfait. D'après la Propriété 3.4.3 page 69 le cardinal d'un d-extensible de G ne peut dépasser $2d$.

L'algorithme 4 permet de construire un d-extensible F de cardinal $2d$ dans une grille dont tous les sommets sont libres. On suppose que le nombre de colonnes est pair. Cet algorithme commence par sélectionner les deux sommets voisins v_{11} et v_{21}. Puis, tant que l'ensemble F est de cardinal inférieur à $2d$, l'algorithme ajoute à F des sommets voisins des lignes 1 et 2 de la deuxième colonne, puis de la troisième et ainsi de suite. Une fois que tous les sommets des deux premières lignes ont été sélectionnés, l'algorithme sélectionne des sommets de la troisième ligne sur des colonnes voisines de sorte que tous les sommets d'indice inférieur sur la ligne sont sélectionnés. L'algorithme procède ainsi pour chaque ligne tant que l'ensemble F est de cardinal inférieur à $2d$. Il s'arrête quand F est de cardinal $2d$ ou quand il ne reste plus de sommets à sélectionner.

Prouvons que l'algorithme 4 retourne bien un d-extensible. Pour cela, considérons F un ensemble de sommets retourné par l'algorithme 4 et montrons que ses stables optimaux sont $F \cap B$ et $F \cap R$. L'ensemble F contient d sommets de B et d sommets de R. Si $d = 1$, F ne contient que deux sommets voisins donc la propriété est vérifiée. Supposons que $d > 1$. Soit L un ensemble de l sommets de $B \cap F$ avec $l > 0$. L'ensemble L a au moins $l + 1$ voisins dans $F \cap R$. En effet, chaque sommet est voisin du sommet avec lequel il a été ajouté et d'un sommet qui appartenait déjà F lors de sa sélection. Donc tout stable de $G[F]$ qui contient L et des sommets de $F \cap R$ est au plus de cardinal $d - 1$. Donc les stables optimaux de $G[F]$ sont $F \cap B$ et $F \cap R$. Ils sont de cardinal d et il est évident qu'ils sont extensibles. Donc F est bien un d-extensible. □

Exemple 3.4.2. *Considérons le graphe de la Figure 3.10 et construisons un* 5*-extensible. L'algorithme 4 commence par sélectionner deux sommets voisins quelconques. Ici, comme on peut le voir sur la figure, on sélectionne les sommets* a *et* b*. On cherche ensuite deux sommets voisins qui forment un cycle de longueur* 4 *avec les sommets sélectionnés. Dans notre exemple, il n'y a qu'une possibilité, on sélectionne donc les sommets* e *et* f*. On recommence la même recherche et on trouve cette fois deux paires de sommets possibles :*

Algorithm 4 Algorithme permettant de déterminer un d-extensible optimal dans une grille dont tous les sommets sont libres

```
1: Supposons que le nombre de colonnes m est pair
2: F := ∅
3: j := 1
4: while |F| < 2d ET j ⩽ m do
5:    F := F ∪ {v_1j, v_2j}
6:    j := j + 1
7: end while
8: i := 3
9: while |F| < 2d do
10:   j := 1
11:   while |F| < 2d ET j < m do
12:      F := F ∪ {v_ij, v_i(j+1)}
13:      j := j + 2
14:   end while
15:   i := i + 1
16: end while
17: return F
```

$\{i, j\}$ *et* $\{c, g\}$*. On sélectionne une des deux paires et on recommence l'itération jusqu'à obtenir un* 5*-extensible de* 10 *sommets.*

Propriété 3.4.10. *Soit $G = (B, R, E)$ un cycle pair. Il est possible de déterminer un d-extensible de G en temps polynomial.*

Démonstration. Dans un cycle pair, $\alpha(G) = |B| = |R|$. Toutes les arêtes du cycle G appartiennent à au moins un couplage parfait. Le graphe des composantes de G ne contient donc qu'une composante. Ainsi, un d-extensible de G ne peut contenir plus de $2d$ sommets d'après la Propriété 3.4.3 page 69.

Si $d = 1$, un d-extensible peut être construit en sélectionnant deux sommets voisins. C'est un d-extensible optimal car de cardinal 2.

Considérons le cas $1 < d < \alpha(G)$. D'après la Propriété 3.4.7 page 74, G admet un d-extensible de cardinal $2d-1$. Soit F un ensemble de sommets de cardinal $2d$. Il contient donc d sommets de B et d sommets de R. Or G est un cycle pair et F ne contient pas tous les sommets de G donc il existe $x \in F \cap B$ tel que $|\Gamma(x) \cap F| < 2$. Supposons que x ait un voisin y dans F. L'ensemble $((F \cap R) - \{y\}) \cup \{x\}$ est un stable de cardinal d de $G[F]$

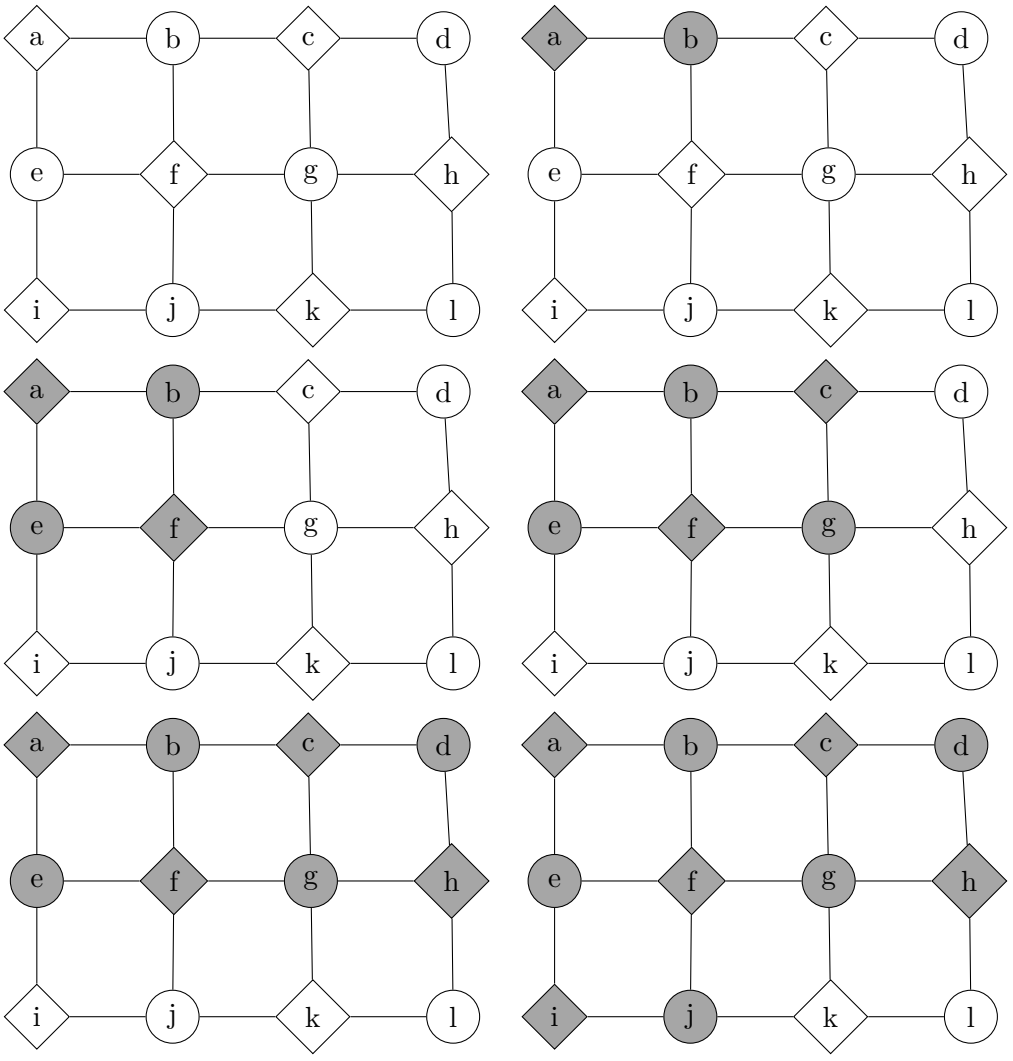


FIGURE 3.10 – Exécution de l'Algorithme 4. Ici, $\alpha(G) = 6$

qui d'après la Propriété 3.4.1 page 64 n'est pas extensible par rapport à F. De même, si $|\Gamma(x) \cap F| = 0$, l'ensemble $(F \cap R) \cup \{x\}$ est un stable de cardinal $d+1$ de $G[F]$ qui n'est pas extensible par rapport à F. Donc F n'est pas un d-extensible. Donc un d-extensible optimal de G peut être déterminé par l'Algorithme 3 page 3. □

Dans la propriété suivante, on s'intéresse au cas des cycles impairs, bien qu'ils ne soient pas bipartis. Les propriété structurelles sur les d-extensibles démontrées dans cette section ne s'appliquent donc pas dans ce cas.

Propriété 3.4.11. *La recherche d'un d-extensible optimal s'effectue en temps polynomial dans le cas où $G = (V, E)$ est un cycle impair.*

- *Si $1 < d < \alpha(G)$ un d-extensible optimal contient $2d$ sommets et peut être obtenu en sélectionnant une chaîne de $2d$ sommets du cycle.*
- *Si $d = 1$, un 1-extensible optimal du graphe contient 3 sommets et peut être obtenu en sélectionnant une chaîne de 3 sommets.*

Démonstration. Soit $G = (V, E)$ un cycle impair de longueur $2k+1$. Dans un cycle impair, $\alpha(G) = \frac{|V|-1}{2} = k$.

Considérons le cas $1 < d < \alpha(G)$. Soit F une chaîne de $2d$ sommets. Tout stable de cardinal d de F contient au moins une des extrémités de F. Le sous-graphe $G[V - F]$ est une chaîne de $2(k-d)+1$ sommets, donc une chaîne de longueur impaire. Si on retire les sommets voisins des extrémités de F, on obtient une chaîne de $2(k-d-1)+1$ sommets, dont le stable S de cardinal maximal forme un stable avec tout sous-ensemble stable de F et est de cardinal $k-d$. Donc tout stable de cardinal d de $G[F]$ peut être complété en un stable de cardinal maximal du graphe en ajoutant les sommets du stable S. Ainsi, F est bien un d-extensible.

Considérons maintenant un ensemble $\hat{F} \subset V$ de $2d+1$ sommets et montrons que ce n'est pas un d-extensible. Supposons tout d'abord que $G[\hat{F}]$ est une chaîne. Dans ce cas, $G[V - \hat{F}]$ est une chaîne paire. Soit $\hat{S}$ un stable de cardinal d de $G[\hat{F}]$ qui contient les deux extrémités de la chaîne. Pour compléter le stable $\hat{S}$, on ne peut sélectionner les sommets des extrémités de $G[V - \hat{F}]$. Donc on ne peut sélectionner que des sommets d'une chaîne

paire de longueur $2(k-d-1)$. Or, le cardinal maximal d'un stable d'une chaîne de longueur $2(k-d-1)$ est $k-d-1$ donc on ne peut pas compléter le stable $\hat{S}$ en un stable optimal du graphe à l'aide d'un stable de $G[V-\hat{F}]$. Donc $\hat{F}$ n'est pas un d-extensible.

Supposons maintenant que $G[\hat{F}]$ n'est pas une chaîne. Dans ce cas, les sommets de $\hat{F}$ sont séparés par i chaînes avec $i \leqslant 2d+1$. La somme du nombre de sommets des i chaînes est paire car $\hat{F}$ contient un nombre impair de sommets. On note u_j ces chaînes. Soit v le nombre de chaînes u_j dont le nombre de sommets est impair. Nécessairement, v est pair donc $v \leqslant d$. Si $v = 0$, alors le cardinal maximal d'un stable de $G[V-\hat{F}]$ est $k-d$. Soit $\hat{S}$ un stable de cardinal d de $G[\hat{F}]$ qui contient les deux extrémités voisines de celles d'une des chaînes u_j. Dans ce cas, il n'est possible de construire un stable de cardinal $k-d$ de $G[V-\hat{F}]$ qui soit aussi un stable avec $\hat{S}$. Donc $\hat{F}$ n'est pas un d-extensible. Supposons maintenant que $v > 0$. On suppose que les chaînes u_j dont le nombre de sommets est impair sont numérotées de 1 à v. Dans ce cas le cardinal maximal d'un stable de $G[V-\hat{F}]$ est $\sum_{j=1}^{v} \frac{|u_j|+1}{2} + \sum_{j=v+1}^{i} \frac{|u_j|}{2} = \sum_{j=1}^{i} \frac{|u_j|}{2} + \frac{v}{2}$. Or, $\sum_{j=1}^{i} \frac{|u_j|}{2} = \frac{2k+1-2d-1}{2} = k-d$. Donc $\alpha(G[V-\hat{F}]) = k-d+\frac{v}{2}$. Ainsi, en construisant un stable $\hat{S}$ de $G[\hat{F}]$ tel que les extrémités d'au moins $\frac{v}{2}+1$ chaînes u_j telles que $1 \leqslant j \leqslant v$ ont pour voisins des sommets de $\hat{S}$, on est assuré que $\hat{S}$ n'est pas extensible. Montrons que l'on peut toujours construire un tel stable $\hat{S}$.

Supposons tout d'abord que $v = 2d$. Dans ce cas, il existe au plus deux sommets de $G[\hat{F}]$ qui ne sont pas isolés. Donc il existe un sommet isolé de $G[\hat{F}]$ dont les voisins appartiennent à des chaînes u_j dont le nombre de sommets est impair. En sélectionnant ce sommet puis $d-1$ autres sommets voisins de chaînes u_j dont le nombre de sommets est impair, on construit un stable $\hat{S}$ de cardinal d tel qu'il existe $\frac{v}{2}+1 = d+1$ chaînes u_j telle que $1 \leqslant j \leqslant v$ qui ont pour voisins des sommets de $\hat{S}$. Donc le stable $\hat{S}$ n'est pas extensible.

Supposons maintenant que $v < 2d$. On sait que v est pair donc $v \leqslant 2d-2$. Dans ce cas, $\frac{v}{2}+1 = d$. On peut donc construire un stable $\hat{S}$ en sélectionnant d sommets de F voisins d'extrémités de d chaînes u_j telle que $1 \leqslant j \leqslant v$. Ainsi le stable $\hat{S}$ n'est pas extensible.

Considérons enfin le cas $d = 1$. Considérons une chaîne de 3 sommets et montrons que

c'est un 1-extensible. Dans ce cas, le reste du graphe est constitué d'une chaîne paire de longueur $2k-2$ donc elle admet des stable optimaux (de cardinal $k-1$) qui ne contiennent qu'une des deux extrémités de la chaîne. Ainsi, quel que soit le sommet de la chaîne de 3 sommets sélectionné, il est possible de le compléter en un stable optimal du graphe à l'aide d'un des stables optimaux de la chaîne de longueur paire.

Montrons maintenant qu'il ne peut pas exister de 1-extensible de cardinal 4. Considérons un ensemble $F = \{x, y, z, t\}$ de quatre sommets et montrons que ce n'est pas un 1-extensible.

— Si $G[F]$ est une chaîne, alors $G[V - F]$ est une chaîne de longueur impaire dont l'unique stable optimal est de cardinal $k - 2$ et contient les deux extrémités de la chaîne. Donc en sélectionnant un sommet dont un voisin est une extrémité de la chaîne $G[V - F]$, il n'est pas possible de construire un stable dans $G[V - F]$ de cardinal supérieur à $k - 2$ donc F n'est pas un d-extensible car un de ses stables de cardinal 1 n'est pas extensible par rapport à F.

— Si $G[F]$ est constitué de deux composantes connexes alors $G[V - F]$ est constitué d'une chaîne de longueur paire et d'une chaîne de longueur impaire. De la même façon qu'au point précédent, un stable qui contient un sommet de F dont un voisin est une extrémité d'une chaîne impaire n'est pas extensible par rapport à F.

— Si $G[F]$ est constitué de trois composantes connexes, $G[V-F]$ est constitué de trois chaînes dont au moins une est impaire. Si une seule est impaire, on sélectionne un sommet voisin d'une extrémité de cette chaîne et on obtient comme dans les points précédents un stable de cardinal 1 qui n'est pas extensible. Sinon, les trois chaînes sont impaires et il existe un sommet de F dont les deux voisins sont des extrémités de chaînes impaires. On sélectionne ce sommet et, comme précédemment, on obtient un stable de cardinal 1 qui n'est pas extensible par rapport à F.

— Si $G[F]$ est constitué de quatre composantes connexes $G[V - F]$ est constitué de quatre chaînes dont au moins une contient un nombre impair de sommets. S'il n'y en a qu'une, on sélectionne un sommet de F voisin d'une de ses extrémités comme précédemment. Sinon, trois chaînes sont impaires et il existe un sommet de F dont les voisins sont les extrémités de deux chaînes impaires. On sélectionne ce sommet

et, comme précédemment, on obtient un stable de cardinal 1 qui n'est pas extensible par rapport à F.

□

Nous avons dans cette section, dans le cas des graphes bipartis dont tous les sommets sont libres, établi une caractérisation des d-extensibles et donné une borne inférieure du cardinal maximal d'un d-extensible. Nous avons également résolu le cas de la recherche d'un 1-extensible dans le cas où le graphe des composantes est un arbre et le cas de la recherche d'un d-extensible si le graphe est une grille ou un cycle. Nous allons maintenant nous intéresser au cas des graphes bipartis qui contiennent des sommets forcés.

3.5 Cas des graphes bipartis qui contiennent des sommets forcés

Dans cette section, on considère un graphe biparti $G(B, R, E)$ qui contient des sommets libres et des sommets forcés. Rappelons que les sommets exclus ont été retirés. Les sommets forcés sont donc isolés.

Propriété 3.5.1. *Soit F un d-extensible de G. Si $F \cap \Xi(G) \neq \varnothing$ alors $|F| \leqslant 2d - |F \cap \Xi(G)|$.*

Démonstration. D'après la Propriété 3.2.1 page 60, tout stable S de $G[F]$ tel que $|S| = d$ vérifie $F \cap \Xi(G) \subseteq S$. Considérons la tri-partition de F $\{F \cap \Xi(G), F \cap B - \Xi(G), F \cap R - \Xi(G)\}$. Puisque $F \cap B - \Xi(G)$ est un stable de $G[F]$, $|F \cap B - \Xi(G)| \leqslant d - |F \cap \Xi(G)|$. De même, $|F \cap R - \Xi(G)| \leqslant d - |F \cap \Xi(G)|$. Donc $|F| \leqslant 2d - |F \cap \Xi(G)|$. □

D'après la Propriété 3.4.7 page 74, on sait qu'il existe un d-extensible de G de cardinal $2d - 1$. On en déduit le corollaire suivant :

Corollaire 3.5.1. *Si $d \leqslant \frac{\phi(G)}{2}$ alors il existe un d-extensible optimal F de G tel que $F \subset \Phi(G)$.*

Démonstration. Supposons que $d \leqslant \frac{\phi(G)}{2}$. Dans ce cas, d'après la Propriété 3.4.7 page 74, il existe un d-extensible de G de cardinal $2d - 1$.

Soit F un d-extensible de G tel que $F \cap \Xi(G) = l$ avec $l \geqslant 1$. D'après la Propriété 3.5.1, $|F| \leqslant 2d - |F \cap \Xi(G)|$. Donc $|F| \leqslant 2d - 1$. Donc il existe un d-extensible de G qui ne contient que des sommets libres de cardinal supérieur ou égal à celui de F. Donc il existe un d-extensible optimal qui ne contient que des sommets libres. □

Propriété 3.5.2. *Si $d \geqslant \frac{\phi(G)}{2}$ alors il existe un d-extensible optimal F^* de G est formé de tous les sommets libres de G et de $(d - \frac{\phi(G)}{2})$ sommets forcés. Ainsi $|F^*| = d + \frac{\phi(G)}{2}$*

Démonstration. Tout stable de cardinal d de $G[\hat{F}]$ contient $\frac{\phi(G)}{2}$ sommets libres et $(d - \frac{\phi(G)}{2})$ sommets forcés. Il peut donc être complété en un stable optimal du graphe en ajoutant les sommets de $\Xi(G) - F^*$. Ainsi, F^* est bien un d-extensible.

Soit $\hat{F} \subset B \cup R$ tel que $|\hat{F}| \geqslant 1 + d + \frac{\phi(G)}{2}$. Comme $\hat{F}$ contient au moins $d+1$ sommets forcés alors $G[\hat{F}]$ contient un stable S formé de d sommets forcés et il existe $x \in \Xi(G) \cap \hat{F}$ tel que $x \notin S$. D'après la Propriété 3.2.1 page 60, $\hat{F}$ n'est pas un d-extensible.

□

Ainsi, pour $d \geqslant \frac{\phi(G)}{2}$, déterminer un d-extensible optimal d'un graphe biparti se fait en temps polynomial car déterminer les sommets libres et forcés se fait en temps polynomial. Dans la suite du chapitre, on étudiera des graphes où tous les sommets sont libres et où $d < \frac{\phi(G)}{2}$.

3.6 d-extensibles de stables dans les arbres

Dans cette section, on considère un arbre $G = (B, R, E)$ dont tous les sommets sont libres, donc $\alpha(G) = |B| = |R|$. Dans ce cas, G_c est un arbre orienté et les composantes C_i de G contiennent deux sommets : un sommet de B et un sommet de R. D'après la Propriété 3.4.8 page 75, déterminer un 1-extensible optimal d'un arbre se fait en temps polynomial car les graphes des composantes des arbres sont des arbres.

Cette section donne une borne inférieure de la cardinalité maximale d'un d-extensible d'un arbre puis étudie deux cas particuliers. Ensuite, une sous-section est consacrée à une famille d'arbres : les homards.

La propriété suivante fournit une meilleure borne inférieure de la cardinalité d'un d-extensible que celle de la Propriété 3.4.7 page 74. La borne est obtenue à l'aide d'un algorithme similaire à l'Algorithme 3.

Propriété 3.6.1. *Tout d-extensible optimal F^* de G vérifie $|F^*| \geqslant 2d$*

Algorithm 5 Algorithme permettant de déterminer un d-extensible de cardinal $2d$ dans un arbre dont tous les sommets sont libres

```
1: On considère que les composantes, notées C_i, sont classées par niveau croissant
2: F := ∅
3: i := 1
4: while |F| < 2d do
5:     F := F ∪ V(C_i)
6:     i :=i+1
7: end while
8: return F
```

Démonstration. Montrons que l'algorithme 5 retourne un d-extensible F de cardinal $2d$.

Classons les composantes en niveaux dans G_c, puis numérotons les composantes par niveau croissant. L'algorithme sélectionne les sommets des d premières composantes. Chaque composante n'a que deux sommets donc $|F| = 2d$. De plus, F respecte les conditions de la Propriété 3.4.3 page 69 donc F est un d-extensible.

□

Puisque qu'il existe d-extensible de cardinal $2d$ dans un arbre, si ce d-extensible n'est pas optimal alors il existe un d-extensible de cardinal strictement supérieur à $2d$. La propriété suivante détaille la structure que peut avoir un d-extensible de cardinal strictement supérieur à $2d$.

Propriété 3.6.2. *Soit F un d-extensible de G tel que $|F| > 2d$.*

- *Pour toute composante C de G, $|V(C) \cap F| \leqslant 1$.*
- *Soient x, y deux sommets de F. Si $C(x) \to C(y)$, $C(x) \neq C(y)$ et si $x \in R$ alors $y \notin B$.*

Démonstration. Soit F un d-extensible de G tel que $|F| > 2d$.

Montrons que pour toute composante C de G, $|V(C) \cap F| \leqslant 1$. Supposons par l'absurde qu'il existe une composante telle que F contient ses deux sommets $x \in B$ et $y \in R$. Dans ce cas, d'après la Propriété 3.4.1 page 64, tout stable de cardinal d de $G[F]$ doit contenir un des deux sommets. Donc si $|F \cap B| > d$ il existe un stable de $G[F]$ qui ne contient que des sommets de B mais qui ne contient ni x, ni y donc qui n'est pas extensible par rapport à F. Donc $|F \cap B| \leqslant d$. On démontre de la même façon que $|F \cap R| \leqslant d$. Mais cela implique que $|F| \leqslant 2d$. Contradiction.

Montrons que pour tous sommets x et y de F, si $C(x) \to C(y)$ et si $x \in R$ alors $y \notin B$. Supposons par l'absurde que $x \in R$ alors $y \in B$. Dans ce cas, d'après la Propriété 3.4.1 page 64, tout stable de cardinal d de $G[F]$ doit contenir un des deux sommets. On démontre de la même façon qu'au cas précédent que $|F| \leqslant 2d$. Contradiction.

□

La Propriété 3.4.2 page 67 montrait qu'un d-extensible était aussi un k-extensible pour $k > d$ et $d > 1$. La propriété suivante montre que des 1-extensibles peuvent aussi être des k-extensibles pour $k > 1$ si le graphe des composantes de l'arbre est une arborescence.

Propriété 3.6.3. *Si G_c est une arborescence, alors il existe un 1-extensible optimal qui est aussi un k-extensible pour $k > 1$.*

Démonstration. Soit $\hat{F}$ un 1-extensible qui n'est pas un k-extensible pour $k > 1$. Montrons qu'il est possible de construire un 1-extensible F de même cardinal que $\hat{F}$ qui soit aussi un k-extensible.

L'ensemble $\hat{F}$ n'est pas un k-extensible donc un des points de la Propriété 3.4.1 page 64 n'est pas respecté pour un stable de cardinal k. Le seul point qui peut être respecté pour un stable de cardinal 1 et pas de cardinal k est le quatrième point de la propriété. Donc il existe deux sommets $x \in B \cap \hat{F}$ et $y \in R \cap \hat{F}$ tels que $C(x) \to C(y)$, $C(x) \neq C(y)$ et $xy \notin E$.

Le graphe G_c est une arborescence donc pour tout sommet $z \in B$ distinct de x tel que $C(z) \to C(y)$, soit $C(z) \to C(x)$, soit $C(x) \to C(z)$. Si $z \in \hat{F}$ et $C(z) \to C(x)$, alors $\{z\}$ doit être extensible par rapport à $\hat{F}$ puisque $\hat{F}$ est un 1-extensible. Mais d'après le premier

point de la Propriété 3.4.1 page 64, pour être extensible par rapport à $\hat{F}$ il faut que $x \in \{z\}$ ce qui n'est pas le cas. Donc il n'est pas possible que $z \in \hat{F}$ et $C(z) \to C(x)$. On démontre de la même façon que le cas $z \in \hat{F}$ et $C(x) \to C(z)$ est impossible. Donc pour tout sommet $y \in R \cap \hat{F}$, il existe au plus un sommet $x \in B \cap \hat{F}$ tel que $C(x) \to C(y)$.

Construisons l'ensemble F selon les instructions suivantes :

— $F := F \cup (\hat{F} \cap R)$

— Pour tout $x \in B \cap \hat{F}$ tel qu'il n'existe pas de sommet $y \in R$ tel que $C(x) \to C(y)$: $F := \{x\} \cup F$

— Pour les autres sommets $x \in B \cap \hat{F}$, soit $y \in R \cap \hat{F}$ tel que $C(y)$ est de plus haut niveau possible. Soit $z \in B$ tel que $C(z) \neq C(y)$ et $(zy) \in E$: $F := \{z\} \cup F$

L'ensemble F vérifie les conditions de la Propriété 3.4.3 page 69, c'est donc un 1-extensible et c'est aussi un k-extensible. De plus $|F| = |\hat{F}|$, donc il existe un 1-extensible optimal qui est aussi un k-extensible. □

Le propriété suivante montre que si le graphe des composantes est une arborescence alors il est possible de déterminer un d-extensible optimal en temps polynomial.

Propriété 3.6.4. *Si G_c est une arborescence alors il est possible de déterminer un d-extensible optimal en $O(n)$.*

Démonstration. Soit F_1 un ensemble de sommets retourné par l'Algorithme 6 page 88. Comme F_1 respecte les conditions de la Propriété 3.4.3 page 69, c'est un 1-extensible.

Montrons que F_1 est un 1-extensible optimal.

Soit $\hat{F_1}$ un 1-extensible tel que $\exists\ x \in \hat{F_1}$ et $x \notin F_1$.

Supposons qu'il existe $y \in \hat{F_1}$ tel que $C(x) \to C(y)$. D'après la preuve de la Propriété 3.6.3 page 86 il existe un 1-extensible de même cardinal que $\hat{F_1}$ tel que si deux sommets u et v de ce 1-extensible sont tels que $C(u) \to C(v)$ alors ils sont voisins. On peut donc supposer que x et y sont voisins. Puisque $x \notin F_1$ alors $\exists\ a, b \in F_1$ tels que $C(x) \to C(y) \to C(a) \to C(b)$. Or $(\hat{F_1} - \{x, y\}) \cup \{a, b\}$ est un 1-extensible de même cardinal que $\hat{F_1}$. Donc $|\hat{F_1}| \leqslant |F_1|$.

Algorithm 6 Algorithme permettant de déterminer un 1-extensible optimal dans un arbre dont le graphe des composantes est une arborescence

1: $F = \varnothing$
2: L = ensemble des composantes du graphe triées par niveau décroissant dans le graphe des composantes
3: **while** $L \neq \varnothing$ **do**
4: C composante en tête de L
5: $L := L - C$
6: **if** C n'a pas de successeurs dans G_C **then**
7: $F := F \cup (V(C) \cap R)$
8: **else**
9: $F := F \cup (V(C) \cap B)$
10: Retirer de L les composantes X telles que $X \rightarrow C$
11: **end if**
12: **end while**
13: **return** F

Algorithm 7 Algorithme permettant de déterminer un d-extensible optimal dans un arbre dont le graphe des composantes est une arborescence pour $d > 1$

1: F = Ensemble retourné par l'application de l'algorithme 6
2: H = ensemble des sommets x tels que $(\Gamma^+(x) \cap F) \neq \varnothing$
3: **while** $|B \cap F| \leqslant d - 1$ ou $H \neq \varnothing$ **do**
4: $x \in H$
5: $H := H - \{x\}$
6: **if** $|(F \cap R) - \Gamma^+(x)| \leqslant d - 2$ **then**
7: $F := F \cup \{x\}$
8: **end if**
9: **if** $|F| < 2d$ **then**
10: $F :=$ d-extensible de cardinal $2d$ déterminé à l'aide de l'Algorithme 5 page 85
11: **end if**
12: **end while**
13: **return** F

De même, s'il n'existe pas de sommets $y \in \hat{F_1}$ tel que $C(x) \rightarrow C(y)$ alors il existe $a, b \in F_1$ tels que $C(x) \rightarrow C(a) \rightarrow C(b)$ et $(\hat{F_1} - \{x\}) \cup \{a, b\}$ est un 1-extensible de cardinal supérieur à $\hat{F_1}$. Donc $|\hat{F_1}| \leqslant |F_1|$. Donc F_1 est optimal.

Pour le cas $d > 1$, considérons F un ensemble de sommets retourné par l'Algorithme 7 page 88. F respecte les conditions de la Propriété 3.4.3 page 69, c'est donc un d-extensible. Montrons qu'il est optimal.

Soit H un d-extensible tel que $|H| > 2d$. Soit $x \in H$ tel que $x \notin F_1$ et $\forall y \in H$, $C(y) \notin \Gamma^+_{G_c}(C(x))$.

Supposons que $x \in B$. D'après la Propriété 3.4.3 page 69, $\nexists y \in H$ tel que $C(x) \rightarrow C(y)$. Considérons l'ensemble $\hat{H} = (H - \{x\}) \cup \{y | y \in V^z \cap B, C(z) \in \Gamma^+_{G_c}(C(x))\}$. D'après la Propriété 3.4.3 page 69, $\hat{H}$ est un d-extensible et $|\hat{H}| \geqslant |H|$ donc H n'est pas un d-extensible optimal. De la même façon, on démontre que le cas $x \in R \cap H$ tel que $x \notin F_1$ n'est pas possible.

□

Exemple 3.6.1. *Considérons le graphe de la Figure 3.11 et déterminons un* 4*-extensible optimal. Le graphe des composantes est bien une arborescence. La racine est la composante qui contient les sommets a et b. On commence par déterminer un* 1*-extensible optimal à l'aide de l'Algorithme 6. On sélectionne ainsi les sommets de R des composantes sans successeurs, c'est-à-dire les sommets i, x, o, m, k, q, s et u.*

Puis on considère les sommets de B voisins des sommets sélectionnés et qui ne sont pas dans la même composante. Pour chacun de ces sommets $x \in B$, s'il n'existe pas de sommet $y \in R$ sélectionné tels que $C^x \rightarrow C^y$ et $xy \notin E$, alors le sommet est sélectionné dans le 1*-extensible. Dans l'exemple, seul le sommet h correspond aux critères et peut être sélectionné. Le* 1*-extensible déterminé par l'algorithme est indiqué dans la Figure 3.12 où les sommets sélectionnés sont en gris.*

Enfin, avec l'Algorithme 7, on cherche à déterminer un 4*-extensible optimal en complétant le* 1*-extensible F précédent. On considère donc les sommets de $x \in B$ voisins de sommets de R du* 1*-extensible. Pour chacun de ces sommets x, si $|(F \cap R) - \Gamma^+(x)| \leqslant d-2$ on l'ajoute au d-extensible. Dans l'exemple, seul le sommet b est ajouté. Le* 4*-extensible dé-*

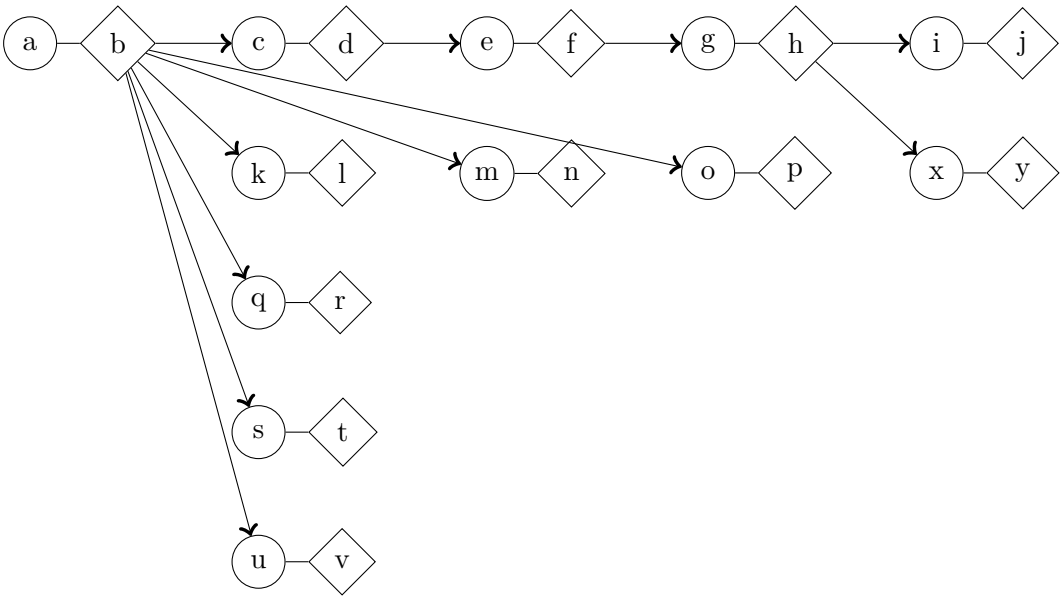


FIGURE 3.11 – Arbre dont le graphe des composantes est une arborescence. Dans ce graphe $\alpha(G) = 12$

terminé par l'algorithme est indiqué dans la Figure 3.13 où les sommets sélectionnés sont en gris. Ce 4*-extensible est de cardinal* 10*. On compare ce cardinal à celui du d-extensible de cardinal* $2d = 8$ *qui est déterminé par la Propriété 3.6.1 page 85. On constate que* $10 > 8$ *donc l'algorithme retourne le* 4*-extensible qu'il a déterminé. D'après la Propriété 3.6.4, c'est un* 4*-extensible optimal.*

La propriété suivante utilise le résultat précédent pour déterminer des d-extensibles dans les arbres qui contiennent une composante C telle que pour toute autre composante $\hat{C}$ de G, soit $C \to \hat{C}$, soit $\hat{C} \to C$. La Figure 3.14 est un exemple de ces graphes où la composante C est la composante qui contient les sommets e et f.

Propriété 3.6.5. *Soit* $G = (B, R, E)$ *un arbre dont tous les sommets sont libres et qui contient une composante* C *telle que pour toute autre composante* $\hat{C}$ *de* G*, soit* $C \to \hat{C}$*, soit* $\hat{C} \to C$*. Il est possible de déterminer un d-extensible optimal de* G *en temps polynomial.*

Démonstration. Soit G_1 le sous-graphe de G constitué de toutes les composantes $\hat{C}$ telles que $\hat{C} \to C$ et soit G_2 le sous-graphe de G constitué de toutes les composantes $\hat{C}$ telles que $C \to \hat{C}$. Le graphe des composantes de G_1 est une anti-arborescence et le graphe des

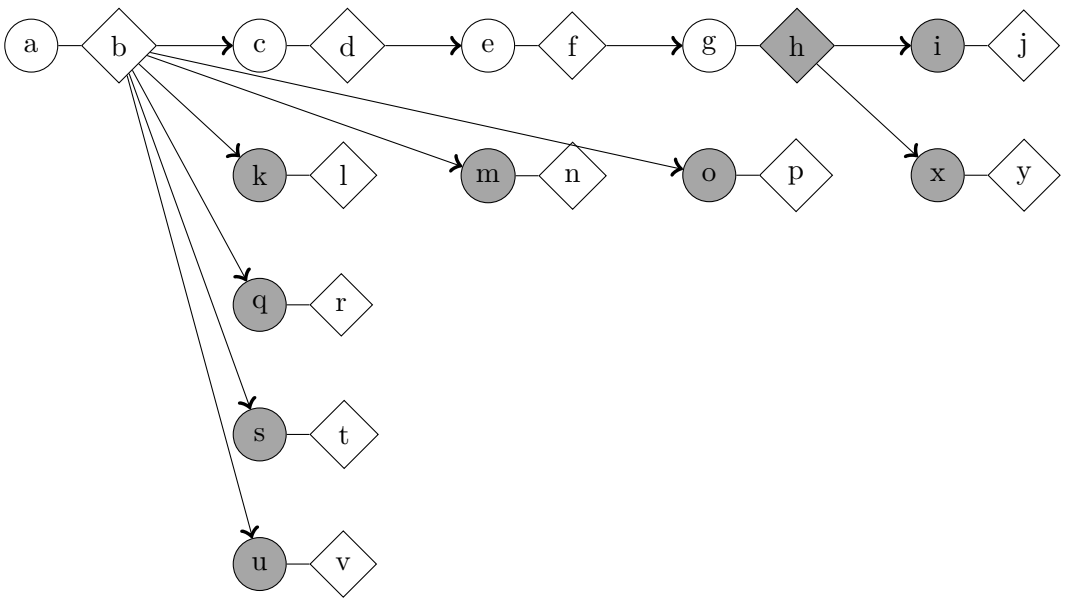


FIGURE 3.12 – 1-extensible d'un arbre dont le graphe des composantes est une arborescence

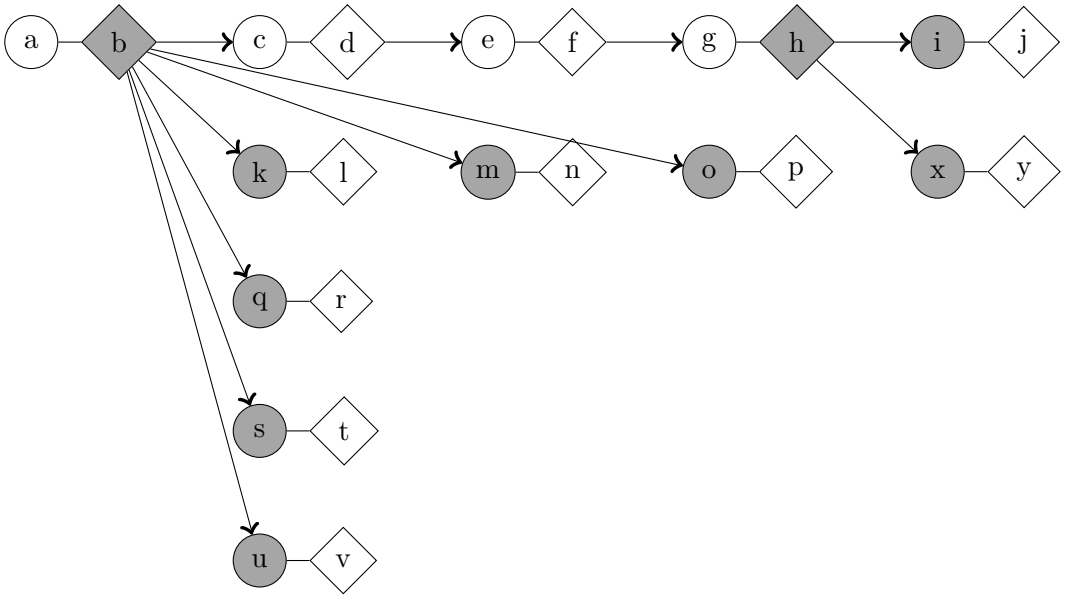


FIGURE 3.13 – 4-extensible d'un arbre dont le graphe des composantes est une arborescence

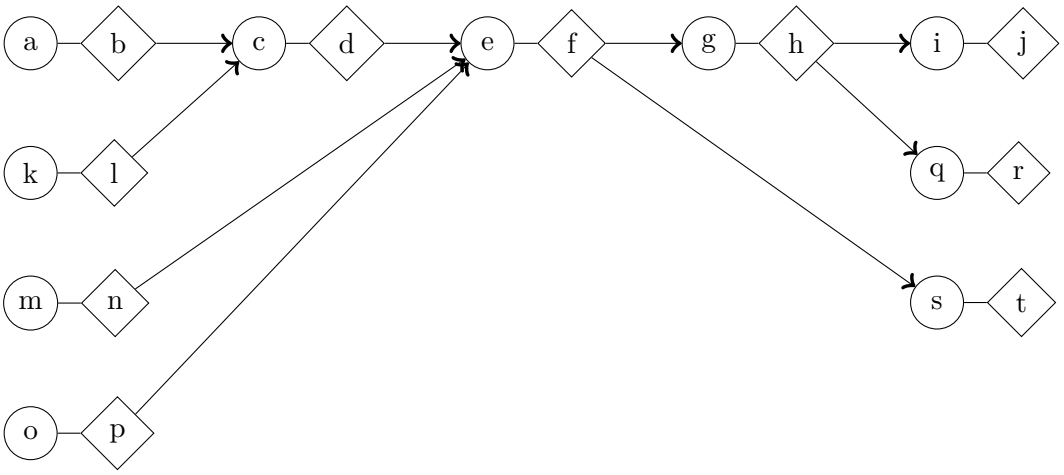


FIGURE 3.14 – Arbre tel que pour toute composantes C, soit $C \to C(e)$, soit $C(e) \to C$

composantes de G_2 est une arborescence.

Supposons que $d > 1$. Montrons qu'il n'est pas possible qu'un d-extensible F de cardinal supérieur à $2d$ de G contienne deux sommets x et y tels que x soit un sommet de G_1, y soit un sommet de G_2 et $C(x) \to C(y)$.

— Si $x \in B$ et $y \in B$ alors d'après la Propriété 3.4.3 page 69, $\alpha(G[F-(\Gamma(x)\cup\{y\})]) < d$. Donc $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d-2+|\Gamma(x)-1|$. Or le graphe des composantes de G_1 est une anti-arborescence donc $|\Gamma(x) = 2|$. Donc $|F \cap R| \leqslant d-1$ ce qui implique que $|F| < 2d$. Contradiction.

— Si $x \in B$ et $y \in R$ alors d'après la Propriété 3.4.3 page 69, $\alpha(G[F-(\Gamma(x)\cup\Gamma(y))]) < d$. Donc $|F \cap R| \leqslant d-2+|\Gamma(x)-1|$ et $|F \cap B| \leqslant d-2+|\Gamma(y)-1|$. Or le graphe des composantes de G_1 est une anti-arborescence donc $|\Gamma(x) = 2|$ et le graphe des composantes de G_2 est une arborescence donc $|\Gamma(y) = 2|$. Ainsi, $|F| \leqslant 2d$. Contradiction.

— Si $x \in R$ et $y \in B$ alors d'après la Propriété 3.4.3 page 69, $\alpha(G[F-(\{x,y\})]) < d$. Ainsi $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d$. Donc $|F| \leqslant 2d$. Contradiction.

— Si $x \in R$, $y \in R$ alors d'après la Propriété 3.4.3 page 69, $\alpha(G[F-(\{x\}\cup\Gamma(y))]) < d$. Donc $|F \cap R| \leqslant d$ et $|F \cap B| \leqslant d-1+|\Gamma(y)-1|$. Or le graphe des composantes de G_2 est une arborescence donc $|\Gamma(y)| = 2$. Donc $|F| \leqslant 2d$. Contradiction.

Ainsi, s'il existe un d-extensible F de cardinal supérieur à $2d$ pour $d > 1$ de G, c'est soit

un d-extensible de G_1, soit un d-extensible de G_2. Or d'après la Propriété 3.6.4 page 87, il est possible de construire un d-extensible optimal de G_1 et de G_2 en temps polynomial. Donc pour $d > 2$, il est possible de déterminer un d-extensible optimal de G en construisant celui de cardinal $2d$ et ceux de G_1 et de G_2 puis de sélectionner celui de plus grand cardinal.

Considérons maintenant le cas $d = 1$. Il est évident que des 1-extensible de G_1et de G_2 sont aussi des d-extensibles de G. Montrons que si un 1-extensible F de G de cardinal supérieur à 2 contient deux sommets x et y tels que $C(x) \to C(y)$, x est un sommet de G_1 et y un sommet de G_2, alors $x \in B$ et $y \in R$.

— Si $x \in R$ et $y \in B$ alors alors d'après la Propriété 3.4.3 page 69, $|F| \leqslant 2d$. Contradiction.

— Si $x \in R$ et $y \in R$ alors d'après la Propriété 3.4.1 page 64 le stable $\{y\}$ n'est pas extensible par rapport à F. Contradiction.

— Si $x \in B$ et $y \in B$ alors d'après la Propriété 3.4.1 page 64 le stable $\{x\}$ n'est pas extensible par rapport à F. Contradiction.

Ainsi, si F contient un sommet R de G_1, il ne peut contenir aucun sommet de G_2 car pour tout sommet x de G_1 et pour tout sommet y de G_2, $C(x) \to C(y)$. De même, si F contient un sommet de B de G_2 il ne peut contenir aucun sommet de G_1. Si un 1-extensible contient des sommets de G_1 et de G_2 alors tous les sommets de G_1 qu'il contient sont dans B et tous les sommets de G_2 qu'il contient appartiennent à R.

Considérons l'ensemble F qui contient les sommets de B des composantes sans prédécesseurs et les sommets de R des composantes sans successeurs. On obtient ainsi un 1-extensible dont les sommets de B sont dans G_1 et les sommets de R dans G_2. De plus G_1 est une anti-arborescence donc chaque composante a au moins un prédécesseur sauf les composantes sans prédécesseurs et G_2 est une arborescence donc chaque composante a au moins un successeur sauf les composantes sans successeurs. Ainsi, il n'est pas possible de construire un 1-extensible qui contient des sommets de G_1 et de G_2 de plus grand cardinal que F.

Ainsi, en construisant F et les 1-extensible optimaux de G_1 et de G_2 puis en conservant celui de plus grand cardinal, on obtient un 1-extensible optimal de G.

□

3.6.1 d-extensibles de stables de homards

Cette sous-section traite d'un famille d'arbres : les homards. Dans un premier temps nous traitons le cas des chenilles. Puis nous montrons que déterminer un d-extensible optimal d'un homard se fait en temps polynomial en examinant successivement trois cas : $d = 1$, $d = 2$ et $d > 2$.

Considérons tout d'abord une chenille $G = (B, R, E)$ dont tous les sommets sont libres. Une chenille est un arbre dont tous les sommets sont au plus à une distance 1 d'une chaîne centrale. Par exemple, sur la Figure 3.15 page 100, la chaîne centrale est constituée des sommets a, b, c, d, e, f, g, j, k, l, m, r, q, p, s et t ; les pattes sont les sommets h, i, n et o.

Le propriété suivante est similaire à la Propriété 3.6.3 car elle démontre qu'il existe un 1-extensible optimal qui est aussi un k-extensible pour $k > 1$ dans les chenilles. Mais ce 1-extensible est aussi un d-extensible optimal s'il contient plus de $2d$ sommets, donc plus de sommets que le d-extensible déterminé par la Propriété 3.6.1 page 85.

Propriété 3.6.6. *Soit $G = (B, R, E)$ une chenille dont tous les sommets sont libres.*

- *Il existe un 1-extensible optimal F_1 qui est aussi un k-extensible pour $k > 1$. Ce un 1-extensible optimal est détermine par l'Algorithme 8 page 98.*
- *Pour $d > 1$, s'il existe un d-extensible optimal de cardinal strictement supérieur à $2d$, il est déterminé par l'algorithme 8 page 98. Sinon un d-extensible optimal est de cardinal $2d$ et est déterminé par l'algorithme 5 page 85*
- *Il est possible de déterminer un d-extensible optimal de G en temps polynomial.*

Démonstration. Une chenille est composée d'une chaîne centrale et de sommets qui sont voisins de ceux de la chaîne. Pour que ces sommets soient libres chaque sommet de la chaîne ne peut pas avoir plus d'un voisin qui n'est pas dans la chaîne, sans quoi il sera exclu. Dans le graphe des composantes, les composantes correspondant aux "pattes" de la chenille sont celles qui n'ont soit pas de successeurs, soit pas de prédécesseurs. Les composantes

dont tous les sommets sont sur la chaîne centrale sont celles qui ont un prédécesseur et un successeur.

Une chenille est un arbre donc il existe un d-extensible de cardinal $2d$. C'est pourquoi on recherche s'il existe un d-extensible de cardinal strictement supérieur à $2d$.

Chaque sommet est au plus de degré 3 donc d'après la Propriété 3.4.3 page 69, si $d > 1$ un d-extensible F de G de cardinal supérieur à $2d$ ne peut contenir de sommets $x \in F$ et $y \in F$ tels que $C(x) \to C(y)$. En effet :

— Si $C(x) = C(y)$, $|F| \geqslant 2d$

— Si $x \in B$ et $y \in B$, alors si $|F \cap B| > d$ ou $|F \cap R| > d - 2 + |\Gamma(x) - 1|$ alors il existe un stable de cardinal d qui contient x et pas y. Or $|\Gamma(x)| \geqslant 3$ donc $|F| \geqslant 2d$. Contradiction.

— Si $x \in R$ et $y \in B$, $|F \cap B| \geqslant d$ et $|F \cap R| \geqslant d$ donc $|F| \geqslant 2d$. Contradiction.

— Si $x \in R$ et $y \in R$, alors si $|F \cap R| > d$ ou $|F \cap B| > d - 2 + |\Gamma(y) - 1|$ alors il existe un stable de cardinal d de $G[F]$ qui contient y et pas x. Or $|\Gamma(y)| \geqslant 3$ donc $|F| \geqslant 2d$. Contradiction.

— Si $x \in B$ et $y \in R$, si $|F \cap R| > d - 2 + |\Gamma(x) - 1|$ ou $|F \cap B| > d - 2 + |\Gamma(y) - 1|$ alors il existe un stable de cardinal d de $G[F]$ qui contient x et y. Or $|\Gamma(y)| \geqslant 3$ et $|\Gamma(x)| \geqslant 3$ donc $|F| \geqslant 2d$. Contradiction.

Ainsi, sur chaque chemin dans le graphe des composantes, si $d > 1$ un d-extensible de cardinal supérieur à $2d$ ne peut contenir plus de deux sommets. C'est aussi le cas si $d = 1$. En effet, d'après la Propriété 3.4.3 page 69, si un 1-extensible contient deux sommets x et y tel que $C(x) \to C(y)$ alors soit $x \in B$ et $y \in R$, soit $x \in R$ et $y \in B$.

Nous allons prouver que l'Algorithme 8 page 98 retourne un ensemble de sommets qui est un 1-extensible optimal et pour $d > 1$, et un d-extensible optimal s'il est de cardinal supérieur ou égal à $2d$. Le principe de l'algorithme est le suivant : le graphe des composantes d'une chenille est constitué de chemins qui sont reliés les uns aux autres par leurs extrémités. On sait que sur chaque chemin, on ne peut pas sélectionner plus de deux sommets. On souhaite identifier chacun de ces chemins et sélectionner si possible deux sommets voisins sur chaque chemin. Pour cela, on considère séparément les composantes des pattes de la

chenille et de la chaîne centrale car les extrémités des chemins dans le graphe des composantes sont les composantes des "pattes" de la chenille. Donc les composantes de la chaîne centrale n'appartiennent qu'à un chemin et on peut facilement sélectionner des sommets dans ces composantes. On traite ensuite le cas des composantes des "pattes" de la chenille.

L'Algorithme 8 commence par partitionner les composantes dans les ensembles suivants :

— Soit L_1 l'ensemble des composantes qui ont deux prédécesseurs et dont les prédécesseurs ont deux successeurs

— Soit L_2 l'ensemble des composantes qui ont deux successeurs et dont les successeurs ont deux prédécesseurs

— Soit L_3 l'ensemble des composantes qui ont deux prédécesseurs et dont les prédécesseurs ne sont pas dans L_2

— Soit L_4 l'ensemble des composantes qui ont deux successeurs et dont les successeurs ne sont pas dans L_1

— Soit L_5 l'ensemble des composantes qui ont au plus un successeur et un prédécesseur et qui ont au moins un voisin qui n'est pas dans $L_1 \cup L_2 \cup L_3 \cup L_4$

— Soit L_6 l'ensemble des composantes qui ne sont pas dans les ensemble précédents

Les composantes des ensembles L_1 et L_2 appartiennent à des chemins de longueur 1 dans G_c. Un d-extensible de cardinal supérieur à $2d$ ne peut contenir plus de deux sommets sur chaque chemin de G_c et pas plus d'un sommet par composante. L'algorithme sélectionne donc un sommet par composante de L_1 et un sommet par composante de L_2. Les composantes de l'ensemble L_5 appartiennent aux chemins de longueur supérieure ou égale à 3. Sur ces chemins, on sélectionne deux sommets sur des composantes qui ne sont pas aux extrémités du chemin donc dans deux composantes de L_5. Il reste à considérer les chemins de longueur 2 dans G_c. Sur ces chemins, on sélectionne un sommet sur la composante centrale, c'est-à-dire les composantes de L_6 et, si possible un sommet à une des extrémités donc dans L_3 ou L_4.

Soit F un ensemble de sommets retourné par l'Algorithme 8 page 98. D'après la Propriété 3.4.3 page 69, F est bien un d-extensible pour toute valeur de d. Montrons que s'il

existe un d-extensible de cardinal $2d$, alors F est un d-extensible optimal de G.

Soit $\hat{F}$ un 1-extensible tel que $|\hat{F}| > |F|$. Un 1-extensible ne contient au plus qu'un sommet par composante et F contient un sommet pour chaque composante des ensembles L_1, L_2 et L_6 donc $|\hat{F} \cap L_1| \leqslant |F \cap L_1|$, $|\hat{F} \cap L_2| \leqslant |F \cap L_2|$ et $|\hat{F} \cap L_6| \leqslant |F \cap L_6|$. Nous avons démontré plus haut qu'un 1-extensible ne pouvait contenir plus de 2 sommets sur chaque chemin du graphe des composantes donc en notant L_r l'ensemble des composantes retirées de L_3, L_4 et L_5 lors de l'exécution de l'algorithme, $|\hat{F} \cap (L_5 \cup L_r)| \leqslant |F \cap L_5|$. Enfin, sur chaque chemin de longueur 3 de G_c, un 1-extensible ne peut contenir plus de 2 sommets donc en sélectionnant tous les sommets de L_6, on ne peut sélectionner plus de sommets que le cardinal d'un stable de cardinal maximal du graphe $\hat{G} = (L_3 - L_r, L_4 - L_r), \hat{E})$ avec l'arête $C(x)C(y) \in \hat{E}$ si et seulement si $C(x) \to C(y)$. Donc $|\hat{F} \cap ((L_3 \cup L_4) - L_r)| \leqslant |F \cap ((L_3 \cup L_4) - L_r|$. Donc $|\hat{F}| \leqslant |F|$. Contradiction.

Ainsi, s'il existe un d-extensible de cardinal supérieur à $2d$, l'Algorithme 8 page 98 retourne un d-extensible optimal. Sinon, le cardinal maximal d'un d-extensible est $2d$ et un tel d-extensible peut être déterminé selon la méthode décrite dans la preuve de la Propriété 3.6.1 page 85.

□

Exemple 3.6.2. *Considérons la chenille de la figure 3.15 :*

- $L_1 = \{C(s)\}$ *est l'ensemble des composantes qui n'ont pas de successeurs et dont les prédécesseurs ont deux successeurs.*
- $L_2 = \varnothing$ *est l'ensemble des composantes qui n'ont pas de prédécesseurs et dont les successeurs ont deux prédécesseurs.*
- $L_3 = \{C(m), C(g)\}$ *est l'ensemble des composantes qui ont deux prédécesseurs et dont les prédécesseurs ne sont pas dans* L_2.
- $L_4 = \{C(o), C(i), C(a)\}$ *est l'ensemble des composantes qui ont deux successeurs et dont les successeurs ne sont pas dans* L_1.
- $L_5 = \{C(c), C(e)\}$ *est l'ensemble des composantes qui ont au plus un successeur et un prédécesseur et qui ont au moins un voisin qui n'est pas dans* $(L_1 \cup L_2 \cup L_3 \cup L_4)$.

Algorithm 8 Algorithme permettant de déterminer un 1-extensible optimal d'une chenille dont tous les sommets sont libres

$F = \varnothing$
Pour toute composante C, on note C_p son prédécesseur et C_s son successeur
Soit L_1 l'ensemble des composantes qui ont deux prédécesseurs et dont les prédécesseurs ont deux successeurs
Soit L_2 l'ensemble des composantes qui ont deux successeurs et dont les successeurs ont deux prédécesseurs
Soit L_3 l'ensemble des composantes qui ont deux prédécesseurs et dont les prédécesseurs ne sont pas dans L_2
Soit L_4 l'ensemble des composantes qui ont deux successeurs et dont les successeurs ne sont pas dans L_1
Soit L_5 l'ensemble des composantes qui ont au plus un successeur et un prédécesseur et qui ont au moins un voisin qui n'est pas dans $L_1 \cup L_2 \cup L_3 \cup L_4$
Soit L_6 l'ensemble des composantes qui ne sont pas dans $L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5$
for all $C \in L_1$ **do**
    $F := F \cup (V(C) \cap R)$
end for
for all $C \in L_2$ **do**
    $F := F \cup (V(C) \cap B)$
end for
for all $C \in L_5$ **do**
    if $C_p \in L_5$ **then**
        $F := F \cup (V(C_p) \cap B) \cup (V(C) \cap R)$
    else
        $F := F \cup (V(C) \cap B) \cup (V(C_s) \cap R)$
    end if
    Retirer C, ses prédécesseurs et ses successeurs de L_5, L_3, L_4
end for
Soit le graphe biparti $\hat{G} = (L_3, L_4, \hat{E})$ avec $C(x)C(y) \in \hat{E}$ si et seulement si $C(x) \rightarrow C(y)$
Soit S^* un stable optimal de $\hat{G}$
for all $C \in L_3$ **do**
    if $C \in S^*$ **then**
        $F := F \cup (V(C) \cap R)$
    end if
end for
for all $C \in L_4$ **do**
    if $C \in S^*$ **then**
        $F := F \cup (V(C) \cap B)$
    end if
end for
for all $C \in L_6$ **do**
    if $F \cap V(C_p) \neq \varnothing$ **then**
        $F := F \cup (V(C) \cap R)$
    else
        $F := F \cup (V(C) \cap B)$
    end if
end for
return F

— $L_6 = \{C(k), C(q)\}$ *est l'ensemble des composantes qui ne sont pas dans les ensemble précédents.*

L'Algorithme 8 commence par sélectionner les sommets de R *des composantes de* L_1 *et les sommets de* B *des composantes de* L_2*. Ici, il ne sélectionne donc que le sommet* s *de* $C(s)$*.*

L'algorithme considère ensuite les composantes de L_5*. Chacune des ces composantes a un voisin dans* L_5*. Pour chaque composante de* L_5*, l'algorithme sélectionne un sommet* x *de la composante et un sommet* y *d'une composante dans* L_5 *voisine tels que* $xy \in E$*. Puis il retire de* $L_3 \cup L_4 \cup L_5$ *les composantes qui appartiennent au chemin dans lequel les sommets ont été sélectionnés. Ici, les sommets* d *et* e *des composantes* $C(c)$ *et* $C(e)$ *sont sélectionnés. Les sommets sélectionnés à ce stade de l'exécution de l'algorithme sont indiqués dans la Figure 3.16.*

Puis, l'algorithme construit un graphe biparti $\hat{G}$ *qui est indiqué dans la Figure 3.17. L'un des ensemble de la bipartition contient* $C(m)$*, l'autre contient* $C(o)$ *et* $C(i)$ *car* $C(a)$ *et* $C(g)$ *ont été retirés lors de la sélection d'un sommet de* $C(e)$*. Les arêtes sont* $C(i)C(m)$ *et* $C(o)C(m)$*. Le stable optimal du graphe contient donc* $C(i)$ *et* $C(o)$*. On sélectionne donc des sommets dans ces composantes. Elles appartiennent toutes les deux à* L_4 *donc on sélectionne les sommets de* B *c'est-à-dire* p *et* j*.*

Enfin, dans les composantes de L_6*,* $C(k)$ *et* $C(q)$*, on sélectionne des sommets voisins de sommets déjà sélectionnés donc ici* q *et* k*. Le d-extensible final est indiqué dans la Figure 3.18.*

On considère maintenant un homard $G = (B, R, E)$ dont tous les sommets sont libres. Un homard est un arbre dont les sommets sont au plus à distance 2 d'une chaîne centrale.

Soit la partition de l'ensemble des composantes de G suivante :

— L_1 ensemble des composantes avec un unique prédécesseur et pas de successeur

— L_2 ensemble des composantes avec un unique successeur et pas de prédécesseur

— L_3 ensemble des composantes avec au moins deux prédécesseurs et pas de successeurs

— L_4 ensemble des composantes avec au moins deux successeurs et pas de prédécesseurs

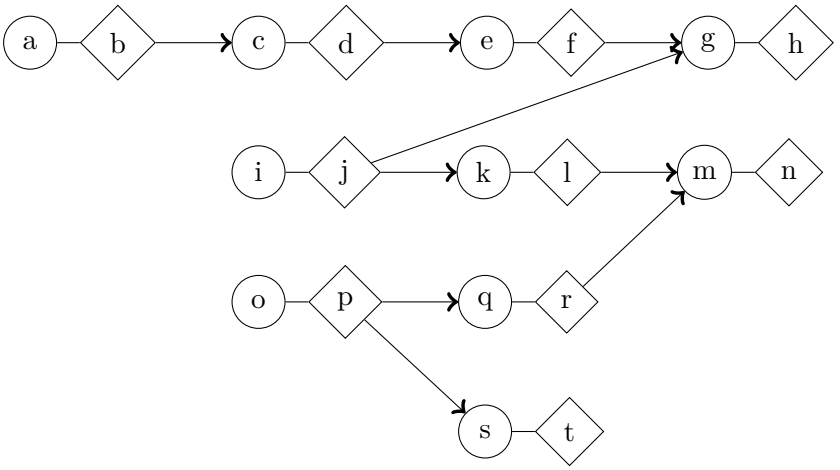


FIGURE 3.15 – Chenille. Dans ce graphe, $\alpha(G) = 10$

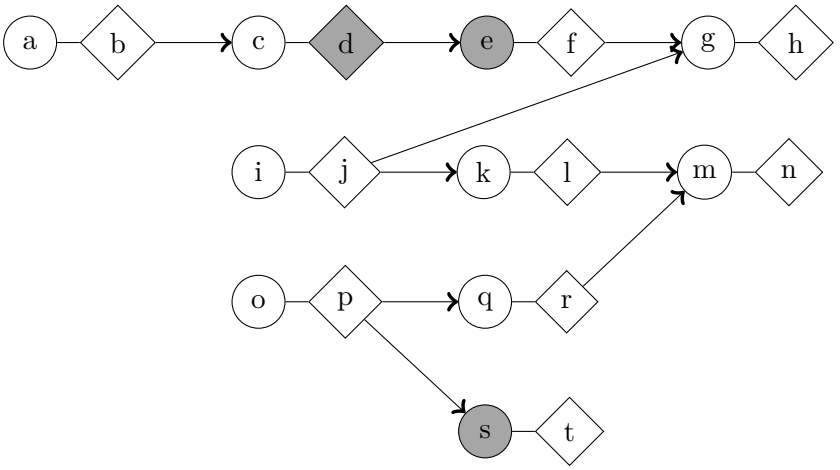


FIGURE 3.16 – Chenille après sélection de sommets de L_1, L_2 et L_5 au cours de l'exécution de l'Algorithme 8

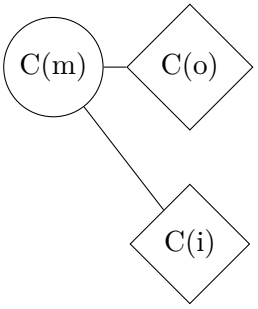


FIGURE 3.17 – Graphe auxiliaire construit par l'algorithme 8

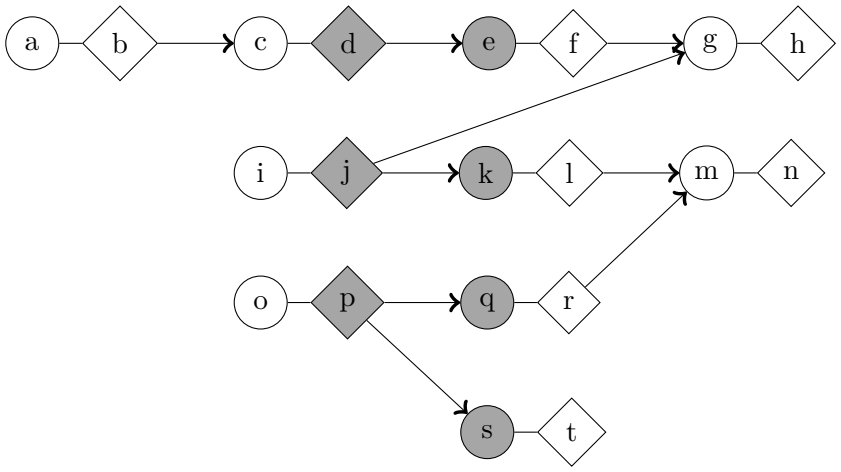


FIGURE 3.18 – Résultat de l'exécution de l'algorithme 8

— L_5 ensemble des composantes avec au moins un prédécesseur et au moins un successeur

Exemple 3.6.3. *Considérons le homard de la Figure 3.19. La chaîne centrale est constituée des sommets c, d, m, n, o, p, q, r, u, x, y,* h_2 *et* g_2*.*

— $L_1 = \{C(k), C(i), C(j), C(e), C(k_2)\}$

— $L_2 = \{C(a), C(s), C(a_2), C(c_2), C(e_2)\}$

— $L_3 = \{C(u), C(y)\}$

— $L_4 = \{C(w)\}$

— $L_5 = \{C(c), C(m), C(o), C(q), C(g_2)\}$

La suite de cette section démontre que la recherche d'un d-extensible dans un homard dont tous les sommets sont libres est polynomiale. D'après la Propriété 3.6.3 page 86, c'est le cas si $d = 1$ car le graphe des composantes d'un homard est un arbre. Les deux propriétés suivantes traitent le cas $d = 2$. Le cas $d > 2$ est traité ensuite.

Propriété 3.6.7. *Soit* $G = (B, R, E)$ *un homard dont tous les sommets sont libres et tel que pour tout couple de composantes* C_1 *et* C_2 *de* L_5*, soit* $C_1 \to C_2$*, soit* $C_2 \to C_1$*. Un* 2*-extensible optimal de G peut être déterminé en temps polynomial.*

Démonstration. Un homard est un arbre donc il existe un 2-extensible de cardinal 4 d'après

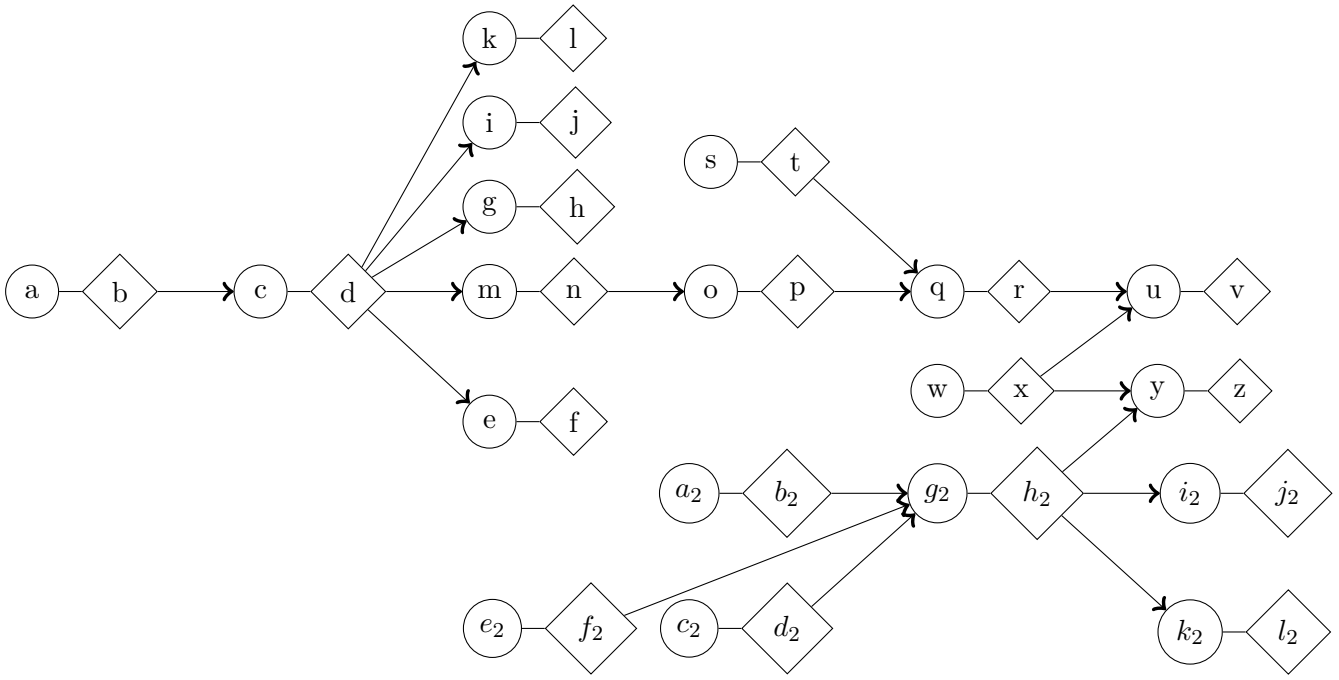


FIGURE 3.19 – Un homard. Dans ce graphe, $\alpha(G) = 19$

la Propriété 3.6.1 page 85. On cherche donc s'il existe un 2-extensible de cardinal strictement supérieur à 4.

Montrons qu'il existe un 2-extensible optimal tel que tout couple de sommets x et y de ce 2-extensible, si $C(x) \to C(y)$ alors $xy \in E$.

— Si $x \in B$ et $y \in R$ alors d'après la Propriété 3.4.1 page 64, le stable $\{x, z\}$ n'est pas extensible par rapport à F. Contradiction.

— Si $x \in R$ et $y \in B$ alors $|F| \leqslant 2d$. Contradiction.

— Si $x \in B$ et $y \in B$ alors il n'existe pas de sommets $z \in F$ distinct de y tel que $C(x) \to C(z)$ et $xz \notin E$. En effet, dans ce cas, d'après la Propriété 3.4.1 page 64, le stable $\{x, z\}$ n'est pas extensible par rapport à F. De même, il n'existe pas de sommets $z \in F$ tel que $C(z) \to C(x)$. Soit $a \in R$ le sommet tel que $C(a) \to C(y)$ et $xa \in E$. D'après le point précédent, $a \notin F$. Ainsi, d'après la Propriété 3.4.3 page 69, l'ensemble $(F - \{y\}) \cup \{a\}$ est un 2-extensible de même cardinal que F.

— Si $x \in R$ et $y \in R$ alors il n'existe pas de sommets $z \in F$ distinct de y tel que $C(z) \to C(y)$ et $xz \notin E$. En effet, dans ce cas, d'après la Propriété 3.4.1 page 64,

le stable $\{x, z\}$ n'est pas extensible par rapport à F. De même, il n'existe pas de sommets $z \in F$ tel que $C(y) \to C(z)$. Soit $a \in B$ le sommet tel que $C(a) \to C(y)$ et $ay \in E$. D'après le premier point, $a \notin F$. Ainsi, d'après la Propriété 3.4.3 page 69, l'ensemble $(F - \{x\}) \cup \{a\}$ est un 2-extensible de même cardinal que F.

S'il existe un 2-extensible optimal F avec deux sommets $x \in F$ et $y \in F$ tels que $xy \in E$, alors $C(x) \in L_5$ ou $C(y) \in L_5$ car les prédécesseurs de L_1 et L_3 sont dans L_5, de même que les successeurs de L_2 et L_4.

Montrons que l'Algorithme 9 détermine pour chaque composante C de L_5 et de L_4 le 2-extensible maximal (au sens de l'inclusion) qui contient le sommet $x \in B \cap V(C)$, ainsi que le 2-extensible maximal qui ne contient aucun sommet de L_4 et de L_5.

Soit $x \in B$ tel que $C(x) \in L_5$ ou $C(x) \in L_4$. Montrons que tout 2-extensible optimal F tel que $x \in F$ contient aussi tous les sommets y tels que $xy \in E$ et $C(x) \neq C(y)$. Puisque $x \in F$, il n'existe pas de $z \in F$ tel que $C(z) \to C(x)$ ou tel que $C(x) \to C(z)$ et $C(x)$ et $C(z)$ ne sont pas voisins dans G_c. Donc pour tout $y \in R$ tel que $y \notin F$, $xy \in E$ et $C(x) \neq C(y)$, $F \cup \{y\}$ est aussi un 2-extensible d'après la Propriété 3.4.3 page 69.

Montrons maintenant que tout 2-extensible F qui contient x déterminé par l'algorithme est de cardinal maximal. Soit $\hat{F}$ un 2-extensible tel que $x \in \hat{F}$ et $|\hat{F}| > |F|$. On sait que F contient x et les sommets z tels que $xy \in E$ et $C(x) \neq C(z)$. L'ensemble $\hat{F}$ ne peut contenir plus de sommets des composantes de $L_3 \cup L_4 \cup L_5$ que ceux contenus dans F. Considérons $\widehat{L_1}$ constitué de l'ensemble L_1 auquel on a retiré les composantes C telles que $C(x) \to C$ et l'ensemble $\widehat{L_2}$ constitué de l'ensemble L_2 auquel on a retiré les composantes C telles que $C \to C(x)$. Un 2-extensible qui contient x ne peut contenir de sommets de $L_1 - \widehat{L_1}$ ou de $L_2 - \widehat{L_2}$. Considérons le graphe biparti $\hat{G} = (\widehat{L_1}, \widehat{L_2}, \hat{E})$ tel qu'il existe une arête dans $\hat{E}$ entre deux composantes C_1 de $\widehat{L_1}$ et C_2 de $\hat{L_1}$ si et seulement si $C_2 \to C_1$. Un 2-extensible ne peut contenir des sommets de deux composantes de $\widehat{L_1}$ et de $\widehat{L_2}$ si elles sont voisines dans $\hat{G}$. Donc $\hat{F}$ ne peut contenir plus de sommets de $\widehat{L_1} \cup \widehat{L_2}$ que le cardinal maximal d'un stable de $\hat{G}$. Or, F contient exactement autant de sommets de $\widehat{L_1} \cup \widehat{L_2}$ que le cardinal maximal d'un stable de $\hat{G}$. Donc il n'est pas possible que $|\hat{F}| > |F|$.

Considérons maintenant le 2-extensible F déterminé par l'algorithme qui ne contient

aucun sommet des composantes de L_5, L_4 et L_3. Soit $\hat{F}$ un 2-extensible qui ne contient aucun sommet des composantes de L_5, L_4 et L_3 et tel que $|\hat{F}| > |F|$. Dans ce cas, $\hat{F}$ contient plus de sommets des composantes de L_1 et L_2 que F. Considérons le graphe $\hat{G} = (L_1, L_2, \hat{E})$ dans lequel il existe une arête entre deux composantes C_1 de L_1 et C_2 de L_1 si et seulement si $C_2 \to C_1$. Un 2-extensible ne peut contenir de sommets de deux composantes voisines dans $\hat{G}$ donc un $\hat{F}$ ne peut contenir plus de sommets que le cardinal maximal d'un stable de $\hat{G}$. Or F contient autant de sommets de $L_1 \cup L_2$ que le cardinal maximal de $\hat{G}$ donc $\hat{F}$ ne peut pas contenir plus de sommets que F.

Soit un 2-extensible optimal de G ne contient pas de sommets de L_4 et L_5, soit il en contient un, donc l'Algorithme 9 détermine un 2-extensible optimal car il détermine les 2-extensible maximaux, dans le cas où aucun sommet de $L_4 \cup L_5$ n'est sélectionné et, dans les cas où chaque élément de $L_4 \cup L_5$ est dans l'extensible.

□

Exemple 3.6.4. *Considérons le homard de la Figure 3.20. On constate que dans ce homard, pour toute paire de composantes C_1 et C_2 de L_5, soit $C_1 \to C_2$, soit $C_2 \to C_1$. Autrement dit, il ne contient pas de pattes de longueur* 1*. Les sommets à distance* 1 *de la chaîne centrale ont tous un unique voisin à distance* 2 *de la chaîne.*

L'Algorithme 9 commence par construire un 2*-extensible maximal au sens de l'inclusion qui ne contient pas de sommets sur la chaîne centrale. Il construit le graphe biparti qui contient dans l'un des ensembles de la bipartition $C(a)$ et $C(s)$ et dans l'autre ensemble $C(k)$, $C(i)$, $C(g)$, $C(e)$, $C(u)$. Ce graphe est représenté dans la Figure 3.21. Il existe une arête entre $C(a)$ et tous les autres sommets sauf $C(s)$, et une arête entre $C(s)$ et $C(u)$. Un stable optimal est celui qui contient tous les sommets sauf $C(a)$ et $C(u)$. En sélectionnant un sommet du homard par composante du stable du graphe auxiliaire, on obtient un* 2*-extensible de cardinal* 5*.*

L'Algorithme 9 examine ensuite les 2*-extensibles qui contiennent des sommets de la chaîne centrale. Pour chacune des composantes de L_5 et de L_4, l'algorithme détermine un* 2*-extensible maximal au sens de l'inclusion qui contient un sommet de cette composante :*

— *pour la composante $C(d)$, l'algorithme détermine le* 2*-extensible de cardinal* 7 *:*

Algorithm 9 Algorithme permettant de déterminer un 2-extensible optimal d'un homard dans lequel il existe un chemin dans G_c entre tout couple de composantes de L_5

1: $F = \varnothing$
2: $F_{cur} = \varnothing$
3: Soit le graphe biparti $\hat{G} = (L_1, L_2, \hat{E})$ avec $C(x)C(y) \in \hat{E}$ si et seulement si $C(x) \rightarrow C(y)$
4: Soit S^* un stable optimal de $\hat{G}$
5: **for all** $C \in S^*$ **do**
6: $\quad F := F \cup (V(C) \cap B)$
7: **end for**
8: **for all** $C \in L_5$ ou $C \in L_4$ **do**
9: $\quad F_{cur} := \varnothing$
10: $\quad F_{cur} := V(C) \cap B$
11: $\quad$ **for all** C_s successeur immédiat de C **do**
12: $\quad\quad F_{cur} := F_{cur} \cup (V(C_s) \cap R)$
13: $\quad$ **end for**
14: $\quad$ Soit $\hat{L_1}$ l'ensemble des composantes C_i de L_1 telles que $C(i) \nrightarrow C$
15: $\quad$ Soit $\hat{L_2}$ l'ensemble des composantes C_i de L_2 telles que $C \nrightarrow C(i)$
16: $\quad$ Soit le graphe biparti $\hat{G} = (\hat{L_1}, \hat{L_2}, \hat{E})$ avec $C(x)C(y) \in \hat{E}$ si et seulement si $C(x) \rightarrow C(y)$
17: $\quad$ Soit S^* un stable optimal de $\hat{G}$
18: $\quad$ **for all** $C_j \in S^*$ **do**
19: $\quad\quad F_{cur} := F_{cur} \cup (V(C_j) \cap B)$
20: $\quad$ **end for**
21: $\quad$ **if** $|F_{cur}| > |F|$ **then**
22: $\quad\quad F := F_{cur}$
23: $\quad$ **end if**
24: **end for**
25: **return** F

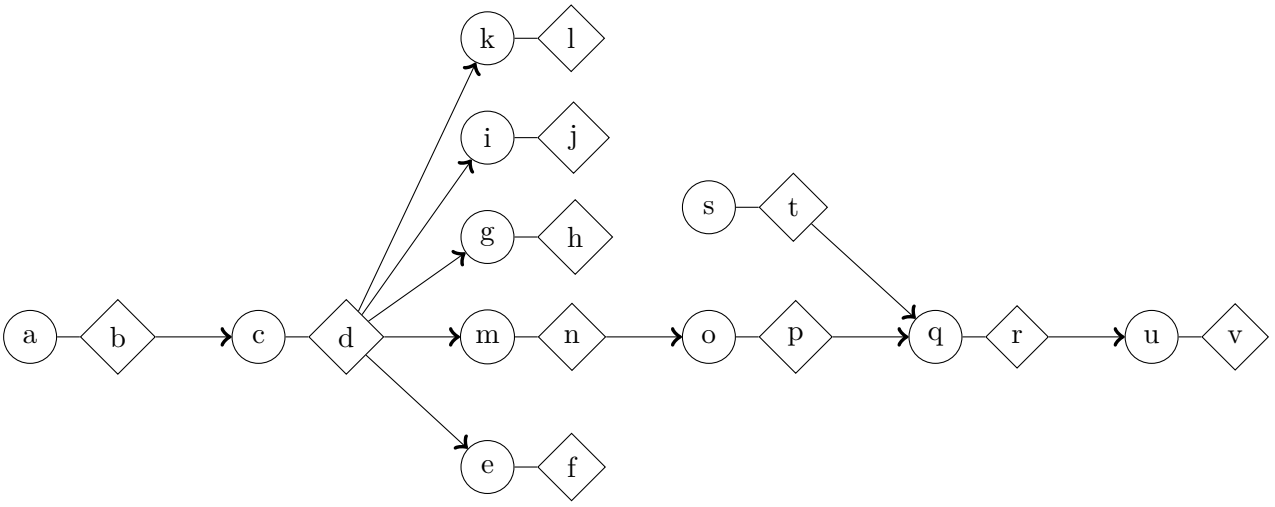


FIGURE 3.20 – Homard tel qu'il existe un chemin entre toute composante d'un sommet de la chaîne centrale. Dans ce graphe, $\alpha(G) = 11$

$\{d, m, k, i, g, e, s\}$. *Il est indiqué dans la Figure 3.22.*

- *pour la composante* $C(m)$*, l'algorithme détermine le* 2*-extensible de cardinal* 7 *:* $\{n, o, k, i, g, m, e, s\}$
- *pour la composante* $C(o)$*, l'algorithme détermine le* 2*-extensible de cardinal* 7 *:* $\{p, q, k, i, g, m, e, s\}$
- *pour la composante* $C(q)$*, l'algorithme détermine le* 2*-extensible de cardinal* 6 *:* $\{r, u, k, i, g, e\}$

Le plus grand cardinal de 2*-extensible déterminé est* 7*. il est supérieur à* 4*. L'algorithme retourne donc un* 2*-extensible de cardinal* 7*,* $\{d, m, k, i, g, e, s\}$.

Propriété 3.6.8. *Soit* $G = (B, R, E)$ *un homard dont tous les sommets sont libres. Un* 2*-extensible optimal de* G *peut être déterminé en temps polynomial.*

Démonstration. Soient C_1 et C_2 deux composantes de L_3. Elles n'ont pas de successeurs donc il est impossible que $C_1 \to C_2$ ou que $C_2 \to C_1$. De même, si C_1 et C_2 sont deux composantes de L_4 alors il n'est pas possible que $C_1 \to C_2$ ou que $C_2 \to C_1$.

Considérons maintenant $C_1 \in L_4$ et $C_2 \in L_3$. Considérons le sous-graphe G_s de G constitué des composantes C telles que soit $C \to C_2$, soit $C_1 \to C$. Le sous-graphe ainsi

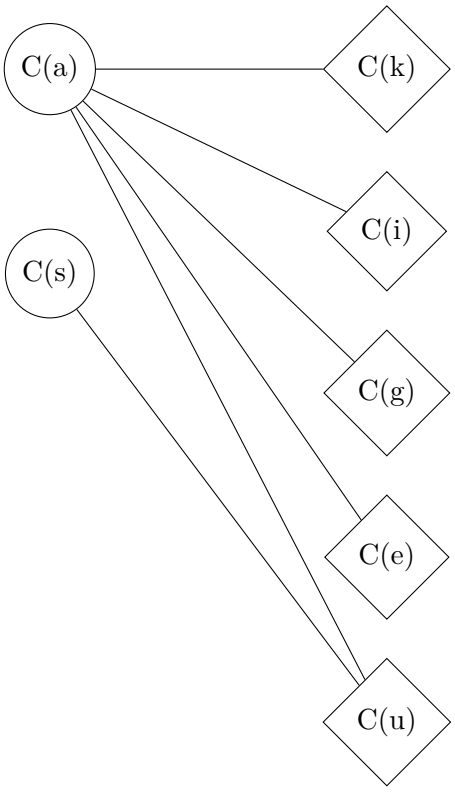


FIGURE 3.21 – Graphe auxiliaire construit par l'Algorithme 9

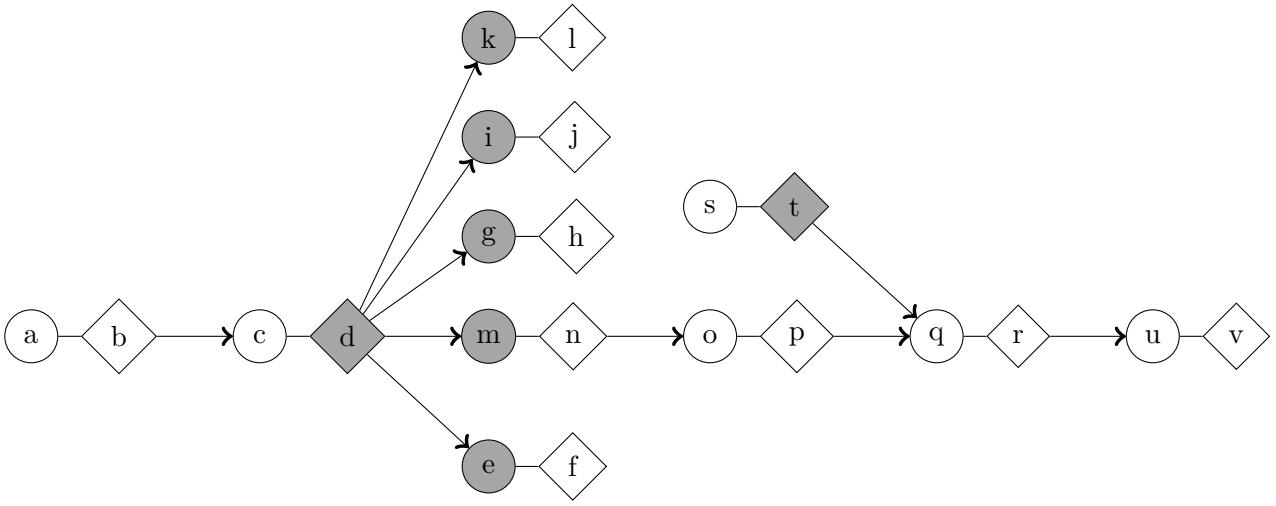


FIGURE 3.22 – 1-extensible obtenu par l'Algorithme 9

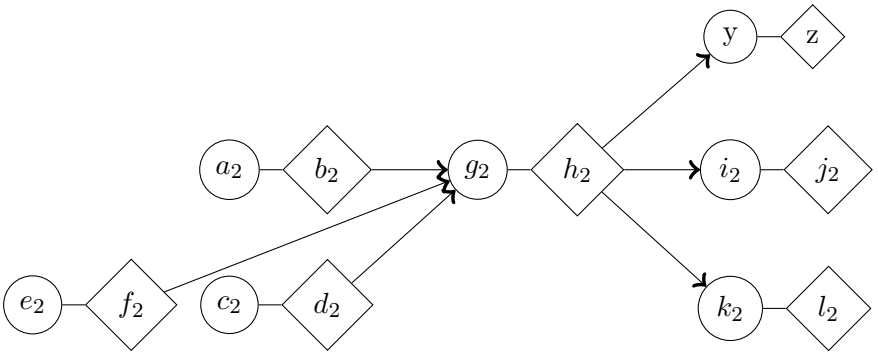


FIGURE 3.23 – Homard tel qu'il existe un chemin entre les composantes de sommets de la chaîne centrale. Dans ce graphe $\alpha(G) = 7$

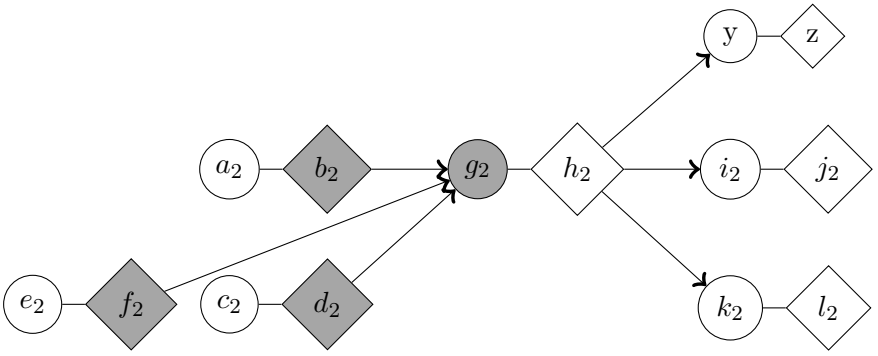


FIGURE 3.24 – 2-extensible optimal d'un homard dans lequel il existe un chemin entre les composantes des sommets de la chaîne centrale

formé est un homard tel que pour tout couple de composantes de L_5, il existe un chemin entre ces composantes dans le graphe des composantes. Donc d'après la Propriété 3.6.7 page 101, il est possible de déterminer un 2-extensible optimal à l'aide de l'algorithme 9.

Ainsi, un homard peut être décomposé en plusieurs homards dans lesquels pour tout couple de composantes de L_5, il existe un chemin entre ces composantes dans le graphe des composantes. Cette décomposition s'obtient en considérant les sous-graphes entre les couples de composantes $C_1 \in L_4$ et $C_2 \in L_3$ tels que $C_1 \rightarrow C_2$. Pour chacun de ces sous-graphes, l'Algorithme 9 détermine un 2-extensible optimal. Construisons maintenant un 2-extensible de G à l'aide de cette décomposition. On note $H = (H_1, H_2, ..., H_m)$ les sous-graphes obtenus par cette décomposition.

Soient H_i et H_j deux homards de H qui ont une composante $C \in (L_3 \cup L_4)$ en commun.

— S'il existe un 2-extensible optimal F_i de H_i qui ne contient pas de sommets de C et un 2-extensible optimal F_j de H_j qui ne contient pas de sommets de C, alors d'après la propriété 3.4.3 page 69, $F_i \cup F_j$ est un 2-extensible du graphe composé de H_i et H_j. C'est également un 2-extensible optimal car sinon l'un des deux 2-extensibles F_i ou F_j ne serait pas optimal pour son sous-graphe respectif.

— Si tout 2-extensible optimal F_i de H_i contient un sommet x de C et si tout 2-extensible optimal F_j de H_j contient un sommet x de C alors de même $F_i \cup F_j$ est un 2-extensible optimal du graphe composé de H_i et H_j.

— Si tout 2-extensible optimal F_i de H_i contient un sommet x de C et s'il existe un 2-extensible F_j de H_j qui ne contient aucun sommet de C alors de même $(F_i \cup F_j) - \{x\}$ est un 2-extensible du graphe composé de H_i et H_j. C'est aussi un 2-extensible optimal car il est de cardinal $|F_i| + |F_j| - 1$ donc de même cardinal que si F_j avait contenu x.

Ainsi, on peut déterminer un 2-extensible optimal de G par l'Algorithme 10 en considérant pour chaque composante C de $L_3 \cup L_4$ quels sont les sous-graphes H_i qui contiennent C. Puis, pour chacun de ces sous-graphes H_i, on détermine à l'aide de l'Algorithme 9 s'il existe un 2-extensible optimal F_i qui ne contient pas de sommets de C. Si c'est le cas, on sélectionne les sommets de F_i. Sinon, soit tous les 2-extensibles optimaux des H_i doivent contenir un sommet de C, auquel cas on sélectionne les sommets de ces 2-extensible, soit

il existe un 2-extensible optimal d'un des sous-graphe qui ne contient pas de sommets de C, auquel cas on sélectionne les sommets des 2-extensibles sauf ceux de C. On détermine ainsi un 2-extensible de G. Si ce 2-extensible a un cardinal strictement supérieur à 4, c'est un 2-extensible optimal et sinon le 2-extensible optimal est celui obtenu à l'aide de la Proposition 3.6.1 page 85.

□

Exemple 3.6.5. *Considérons le homard de la Figure 3.19 et appliquons l'Algorithme 10. La décomposition du homard en sous-graphes en quatre composantes de L_3 et de L_4 aboutit à quatre sous-graphes induits par les ensembles de sommets : $\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v\}$ (Figure 3.20), $\{u, v, w, x\}$, $\{w, x, y, z\}$ et $\{y, z, a_2, b_2, c_2, d_2, e_2, f_2, g_2, h_2, i_2, j_2, k_2, l_2\}$ (Figure 3.23).*

On commence par parcourir les composantes de L_3. Pour chacune des ces composantes, on considère les deux sous-graphes qui contiennent cette composante. On commence par la composante $C(u)$. Le sous-graphe de la Figure 3.20 contient un 2-extensible optimal (3.22) qui ne contient ni de sommets de composantes de L_3, ni de sommets de composantes de L_4, on sélectionne donc les sommets de ce 2-extensible. Le 2-extensible optimal $\{u, x\}$ du sous-graphe induit par l'ensemble $\{u, v, w, x\}$ contient des sommets de composantes de L_3 et de L_4, on ne sélectionne donc pas de sommets de ce sous-graphe.

On considère ensuite la composante $C(y)$. Le sous-graphe de la Figure 3.23 contient un 2-extensible optimal (Figure 3.24) qui ne contient aucun sommet de L_3 ou de L_4. On sélectionne donc les sommets de ce 2-extensible. Le 2-extensible optimal $\{x, y\}$ du sous-graphe induit par $\{y, z, a_2, b_2, c_2, d_2, e_2, f_2, g_2, h_2, i_2, j_2, k_2, l_2\}$ contient des sommets de L_3 et de L_4, on ne sélectionne donc pas de sommets de ce sous-graphe.

On considère maintenant les composantes de L_4. Pour chacune de ces composantes, on considère les deux sous-graphes qui contiennent cette composante. La seule composante de L_4 contenu dans des sous-graphes dont on n'a pas encore sélectionné des sommets est $C(x)$. Mais chacun des deux sous-graphes qui contiennent $C(x)$ contient une composante de L_3 en commun avec un autre sous-graphe dans lequel des sommets ont été sélectionnés donc on ne peut sélectionner de sommet des composantes de L_3 On retire donc les composantes de

Algorithm 10 Algorithme permettant de déterminer un 2-extensible optimal d'un homard dont les sommets sont libres

1: $F = \varnothing$
2: **for all** $C \in L_3$ **do**
3: Déterminer les sous-graphes H_i qui contiennent C, tels que dans chaque H_i, il existe un chemin entre tout couple de composantes de L_5
4: **for all** H_i **do**
5: Déterminer s'il existe un 2-extensible optimal F_i qui ne contient pas de sommet de C ni de sommet d'une composante de L_4
6: **if** F_i existe **then**
7: $F := F \cup F_i$
8: **end if**
9: **end for**
10: **if** Tous les 2-extensibles optimaux des H_i contiennent un sommet de C **then**
11: **for all** H_i **do**
12: Déterminer un 2-extensible optimal $\tilde{F}_i$ de H_i qui contient un sommet de C
13: $F := F \cup \tilde{F}_i$
14: **end for**
15: **end if**
16: **end for**
17: **for all** $C \in L_4$ **do**
18: Déterminer les H_i qui contiennent C et donc aucun sommet n'est dans F
19: **for all** H_i **do**
20: Déterminer s'il existe un 2-extensible optimal F_i qui ne contient pas de sommet de C ni de sommet d'une composante de L_3.
21: **if** F_i existe **then**
22: $F := F \cup F_i$
23: **end if**
24: **end for**
25: **if** Si les sommets d'un des H_i sont dans F **then**
26: **for all** H_i dont les sommets ne sont pas encore dans F **do**
27: Soit $\hat{F}_i$ un 2-extensible optimal de H_i auquel on a retiré les sommets des composantes de L_4 et de L_3
28: $F := F \cup \hat{F}_i$
29: **end for**
30: **else**
31: **for all** H_i **do**
32: Déterminer un 2-extensible optimal $\tilde{F}_i$ de H_i qui contient un sommet de C
33: $F := F \cup \tilde{F}_i$
34: **end for**
35: **end if**
36: **end for**
37: **return** F

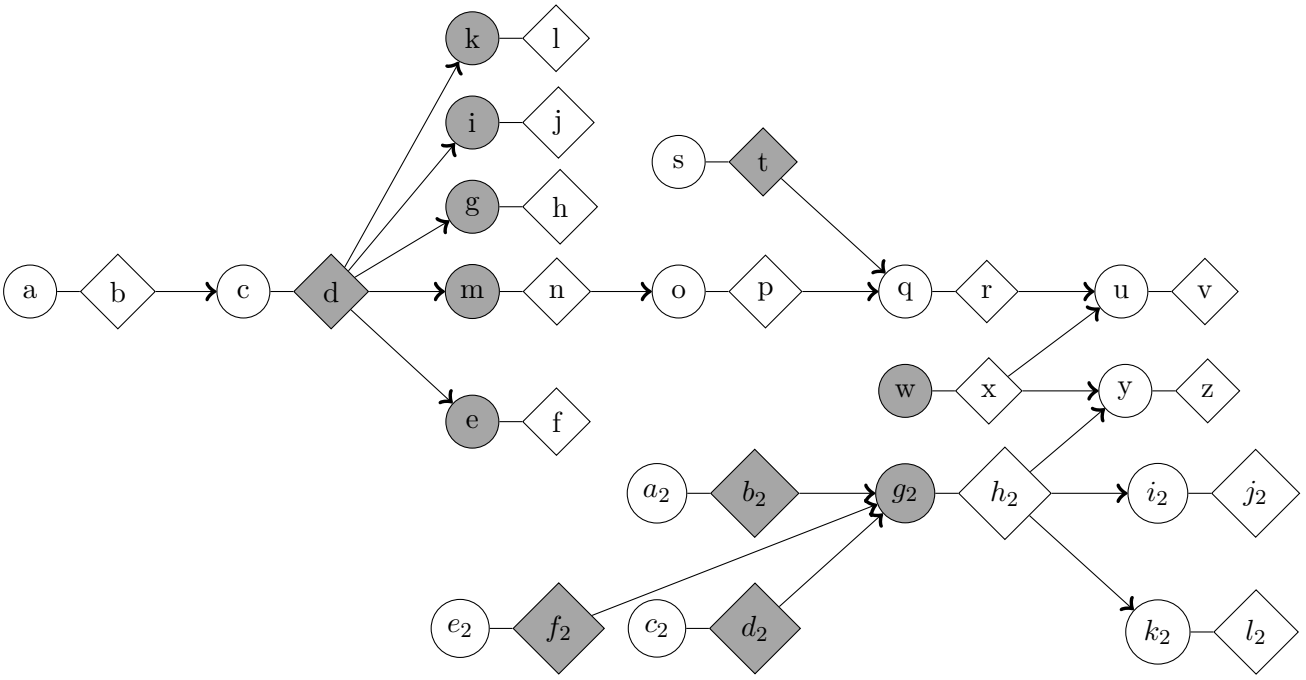


FIGURE 3.25 – 2-extensible optimal du homard de la Figure 3.19 par l'Algorithme 10

L_3 de ces sous-graphes et on recherche des 2-extensibles qui contiennent un sommet de la composante $C(x)$. Puisque les sous-graphes ne contiennent plus que la composante $C(x)$, on sélectionne le sommet x. On obtient le 2-extensible $\{d, k, i, g, m, e, t, x, b_2, d_2, f_2, g_2\}$, indiqué dans la Figure 3.25. Ce 2-extensible est de cardinal supérieur à 4, il est donc optimal.

Dans les trois propriétés suivantes, nous montrons que dans le cas d'un d-extensible de homard pour $d > 2$, certaines structures ne peuvent pas exister et que des structures optimales impliquent l'existence d'autres structures optimales. Ainsi, on pourra par la suite énumérer certaines structures en étant certains que l'une d'entre elles est optimale.

Propriété 3.6.9. *Soit $G = (B, R, E)$ un homard et soit F un d-extensible de G tel que $|F| > 2d$ et $d > 2$. Soient x,y et z trois sommets de F. Si $C(x) \to C(y)$ et $C(y) \to C(z)$ alors soit $yz \in E$, soit $xy \in E$.*

Démonstration. Supposons tout d'abord que $xy \notin E$.

D'après la Propriété 3.4.3 page 69, s'il existe deux sommets $a \in R \cap F$ et $b \in B \cap F$

tels que $C(a) \to C(b)$ alors $|F| \leqslant 2d$. Or $|F| > 2d$ donc ce cas n'est pas possible.

Supposons par l'absurde qu'il existe trois sommets x, y et z tels que $C(x) \to C(y)$, $C(y) \to C(z)$, $xy \notin E$ et $yz \notin E$. Dans ce cas, la composante $C(y)$ a au moins un prédécesseur et au moins un successeur. De plus il n'existe pas de sommet $a \in F \cap R$ tel que $ya \in E$ et $C(a) \to C(z)$, sinon $|F| \leqslant 2d$. Donc y est l'unique voisin dans $F \cap B$ de $\Gamma(y) \cap R$.

Considérons le cas $x \in B$, $y \in B$, $z \in B$. Tout stable S de $G[F]$ de cardinal d qui contient x doit contenir y et z.

Soit l'ensemble $L = (B \cap F) - \{x, y, z\}$. L'ensemble $\{x, z\} \cup (\Gamma(y) \cap F) \cup L$ est un stable donc $|\{x, z\} \cup (\Gamma(y) \cap F) \cup L| < d$ sinon il existe un stable de $G[F]$ qui n'est pas extensible par rapport à F. Or $|L| = |F \cap B| - 3$ donc $|F \cap B| < d + 1 - |\Gamma(y) \cap F|$.

Soit l'ensemble $K = (R \cap F) - (\Gamma(y) \cap F)$. L'ensemble $\{y\} \cup K$ est un stable donc $|K| + 1 < d$ sinon il existe un stable de $G[F]$ de cardinal d qui n'est pas extensible par rapport à F. Or $|K| = |R \cap F| - |\Gamma(y) \cap F|$. Donc $|F \cap R| < d - 1 + |\Gamma(y) \cap F|$. Ainsi $|F| = |F \cap R| + |F \cap B| < 2d$. Contradiction.

Considérons le cas $x \in B$, $y \in B$, $z \in R$. Tout stable S de $G[F]$ de cardinal d qui contient x doit contenir y et tout stable S de $G[F]$ de cardinal d qui contient z ne peut contenir ni x, ni y.

Soit $L = (F \cap B) - \{x, y\}$. L'ensemble $\{x\} \cup (\Gamma(y) \cup F) \cap L$ est un stable. Donc $|\{x\} \cup (\Gamma(y) \cap F) \cup L| < d$ sinon il existe un stable de $G[F]$ de cardinal d qui n'est pas extensible par rapport à F. Or, $|L| = |F \cap B| - 2$. Donc $|F \cap B| < d - |\Gamma(y) \cap F| + 1$.

Soit $K = (R \cap F) - (\Gamma(y) \cap F) - \{z\}$. L'ensemble $\{y, z\} \cup K$ forme un stable donc $|\{y, z\} \cup K| < d$ sinon il existe un stable de $G[F]$ de cardinal d qui n'est pas extensible par rapport à F. Or $|K| = |R \cap F| - |\Gamma(y) \cap F| - 1$. Donc $|R \cap F| < d + |\Gamma(y) \cap F| - 1$. Ainsi $|F| = |F \cap R| + |F \cap B| < 2d$. Contradiction.

Le cas $x \in B$, $y \in R$, $z \in R$ correspond au cas $x \in B$, $y \in B$, $z \in R$ après inversion de l'orientation du graphe des composantes. De même, le cas $x \in R$, $y \in R$, $z \in R$ correspond au cas $x \in B$, $y \in B$, $z \in B$. Ces cas ne peuvent donc pas se présenter.

Supposons maintenant que $yz \notin E$. Par inversion de l'orientation du graphe des com-

posantes, on retrouve le cas précédent donc $xy \in E$.

□

Propriété 3.6.10. *Soit $G = (B, R, E)$ un homard dont tous les sommets sont libres et soit $d > 1$. S'il existe un d-extensible optimal de G de cardinal strictement supérieur à $2d$ tel qu'il n'existe aucun triplet de sommet x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$, alors il existe un d-extensible optimal F qui ne contient aucun triplet de sommets x, y et z tels que :*

- *$C(x) \to C(y)$, $C(x) \to C(z)$, $C(y) \nrightarrow C(z)$ et $C(z) \nrightarrow C(y)$, $xy \notin E$, $xz \notin E$*
- *$C(x) \to C(z)$, $C(y) \to C(z)$, $C(y) \nrightarrow C(x)$ et $C(x) \nrightarrow C(y)$, $xz \notin E$, $yz \notin E$*

Démonstration. Soient x, y et z trois sommets de F tels que $C(x) \to C(y)$, $C(x) \to C(z)$, $C(y) \nrightarrow C(z)$ et $C(z) \nrightarrow C(y)$, $xy \notin E$, $xz \notin E$.

Puisque $C(y) \nrightarrow C(z)$ et $C(z) \nrightarrow C(y)$, soit $C(y) \in L_1$ et $C(z) \in L_5 \cup L_3$, soit $C(z) \in L_1$ et $C(y) \in L_5 \cup L_3$, soit $C(y) \in L_1$ et $C(z) \in L_1$. Supposons sans perte de généralités que $C(y) \in L_1$, ce qui implique $\Gamma(y) = 2$ si $y \in R$ et $\Gamma(y) = 1$ sinon.

1. Si $x \in R$ et $y \in B$ ou $z \in B$ alors d'après la Propriété 3.4.3 page 69, $|F| \leqslant 2d$. Contradiction.
2. Si $x \in R$, $y \in R$ et $z \in R$ alors d'après la Propriété 3.4.3 page 69, $\alpha(G[F - (\{x\} \cup \Gamma(y))]) < d$. Donc $|F \cap R| \leqslant d$ et $|F \cap B| \leqslant d - 1 + |\Gamma(y) - 1|$. Or $\Gamma(y) = 2$ donc $|F \cap B| \leqslant d$ et $|F| \leqslant 2d$. Contradiction.
3. Si $x \in B$, $y \in R$ et $z \in R$ alors soit $k \in R \cap F$ tel que $C(x) \to C(k)$, $V(C(k)) \cap F \neq \varnothing$ et le chemin entre $C(x)$ et $C(k)$ est le plus long possible. Soit $p \in B$ tel que $C(x) \to C(p)$, $C(p) \in L_5 \cup L_3$ et $pk \in E$. Il n'est pas possible que $C(z) \to C(p)$ car sinon $C(z) \to C(y)$. Soit $s \in R$ tel que $C(s) \in L_5$, $C(p) \to C(s)$ et $ps \in E$. Montrons que $(F - \{x, z\}) \cup \{p, s\}$ est un d-extensible. D'après la Propriété 3.6.9, il n'existe pas de sommet $a \in B \cap F$ tel que $C(p) \to C(a)$ ou que $C(a) \to C(x)$. Supposons qu'il existe un sommet $a \in B \cap R$ tel que $C(a) \to C(p)$. Dans ce cas, $C(a) \to C(k)$ et d'après la Propriété 3.4.3 page 69, $\alpha(G[F - (\Gamma(a) \cup \Gamma(k))]) < d$ donc $|F \cap B| < d + |\Gamma(k) - 1|$ et $|F \cap R| < d + |\Gamma(a) - 1|$. Or, puisque $C(a) \to C(x)$, $C(a) \in L_2$ donc $\Gamma(a) = 2$. De plus,

$\Gamma(k) = 2$ donc $|F| \leqslant 2d$. Contradiction. Donc il n'existe pas de sommet $a \in B \cap F$ tel que $C(a) \to C(p)$. Par définition de k, il n'existe pas de sommet $b \in R \cap F$ tel que $C(p) \to C(b)$. Montrons maintenant qu'il n'existe pas de sommet $b \in R \cap F$ tel que $C(b) \to C(p)$. D'après la Propriété 3.4.3 page 69, $\alpha(G[F - (\{b\} \cup \Gamma(k))]) < d$. Donc $|F \cap R| \leqslant d$ et $|F \cap B| \leqslant d - 1 + |\Gamma(k) - 1|$. Or $\Gamma(k) = 2$ donc $|F| \leqslant 2d$. Contradiction. Ainsi, il n'existe pas dans F de sommets a tels que $C(a) \to C(p)$ ou $C(p) \to C(a)$ sans que $pa \in E$. Donc $(F - \{x, z\}) \cup \{p, s\}$ est un d-extensible.

4. Si $x \in B$, $y \in B$ et $z \in R$ alors soit $k \in R \cap F$ tel que $C(x) \to C(k)$, $V(C(k)) \cap F \neq \varnothing$ et le chemin entre $C(x)$ et $C(k)$ est le plus long possible. Soit $p \in B$ tel que $C(x) \to C(p)$, $C(p) \in L_5 \cup L_3$ et $pk \in E$. Il n'est pas possible que $C(z) \to C(p)$ car sinon $C(z) \to C(y)$. Soit $s \in R$ tel que $C(s) \in L_5$, $C(p) \to C(s)$ et $ps \in E$. On montre de la même façon que le point précédent que $(F - \{x, z\}) \cup \{p, s\}$ est un d-extensible.
5. Si $x \in B$, $y \in R$ et $z \in B$, la démonstration est identique au cas précédent.
6. Si $x \in B$, $y \in B$ et $z \in B$, la démonstration est identique au cas $x \in R$, $y \in R$ et $z \in R$.

Soient x, y et z trois sommets de F tels que $C(x) \to C(y)$, $C(y) \to C(z)$, $C(y) \nrightarrow C(x)$ et $C(x) \nrightarrow C(y)$, $xy \notin E$, $yz \notin E$.

7. Si $x \in R$ et $z \in B$ ou $y \in R$ et $z \in B$ alors $|F| \leqslant 2d$. Contradiction.
8. Si $x \in R$, $y \in B$, $z \in R$, on retrouve le cas 5. après inversion de l'orientation du graphe des composantes

□

Propriété 3.6.11. *Soit $G = (B, R, E)$ un homard dont tous les sommets sont libres et soit $d > 2$. S'il existe un d-extensible optimal F de G de cardinal strictement supérieur à $2d$ qui ne contient pas de sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ alors il existe un d-extensible de même cardinal que F tel que pour tout couple de sommets x et y tels que $C(x) \to C(y)$, $xy \in E$.*

Démonstration. Soient deux sommets x et y de F tels que $C(x) \to C(y)$ et $xy \notin E$. D'après

la Propriété 3.6.10 page 114, il n'existe pas de sommets $z \in F$ tels que $C(z) \to C(y)$ ou $C(x) \to C(z)$.

Considérons les différents cas possibles :

— Si $x \in R$ et $y \in B$ alors d'après la propriété 3.4.3 page 69, $|F| \leqslant 2d$. Contradiction.

— Si $x \in B$ et $y \in B$ alors il existe $p \in R$ tel que $C(x) \to C(p)$ et $xp \in E$. Il n'existe pas de sommet $z \in F$ distinct de y tel que $C(z) \to C(p)$ ou $C(p) \to C(z)$ donc d'après la propriété 3.4.3, $(F - \{y\}) \cup \{p\}$ est un d-extensible.

— Si $x \in R$ et $y \in R$ alors on retrouve le cas précédent après inversion de l'orientation du graphe des composantes.

— Si $x \in B$ et $y \in R$ alors il existe $p \in R$ tel que $C(x) \to C(p)$ et $xp \in E$ Il n'existe pas de sommet $z \in F$ distinct de y tel que $C(z) \to C(p)$ ou $C(p) \to C(z)$ donc d'après la propriété 3.4.3, $(F - \{y\}) \cup \{p\}$ est un d-extensible.

□

Les deux propriétés suivantes traitent le cas d'un d-extensible de homard pour $d > 2$ d'une manière similaire à celle utilisée pour le cas du 2-extensible.

Propriété 3.6.12. *Soit $G = (B, R, E)$ un homard dont tous les sommets sont libres et tel que pour tout couple de composantes C_1 et C_2 de L_5, soit $C_1 \to C_2$, soit $C_2 \to C_1$. Un d-extensible optimal de G peut être déterminé en temps polynomial pour $d \geqslant 3$.*

Démonstration. D'après la Propriété 3.6.11, s'il existe un d-extensible optimal qui ne contient pas de sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ alors s'il existe deux sommets x et y tels que $C(x) \to C(y)$, $xy \in E$. Ainsi, s'il existe un d-extensible optimal qui ne contient pas de sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ alors c'est aussi un 2-extensible et d'après la Propriété 3.6.8 il peut être déterminé en temps polynomial.

Considérons le cas où le d-extensible optimal de G n'est pas un 2-extensible. On sait que si un d-extensible ne contient pas de sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ alors il existe un 2-extensible de même cardinal. Donc pour qu'un d-extensible de G ne soit pas un 2-extensible, il doit contenir trois sommets x, y et z tels que $C(x) \to C(y)$

et $C(y) \to C(z)$. Or, d'après la Propriété 3.6.9, il n'est pas possible que $xy \notin E$ et $yz \notin E$. Donc soit $xy \in E$, soit $yz \in E$.

Montrons que l'Algorithme 11 retourne un d-extensible optimal de G s'il existe un d-extensible de cardinal supérieur à $2d$. Cet algorithme commence par déterminer un 2-extensible optimal. Il cherche ensuite à déterminer s'il existe un d-extensible qui contient trois sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$. Si $yz \in E$ alors $y \in B$, $C(y) \in L_5$ et $z \in R$. Donc $x \in B$ car si $x \in R$, comme $C(x) \to C(y)$, d'après la Propriété 3.4.3 page 69, $|F| \leqslant 2d$. De même, si $xy \in E$ alors $x \in B$, $y \in R$ et $z \in R$.

Considérons le cas $yz \in E$. La composante $C(x)$ doit appartenir à L_4 ou L_5. En effet, elle ne peut appartenir à L_1 ou à L_3 puisqu'elle doit avoir des successeurs et si elle appartient L_2, d'après la Propriété 3.4.3 page 69 $\alpha(G[F-(\Gamma(x)\cup\{y\})]) < d$. Or $|\Gamma(x)| = 1$ donc $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d$ donc $|F| \leqslant 2d$ ce qui n'est pas possible. La composante $C(y)$ doit appartenir à L_5 ou L_3 car $yz \in E$ et $C(x) \to C(z)$. Pour tous les sommets de $L_4 \cup L_5$, l'Algorithme 11 parcourt les sommets y de $L_5 \cup L_3$ tels que $C(x) \to C(y)$, sélectionne les voisins de y et détermine le d-extensible de plus grand cardinal possible qui contient x, y et les voisins de y. En effet, d'après la Propriété 3.6.9, un d-extensible F qui contient x, y et z ne peut pas contenir de sommets de composantes C telles que $C \to C(y)$ ou que $C(y) \to C$. De plus, d'après la Propriété 3.4.3 page 69, $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d + |\Gamma(x)| - 1$. Si on note $\hat{L_1}$ le sous-ensemble de L_1 auquel on a retiré les composantes C telles que $C(y) \to C$ et $\hat{L_2}$ le sous-ensemble de L_2 auquel on a retiré les composantes C telles que $C \to C(y)$, on peut sélectionner un sommet par composante de $\hat{L_1} \cup \hat{L_2}$ tant que $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d + |\Gamma(x)| - 1$. Il reste à considérer les composantes C de L_1 telles que $C(x) \to C$ et $C(y) \nrightarrow C$. Dans ces composantes, on peut à nouveau sélectionner un sommet tant que $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d + |\Gamma(x)| - 1$. L'algorithme 11 construit donc le plus grand d-extenisble possible qui contient x, y et z.

Le cas $xy \in E$ revient au cas $yz \in E$ après inversion de l'orientation du graphe des composantes de G.

L'Algorithme 11 détermine un 2-extensible optimal et détermine pour tous les triplets x, y et z un d-extensible de plus grand cardinal possible qui les contient, si c'est possible. Il détermine donc un d-extensible optimal de G.

□

Algorithm 11 Algorithme permettant de déterminer un 3-extensible optimal d'un homard dans lequel il existe un chemin dans G_c entre tout couple de composantes de L_5

1: $F :=$ 2-extensible retourné par l'algorithme 10
2: $F_{cur} := \varnothing$
3: **for all** $C \in L_5 \cup L_4$ **do**
4: $\quad F_{cur} := V(C) \cap B$
5: $\quad$ **for all** $C_2 \in L_5 \cup L_3$ tel que $C \to C_2$ **do**
6: $\quad\quad F_{cur} := F_{cur} \cup (V(C_2) \cap B)$
7: $\quad\quad$ **for all** C_3 tel que $C_2 \to C_3$ et C_2 et C_3 sont voisins dans G_c **do**
8: $\quad\quad\quad F_{cur} := V(C_3) \cap R$
9: $\quad\quad$ **end for**
10: $\quad\quad$ Soit $\hat{L_1}$ l'ensemble des composantes C_i de L_1 telles que $C_i \nrightarrow C_2$
11: $\quad\quad$ Soit $\hat{L_2}$ l'ensemble des composantes C_i de L_2 telles que $C \nrightarrow C_i$
12: $\quad\quad$ **for all** $C_i \in (\hat{L_1} \cup \hat{L_2})$ **do**
13: $\quad\quad\quad$ **if** $|F_{cur} \cap B| < d$ **then**
14: $\quad\quad\quad\quad F_{cur} := F_{cur} \cup (V(C_i) \cap B)$
15: $\quad\quad\quad$ **else if** $|(F_{cur} \cap R) - V(\Gamma(C))| < d$ **then**
16: $\quad\quad\quad\quad F_{cur} := F_{cur} \cup (V(C_i) \cap R)$
17: $\quad\quad\quad$ **end if**
18: $\quad\quad$ **end for**
19: $\quad\quad$ **for all** $C_i \in L_2$ tel que $C \to C_i$ et $C_2 \nrightarrow C_i$ **do**
20: $\quad\quad\quad$ **if** $|F_{cur} \cap B| < d$ **then**
21: $\quad\quad\quad\quad F_{cur} := F_{cur} \cup (V(C_i) \cap B)$
22: $\quad\quad\quad$ **else if** $|(F_{cur} \cap R) - V(\Gamma(C))| < d$ **then**
23: $\quad\quad\quad\quad F_{cur} := F_{cur} \cup (V(C_i) \cap R)$
24: $\quad\quad\quad$ **end if**
25: $\quad\quad$ **end for**
26: $\quad\quad$ **if** $|F_{cur}| > |F|$ **then**
27: $\quad\quad\quad F := F_{cur}$
28: $\quad\quad$ **end if**
29: $\quad$ **end for**
30: **end for**
31: Inverser l'orientation du graphe des composantes de G et exécuter la boucle une nouvelle fois
32: **return** F

Exemple 3.6.6. *Considérons le homard de la Figure 3.20 et appliquons l'Algorithme 11 pour déterminer un* 3*-extensible optimal. L'algorithme commence par effectuer une boucle sur les composantes de* $L_4 \cup L_5$ *puis détermine pour chacune des composantes le meilleur* 3*-extensible qui contient trois sommets* x*,* y *et* z *tels que* $C(x)$ *soit la composante examinée*

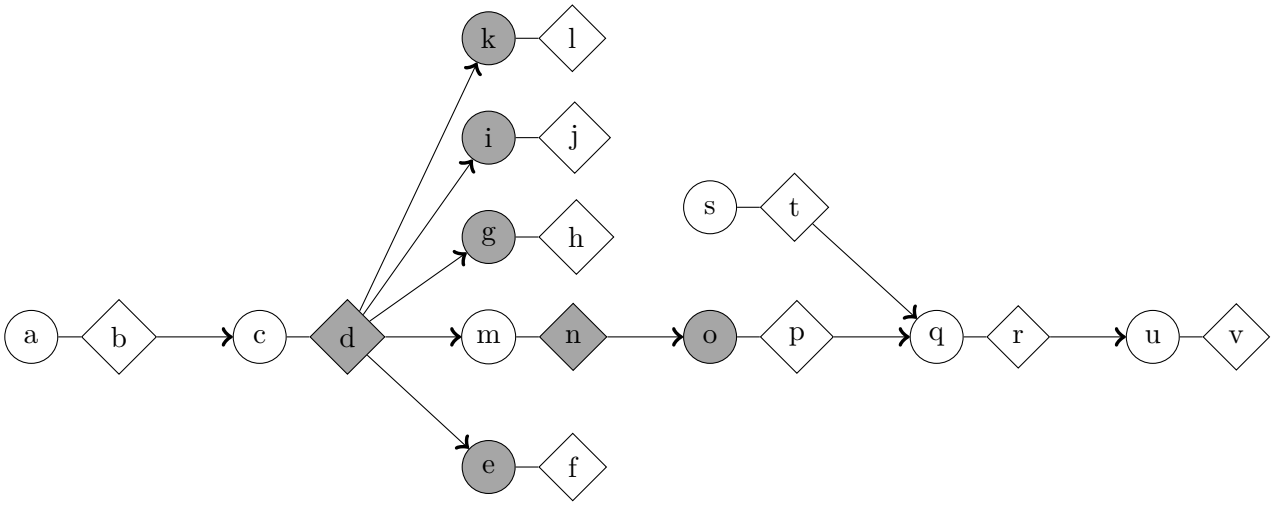


FIGURE 3.26 – 3-extensible optimal d'un homard

dans la boucle, $C(x) \to C(y)$, $C(y) \to C(z)$, $xy \notin E$ *et* $yz \in E$.

— *Pour* $C(b)$ *le* 3*-extensible maximum déterminé est* $\{b, d, m\}$.

— *Pour* $C(d)$ *le* 3*-extensible maximum déterminé est* $\{d, n, o, k, i, g, e\}$.

— *Pour* $C(n)$ *le* 3*-extensible maximum déterminé est* $\{n, p, q, t\}$.

— *Pour* $C(p)$ *le* 3*-extensible maximum déterminé est* $\{p, r, u\}$.

— *Pour* $C(r)$ *le* 3*-extensible maximum déterminé est* $\{r, u, k, i, g, e\}$.

Le meilleur 3*-extensible déterminé est* $\{d, n, o, k, i, g, e\}$*, de cardinal* 7*. Le* 2*-extensible optimal est de cardinal* 7 *donc* $\{d, n, o, k, i, g, e\}$ *est un* 3*-extensible optimal. Il est indiqué dans la Figure 3.26.*

Notons que si on retire du homard la composante $C(t)$*, un* 2*-extensible optimal est alors de cardinal* 6 *et dans ce cas, le* 3*-extensible déterminé est optimal.*

Pour $d = 4$*, l'algorithme rechercherait un* 4*-extensible et retournerait l'ensemble* $\{d, n, o, k, i, g, e, t\}$ *de cardinal* 8*. Il ne retournerait donc pas un meilleur* 4*-extensible que celui de par la Propriété 3.6.1 page 85.*

Propriété 3.6.13. *Soit* $G = (B, R, E)$ *un homard dont tous les sommets sont libres. Un* d*-extensible optimal de* G *peut être déterminé en temps polynomial pour* $d \geqslant 3$.

Démonstration. D'après la Proposition 3.6.1 page 85, il existe un d-extensible de G de cardinal $2d$. On cherche donc à déterminer s'il existe un d-extensible de cardinal supérieur à $2d$. De façon analogue à la preuve de la Propriété 3.6.8, on considère les sous-graphes de G entre deux composantes de L_3 et de L_4. Dans ces sous-graphes, on peut déterminer un d-extensible optimal d'après la Propriété 3.6.12.

Soient deux sous-graphes G_1 et G_2. Supposons qu'il existe un d-extensible F de G qui contient deux sommets $x_1 \in B$ et $y_1 \in R$ de G_1 tels que $x_1y_1 \notin E_1$ et $C(x_1) \to C(y_1)$ et deux sommets $x_2 \in B$ et $y_2 \in R$ de G_2 tels que $x_2y_2 \notin E_2$ et $C(x_2) \to C(y_2)$. D'après la Propriété 3.4.3 page 69, $\alpha(G[F - (\Gamma(x_1) \cup \Gamma(y_1))]) < d$ et $\alpha(G[F - (\Gamma(x_2) \cup \Gamma(y_2))]) < d$. Donc $|F \cap R| \leqslant d-1+|\Gamma(x_1)|$. Considérons le stable de $G[F]$ qui contient x_2, y_2, $\Gamma(y_1) \cap F$ et un ensemble K de sommets de R autres que y_1 et y_2 tel que $F \cap R = K \cup \{y_1, y_2\}$. Ce stable n'est pas extensible donc $|\Gamma(y_1) \cap F|+|K|+2 < d$. Donc $|\Gamma(y_1) \cap F|+|F \cap R|-2+2 < d$. Donc $|F \cap R| < d-|\Gamma(y_1) \cap F|$. Or $|F \cap B| < d+|(\Gamma(y_1) \cap F)|$ donc $|F| = |F \cap B|+|F \cap R| < 2d$. Contradiction. Ainsi, un seul sous-graphe peut contenir deux sommets x et y tels que $C(x) \to C(y)$ et $xy \notin E$.

Montrons que pour chaque sous-graphe G_i de G entre deux composantes de L_3 et de L_4, l'Algorithme 12 détermine le plus grand d-extensible de G qui contient trois sommets de G_i x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ s'il existe un tel d-extensible de G_i de cardinal supérieur à un 2-extensible de G_i.

Pour chaque sous-graphe G_i, l'Algorithme 12 commence par utiliser l'Algorithme 11 pour déterminer s'il existe un d-extensible F_i de G_i avec un cardinal supérieur à un 2-extensible de G_i. Si c'est le cas, on considère $\hat{G}$ le sous-graphe de G qui contient tous les sommets de G sauf ceux de G_i. Un d-extensible de G ne peut contenir de sommets x, y et z tels que $C(x) \to C(y)$ et $C(y) \to C(z)$ dans deux sous-graphes entre deux composantes de L_3 et de L_4. On détermine donc un 2-extensible $\hat{F}$ de $\hat{G}$. D'après la Propriété 3.4.3 page 69, un d-extensible F qui contient F_i doit respecter $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d + |\Gamma(x) \cap F|$. De plus, puisque F contient F_i, $|F| \leqslant |F_i \cup \hat{F}|$. Dans $\hat{F}$, pour tout sommet a tel que $\Gamma(a) \cap F = \varnothing$, il n'existe pas de sommet $b \in \hat{F}$ tel que $C(a) \to C(b)$ ou que $C(b) \to C(a)$. Donc ces sommets a peuvent appartenir aussi bien à B qu'à R. Pour construire F, on souhaite ajouter des sommets de $\hat{F}$ à F_i. L'algorithme commence par ajouter les sommets

de $\hat{F}$ dont un de leur voisin est aussi dans $\hat{F}$. Puis tant que c'est possible, on considère les sommets restants de $\hat{F}$ et on ajoute à F_i des sommets de B et de R, tant que $|F \cap B| \leqslant d$ et $|F \cap R| \leqslant d + |\Gamma(x) \cap F|$. On construit ainsi le d-extensible qui contient F_i de plus grand cardinal possible. En considérant tous les F_i possibles, l'Algorithme 12 détermine ainsi un d-extensible optimal de G.

□

Exemple 3.6.7. *Considérons le homard de la figure 3.19 pour déterminer un* 3*-extensible optimal. L'algorithme 10 commence par déterminer un* 2*-extensible optimal. Comme indiqué dans l'exemple qui suit l'algorithme 10, il trouve un* 2*-extensible de cardinal* 12*.*

L'algorithme considère ensuite le 3*-extensible de la figure 3.26. On ne peut ajouter aucun sommet de B car sinon on aurait un stable qui ne serait pas extensible avec d et o. De même, on ne peut ajouter aucun sommet de R. Donc le* 2*-extensible optimal est aussi un* 3*-extensible optimal.*

Corollaire 3.6.1. *D'après les Propriétés 3.6.3, 3.6.8 et 3.6.13, déterminer un d-extensible d'un homard dont tous les sommets sont libres se fait en temps polynomial.*

3.7 Perspectives

Dans ce chapitre, nous avons caractérisé les stables extensibles et les d-extensibles dans les graphes bipartis. Nous avons proposé des bornes inférieures du cardinal maximal d'un d-extensible dans le cas d'un graphe biparti quelconque et dans le cas d'un arbre. Des cas particuliers ont été résolus, les grilles, les cycles et certaines classes d'arbres comme les homards. Si le graphe biparti contient des sommets forcés, nous avons montré comment déterminer un d-extensible optimal pour $d \geqslant \frac{\phi(G)}{2}$. Nous avons également montré que pour les graphes bipartis dont le graphe des composantes est un arbre, déterminer un 1-extensible optimal est un problème polynomial.

Le problème reste ouvert dans de nombreux cas. Les d-extensibles tels que $d > 1$ des arbres arbitraires ne sont pas traités. La recherche d'un 1-extensible dans le cas où le graphe des composantes est un arbre utilise la programmation mathématique et un algorithme

Algorithm 12 Algorithme permettant de déterminer un d-extensible optimal d'un homard dont tous les sommets sont libres pour $d > 2$

$F :=$ 2-extensible retourné par l'Algorithme 10
$F_{cur} := \varnothing$
for all Sous-graphe G_i entre deux composantes de L_3 et de L_4 **do**
  $F_{cur} :=$ d-extensible retourné par l'utilisation de l'Algorithme 11 sur le sous-graphe G_i
  if F_{cur} contient deux sommets x et y tels que $C(x) \to C(y)$ et $xy \notin E$ **then**
    Soit $\hat{G}$ le sous-graphe constitué des sommets de G qui n'appartiennent pas à G_i
    Soit $\hat{F}$ le 2-extensible optimal de $\hat{G}$ retourné par l'algorithme 10
    Soit (X, Y, Z) la tri-partition de G telle que :
    — X contient les sommets $a \in B \cap \hat{F}$ tel qu'il existe un sommet $b \in F$ tel que $ab \in E$
    — Y contient les sommets $a \in R \cap \hat{F}$ tel qu'il existe un sommet $b \in F$ tel que $ab \in E$
    — Z contient les sommets de $\hat{F}$ qui ne sont ni dans X, ni dans Y
    for all $a \in X$ **do**
      if $|F \cap B| < d$ **then**
        $F_{cur} := F_{cur} \cup \{a\}$
      end if
    end for
    for all $a \in Y$ **do**
      if $|F \cap R| < d + |\Gamma(x) \cap F|$ **then**
        $F_{cur} := F_{cur} \cup \{a\}$
      end if
    end for
    for all $a \in Z$ **do**
      if $|F \cap B| < d$ **then**
        $F_{cur} := F_{cur} \cup (V(C(a)) \cap B)$
      else if $|F \cap R| < d + |\Gamma(x) \cap F|$ **then**
        $F_{cur} := F_{cur} \cup (V(C(a)) \cap B)$
      end if
    end for
    if $|F_{cur}| > |F|$ **then**
      $F := F_{cur}$
    end if
  end if
end for
return F

dédié reste encore à écrire. Dans le cas des graphes bipartis quelconques, il reste à traiter les d-extensibles de cardinal strictement supérieur à $2d$ et les d-extensibles de cardinal $2d$ qui contiennent plusieurs sommets d'une même composante.

Chapitre 4

d-bloqueurs de coût minimal de stables de cardinal maximal dans les arbres

4.1 Introduction

Dans le chapitre précédent, nous avons considéré un système dans lequel nous tentions de résister aux pannes. Les d-extensibles permettent ainsi de maintenir le système dans un état de fonctionnement optimal jusqu'à un certain seuil de panne. Dans ce chapitre, nous considérons cette fois le point de vue d'un attaquant qui veut s'en prendre au système.

Pour dégrader le fonctionnement optimal du système, un attaquant peut retirer des éléments. On suppose ici que l'agresseur a un objectif précis et souhaite réduire la valeur optimale du système d'un certain seuil. Un d-bloqueur est un ensemble d'éléments qui, s'ils sont retirés, diminue la valeur optimale du système d'une valeur d. L'assaillant doit donc rechercher un d-bloqueur pour atteindre son objectif. Mais chaque retrait a un coût, l'attaquant a donc intérêt à déterminer un d-bloqueur de coût minimal pour accomplir son but le plus facilement possible.

Dans ce chapitre, nous nous intéressons aux d-bloqueurs de stables de coût minimal. Après un rappel des travaux sur différents problèmes de d-bloqueurs, nous nous intéressons aux arbres. Nous caractérisons les d-bloqueurs de stable dans les arbres puis nous montrons que déterminer un d-bloqueur de coût minimal dans un arbre qui ne contient pas de sommets exclus est polynomial.

4.2 Définitions

Soit Π un problème d'optimisation combinatoire défini sur un ensemble fini M=(m_1, m_2, ... , m_n). Soit P une propriété sur cet ensemble telle que $X \subset M$ satisfait P si et seulement si X est une solution admissible de Π. Étant donné $L \subseteq M$, on définit C_L l'ensemble des sous-ensembles de L qui satisfont P. Ainsi, l'ensemble des solutions admissibles de M, c'est à dire l'ensemble des sous-ensembles de M qui satisfont P, est noté C_M.

Á chaque élément m de M sont associées deux valeurs positives : un coût $c(m)$ et un poids $w(m)$. Le poids (respectivement le coût) d'un sous-ensemble de M est la somme des poids (respectivement des coûts) des éléments de ce sous-ensemble. On note $\nu_w(C_M)$ le poids maximum d'un élément de C_M et $\delta_w(C_M)$ le cardinal maximum d'un élément de C_M de poids maximum. Autrement dit, $\nu_w(C_M) = max\{w(C_j)|C_j \in C_M\}$ et $\delta_w(C_M) = max\{|C_j||C_j \in C_M, w(C_j) = \nu_w(C_M)\}$.

Soit un entier $1 \leqslant d \leqslant \nu_w(C_M)$. Un *d-bloqueur* de Π est un sous-ensemble $B \subset M$ tel que, si on retire les éléments de B de V, on obtient $\nu_w(C_{M-B}) \leqslant \nu_w(C_M) - d$. On s'intéresse au problème suivant :

— BLOCK_Π(w, c, d) :

  — Instance : Un ensemble fini M=(m_1, m_2,..., m_n), une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M| * \max_{i=1..n} c(m_i)$.

  — Question Est-ce qu'il existe un ensemble $B \subseteq M$ tel que $\nu_w(C_{M-B}) \leqslant \nu_w(C_M) - d$ et $|B| \leqslant k$? Autrement dit, existe-t-il un d-bloqueur de problème Π de coût inférieur ou égal à k ?

4.3 État de l'art

Des d-bloqueurs appliqués à différents problèmes ont été étudiés. Dans [64], il est montré que le problème des d-bloqueurs de cliques est NP-difficile. Un algorithme de détermination des d-bloqueurs de cliques y est également proposé. Dans [81], il est montré

que déterminer un d-bloqueur de couplage est NP-difficile dans un graphe biparti mais polynomial dans le cas des grilles et des arbres [68]. Il existe une modélisation générale du problème de recherche de d-bloqueur par la programmation biniveau [32]. Le problème du d-bloqueur pour le p-centre et le p-médian ne sont pas 1.36-approximable [14]. Le problème du d-bloqueur pour le nombre chromatique est NP-difficile, de même que le d-bloqueur pour le nombre de stabilité [12]. L'ensemble des travaux sur les d-bloqueurs de stables est résumé dans le tableau à la fin de la section.

Une variante du problème du d-bloqueur qui consiste à retirer exactement k éléments en maximisant la dégradation du système a aussi été étudiée. Le cas où on cherche à retirer exactement k éléments et à maximiser la dégradation est le problème des k éléments les plus vulnérables. Le cas où on cherche à retirer au plus k éléments et à maximiser la dégradation est un problème d'interdiction.

La majorité des problèmes d'interdiction étudiés porte sur l'interdiction de plus courts chemins [79], [45], [47] et de flots [69], [2]. Ces problèmes sont NP-difficiles [6], [80]. Dans [52], un algorithme, basé sur une décomposition de Benders, de résolution d'interdiction de chemins est proposé. Dans [24], les auteurs étudient l'interdiction d'un projet de fabrication d'armes nucléaires. Un modèle général des problèmes d'interdiction par un programme mathématique biniveau a été proposée dans [38]. Ce modèle est ensuite appliqué à plusieurs problèmes d'interdiction qui sont résolues par l'approche de [39].

Le problème des k éléments les plus vulnérables a été étudié dans [14] pour les problèmes de p-centre et du p-médian où il est démontré qu'il est NP-difficile d'approximer ces problèmes avec un rapport respectivement de $(\frac{4}{3} - \epsilon)$ et $(\frac{7}{5} - \epsilon)$ pour tout $\epsilon > 0$. Le problème des k éléments les plus vulnérables est NP-difficile pour les problèmes de l'arbre couvrant minimal [44], du plus court chemin [7], du flot maximum [80], du stable de poids maximal [13] mais est polynomial pour celui du stable de cardinal maximal.

Dans ce chapitre, nous nous intéressons au problème $BLOCK_{STABLE}(w, c, d)$. L'ensemble des résultats de complexité concernant les d-bloqueurs de stables optimaux est résumé dans le tableau suivant.

Graphe	$w = c = 1$	$w = 1$, c	w, $c = 1$	w, c
Biparti	P [32]	?	NP-difficile [11]	NP-difficile [31]
Arbre	P [32]	?	P [11]	NP-difficile [31]
Largeur d'arbre bornée	P [11]	?	P [11]	NP-difficile [31]
Cographe	P [11]	?	P [11]	NP-difficile [31]
Cobiparti	P [31]	P [31]	P [31]	P [31]
Split	NP-difficile [31]	NP-difficile [31]	NP-difficile [31]	NP-difficile [31]

TABLE 4.1 – Complexité de $BLOCK_{STABLE}(w, c, d)$

On constate que la colonne "$w = 1, c$" contient encore des problèmes ouverts. Nous allons donc dans la suite du chapitre nous intéresser à $BLOCK_{STABLE}(w = 1, c, d)$ lorsque le graphe est un arbre.

4.4 d-bloqueurs de coût minimal de stables dans les arbres

Dans cette section, on considère le problème $BLOCK_{STABLE}(w = 1, c, d)$. La caractérisation des stables optimaux dans les graphes bipartis décrite au chapitre 1 sera réutilisée. Nous prouvons dans un premier temps une caractérisation des d-bloqueurs dans les arbres puis nous montrons que, la recherche d'un d-bloqueur de coût minimal peut être réalisée en temps polynomial lorsque le graphe est un arbre dont aucun sommet n'est exclus.

Orientons les arêtes du graphe G dans le sens du graphe des composantes G_c : soit $\vec{G}$ le graphe orienté obtenu. On remarque que les extrémités des arcs $(u, v) \in R \times B$ de $\vec{G}$ définissent les couples de sommets d'une même composante et que les arêtes correspondantes forment un couplage parfait, noté M, du graphe G. Les arcs $(v, u) \in B \times R$ de $\vec{G}$ correspondent aux arêtes de G exclues dans un couplage optimal.

Soient deux sommets u et v de G, on définit la relation suivante :

$$u \prec v \Leftrightarrow \text{il existe un chemin de } u \text{ à } v \text{ dans } \vec{G}$$

Propriété 4.4.1. *Soit $G = (B, R, E)$ un arbre dont tous les sommets sont libres et $\vec{G} = (B, R, A)$ le graphe orienté associé. Tout d-bloqueur contient d couples $(u, v) \in R \times B$ vérifiant $u \prec v$, qui sont les extrémités de d chemins sommets-disjoints de $\vec{G}$. De plus la cardinalité minimale d'un d-bloqueur est $2d$.*

Démonstration. Montrons que d couples définis comme dans la proposition forment un d-bloqueur.

Tous les sommets de G sont libres donc $\mu(G) = |R| = |B|$. Puisque G est un arbre qui admet un couplage parfait, M est unique. Rappelons que G est biparti donc $\alpha(G)+\mu(G) = |R| + |B|$ et finalement, $\alpha(G) = \mu(G) = |R| = |B|$.

Tout chemin de u à v, $(u, v) \in R \times B$, dans $\vec{G}$ correspond à une chaîne de longueur impaire de G. Les deux arêtes à l'extrémité de cette chaîne appartiennent à M (les deux arêtes sont confondues si $(u, v) \in A$). En retirant les deux sommets extrémités u et v on obtient un couplage parfait M' du graphe $G - \{u, v\}$ en conservant les arêtes de M n'appartenant pas à la chaîne $\pi[u, v]$ et en prenant les arêtes qui ne sont pas dans M dans cette chaîne. On a $|M'| = |M| - 1$. Les d chaînes sont disjointes donc en retirant tous les couples (u, v) de G, le graphe résultant G' a un couplage parfait de cardinalité $\mu(G') = \mu(G) - d$ pour $|R| + |B| - 2d$ sommets. Donc $\alpha(G') = \mu(G') = |R| - d$ et ainsi $\alpha(G') = \alpha(G) - d$.

Montrons que tout d-bloqueur optimal L_d est constitué d'exactement d couples $(u, v) \in R \times B,\ u \prec v$ qui sont les extrémités de d chemins sommets-disjoints de $\vec{G}$.

Pour $d = \alpha(G)$ un d-bloqueur optimal contient tous les sommets de G donc les $\alpha(G)$ couples (u, v) tels que $C(u) = C(v)$. Considérons $d < \alpha(G)$.

La preuve se fait par récurrence sur le nombre de sommets n du graphe.

Si $n = 2$, G_c contient une composante unique, $\alpha(G) = 1$ et la propriété est vérifiée.

Supposons qu'elle est vérifiée pour tout arbre G' de moins de n sommets, tous libres, n pair, $n \geqslant 4$: cela signifie que pour un ensemble L de sommets qui contient k couples $(u, v) \in R \times B, u \prec v$, on a $\alpha(G' - L) = \alpha(G') - k$ (même si $|L| > k$).

Considérons un arbre de n sommets, tous libres, et un couple $(a, b) \in R \times B$ tel que a n'a aucun prédécesseur dans $\vec{G}$. Notons G' le sous graphe obtenu en enlevant a et b de G, et $\vec{G'}$ le graphe orienté associé (notons que l'orientation des arcs dans $\vec{G}$ et $\vec{G'}$ est la même). Remarquons que $\alpha(G) = \alpha(G') + 1$.

Soit k (resp. k') le nombre de couples $(u, v) \in R \times B,\ u \prec v$, qui sont les extrémités de chemins sommets-disjoints de $\vec{G}$ (resp. de $\vec{G'}$) dans un d-bloqueur L_d de $\vec{G}$. Notons L'_d la

restriction de L_d à G'. Supposons que $k < d$. Il y a quatre cas.

— $\{a, b\} \subset L_d$: on a $k' = k - 1$ et $\alpha(G - L_d) = \alpha(G' - L'_d) = \alpha(G') - k' = \alpha(G) - k > \alpha(G) - d$;

— $\{a, b\} \cap L_d = \varnothing$: on a $k' = k$ et $\alpha(G - L_d) = \alpha(G' - L'_d) + 1 = \alpha(G') - k' = \alpha(G) - k > \alpha(G) - d$;

— $\{a, b\} \cap L_d = \{b\}$: $a \notin L_d$ donc les k couples de L_d sont dans G'. Ainsi, $k' = k$ et $\alpha(G - L_d) = \alpha(G' - L'_d) + 1 = \alpha(G') - k' + 1 = \alpha(G) - k > \alpha(G) - d$;

— $\{a, b\} \cap L_d = \{a\}$.

Si a n'est pas sommet de l'un des k couples de L_d alors tous les couples de L_d sont dans G' et $k = k'$. Et on obtient : $\alpha(G - L_d) \geqslant \alpha(G' - L_d) = \alpha(G') - k' = \alpha(G) - k > \alpha(G) - d$.

Si (a, v) est l'un des k couples de L_d, $v \neq b$, soit u le prédécesseur de v. Si $u \in L_d$ alors on remplace (a, v) par (u, v) dans l'ensemble des k couples de L_d, et on est ramené au cas $\{a, b\} \cap L_d = \varnothing$. Supposons enfin que $u \notin L_d$: $\alpha(G' - L_d) \leqslant \alpha(G - L_d) \leqslant \alpha(G) - d = \alpha(G') - (d - 1)$; donc les $k' = k - 1$ couples de L_d qui sont dans G' forment un $(d-1)$-bloqueur de G' et si on leur ajoute le couple (u, v) on obtient un d-bloqueur de G' qui a $k < d$ couples, ce qui contredit l'hypothèse de récurrence.

Nous venons de montrer que tout d-bloqueur optimal contient nécessairement d couples $(u, v), u \prec v$, extrémités de d chemins sommets-disjoints du graphe des composantes G_c. Nous avons montré que d tels couples forment un d-bloqueur. Un d-bloqueur optimal est donc constitué uniquement de ces d couples et tout autre d-bloqueur contient d tels couples. □

Exemple 4.4.1. *Considérons le graphe G de la figure 4.1 dont le cardinal maximal d'un stable est $\alpha(G) = 9$. L'ensemble $\{c, d, g, l\}$ est un 2-bloqueur : dans le graphe de la figure 4.2 où les sommets retirés sont en gris, il n'est pas possible de construire un stable de cardinal supérieur à 7. Considérons maintenant l'ensemble $\{a, g, l, r\}$. Les chemins de a à r et de g à l ne sont pas disjoints donc d'après la Propriété 4.4.1, ce n'est pas un 2-bloqueur. En effet, dans le graphe de la figure 4.3 où les sommets retirés sont en gris, il existe un stable de cardinal 8 : $\{b, d, f, h, k, m, o, q\}$.*

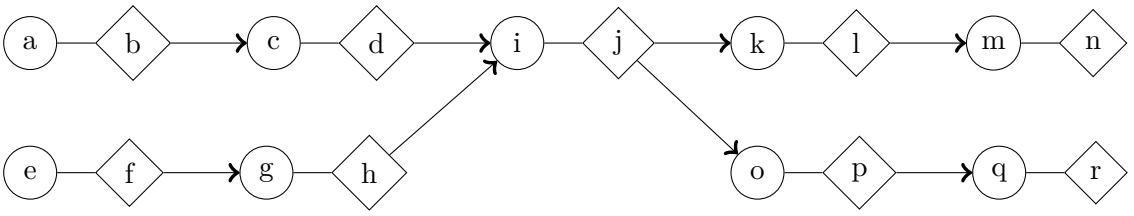


FIGURE 4.1 – Arbre G

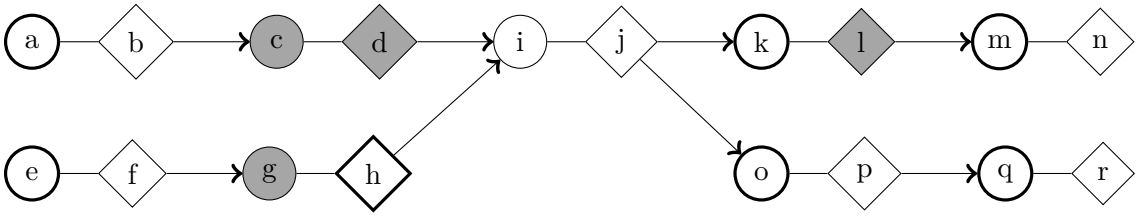


FIGURE 4.2 – Les sommets en gris sont retirés de G et les sommets dont la bordure est plus large forment un stable de cardinal 7

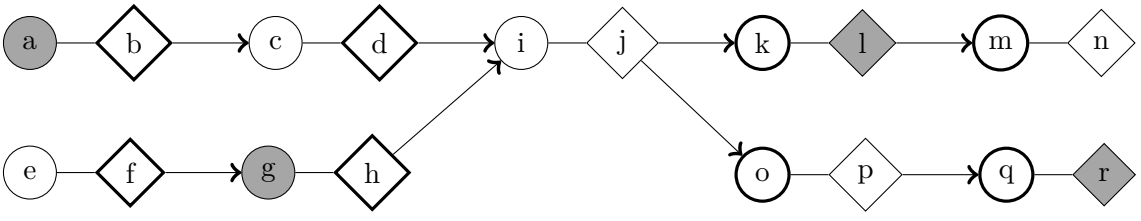


FIGURE 4.3 – Les sommets en gris sont retirés de G et les sommets dont la bordure est plus large forment un stable de cardinal 8

A l'aide de la caractérisation précédente, déterminer un d-bloqueur de coût minimal est polynomial dans les arbres dont tous les sommets sont libres.

Propriété 4.4.2. *Le problème du d-bloqueur de coût minimal de stables, $BLOCK\_STABLE(w = 1, c, d)$, se résout en temps polynomial si le graphe est un arbre dont tous les sommets sont libres.*

Démonstration. Pour trouver un d-bloqueur de coût minimal dans un arbre, if faut trouver d couples respectant les conditions de la Propriété 4.4.1 et tels que la somme des coûts des sommets des couples retenus soit minimale.

Soit $G = (R, B, E)$ un arbre dont tous les sommets sont libres. Soit $\vec{G}$ le graphe obtenu en orientant les arêtes de G dans le sens du graphe des composantes. Il s'agit de trouver d chemins sommets-disjoints de $\vec{G}$ reliant des sommets u de R à des sommets v de B pour un coût total minimal, le coût associé à un chemin de u à v dans $\vec{G}$ étant égal à $c(u)+c(v)$. Nous allons montrer comment obtenir une solution en résolvant un problème de flot maximal de coût minimal dans un nouveau graphe orienté $\vec{G'}$ que nous allons construire.

Appelons u_i et v_i les deux sommets de la composante C_i de G_c. Remarquons que, dans $\vec{G}$, tout sommet $u_i \in R$ (respectivement $v_i \in B$) a pour seul successeur (respectivement prédécesseur) v_i (resp. u_i). Donc, tous les chemins allant de $u \in R$ à $v \in B$ et passant par u_i (ou par v_i) contiennent obligatoirement l'arc (u_i, v_i) (et réciproquement évidemment). Dans $\vec{G}$, chercher des chemins sommets-disjoints est donc équivalent à chercher des chemins arcs-disjoints.

Ajoutons une source s et un puits t à G, et pour tout sommet $u \in R$ relions s à u par un arc (s, u) de coût $c(u)$, et pour tout sommet $v \in B$ relions v à t par un arc (v, t) de coût $c(v)$. Tous les autres arcs ont un coût nul. Tous les arcs ont une capacité égale à 1. Soit $\vec{G'}$ le graphe valué ainsi obtenu. Le problème revient maintenant à chercher un flot de valeur d de coût minimal entre s et t. Pour obtenir un flot de valeur d, ajoutons à $\vec{G'}$ un sommet s' et un arc (s', s) de capacité d et coût 0.

Il s'agit maintenant de chercher un flot maximal de coût minimal de s' à t dans $\vec{G'}$.

C'est un problème polynomial [46].

Vérifions qu'il y a bien équivalence entre les solutions du problème de flot et celles du problème du d-bloqueur BLOCK_$STABLE(w = 1, c, d)$, et que ces solutions ont les mêmes coûts. Soit la solution du problème de flot obtenue : elle comporte d unités routées de s' à t et les capacités égales à 1 imposent que ces unités soient routées sur d chemins disjoints à partir de s. On a donc obtenu d chemins arcs-disjoints de s à t, allant de $u(j) \in R$ à $v(j) \in B$, $j = 1, ..., d$: chaque chemin commence par un arc $(s, u(j))$ et finit par un arc $(v(j), t)$ qui sont les seuls arcs de coûts non nuls sur ce chemin. Le coût du flot est donc égal à $\sum_{j=1,...,d} c(u(j)) + c(v(j))$. Les d chemins arcs-disjoints de $\vec{G'}$ correspondent à d chemins sommets-disjoints dans le graphe $\vec{G}$: on a donc bien défini un d-bloqueur.

Prenons maintenant une solution du problème de d-bloqueur, elle est formée de d couples $(u(j), v(j)) \in R \times B$, $j = 1, ..., d$, et a une valeur de $\sum_{j=1,...,d} c(u(j)) + c(v(j))$. Ces d couples définissent d chemins sommets-disjoints de $\vec{G}$, donc d chemins arcs-disjoints de s à t dans $\vec{G'}$: en routant une unité de flot sur chacun de ces chemins et d unités sur (s', s) on obtient un flot d ayant le même coût que le d-bloqueur.

Une solution optimale du problème de flot définit donc un d bloqueur optimal.

□

Propriété 4.4.3. *Soit $G = (B, R, E)$ un arbre qui ne contient pas de sommets exclus. Déterminer un d-bloqueur de coût minimal de stables de G est un problème polynomial.*

Démonstration. Puisque G ne contient pas de sommets exclus, tous les sommets forcés sont isolés. Un d-bloqueur optimal de G contient au plus $min(|\Xi(g)|, d)$ sommets forcés. Soit k un entier tel que $0 \leqslant k \leqslant min(|\Xi(g)|, d)$. Déterminer un d-bloqueur de coût minimal de G qui contient k sommets forcés est un problème polynomial : on classe les sommets forcés par coût croissant, on sélectionne les k premiers et on détermine un $(d-k)$-bloqueur du graphe réduit aux sommets libres de G grâce à la propriété précédente.

Ainsi, si on construit $(min(|\Xi(g)|, d) + 1)$ k-bloqueurs pour toutes les valeurs de k possibles, cela peut être réalisé en temps polynomial. Et comme un d-bloqueur optimal

de G contient au plus $min(|\Xi(g)|, d)$ sommets forcés, on est certain de déterminer un d-bloqueur optimal de G. □

4.5 Perspectives

Dans ce chapitre, nous avons caractérisé les d-bloqueurs de stables dans les arbres et proposé un algorithme polynomial pour le problème $BLOCK_{STABLE}(w = 1, c, d)$ dans le cas d'un arbre dont tous les sommets sont libres ou forcés. Mais le problème $BLOCK_{STABLE}(w = 1, c, d)$ est encore ouvert dans les arbres qui contiennent des sommets exclus.

Chapitre 5

d-transversaux de problèmes modélisés par un programme mathématique en variables binaires

5.1 Introduction

Dans le chapitre précédent, nous avons considéré un problème d'attaque-défense sur un système avec un nombre fini d'éléments dans lequel le défenseur connaissait les éléments attaqués par l'agresseur et pouvait adapter sa décision en conséquence. Pour s'assurer que l'efficacité de la stratégie du défenseur était bien diminuée, l'attaquant devait considérer les réponses possibles du défenseur après son attaque. Dans ce chapitre, nous considérons cette fois que le défenseur ne sait pas qu'il est attaqué. L'attaquant souhaite toujours réduire l'efficacité de la réponse du défenseur mais puisque son attaque n'est pas connue, il sait que le défenseur choisira une stratégie optimale dans un système avec tous ses éléments. Le but de l'attaquant est donc de déterminer un ensemble minimum d'éléments dont l'intersection avec toute stratégie du défenseur contient un nombre fixe d'éléments. De cette façon, l'attaquant peut interférer efficacement avec toutes les stratégies du défenseur.

Ce chapitre expose tout d'abord les définitions et travaux existants sur les d-transversaux. Puis une section porte sur les d-transversaux pondérés de stables [1] dans les graphes bipartis.

1. On considère ici que les d-transversaux de stables sont des d-transversaux de stables de cardinal maximal

Enfin, dans une dernière partie, nous nous intéressons aux d-transversaux de problèmes modélisés par des programmes mathématiques en variables binaires, proposons une approche de résolution et appliquons cette approche pour déterminer des d-transversaux de cardinal minimal de stables et des d-transversaux de cardinal minimal de couplages.

5.2 Définitions

Soit Π un problème d'optimisation combinatoire défini sur un ensemble fini $M\{m_1, m_2, ..., m_n\}$. Soit P une propriété sur cet ensemble telle que $X \subseteq M$ satisfait P si et seulement si X est une solution admissible de Π. Étant donné $L \subseteq M$, on définit C_L l'ensemble des sous-ensembles de L qui satisfont P. Ainsi, l'ensemble des solutions admissibles de M, c'est-à-dire l'ensemble des sous-ensembles de M qui satisfont P, est noté C_M.

À chaque élément m_i de M sont associées deux valeurs positives : un coût $c(m_i)$ et un poids $w(m_i)$. Le poids (respectivement le coût) d'un sous-ensemble de M est la somme des poids (respectivement des coûts) des éléments de ce sous-ensemble. On note $\nu_w(C_M)$ le poids maximum d'un élément de C_M, c'est-à-dire que $\nu_w(C_M) = max\{w(C_j)|C_j \in C_M\}$. On note également $\delta_w(C_M)$ le cardinal minimum d'un élément de C_M de poids maximum, c'est-à-dire que $\delta_w(C_M) = min\{|C_j| \ \ |C_j \in C_M, w(C_j) = \nu_w(C_M)\}$.

Soit $d \geqslant 1$ un entier.

— Pour $d \leqslant \nu_w(C_M)$, on appelle *d-transversal pondéré* un sous-ensemble $T \subseteq M$ tel que pour tout $C \in C_M$ vérifiant $w(C) = \nu_w(C_M)$, on a $w(C \cap T) \geqslant d$.

— Pour $d \leqslant \delta_w(C_M)$, on appelle *d-transversal* un sous-ensemble $T \subseteq M$ tel que pour tout $C \in C_M$ vérifiant $w(C) = \nu_w(C_M)$, on a $|C \cap T| \geqslant d$.

On considère les problèmes suivants :

— $WTRANS_\Pi(w, c, d)$:

  — Instance : Un ensemble fini $M\{m_1, m_2, ..., m_n\}$, une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M| * \max_{i=1..n} c(m_i)$.

  — Question : Est-ce qu'il existe ensemble $T \subseteq M$ tel que $c(T) \leqslant k$ et $\forall C \in C_M$

vérifiant $w(C) = \nu_w(C_M)$ on a $w(C \cap T) \geqslant d$? Autrement dit, existe-t-il un d-transversal pondéré du problème Π de coût inférieur ou égal à k ?

— $TRANS_{\Pi}(w, c, d)$:

— Instance : Un ensemble fini M=(m_1, m_2, ..., m_n), une fonction de coût c sur M, une fonction de poids w sur M, un entier d, un entier positif $k \leqslant |M| * \max_{i=1..n} c(m_i)$.

— Question : Est-ce qu'il existe un ensemble $T \subseteq M$ tel que $c(T) \leqslant k$ et $\forall C \in C_M$ vérifiant $w(C) = \nu_w(C_M)$, on a $|C \cap T| \geqslant d$ e ? Autrement dit, existe-t-il un d-transversal du problème Π de coût inférieur ou égal à k ?

Le problème $TRANS_{\Pi}(w, c, d)$ peut être modélisé par le programme biniveau suivant [32], dans lequel :

— Les variables x_i valent 1 si l'élément i appartient au d-transversal et 0 sinon

— Les variables y_i valent 1 si l'élément i est choisi dans le sous-problème et 0 sinon.

— C_M désigne l'ensemble des solutions admissible du problème Π.

— $\nu_w(C_M)$ désigne le poids d'une solution optimale de Pi

— w_i désigne le poids de l'élément i

— c_i désigne le poids de l'élément i

$$(Ptrans) \begin{cases} \min\limits_{x} \quad \sum\limits_{i=1}^{n} x_i c_i \\ \quad \sum\limits_{i=1}^{n} x_i y_i \geqslant d \\ \quad x \in \{0,1\}^n \\ \quad \begin{cases} \min\limits_{y} \quad \sum\limits_{i=1}^{n} x_i y_i \\ \quad \sum\limits_{i=1}^{n} w_i y_i = \nu_w(C_M) \\ \quad C = \{v_i \ s.t. \ y_i = 1\} \in C_M \\ \quad y \in \{0,1\}^n \end{cases} \end{cases}$$

La fonction économique de premier niveau minimise le coût des éléments sélectionnés par le meneur. La fonction économique de deuxième niveau minimise le nombre d'éléments en commun entre ceux choisis par le suiveur et ceux choisis par le meneur. La deuxième

contrainte de deuxième niveau assure que les variables y fournissent une solution du problème Π et la première contrainte assure que cette solution est optimale. Le contrainte de liaison impose que l'intersection entre les éléments choisis par le suiveur et ceux choisis par le meneur est bien de d éléments donc que les variables x définissent bien un d-transversal.

Notons que le problème $WTRANS_{\Pi}(w, c, d)$ peut aussi être modélisé par un programme mathématique biniveau. Il suffit pour cela de remplacer la contrainte $\sum_{i=1}^{n} x_i y_i \geqslant d$ par la contrainte $\sum_{i=1}^{n} w_i x_i y_i \geqslant d$ dans le programme précédent.

Les trois propriétés suivantes concernent les liens entre les d-transversaux et les éléments forcés et exclus du système. Elles ont été données dans [32] pour les d-transversaux de stables, nous donnons ici des versions généralisées.

Propriété 5.2.1. *[32] Un d-transversal optimal ne contient aucun élément exclu.*

Propriété 5.2.2. *[32] Si $d \leqslant \xi(V)$ alors un d-transversal T tel que $T \subset \Xi(V)$ et $|T| = d$ est une solution optimale de $TRANS_{\Pi}(w = 1, c = 1, d)$.*

Propriété 5.2.3. *[32] Si $d \geqslant \xi(V)$ alors il existe un d-transversal optimal qui contient tous les éléments forcés.*

Les d-transversaux ont été étudiés pour le problème du stable optimal. Le problème $TRANS\_Stable(w, c, d)$ est polynomial dans les graphes bipartis [17]. Dans les splits graphes, le problème $TRANS\_Stable(w = 1, c, d)$ est polynomial [59] et le problème $TRANS\_Stable(w, c = 1, d)$ est APX-difficile [31].

En ce qui concerne les cliques de taille maximale, $TRANS\_Clique(w = 1, c = 1, d)$ est NP-difficile dans les graphes planaires, les graphes cordaux et les graphes sans triangles [59].

Pour les couplages optimaux, le problème $TRANS\_Couplage(w, c, d)$ est NP-difficile dans les graphes biparti [81] et $TRANS\_Couplage(w = 1, c = 1, d)$ polynomial dans les arbres, les grilles, les cycles et les graphes complets [68].

Pour les plus courts chemins, $TRANS\_Chemins(w = 1, c = 1, d)$ est polynomial [56].

$TRANS\_VertexCover(w, c, d)$ est polynomial dans les graphes bipartis [17].

Pour les d-transversaux pondérés, c'est à dire lorsque l'intersection entre le d-transversal et toute solution optimale est de poids d et non de cardinal d, le problème $WTRANS\_Stable(w, c = 1, d)$ est APX-difficile dans les splits graphes et le problème $WTRANS\_Stable(w, c, d)$ est NP-difficile dans les graphes qui ne contiennent que des sommets isolés [31]. Le problème $WTRANS\_Stable(w, c = 1, d)$ est encore ouvert dans les graphes bipartis. Dans la section suivante, nous étudions quelques cas particuliers qui peuvent être résolus en temps polynomial.

5.3 d-transversaux pondérés de stables optimaux dans les graphes bipartis

Dans cette section, on s'intéresse au problème $WTRANS\_Stable(w, c = 1, d)$ dans les graphes bipartis dont tous les sommets sont libres. Nous réutiliserons la caractérisation des stables optimaux dans les graphes bipartis rappelée dans le chapitre 1. Rappelons que dans les graphes bipartis dont tous les sommets sont libres, les sommets peuvent être partitionnés en composantes C_i, qui contiennent exactement deux stable optimaux : $V(C_i) \cap R$ et $V(C_i) \cap B$.

Nous étudions ici des cas particuliers pour lesquels le problème $WTRANS\_Stable(w, c = 1, d)$ peut être résolu en temps polynomial. Dans un premier temps nous déterminons des d-transversaux pondérés pour des valeurs particulières de d. Puis, dans un deuxième temps, nous considérons le cas des graphes bipartis dont le graphe des composantes du sous-graphe induit par les sommets libres est un chemin.

5.3.1 Valeurs particulières de d

Dans cette sous-section, nous étudions des valeurs particulières de d pour lesquelles le problème $WTRANS\_Stable(w, c = 1, d)$ est résolu en temps polynomial dans un graphe biparti $G = (B, R, E)$ dont tous les sommets sont libres.

Soient les notations suivantes :

— Soit $\alpha_w(G)$ le poids d'un stable de poids maximum de G

— Soit w_1^R le plus petit poids d'un sommet de R et w_1^B le plus petit poids d'un

sommet de B. On supposera sans perte de généralité que $min(w_1^B, w_1^R) = w_1^R$ et $max(w_1^B, w_1^R) = w_1^B$, donc que $w_1^B \geqslant w_1^R$. Si ce n'est pas le cas, il suffit d'inverser l'orientation du graphe des composantes car ainsi B devient R et inversement.

Si $\alpha_w(G) - w_1^R < d \leqslant \alpha_w(G)$, c'est à dire si $\alpha_w(G)$- poids minimum des sommets du graphe $< d$, un d-transversal pondéré optimal de G contient tous les sommets du graphe.

Supposons que $w_1^B \neq w_1^R$ et que $\alpha_w(G) - w_1^B < d \leqslant \alpha_w(G) - w_1^R$. Dans ce cas il est trivial qu'un d-transversal pondéré contient tous les sommets de B, sinon l'intersection avec le stable B n'a pas un poids de d. Pour construire un d-transversal pondéré optimal T, on sélectionne tous les sommets de B et, tant que $w(T \cap R) < d$, on sélectionne les sommets de R par ordre croissant de poids. En construisant T de cette façon, pour toute composante C_i, on a $w(T \cap R \cap V(C_i)) \leqslant w(T \cap B \cap V(C_i))$. Ainsi, R est le stable optimal de G tel que l'intersection avec T est de plus petit poids. Or, $w(T \cap R) \geqslant d$ donc T est bien un d-transversal pondéré. Il est aussi optimal car il contient le minimum de sommets de R et il doit contenir tous les sommets de B.

Considérons le cas dans lequel $d = \alpha_w(G) - w_1^B$. Il faut aussi considérer deux cas : soit il existe un d-transversal optimal T tel que $w(T \cap B) = w(B) - w_1^B$, soit il n'en existe pas.

— Considérons le premier cas. Soit $x \in B$ tel que $w(x) = w_1^B$ et $C(x)$ a le moins de prédécesseurs possible. Si $x \notin T$ alors T doit contenir tous les sommets $y \in R$ tels que $C(y) \to C(x)$ et $C(y) \neq C(x)$ sinon il existe un stable optimal de G dont l'intersection avec T est de poids strictement inférieur à d. Pour déterminer le reste de $T \cap R$, on commence par supposer que $R \subset T$. Puis on classe par ordre de poids croissant les sommets $y \in R$ tels que $C(x) \to C(y)$. Enfin, on retire les sommets y un par un tant que $w((T - \{y\}) \cap R) \geqslant d$. Ainsi, T est un d-transversal pondéré qui ne contient pas x et qui contient le moins de sommets possible.

— Dans le deuxième cas, on construit le d-transversal optimal du paragraphe précédent où $\alpha_w(G) - w_1^B < d \leqslant \alpha_w(G) - w_1^R$. En effet, ce d-transversal pondéré est le plus petit d-transversal pondéré T tel que $T \cap B = B$.

On compare les cardinaux des deux d-transversaux pondérés déterminés. Celui de plus petit cardinal est optimal.

Si $w_1^B \neq w_1^R$ et $d > \alpha_w(G) - 2 \times w_1^B$, on détermine un d-transversal pondéré optimal de la même façon qu'au cas précédent.

Supposons que le graphe des composantes de G est une arborescence. Notons C_r la racine de cette arborescence et w_r le plus grand poids d'un sommet x de $V(C_r) \cap R$. Si $d \leqslant w_r$, alors pour construire un d-transversal pondéré optimal T, on sélectionne x et le minimum de sommets de B tels que $w(T \cap B) \geqslant d$. Pour déterminer $T \cap B$, on classe les sommets de B par poids décroissant et on les sélectionne un par un tant que $w(T \cap B) < d$. Tout stable optimal qui contient x a bien une intersection de poids d avec T et le seul stable optimal qui ne contient pas x est B, qui a aussi une intersection de poids d avec T. Donc T est bien un d-transversal pondéré. Il est aussi optimal : il est évident qu'il ne peut contenir moins de sommets de B ou de R.

5.3.2 Cas des graphes dont le graphe des composantes est un chemin

Dans cette section, on considère un graphe biparti pondéré $G = (B, R, E)$ dont le graphe des composantes du sous graphe induit par les sommets libres est un chemin. Considérons tout d'abord un graphe dont tous les sommets sont libres. Les composantes sont numérotées de 1 à k en partant de la composante sans prédécesseur. Ainsi, C_i désigne la composante i.

Propriété 5.3.1. *Si $G = (B, R, E)$ est un graphe biparti dont le graphe des composantes est un chemin, alors déterminer un d-transversal pondéré optimal de G peut être réalisé en temps polynomial.*

Démonstration. Nous avons vu précédemment que le problème de recherche d'un d-transversal pouvait être modélisé par un programme mathématique biniveau. Nous proposons dans la section 5.4.2 une modélisation par un programme avec un seul niveau mais avec un nombre exponentiel de contraintes. Dans le cas des graphes bipartis dont le graphe des composantes est un chemin, le nombre de contraintes est polynomial.

Considérons le programme suivant, dans lequel w_i désigne le poids du sommet i, $x_i = 1$ si et seulement si le sommet i appartient au d-transversal et S_j désigne le stable optimal du graphe tel que pour toute composante C_i telle que s'il existe i tel que $i > 0$ et $i < j$

on a $V(C_i) \cap S_j = V(C_i) \cap R$ et pour toute composante C_i telle qu'il existe $i \leqslant k$ tel que $i \geqslant j$ on a $V(C_i) \cap S_j = V(C_i) \cap B$.

$$\begin{cases} \min\limits_{x} & \sum\limits_{i=1}^{n} x_i \\ & \sum\limits_{i \in S_j} w_i x_i \geqslant d \quad \forall j \in \{1..k+1\} \\ & x_i \in \{0,1\} \qquad \forall i \in \{1..n\} \end{cases}$$

Dans ce modèle :

— On note n le nombre de sommets du graphe

— On numérote les sommets du graphe de 1 à n. On commence par numéroter les sommets de B de 1 à $|B|$ de façon à ce que le numéro d'un sommet x soit strictement supérieur à ceux des sommets de B dans les composantes qui précèdent $C(x)$. Puis on numérote les sommets de R de $|B|+1$ à n de la même façon.

Le nombre de contraintes du programme est bien polynomial. En effet, il existe autant de contraintes que de stable optimaux dans le graphe. Puisque le graphe des composantes est un chemin, si pour un stable optimal S^* et pour une composante i on a $V(C_i) \cap S^* = V(C_i) \cap B$, alors pour tout j tel que $i \leqslant j \leqslant k$, $V(C_j) \cap S^* = V(C_j) \cap B$. Il existe donc $k+1$ stables optimaux du graphe.

La fonction économique du modèle minimise le nombre de sommets qui appartiennent au d-transversal. La contrainte assure que tout stable optimal du graphe a une intersection de poids supérieur ou égal à d avec le d-transversal. Autrement dit, pour un stable S_i, la composante C_i est la première composante du chemin dont l'intersection des sommets et du stable est incluse dans B. La contrainte assure donc que l'intersection entre le d-transversal et tout stable optimal du graphe est bien de poids d.

Soit A la matrice des contraintes. Elle contient $k+1$ lignes, autant que de stables optimaux du graphe, et n colonnes, autant que de sommets dans le graphe. Dans cette matrice, $a_{ji} = 1$ si et seulement si le sommet i apparaît dans la contrainte liée au stable S_j. Si $1 \leqslant i \leqslant |B|$ alors le sommet i appartient à B et sinon, $|B|+1 \leqslant i \leqslant n$ et le sommet i appartient à R.

Considérons un j quelconque. Le stable S_j contient les sommets de $V(C_l) \cap B$ pour tout

l tel que $j \leqslant l < k$. Soit m_B le cardinal de l'ensemble des sommets de B qui appartiennent à une composante qui précède C_j. Autrement dit, m_B est le nombre de sommets de B qui n'appartiennent pas au stable S_j. Pour tout i tel que $1 \leqslant i \leqslant m_B$, $a_{ji} = 0$ et pour tout i tel que $m_B < i \leqslant |B|$, $a_{ji} = 1$.

De même, le stable S_j contient les sommets de $V(C_l) \cap R$ pour tout l tel que $0 < l < j$. Soit m_R le nombre de sommets de R qui appartiennent à une composante qui précède C_j, c'est-à-dire le nombre de sommets de R qui appartiennent au stable optimal S_j. Donc pour tout i tel que $|B| < i \leqslant |B| + m_R$, $a_{ji} = 1$ et pour tout $i > |B| + m_R$, $a_{ji} = 0$. Ainsi, pour tout i tel que $m_B < i \leqslant |B| + m_R$, $a_{ji} = 1$ et sinon $a_{ji} = 0$. Donc, à chaque ligne j de la matrice A, les a_{ji} égaux à 1 sont consécutifs.

Puisque, dans la matrice des contraintes, sur chaque ligne, les 1 sont consécutifs elle est totalement unimodulaire. La relaxation continue du programme fournit donc une solution optimale entière. Ainsi, déterminer un d-transversal pondéré de G peut être réalisé en temps polynomial.

□

Considérons maintenant un graphe biparti $G = (B, R, E)$ qui contient des sommets forcés et exclus et dont le graphe des composantes du sous-graphe réduit aux sommets libres est un chemin.

Propriété 5.3.2. *Déterminer un d-transversal pondéré optimal de G peut être réalisé en temps polynomial.*

Démonstration. Un d-transversal pondéré optimal T ne contient pas de sommets exclus. Si T contient k sommets forcés avec $0 \leqslant k \leqslant min(d, |\Xi(G)|)$ alors il contient un $(d-k)$-transversal optimal du sous-graphe de G réduit aux sommets libres. De plus, puisque T est optimal alors les k sommets forcés qu'il contient sont ceux de plus grand poids.

Ainsi, en construisant $min(d, |\Xi(G)|)+1$ d-transversaux T_k pour tout $0 \leqslant k \leqslant min(d, |\Xi(G)|)$ tels que les T_k contiennent les k sommets forcés de plus grand poids et un $(d-k)$-transversal optimal du sous-graphe de G réduit aux sommets libres, on obtient un d-transversal optimal. En effet, un des T_k de plus petit cardinal sera un d-transversal optimal. □

5.4 d-transversaux de problèmes modélisés par des programmes mathématiques en variables binaires

Dans cette section, nous nous intéressons au problème $TRANS_{\Pi}(w, c = 1, d)$ dans le cas où Π est un problème modélisé par un programme en variables binaires. Nous souhaitons concevoir une approche générique de résolution de $TRANS_{\Pi}(w, c = 1, d)$ quel que soit le problème en variables binaires choisi. Dans un premier temps, nous chercherons à résoudre le problème par la programmation mathématique biniveau. Dans un deuxième temps, nous présenterons une approche de résolution par génération de contraintes que nous appliquerons aux cas particuliers des d-transversaux de stables et aux d-transversaux de couplages.

Notons que les méthodes de résolution du problème $TRANS_{\Pi}(w, c = 1, d)$ présentées dans ce chapitre peuvent facilement être adaptées pour résoudre le problème $TRANS_{\Pi}(w, c, d)$ en remplaçant le terme $\sum_{i=1}^{n} x_i$ par $\sum_{i=1}^{n} c_i x_i$ dans les fonctions économiques des programmes mathématiques.

5.4.1 Résolution du programme biniveau

Comme indiqué dans l'état de l'art, une modélisation de $TRANS_{\Pi}(w, c = 1, d)$ par un programme mathématique biniveau est proposée dans [32]. Ce programme est détaillé dans la section de Définitions ce ce chapitre.

Ce programme est un programme biniveau avec des variables discrètes aux deux niveaux. Il contient également une contrainte de liaison. Comme indiqué dans le chapitre 2, les approches existantes de résolution des programmes biniveaux entiers ne considèrent que des programmes biniveaux simplifiés. Á notre connaissance, il n'existe actuellement pas d'approches pour résoudre directement ce programme dans la littérature.

Le chapitre 2 propose une extension de l'algorithme présenté dans [10] pour résoudre ce programme. Mais comme indiqué dans le chapitre, il n'est pas efficace. De plus, l'exemple utilisé pour illustrer les inconvénients de cette approche correspond à la recherche d'un 2-transversal dans le petit graphe de la Figure 5.1. Ainsi, l'approche de détermination de d-transversal par la programmation biniveau n'est pas efficace. Nous allons donc chercher

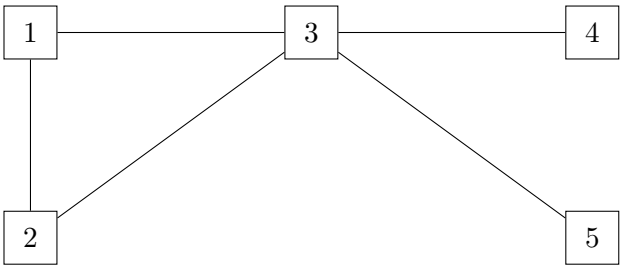


FIGURE 5.1 – Exemple de Graphe dont le cardinal maximal d'un stable est 3

une approche plus efficace pour déterminer des d-transversaux.

5.4.2 Résolution par génération de contraintes

Puisque l'algorithme de la section précédente est inefficace, nous proposons ici une autre approche, fondée sur la génération de contraintes. Nous proposons deux variantes de cette approches et nous les appliquons pour déterminer des d-transversaux optimaux de stables et des d-transversaux optimaux de couplages.

Plutôt que de modéliser le problème $TRANS_{\Pi}(w, c = 1, d)$ par un programme biniveau, nous proposons ici une approche avec un seul niveau. Dans le programme suivant, on note C_M^* l'ensemble des solutions optimales du problème Π. Autrement dit, $C_M^* = \{C \in C_M | w(C) = \nu_w(C_M)\}$. Comme dans le modèle biniveau, les variables x_i valent 1 si l'élément i appartient au d-transversal et 0 sinon. Les variables y_i^k valent 1 si l'élément i appartient à la solution optimale $S_k \in C_M^*$ avec $i \in \{1, .., n\}$ et $k \in \{1, .., |C_M^*|\}$, et 0 sinon.

$$(Ptrans) \begin{cases} \min\limits_{x} & \sum\limits_{i=1}^{n} x_i \\ & \sum\limits_{i=1}^{n} x_i y_i^k \geqslant d \quad \forall k \text{ tel que } \{m_i \text{ tel que } y_i^k = 1\} \in C_M^* \quad (1) \\ & x \in \{0,1\}^n \end{cases}$$

Les contraintes principales assurent que pour chaque solution optimale du problème, l'intersection avec le d-transversal est bien d'au moins d éléments. La fonction économique minimise le nombre d'éléments choisis dans le d-transversal.

Puisque le programme $(Ptrans)$ n'a qu'un niveau, il pourrait être résolu directement par un solveur de programmation linéaire. Mais le nombre de contraintes du programme peut être exponentiel car il y a autant de contraintes que de solutions optimales au pro-

blème Π.

Une autre difficulté de la résolution du programme ($Ptrans$) est qu'il peut être vu comme une modélisation du problème suivant :

— Instance : Un ensemble fini M=(m_1, m_2...m_n), un ensemble C_M de sous-ensembles de M, un entier k, un entier d

— Question : Est-ce qu'il existe un ensemble $T \subseteq M$ tel que $|T| \leqslant k$ et pour tout $C \in C_M$, $|C \cap T| \geqslant d$?

Ce problème est une généralisation du problème Hitting Set dans lequel $d = 1$. Résoudre le problème Hitting Set est un problème NP-difficile [46].

Pour résoudre le programme ($Ptrans$) nous utiliserons une approche de génération de contraintes. Des approches similaires ont été utilisées par [52] pour résoudre un problème d'interdiction de chemins et par [64] pour déterminer des d-bloqueurs de cliques.

L'approche de résolution par génération de contraintes consiste à générer le moins de contraintes possibles du problème. On considère la relaxation du modèle en relâchant une partie des contraintes (1) et on tente de déterminer une solution optimale du problème qui a moins de d éléments communs avec l'ensemble construit. Si on peut générer une solution optimale de ce type, alors l'ensemble n'est pas un d-transversal et on ajoute à la relaxation la contrainte liée à cette solution, puis on résout à nouveau la relaxation jusqu'à ce qu'on ne puisse plus trouver de solution.

Nous proposons deux variantes de cette approche. La première variante résout deux programmes mathématiques en nombres entiers à chaque itération alors que la deuxième variante n'en résout qu'un.

Variante 1

Dans cette variante de l'approche de résolution par génération de contraintes, deux programmes linéaires en nombres entiers sont résolus à chaque itération.

Considérons le problème maître suivant, qui est une relaxation du modèle précédent. Plutôt que de considérer l'ensemble des solutions optimales du problème, on considère un sous-ensemble Y de ces solutions.

$$(PtransMaster) \begin{cases} \min\limits_{x} & \sum\limits_{i=1}^{n} x_i \\ & \sum\limits_{i=1}^{n} x_i y_i^k \geqslant d \quad \forall k \text{ tel que } \{m_i \text{ tel que } y_i^k = 1\} \in Y \\ & x \in \{0,1\}^n \end{cases}$$

On commence par déterminer une solution optimale du problème Π. Ainsi, on connaît $\nu_w(C_M)$ et on peut initialiser Y au singleton constitué de la solution optimale de Π déterminée. On génère ensuite un ensemble T de taille minimum qui a au moins d éléments en commun avec chaque solution dans Y. Puis on recherche une solution optimale du problème qui a moins de d éléments en communs avec T. Si une telle solution n'existe pas, alors on a bien construit un d-transversal. Sinon, on ajoute la solution trouvée à l'ensemble Y et on commence une nouvelle itération.

Pour trouver à chaque itération une solution optimale du problème avec moins de d éléments en commun avec T, on considère le programme suivant. Dans ce programme, les variables y_i valent 1 si et seulement si l'élément i est dans la solution et 0 si ce n'est pas le cas. Le vecteur x est un paramètre où $x_i = 1$ si et seulement si l'élément i appartient à T.

$$(Popt\Pi(x)) \begin{cases} \min\limits_{y} & \sum\limits_{i=1}^{n} x_i y_i \\ s.c & \sum\limits_{i=1}^{n} w(y_i) = \nu_w(C_M) \\ & C = \{v_i \ s.t. \ y_i = 1\} \in C_M \\ & y \in \{0,1\}^n \end{cases}$$

Les contraintes assurent que l'ensemble construit est bien une solution optimale du problème. La fonction économique minimise le nombre d'éléments en commun entre la solution optimale du problème et le transversal. Si la valeur de la fonction économique est supérieure ou égale à d alors l'ensemble T est bien un d-transversal.

L'algorithme de génération de contraintes est résumée dans l'Algorithme 13.

Variante 2

Dans cette variante de l'approche de résolution par génération de contraintes, un seul programme en nombres entiers est résolu à chaque itération. Ce programme permet de déterminer à la fois un d-transversal et une solution optimale qui a le moins d'éléments

Algorithm 13 Algorithme de résolution de $Trans_{\Pi}(w, c = 1, d)$

Déterminer une solution optimale au problème Π pour déterminer $\nu_w(C_M)$ et initialiser Y au singleton correspondant à la solution optimale trouvée.
Soit x une solution de $(PtransMaster)$
Soit y une solution de $(Popt\Pi(x))$
while $\sum_{i=1}^{n} x_i y_i < d$ **do**
    ajouter à Y la contrainte $\sum_{i=1}^{n} x_i y_i \geqslant d$ qui correspond à y
    résoudre $(PtransMaster)$ pour obtenir x
    résoudre $(Popt\Pi(x))$ pour obtenir y
end while

return x

possible en commun avec l'ensemble T.

Considérons le programme $(Ptrans2)$ dans lequel les variables x_i valent 1 si et seulement si l'élément i appartient à T comme dans la variante précédente. De même les variables y_i valent 1 si et seulement si l'élément i appartient à la solution optimale de Π qui a le moins d'éléments en commun avec T. Les variables h_i^k valent 1 si l'élément i appartient à la solution optimale $S_k \in Y$ avec $i \in \{1, .., n\}$ et $k \in \{1, .., |C_M^*|\}$, et 0 sinon.

$$(Ptrans2) \begin{cases} \min\limits_{x,y} & \sum\limits_{i=1}^{n} x_i + \epsilon \sum\limits_{i=1}^{n} x_i y_i & \\ s.c. & \sum\limits_{i=1}^{n} x_i h_i^k \geqslant d & \forall k \text{ tel que } \{m_i \text{ tel que } h_i^k = 1\} \in Y \\ & \sum\limits_{i=1}^{n} w(y_i) = \nu_w(C_M) & \\ & C = \{m_i \text{ s.t. } y_i = 1\} \in C_M & \\ & x \in \{0,1\}^n & \\ & y \in \{0,1\}^n & \end{cases}$$

Ce programme détermine à la fois un ensemble x d'éléments avec d éléments en commun avec les solutions optimales dans Y, et une solution optimale dont l'intersection avec x est aussi petite que possible.

La première contrainte assure que l'intersection entre l'ensemble x et chaque solution optimale de l'ensemble Y contient bien au moins d éléments. Les deuxième et troisième contraintes assurent que y est bien une solution optimale du problème Π. La fonction

économique minimise le nombre d'éléments sélectionnés dans x et la taille de l'intersection entre x et y. Une pénalité $\epsilon > 0$ est appliquée dans la fonction économique à l'intersection entre x et y. Elle doit être suffisamment petite de manière à ce qu'il n'existe pas de solution optimale du problème telle que x contienne plus que le minimum d'éléments nécessaires pour que les premières contraintes soient respectées.

Cette variante de l'approche de résolution est très proche de la précédente. La seule différence est que le d-transversal et la solution optimale avec le minimum d'éléments en commun avec le d-transversal sont déterminés par un seul programme mathématique au lieu de deux. L'Algorithme 14 décrit cette approche.

Algorithm 14 Algorithme de résolution de $Trans_{\Pi}(w, c = 1, d)$

1: Déterminer une solution optimale au problème Π pour déterminer $\nu_w(C_M)$ et initialiser Y au singleton correspondant à la solution optimale trouvée.
2: Soit (x, y) une solution de $(Ptrans2)$
3: **while** $\sum_{i=1}^{n} x_i y_i < d$ **do**
4: ajouter à Y la contrainte $\sum_{i=1}^{n} x_i y_i \geqslant d$ qui correspond à y
5: résoudre $(Ptrans2)$ pour obtenir les nouvelles valeurs de (x, y)
6: **end while**
7: **return** x

5.4.3 d-transversaux de cardinal maximal de stables

Dans cette section, nous testons les deux variantes de l'approche de résolution en recherchant des d-transversaux de stables. Dans une section suivante, nous nous intéresserons aux d-transversaux de couplages. Les instances de graphes testées ont été générées aléatoirement, la probabilité que deux sommets soient voisins suivant une loi uniforme. La taille va de 20 sommets à 200 sommets, avec un pas de 20 sommets. Pour chaque taille trois densités de graphe ont été testées : 25 %, 50 % et 75 %. Pour chaque couple taille-densité, cinq instances ont été générées. Les valeurs de d testées correspondent à 1, 2, 3, le quart de la valeur d'une solution optimale du problème, la moitié de la valeur d'une solution optimale du problème, et les trois quarts de la valeur d'une solution optimale du problème. Le temps de calcul a été limité à une heure par instance. Tous les programmes

mathématiques ont été résolus avec CPLEX 12.

Variante 1

Dans cette section, nous recherchons des d-transversaux optimaux de stables. Le problème maître de l'algorithme est toujours le même et le sous-problème devient le problème suivant, dans lequel les variables y_i valent 1 si et seulement si le sommet i appartient au stable optimal du graphe qui a le moins d'éléments en commun avec le paramètre x.

$$(Popt_{stable}(x)) \begin{cases} \min\limits_{y} & \sum\limits_{i=1}^{n} y_i x_i \\ s.c & y_i + y_j \leqslant 1 \qquad \forall ij \in E \\ & \sum\limits_{i=1}^{n} y_i = \alpha(G) \\ & y \in \{0,1\}^n \end{cases}$$

La première contrainte assure que la solution est bien un stable et la deuxième contrainte assure que le stable est optimal en imposant que son cardinal soit égal au cardinal maximal d'un stable du graphe.

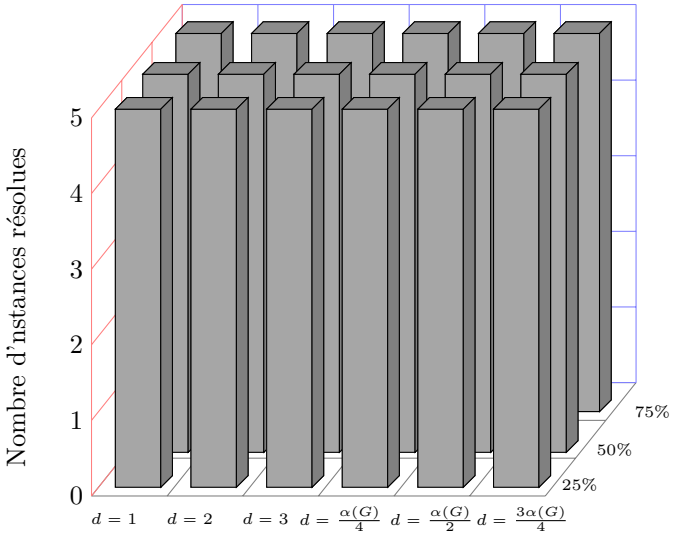
La section suivante présente les résultats des tests de détermination de d-transversaux optimaux de stables sur les instances générées. Les tests sur les instances de 20 à 80 sommets ne sont pas présentés car elles ont été résolues en moins d'une minute et les résultats sont presque identiques à ceux des instances de 100 sommets.

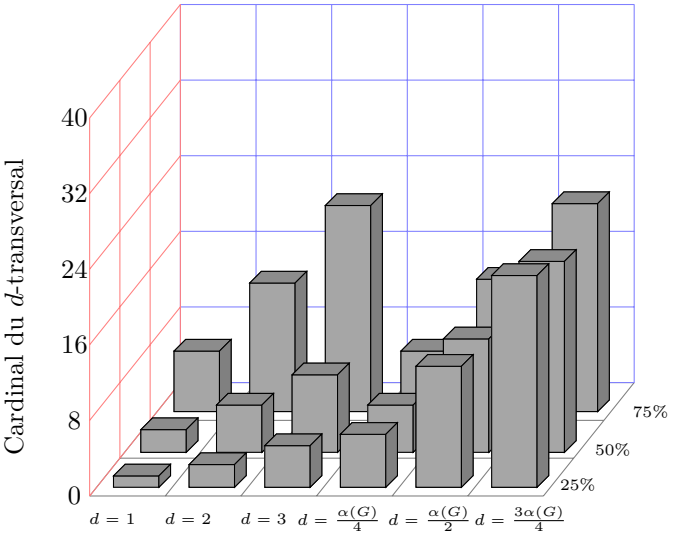
Les résultats des tests sur les instances de 100 à 200 sommets sont présentés sous la forme de six histogrammes. Le premier indique le nombre d'instances résolues. Le deuxième indique le cardinal du d-transversal déterminé. Le troisième indique le nombre d'itérations. Le quatrième indique le temps de calcul total en secondes et est détaillé dans les cinquième et sixième histogrammes qui indiquent respectivement le temps de calcul du problème maître ($PtransMaster$) et le temps de calcul du sous-problème ($Popt_{stable}(x)$).

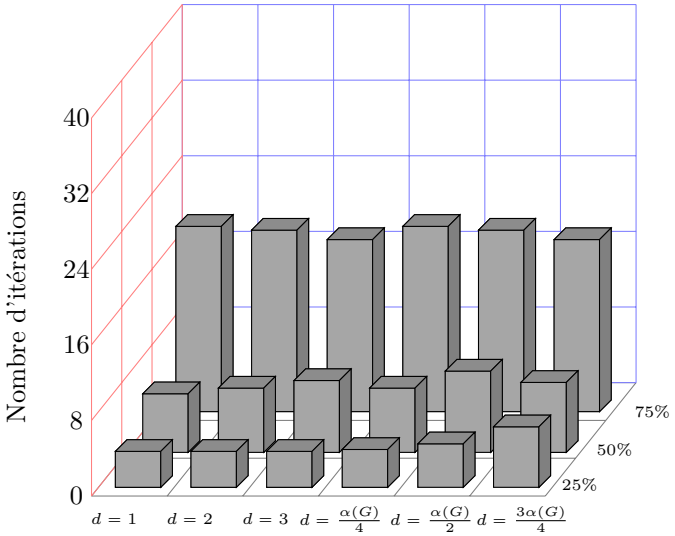
Pour chaque histogramme, l'axe horizontal indique la valeur de d, l'axe vertical indique ce qui est mesuré et le troisième axe indique la densité des instances. Pour une meilleure visibilité, sur les trois premiers histogrammes, la densité est indiquée de façon croissante et dans les trois derniers histogrammes, l'ordre des densités indiquées est décroissant.

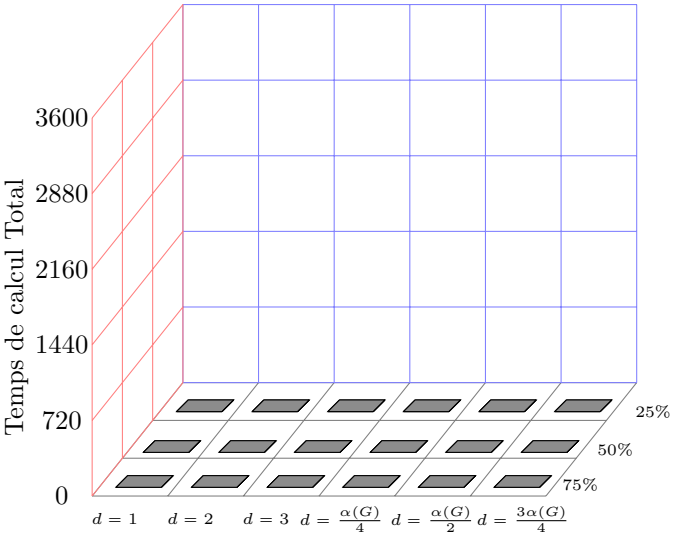
Tests variante 1

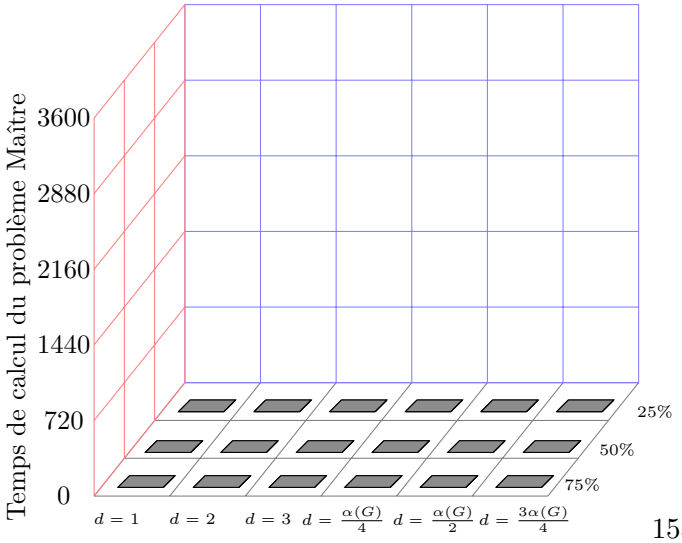
Instances de 100 sommets

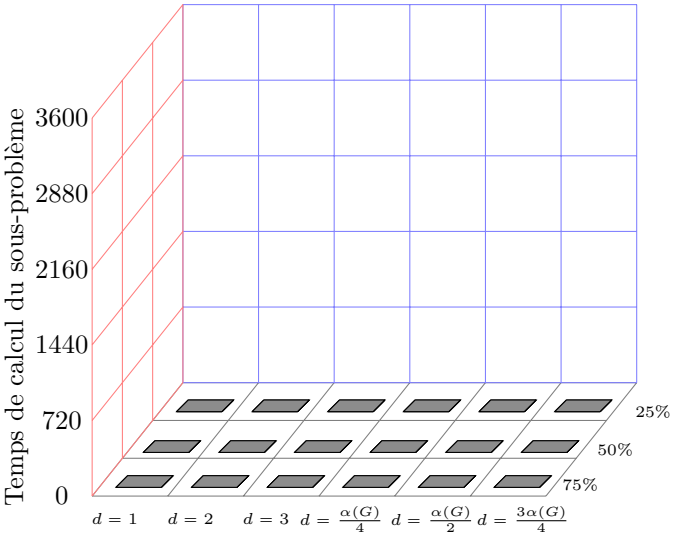




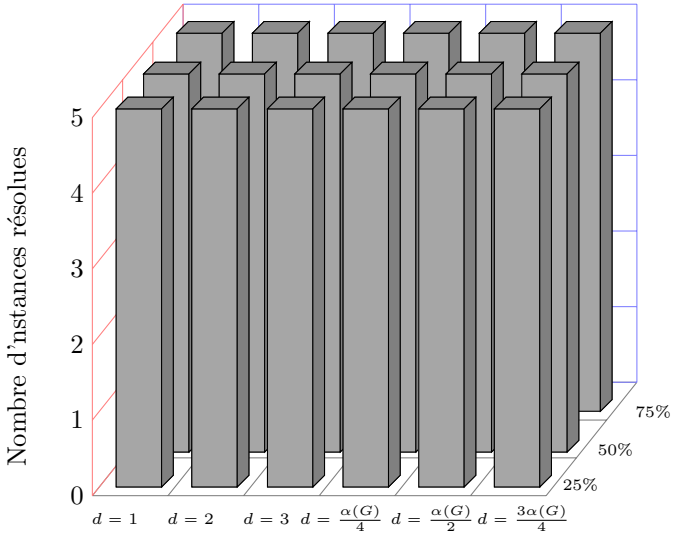


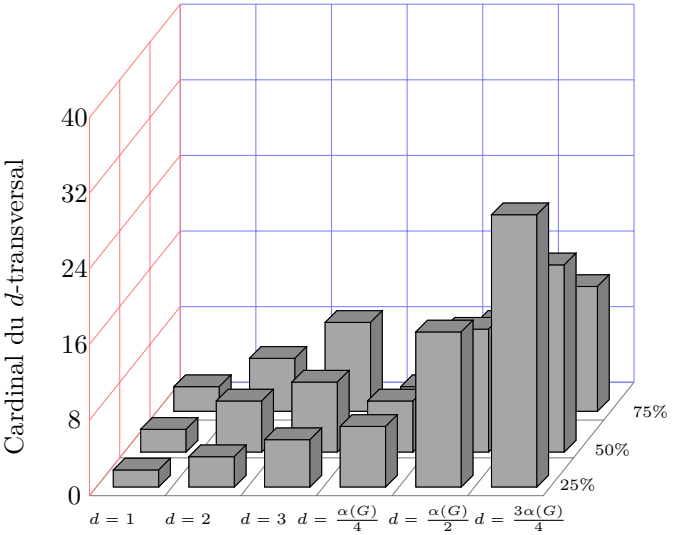


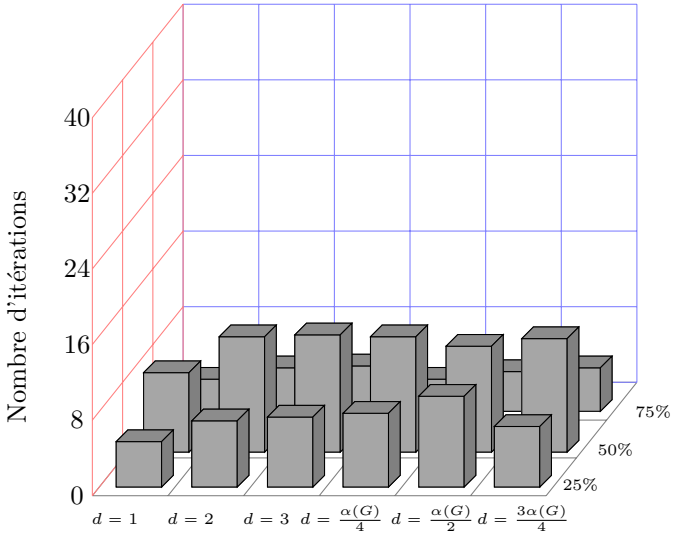


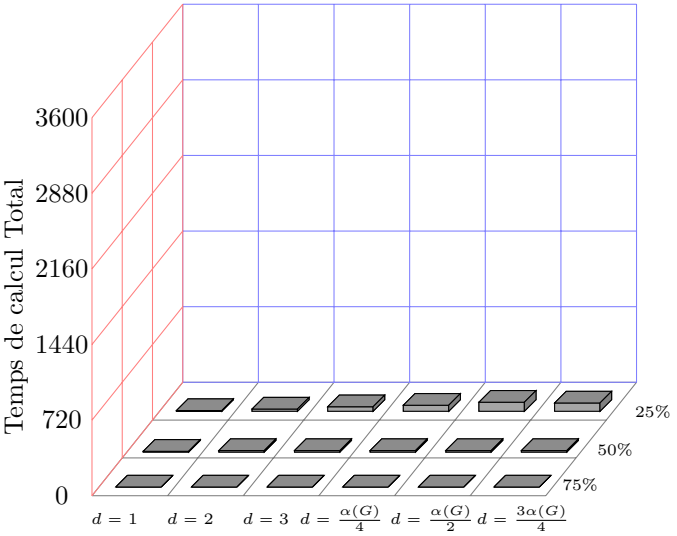


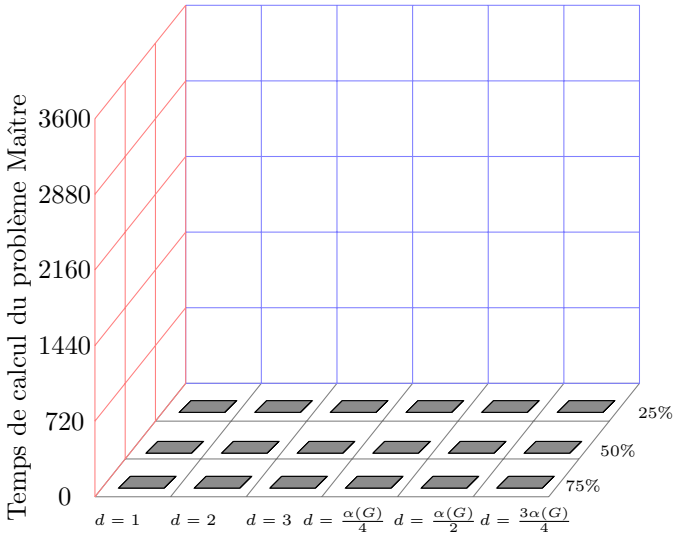
Instances de 120 sommets

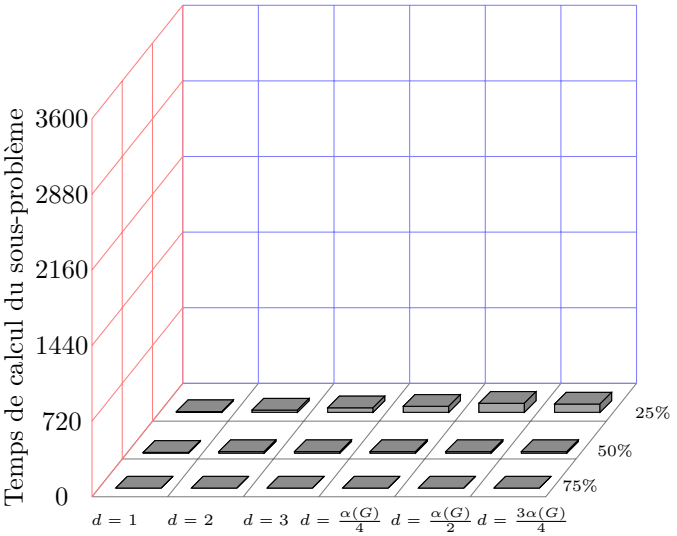




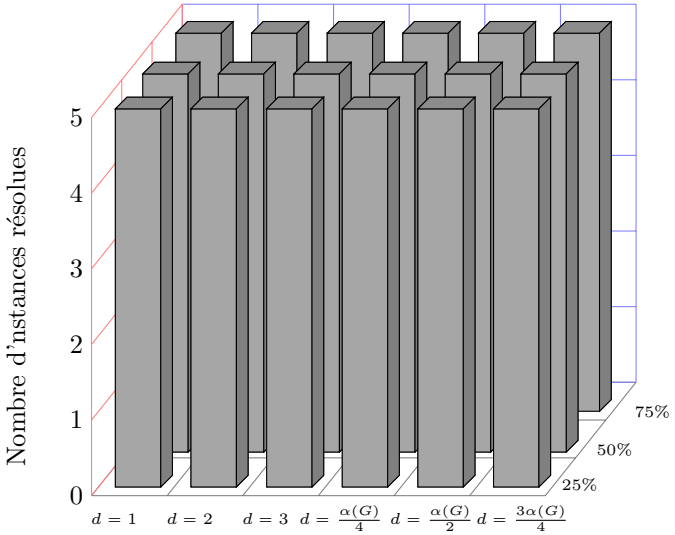


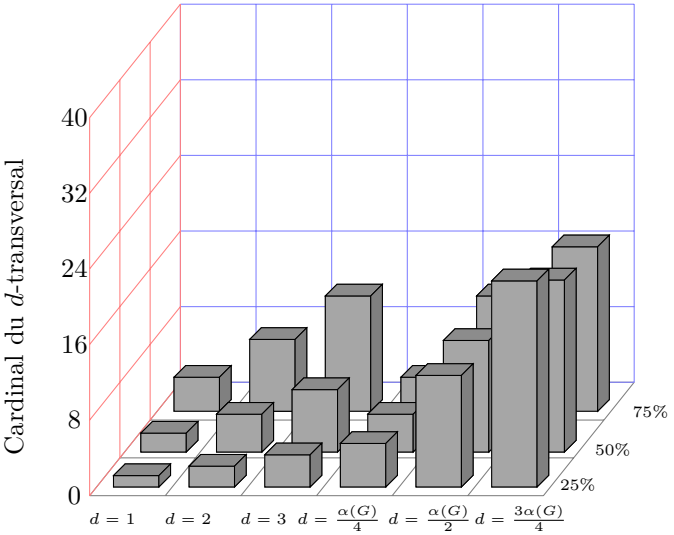


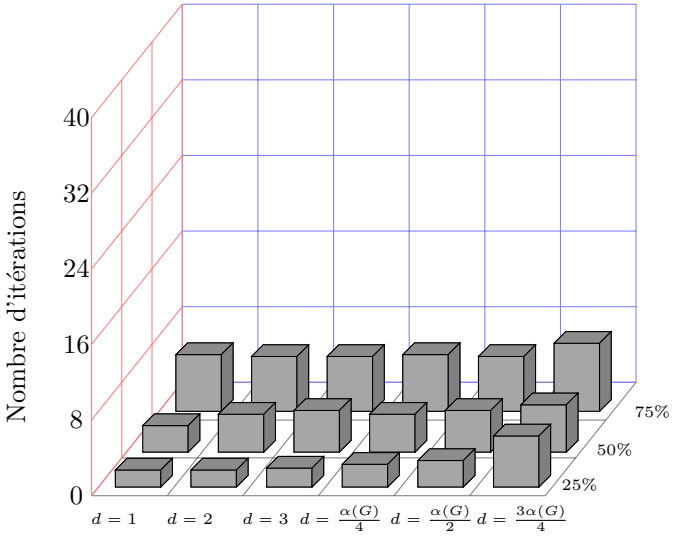


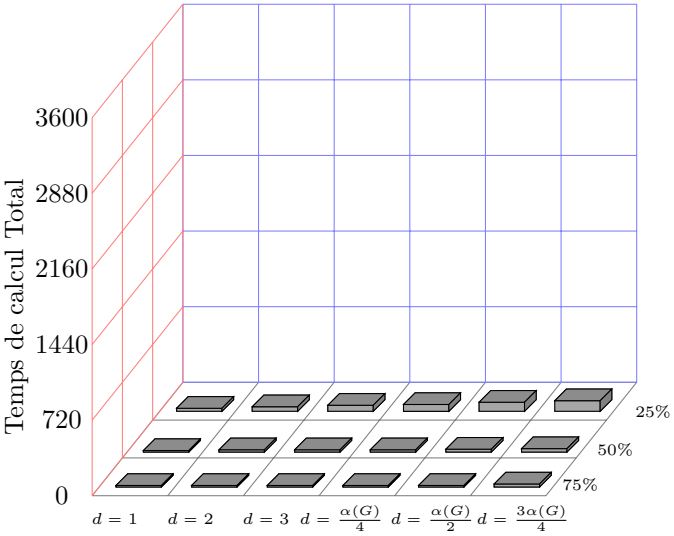


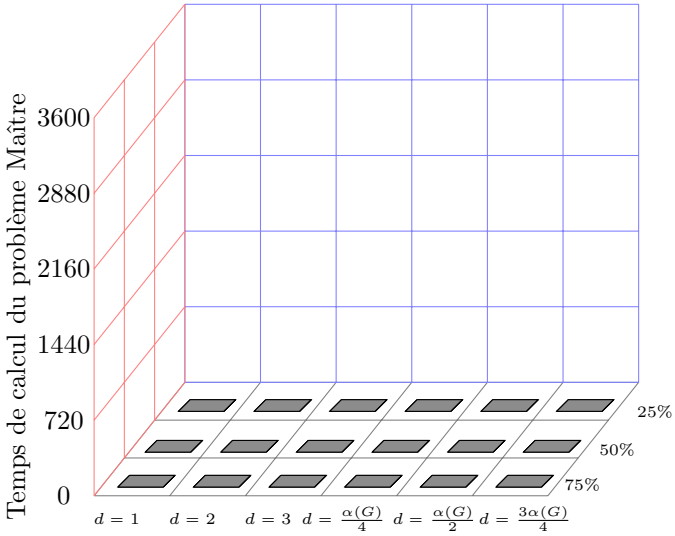
Instances de 140 sommets

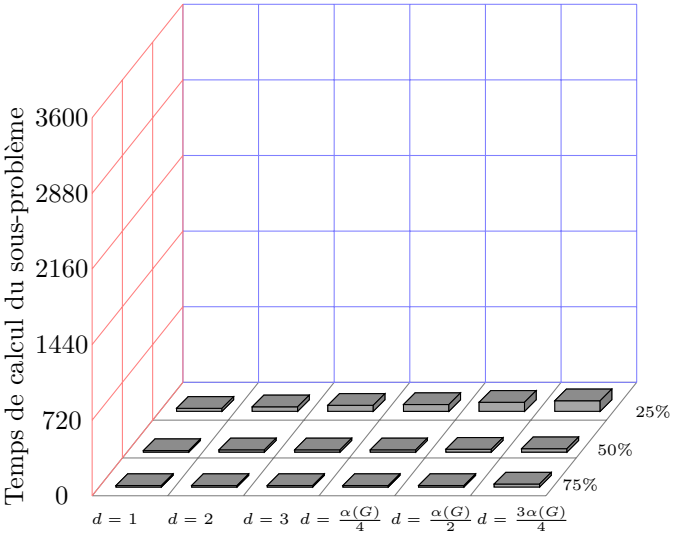




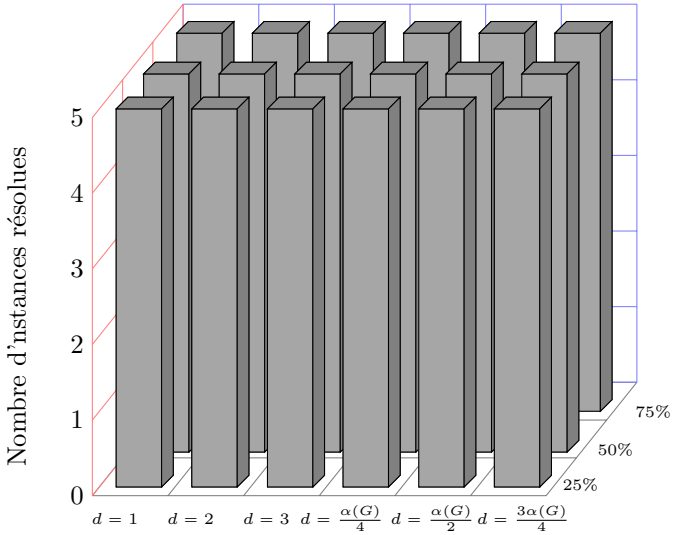


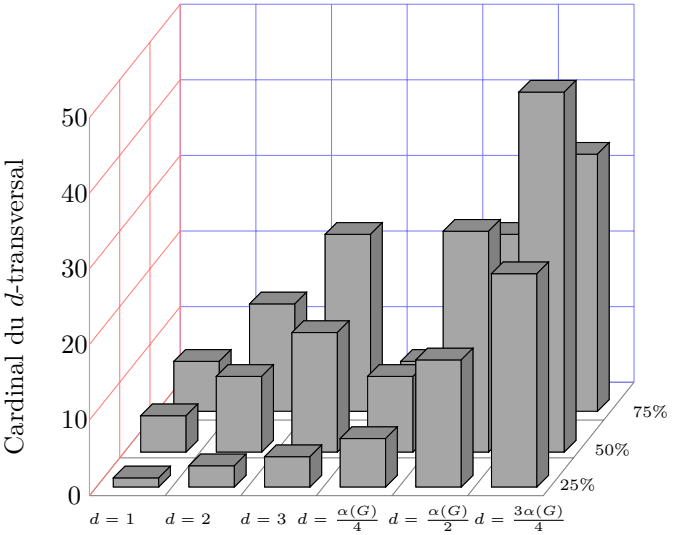


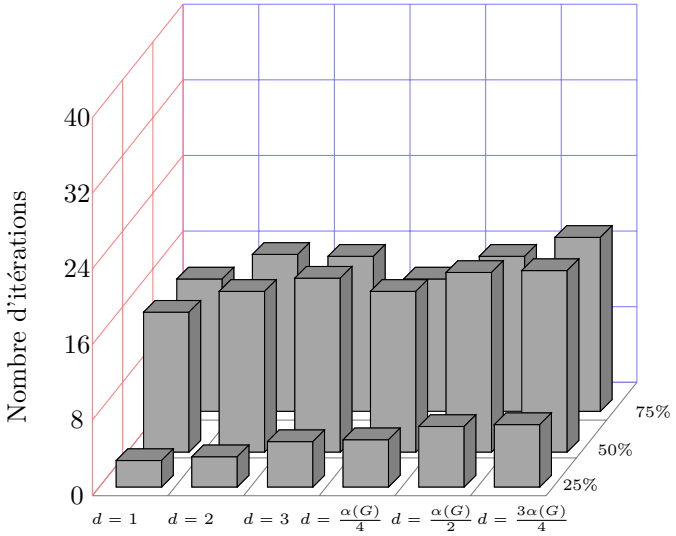


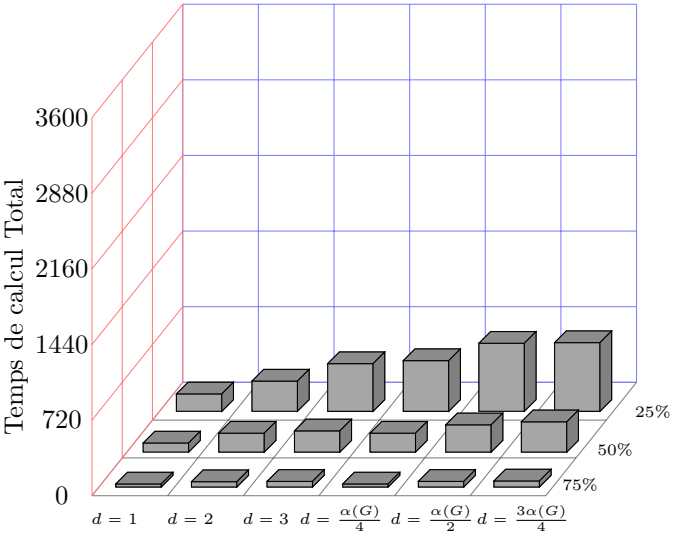


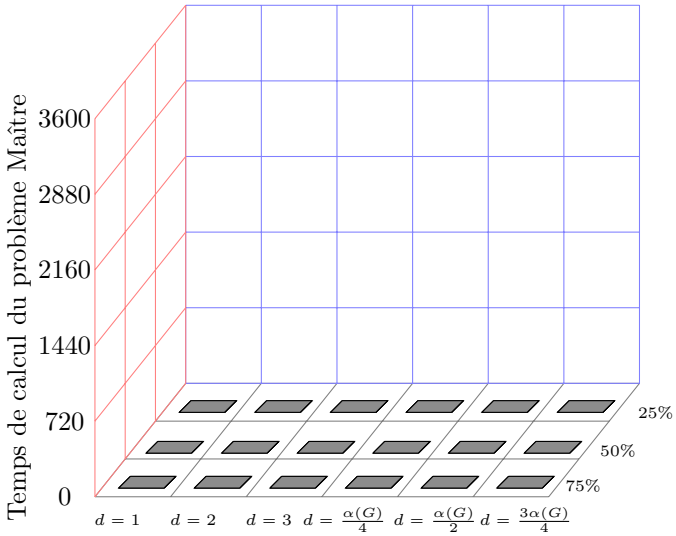
Instances de 160 sommets

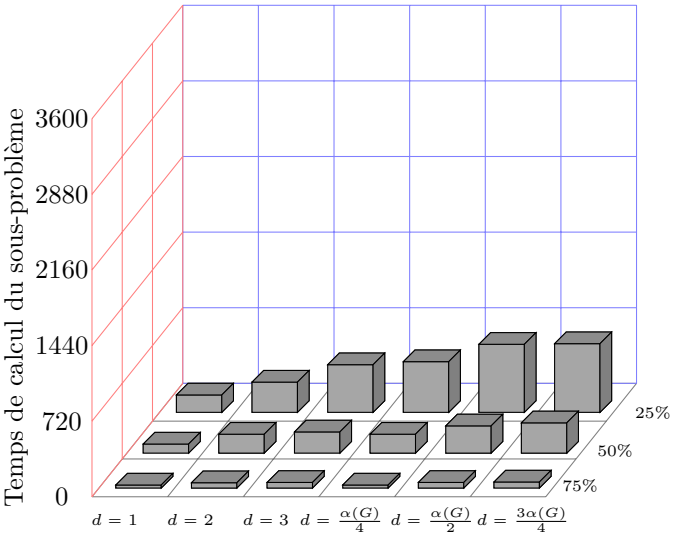




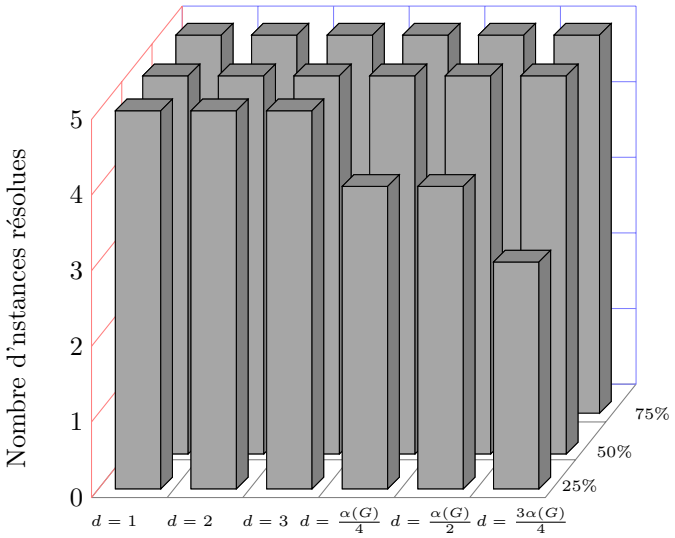


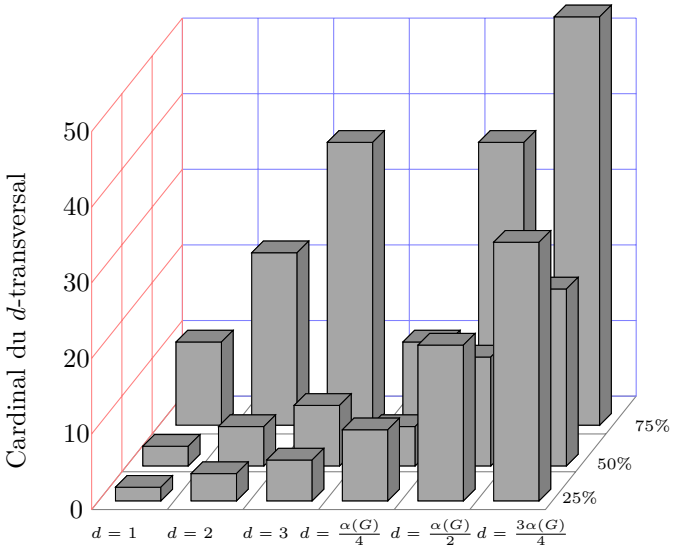


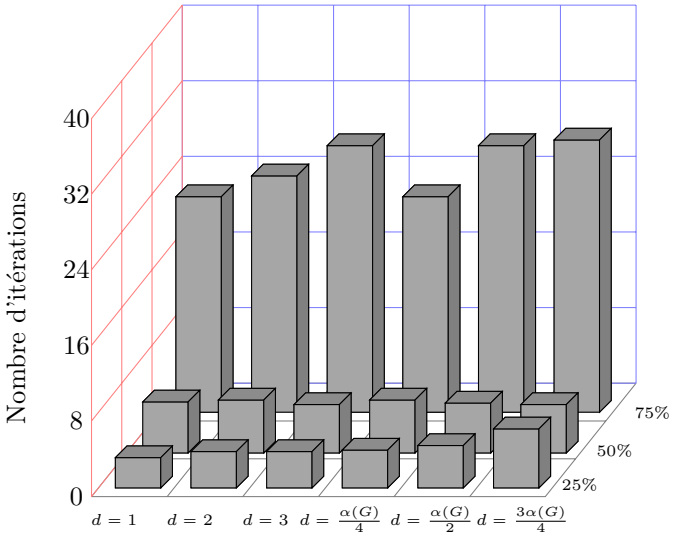


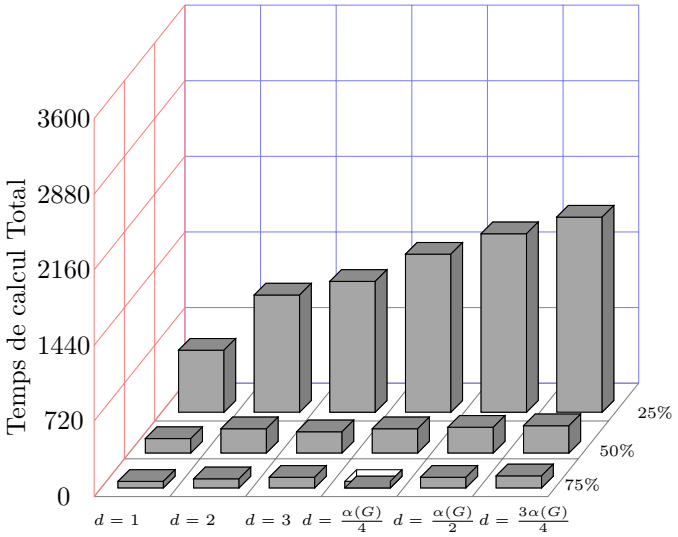


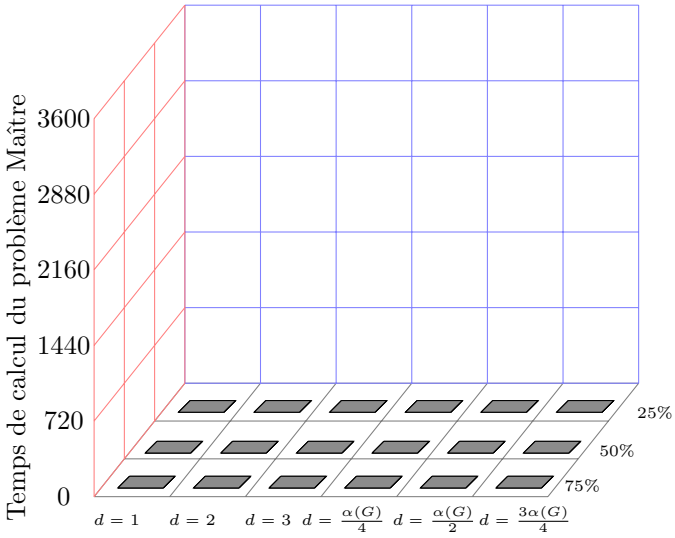
Instances de 180 sommets

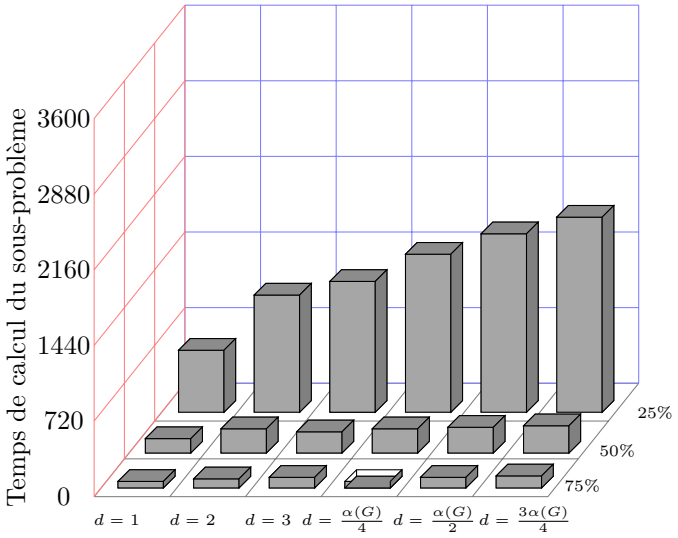




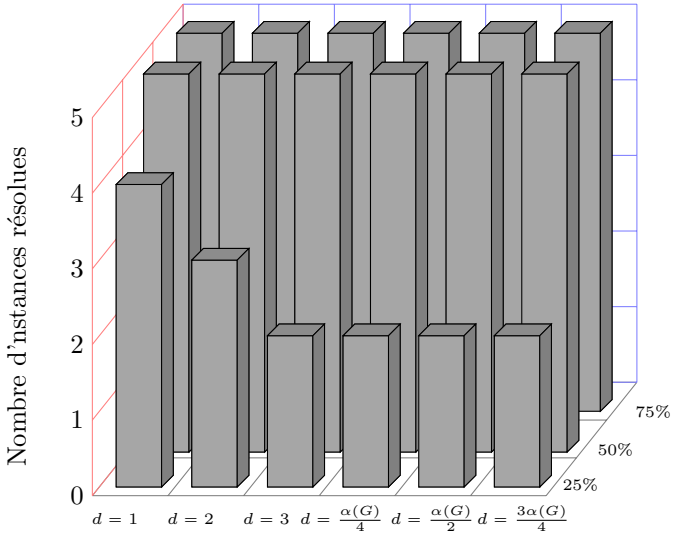


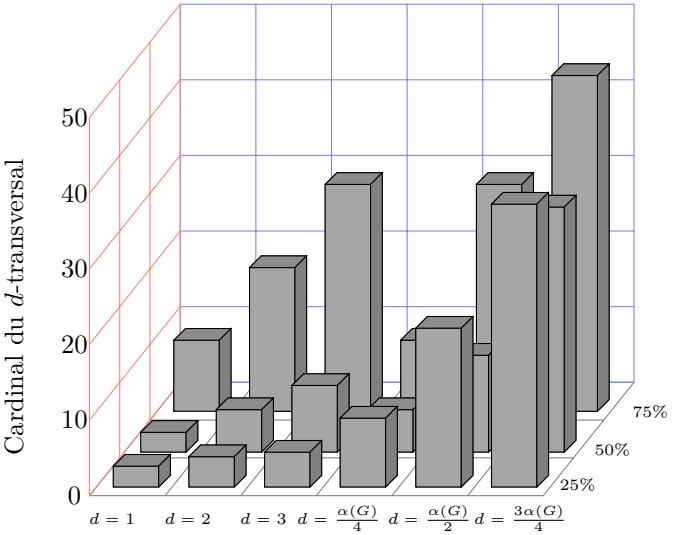


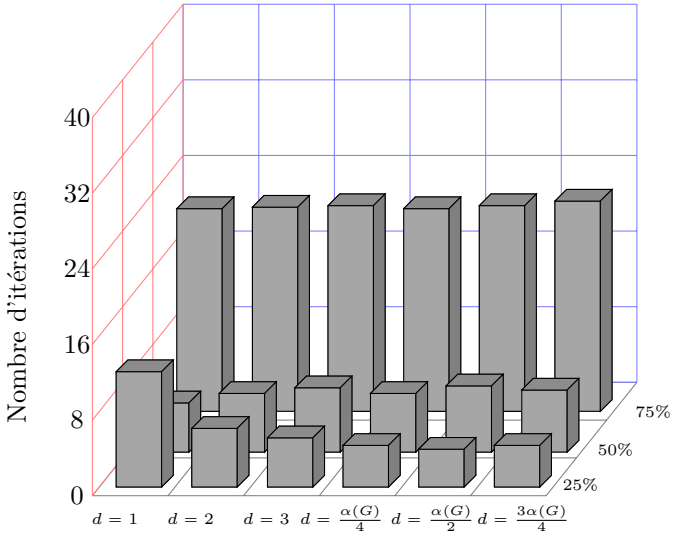


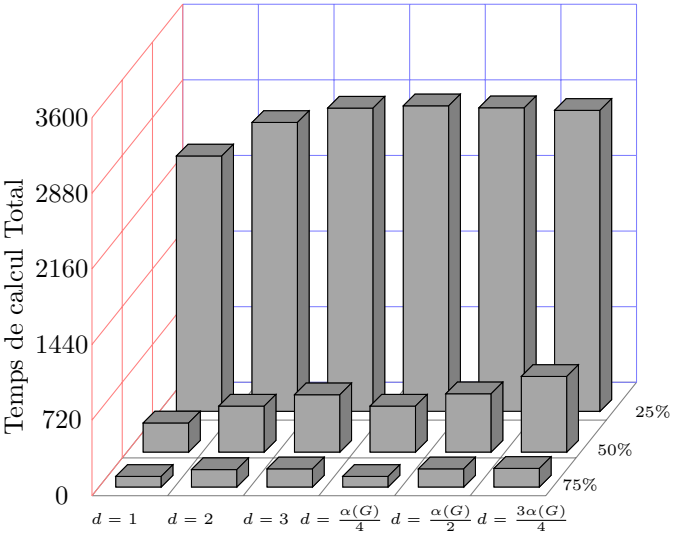


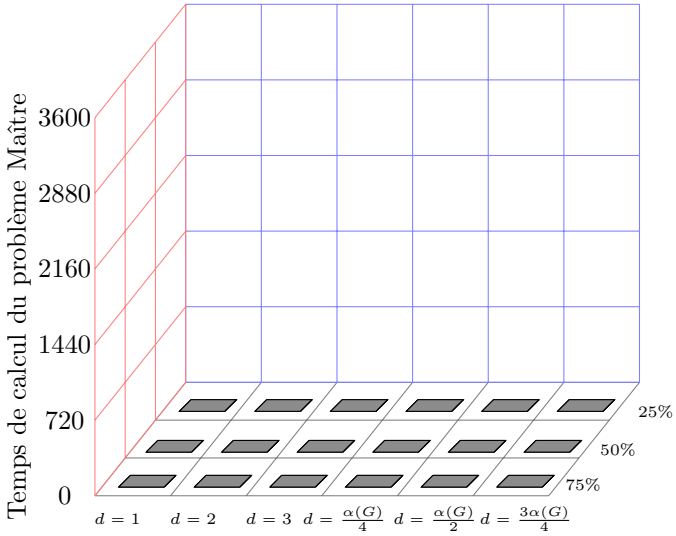
Instances de 200 sommets

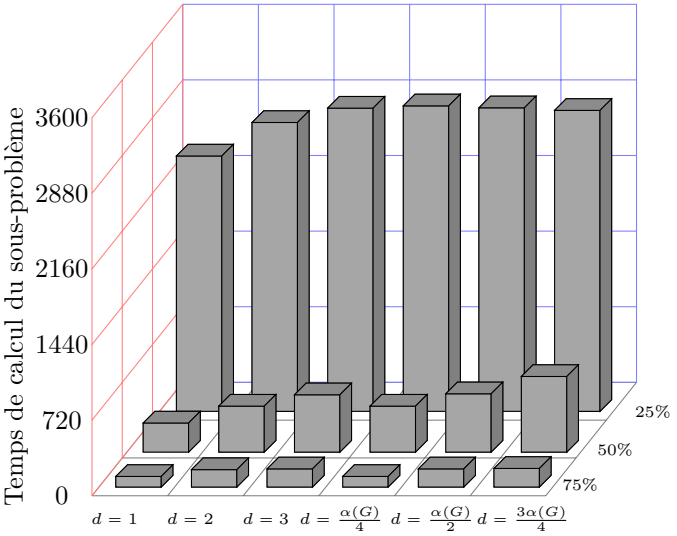












Résultats variante 1

Comme indiqué plus haut, les instances de 20 à 80 sommets ont toutes été résolues en moins d'une minute. Les résultats des instances de 100 à 200 sommets sont présentées sous la forme de six histogrammes. Le premier indique le nombre d'instances résolues. Le deuxième indique le cardinal du d-transversal déterminé. Le troisième indique le nombre d'itérations. Le quatrième indique le temps de calcul total en secondes et est détaillé dans les cinquièmes et sixièmes histogrammes qui indiquent respectivement le temps de calcul de problème maître et le temps de calcul du sous-problème.

On constate dans les résultats que toutes les instances de 160 sommets ou moins ont été résolues. Le temps de calcul du problème maître est instantané, c'est le calcul d'un stable optimal à chaque itération qui prend du temps. Les cardinaux des d-transversaux augmentent avec la valeur de d. Le nombre d'itérations augmente avec la densité du graphe mais les itérations sont plus rapides pour les instances denses que pour les instances peu denses. Jusqu'à 140 sommets, toutes les instances sont résolues en moins de trois minutes. Les temps de calcul sont inversement proportionnels à la densité et si les instances peu denses nécessitent plus d'itérations, celles-ci sont plus rapides que pour les instances plus denses. Les temps de calcul augmentent ensuite rapidement et les première instances non résolues apparaissent à 180 sommets pour les instances peu denses. Toutes les instances de densité 50 et 75 % ont été résolues. Pour les instances de 200 sommets, une analyse des instances de densité 25 % résolues révèle que ces instances ne contiennent aucun sommet libre donc qu'un d-transversal est déterminé en deux itérations. Les instances de 200 sommets de densité 25 % qui ne contenaient pas de sommets forcés n'ont pas été résolues.

Variante 2

Nous allons maintenant tester la deuxième variante de l'approche et comparer son efficacité avec la première variante. Le programme mathématique résolu à chaque itération est le suivant. Les variables x_i valent 1 si et seulement si le sommet i est sélectionné dans T et les variables y_i valent 1 si et seulement si le sommet i appartient à un stable optimal dont l'intersection avec T est la plus petite possible.

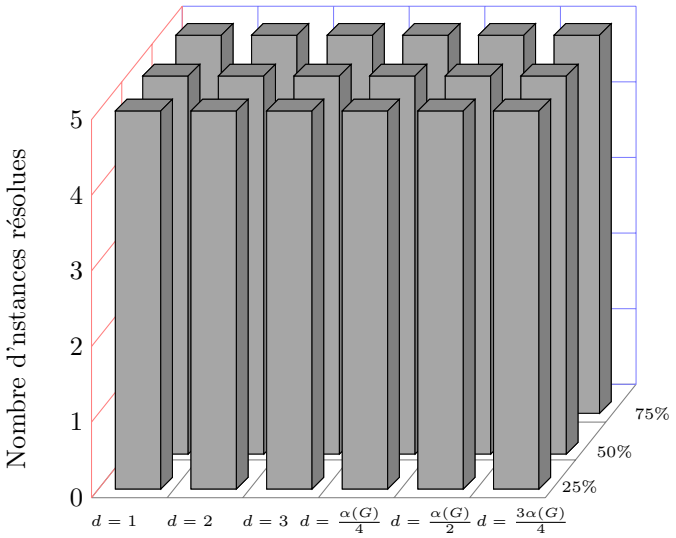
$$(Ptrans2_{Stable}) \begin{cases} \min\limits_{x,y} & \sum\limits_{i=1}^{n} x_i + \dfrac{1}{n^2}\sum\limits_{i=1}^{n} x_i y_i & \\ s.c. & \sum\limits_{i=1}^{n} x_i h_i^k \geqslant d & \forall k \text{ tel que } \{m_i \text{ tel que } h_i^k = 1\} \in Y \\ & y_i + y_j \leqslant 1 & \forall ij \in E \\ & \sum\limits_{i=1}^{n} y_i = \alpha(G) & \\ & x \in \{0,1\}^n & \\ & y \in \{0,1\}^n & \end{cases}$$

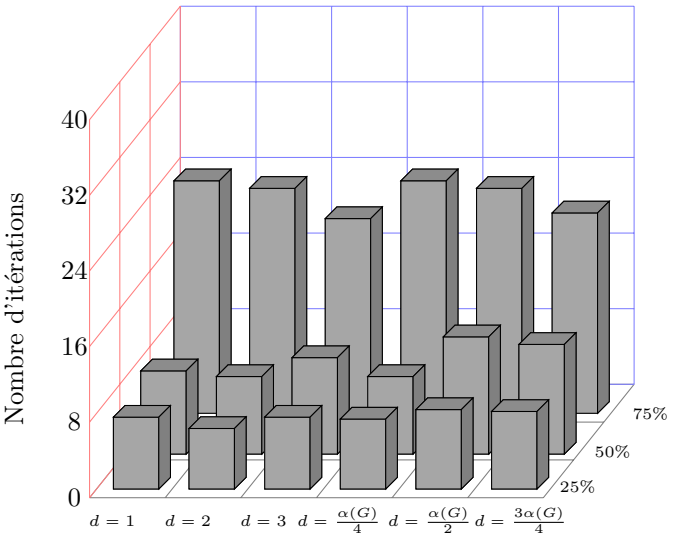
On retrouve les contraintes qui imposent que y est un stable optimal du graphe. La pénalité affectée à l'intersection de x et de y a été fixée à $\epsilon = \frac{1}{n^2}$. Ainsi, on est certain que $\frac{1}{n^2}\sum\limits_{i=1}^{n} x_i y_i < 1$, donc que x sera toujours de plus petite taille possible.

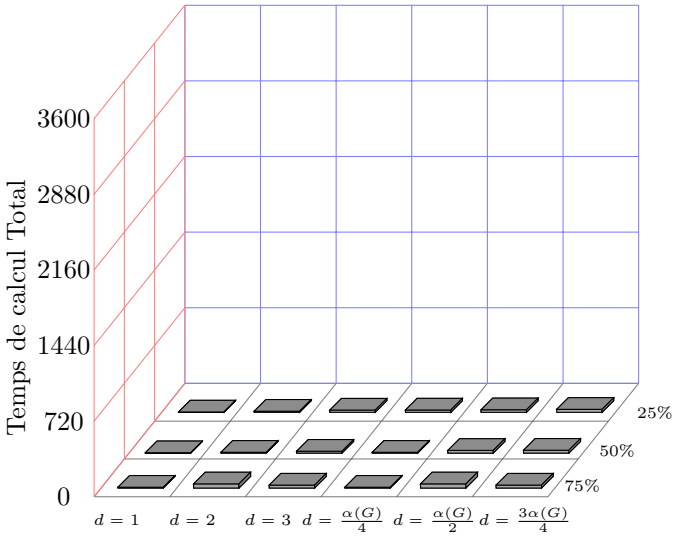
Les tests ont été effectués sur les mêmes instances qu'à la section précédente. Les histogrammes qui indiquaient le nombre de sommets retirés sont identiques à ceux de la section précédente. Ils n'apparaissent donc pas. Puisque un seul programme est résolu à chaque itération, il n'y a plus d'histogrammes détaillant le temps de calcul.

Tests variante 2

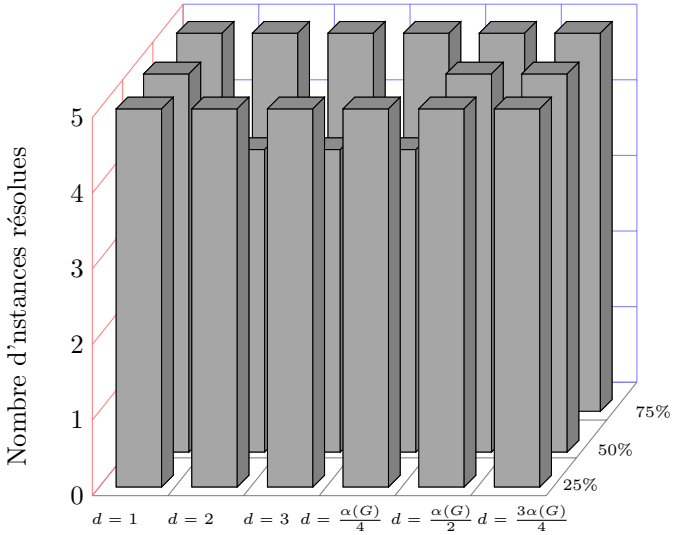
Instances de 100 sommets

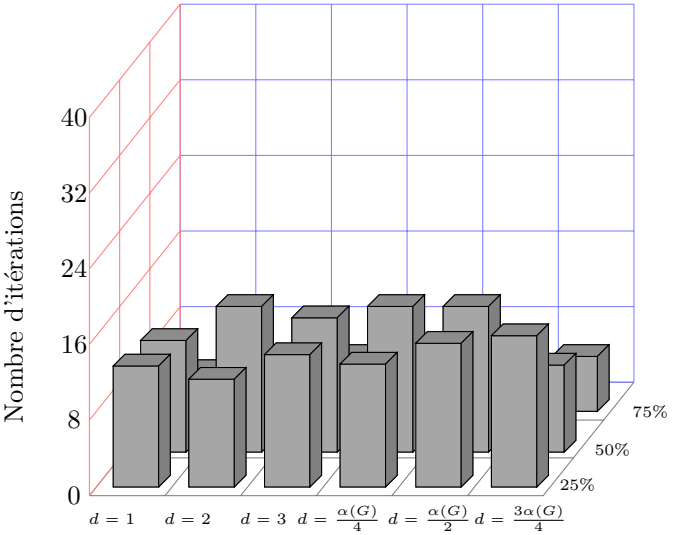


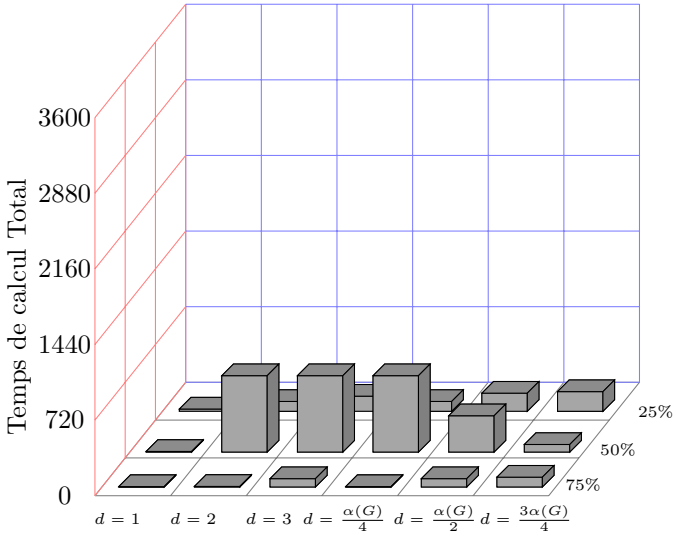




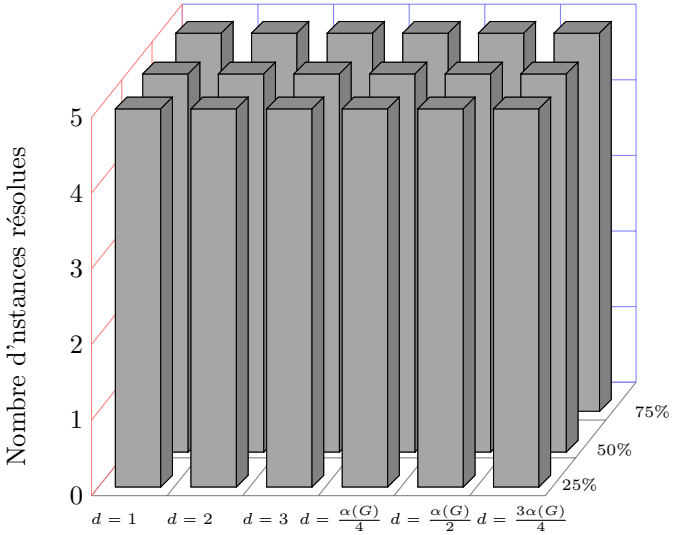
Instances de 120 sommets

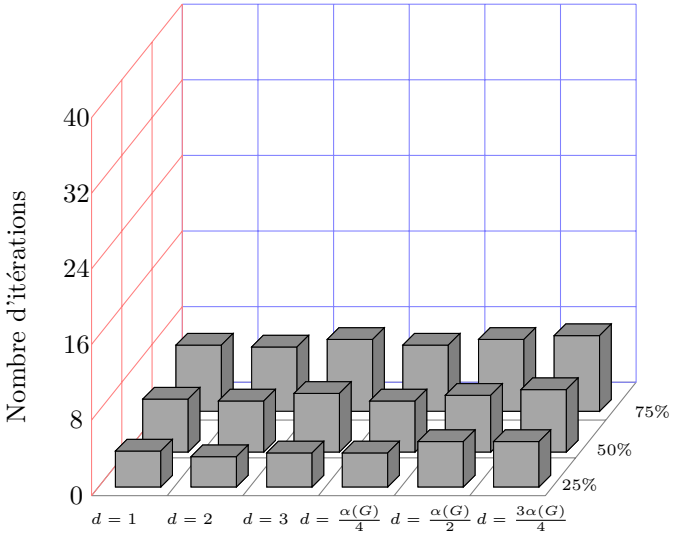


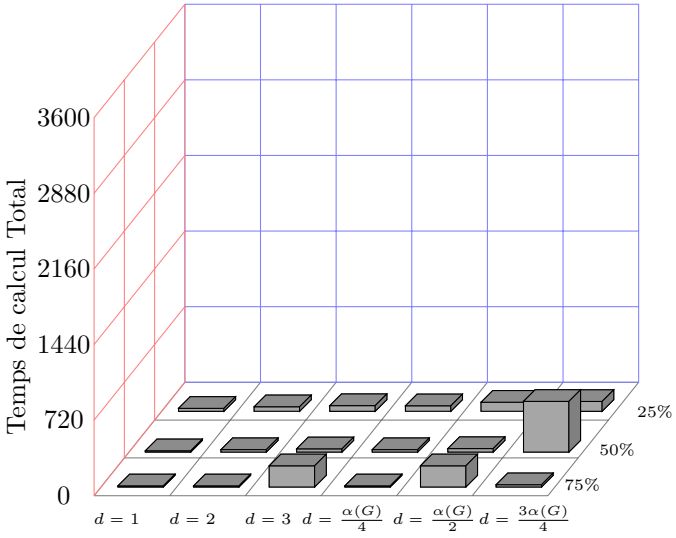




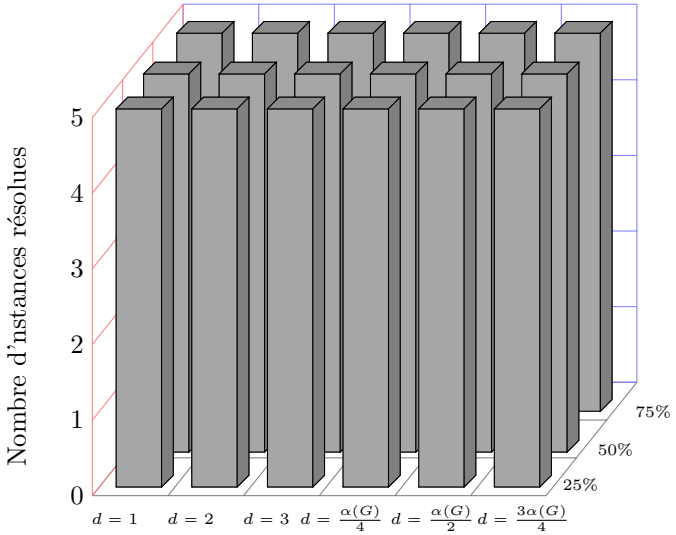
Instances de 140 sommets

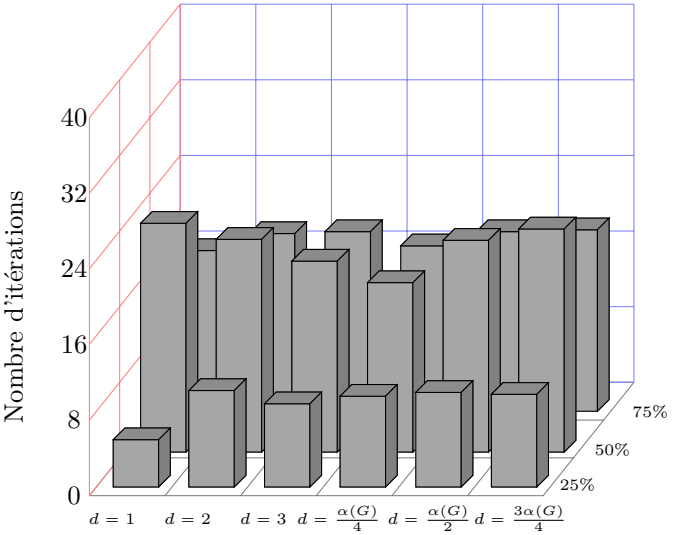


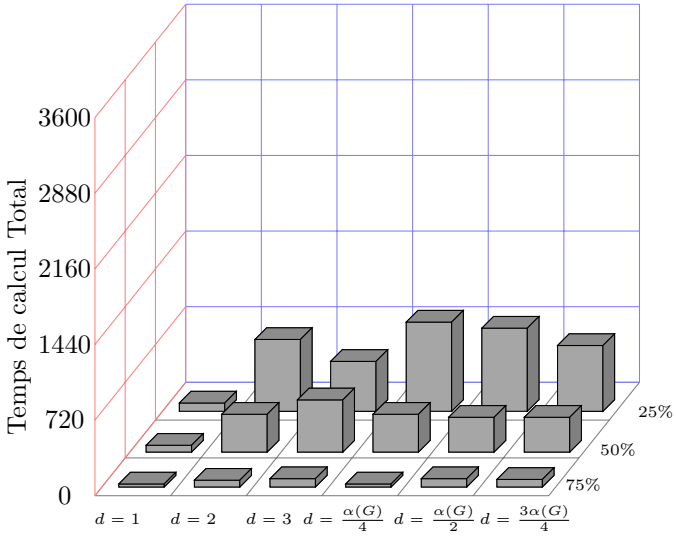




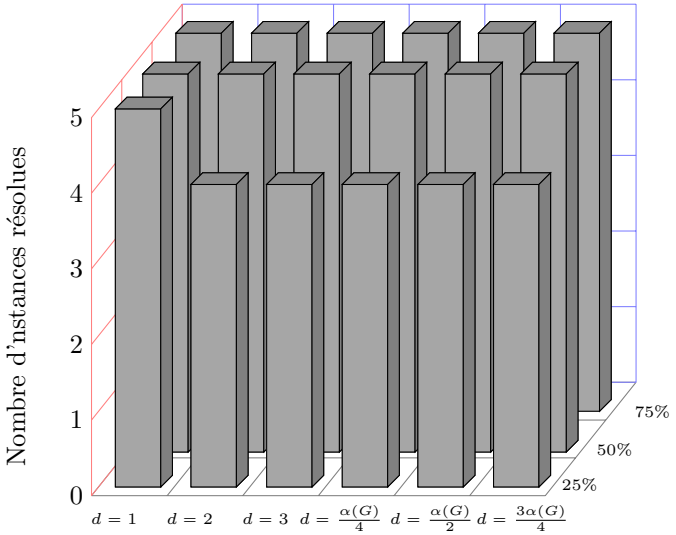
Instances de 160 sommets

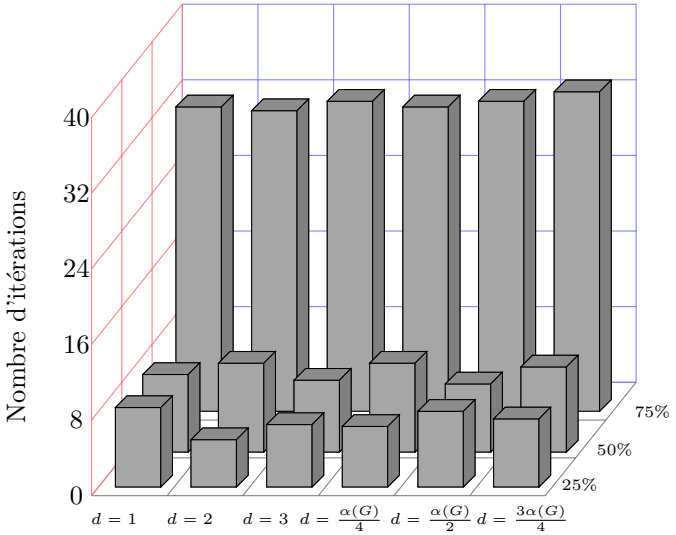


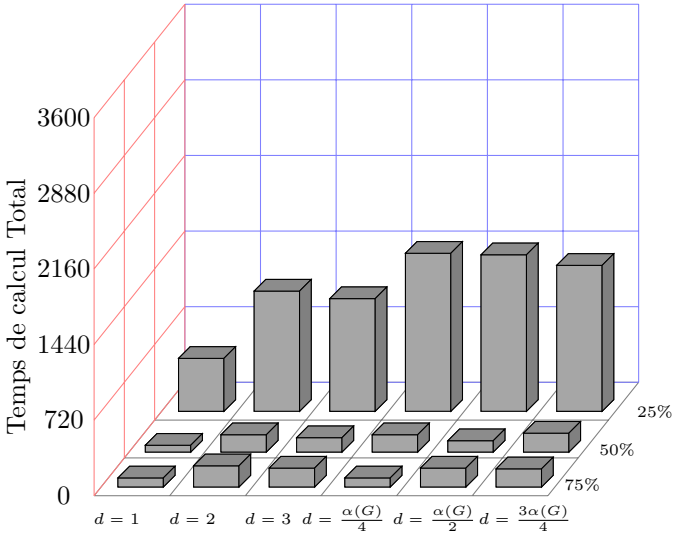




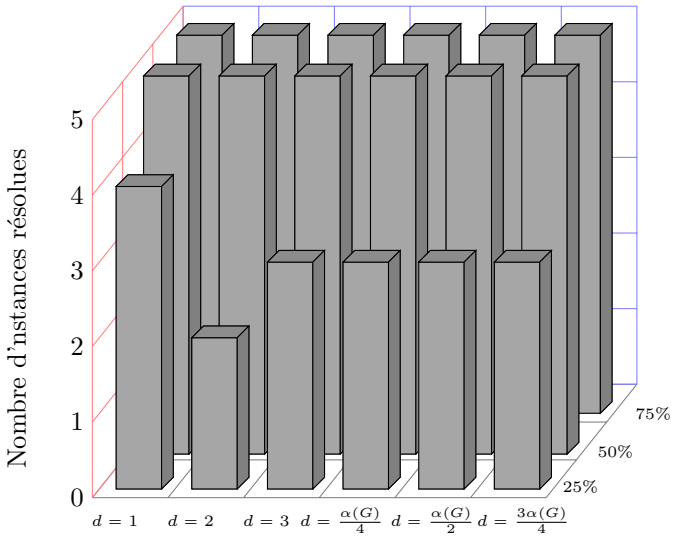
Instances de 180 sommets

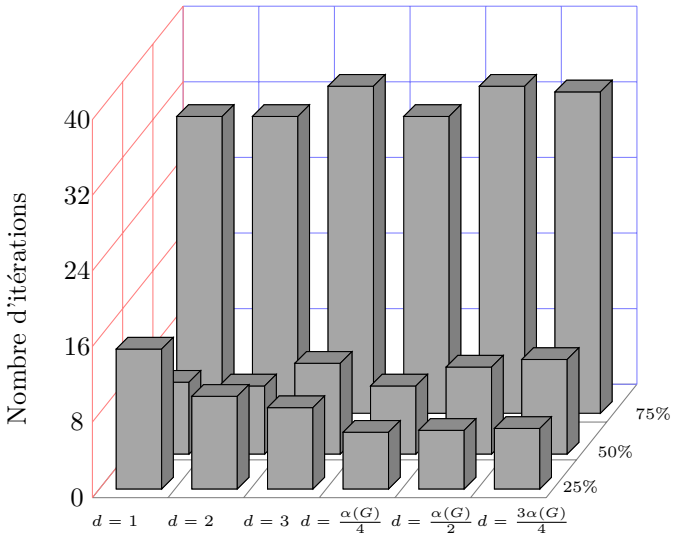


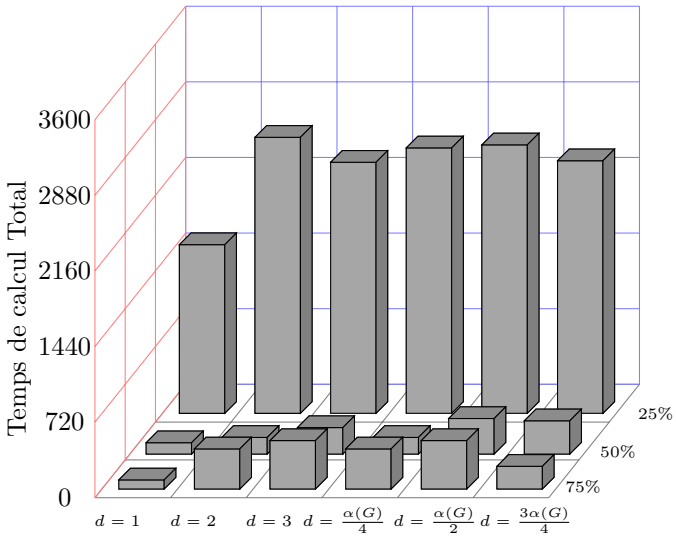




Instances de 200 sommets







Résultats variante 2

On constate que contrairement aux tests de la première variante, le nombre d'itérations est cette fois stable si d varie. Une instance de 120 sommets n'a pas été résolue dans le temps imparti pour plusieurs valeurs de d. Par contre, on observe que certaines instances de 200 sommets sont résolues alors que la première variante ne parvenait pas à les résoudre. Le temps de résolution de la deuxième variante de l'approche est généralement plus important que le temps de résolution de la première variante de l'approche

En étudiant les instances, on observe que dans les quelques cas où la deuxième variante

est plus performante que la première, les instances étaient principalement composées de sommets forcés et étaient résolus en peu d'itérations. Par contre, l'instance de 120 sommets non résolue par la deuxième approche ne contenait pas de sommets forcés, et la recherche s'est arrêtée après plus de 40 itérations. Pour confirmer ou non cette corrélation entre l'efficacité de la deuxième variante et le nombre d'itérations nécessaires à la résolution, il faudrait effectuer plus de tests sur des instances pour lesquels le nombre de sommets forcés et le nombre de solutions optimales sont fixes. Mais générer de telles instances peut être difficile. Cela peut constituer une piste pour de futurs travaux.

Tentatives d'amélioration des approches

Dans cette section, nous évoquons plusieurs modifications de l'approche de résolution qui ont été testées pour augmenter la vitesse de résolution. Toutes ces tentatives ont été infructueuses.

Recherche préliminaire de sommets forcés D'après les Propriétés 5.2.2, 5.2.1 et 5.2.3 page 138, un d-transversal ne contient pas de sommets exclus, mais contient tous les sommets forcés si $d > |\Xi(G)|$ et est inclus dans $\Xi(G)$ sinon. Un pré-traitement possible à l'approche de générations de contraintes présentée est donc de déterminer tous les éléments forcés du graphe. Si $d \leqslant |\Xi(G)|$ alors il est possible de construire un d-transversal. Sinon, on applique la procédure précédente sur le sous-graphe de G qui ne contient que ses sommets libres pour trouver un $(d - |\Xi(G)|)$-transversal.

Pour déterminer tous les sommets forcés, une méthode possible consiste à déterminer un stable optimal du graphe, à considérer l'ensemble X des sommets de ce stable, dont le vecteur caractéristique est x puis à résoudre $(Popt_{stable}(x))$. Puis on retire de X les sommets qui n'appartiennent pas à la solution optimale de $(Popt_{stable}(x))$. On réitère ensuite l'opération jusqu'à ce qu'on ne puisse plus retirer de sommets de x, c'est-à-dire qu'on ne puisse plus trouver de stable optimal qui ne contienne pas les sommets de X. Ainsi, X contient les sommets forcés du graphe.

Cette méthode a été implémentée comme pré-traitement avant l'exécution de la première variante de l'approche. Les tests effectués augmentent le temps de calcul par rapport

aux tests obtenus avec la première variante seule. Ce pré-traitement n'est donc pas efficace.

Une autre méthode pour déterminer les sommets forcés d'un graphe $G = (V, E)$ est de déterminer un stable optimal S et, pour tout sommet $x \in S$, déterminer un stable optimal S_2 du graphe $G[V - \{x\}]$. Si S_2 a un cardinal inférieur à S alors x est un sommet forcé et sinon on retire de l'ensemble des sommets forcés potentiels ceux qui n'appartiennent pas $S \cap S_2$. Cette méthode s'exécute en au plus $\alpha(G)$ itérations mais est moins efficace que la précédente lorsque de nombreux sommets sont forcés. En effet, si tous les sommets de S sont forcés, la méthode présentée plus haut le détermine en deux itérations, et cette méthode en $\alpha(G)$ itérations. Cette méthode n'a donc pas été implémentée.

Formulation plus efficace du sous-problème Une autre piste envisagée est l'utilisation d'une formulation plus efficace pour le problème du stable. Plusieurs formulations sont présentées dans [41]. Nous avons testé la formulation la plus efficace. Les tests ont été effectués pour la première variante de l'approche. Le sous-problème ($Popt_{stable}(x)$) devient le programme suivant, dans lequel les variables et données sont les mêmes que dans le programme original.

$$(Popt_{stable}(x)) \begin{cases} \min\limits_{y} & \sum\limits_{i=1}^{n} y_i x_i \\ s.c & y_i + y_j + \sum\limits_{l \in L_{ij}} y_l \leqslant 1 \quad \forall ij \in E \\ & \sum\limits_{i=1}^{n} y_i = \alpha(G) \end{cases}$$

Dans ce modèle, L_{ij} désigne les sommets d'une clique à laquelle appartient i et j. Cette clique est obtenue par l'Algorithme 15.

Lors des tests, la modification du sous-problème de la première variante n'a pas changé les temps de calcul. Ce changement est donc inutile.

Linéarisation Le programme mathématique résolu à chaque itération de la deuxième variante de l'approche est quadratique. Il est possible de le linéariser en introduisant des variables z_i telles que $z_i = x_i y_i$. Le programme ($Ptrans2_{stable}$) peut se reformuler de la façon suivante avec les mêmes variables et de nouvelles variables z_i qui valent 1 si et seulement si les varaibles x_i et y_i valent 1.

Algorithm 15 Algorithme déterminant une clique qui contient deux sommets i et j, s'il en existe une

```
1: Arguments : deux sommets i et j
2: Soit N l'ensemble des sommets voisins communs à i et à j
3: for k ∈ N do
4:    for l ∈ N, l > k do
5:       if k et l ne sont pas voisins then
6:          N := N − {l}
7:       end if
8:    end for
9: end for
10: return N
```

$$(Ptrans2_{Stable}) \begin{cases} \min\limits_{x,y} & \sum\limits_{i=1}^{n} x_i + \dfrac{1}{n^2}\sum\limits_{i=1}^{n} z_i & \\ & \sum\limits_{i=1}^{n} x_i h_i \geqslant d & \forall C = \{v_i \text{ tel que } h_i = 1\} \in Y \\ & y_i + y_j \leqslant 1 & \forall ij \in E \\ & \sum\limits_{i=1}^{n} y_i = \alpha(G) & \\ & z_i \leqslant x_i & \forall i \in \{1..n\} \\ & z_i \leqslant y_i & \forall i \in \{1..n\} \\ & z_i \geqslant x_i + y_i - 1 & \forall i \in \{1..n\} \\ & x \in \{0,1\}^n & \\ & y \in \{0,1\}^n & \\ & z \in \{0,1\}^n & \end{cases}$$

Les temps de calcul des tests effectués en résolvant ce programme dans la deuxième variante augmentent ou restent les mêmes. Ce changement est donc inefficace.

5.4.4 d-transversaux optimaux de couplages

Dans cette section, nous testons les deux variantes de l'approche de résolution de d-transversaux par génération de contraintes en cherchant à déterminer des d-transversaux de couplage. La structure de cette section est identique à celle de la section précédente. Les instances testées sont les mêmes qu'à la section précédente.

Variante 1

Le sous-problème adapté ($Popt_{\Pi}(x)$) au problème du couplage maximum est le suivant. Les variables y_i valent 1 si et seulement si l'arête i appartient à un couplage optimal du

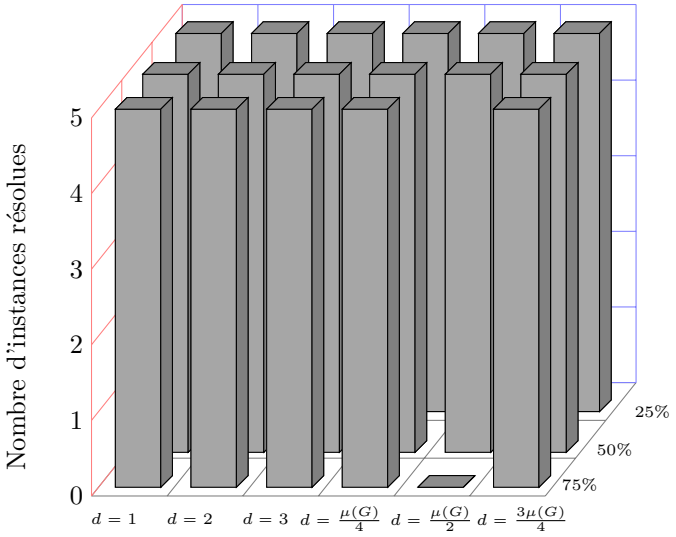
graphe qui a le moins d'éléments en commun avec T.

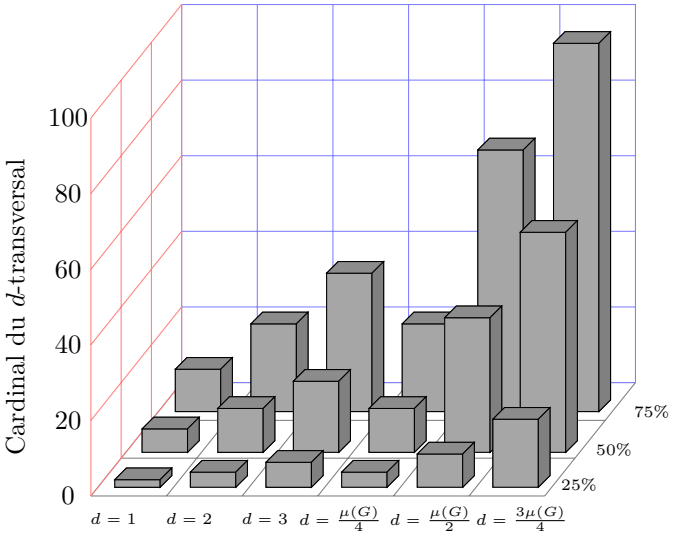
$$(Popt_{couplage}(x)) \begin{cases} \min\limits_{y} & \sum\limits_{i=1}^{n} y_i x_i \\ s.c & \sum\limits_{j \ \ incident \ \ \text{à} \ \ i} y_j \leqslant 1 \quad \forall i \in V \\ & \sum\limits_{i=1}^{n} y_i = \mu(G) \\ & y \in \{0,1\}^n \end{cases}$$

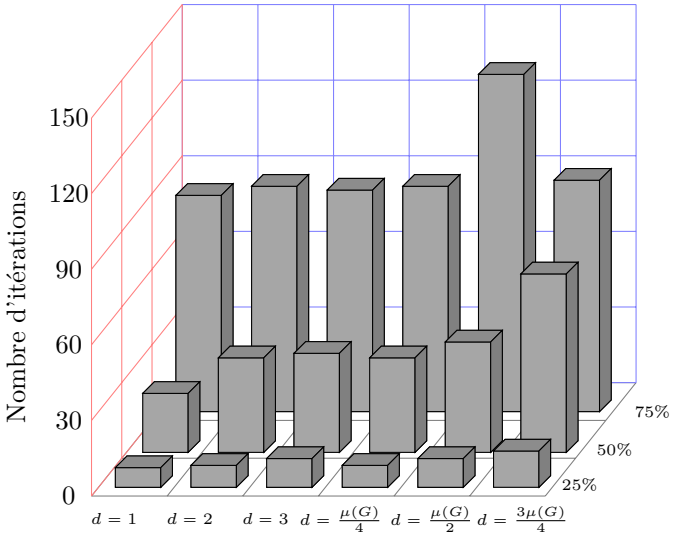
La première contrainte assure que la solution générée est bien un couplage et la deuxième contrainte assure que le couplage est optimal.

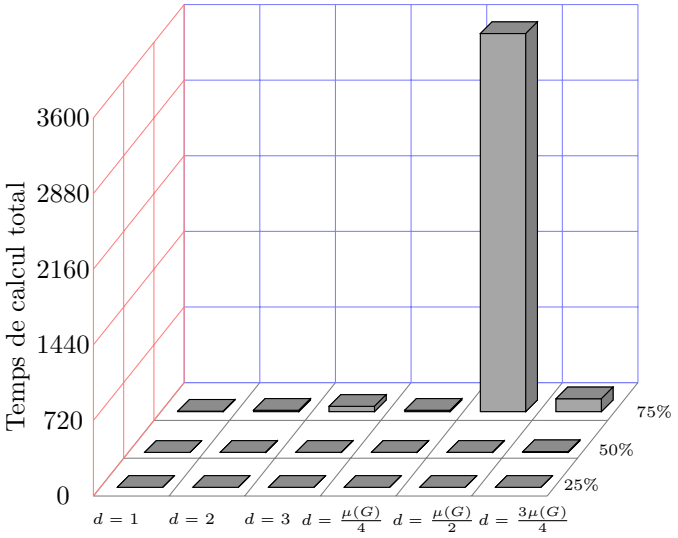
Tests variante 1

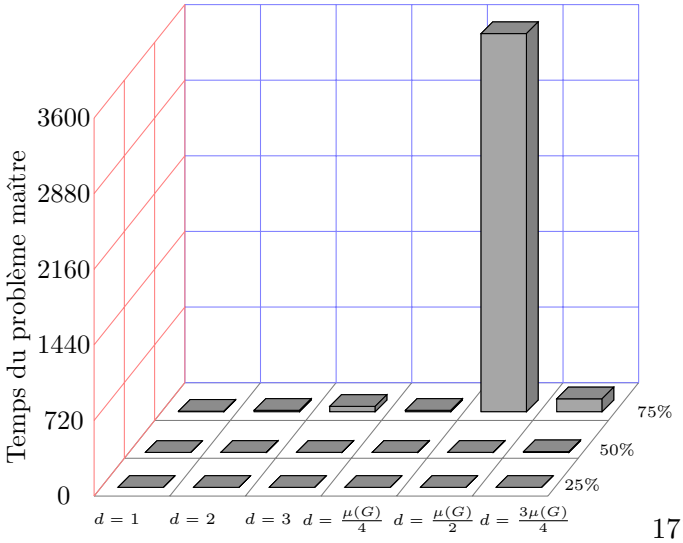
Instances de 20 sommets

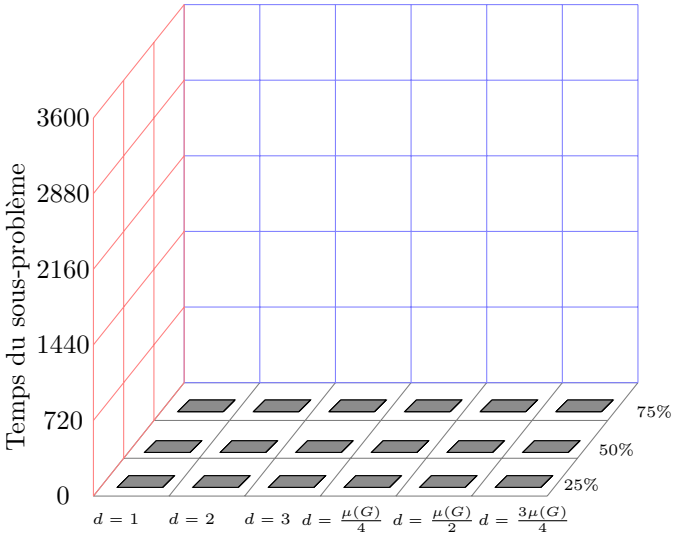




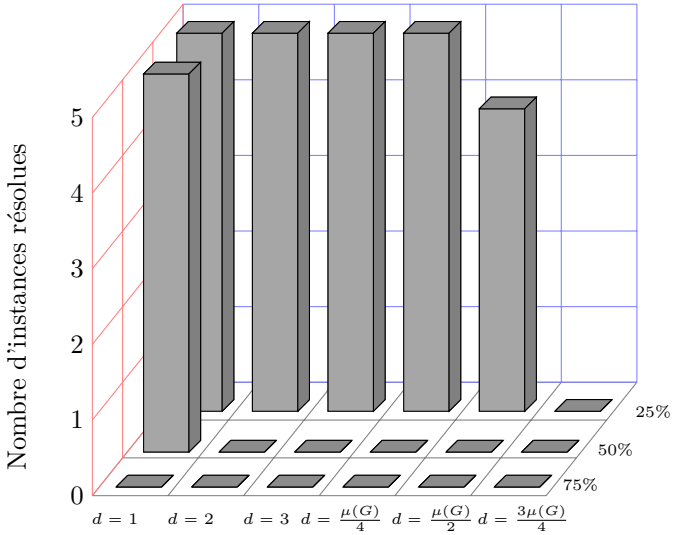


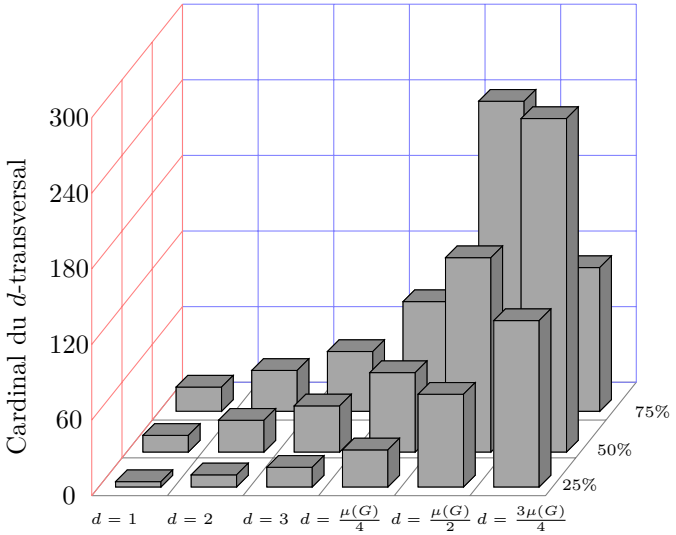


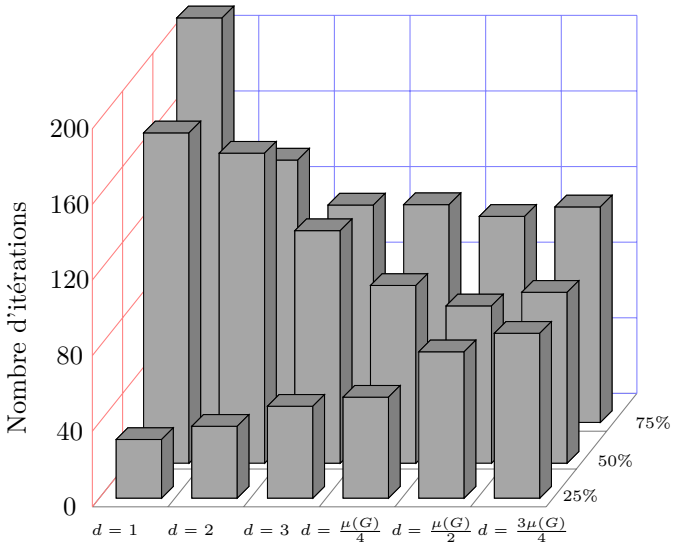


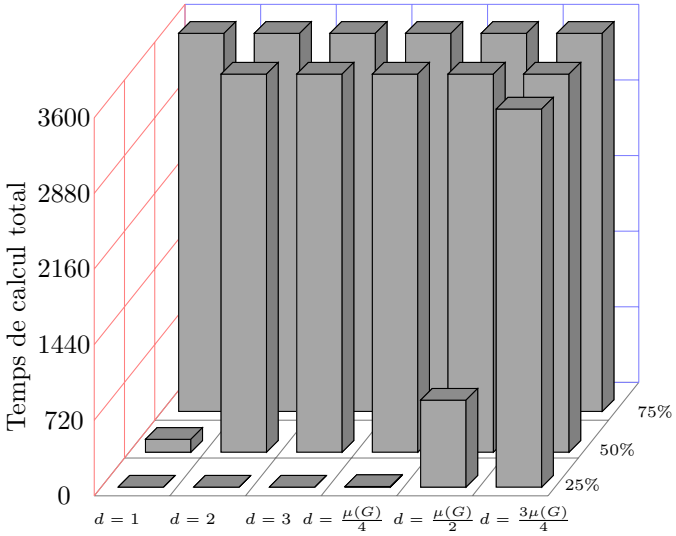


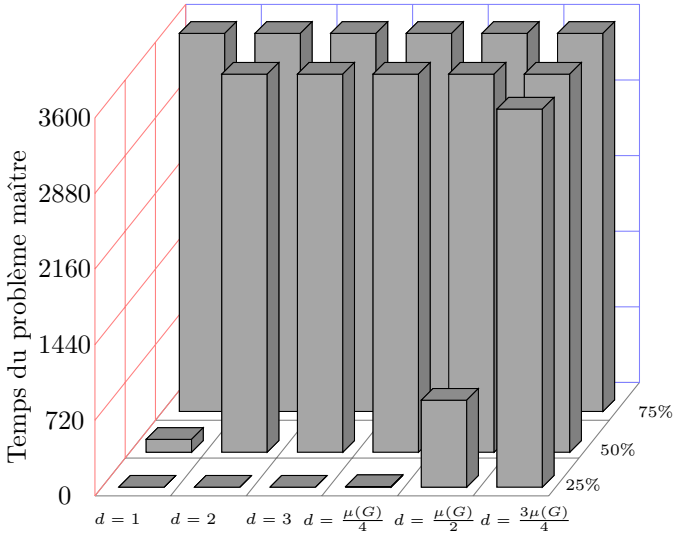
Instances de 40 sommets

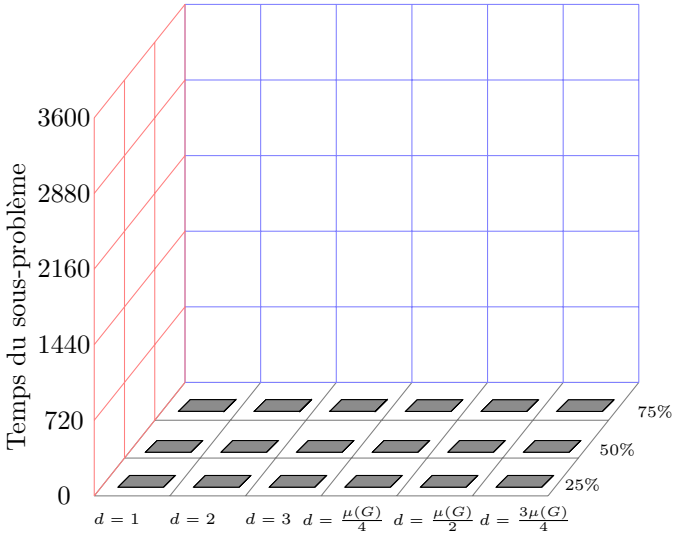












Résultats variante 1

Aucun d-transversal n'a été déterminé pour les instances de 60 sommets ou plus, pour $d > 1$ dans le temps de calcul imparti. On constate que, contrairement aux d-transversaux optimaux de stables, le temps de calcul est principalement consacré au problème maître. En effet, contrairement au problème du stable de cardinal maximal, le problème du couplage de cardinal maximal est polynomial et est résolu rapidement. On constate également que le nombre d'itérations est plus important que pour les d-transversaux de stables, ce qui peut expliquer le temps de calcul du problème maître qui a une contrainte en plus à chaque itération.

Pour les instances de 40 sommets de densité 50 % et 75 %, on remarque que le nombre d'itérations baisse lorsque d augmente. Cela vient du fait que ces instances ne sont pas résolues en une heure donc si d augmente, la résolution du problème maître devient plus longue.

Variante 2

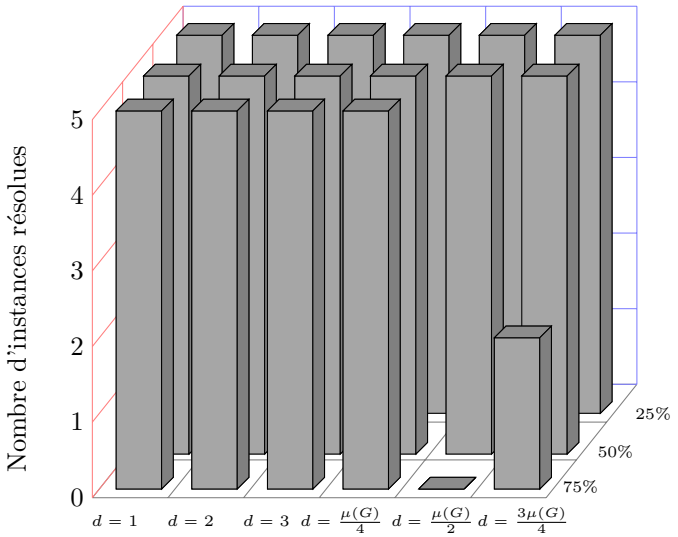
Le programme $(Ptrans2_{\Pi})$ adapté aux couplages donne le programme suivant. Les variables x_i sont égales à 1 si et seulement si l'arête i appartient à T et les variables y_i valent 1 si et seulement si l'arête i appartient à un couplage optimal du graphe qui a le moins d'arêtes en commun avec T.

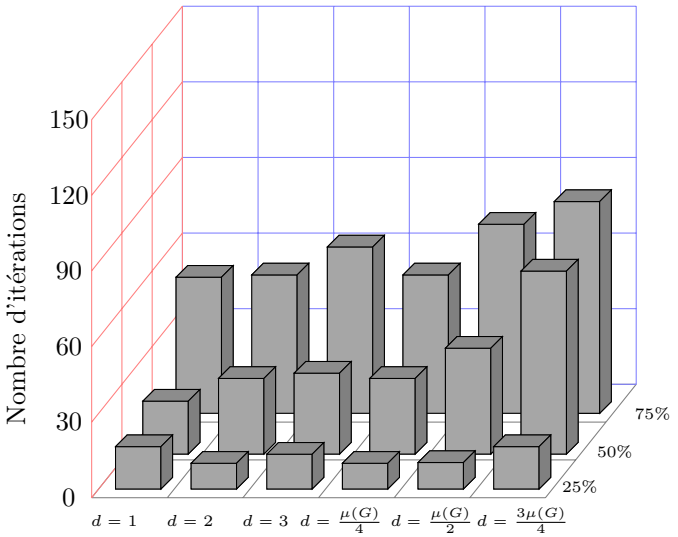
$$(Ptrans2_{Couplage}) \begin{cases} \min\limits_{x,y} & \sum\limits_{i=1}^{n} x_i + \dfrac{1}{n^2}\sum\limits_{i=1}^{n} x_i y_i & \\ & \sum\limits_{i=1}^{n} x_i h_i^k \geqslant d & \forall k \textit{ tel que } \{m_i \textit{ tel que } h_i^k = 1\} \in Y \\ & \sum\limits_{j \ \textit{incident à} \ i} y_j \leqslant 1 & \forall i \in V \\ & \sum\limits_{i=1}^{n} y_i = \mu(G) & \\ & x \in \{0,1\}^n & \\ & y \in \{0,1\}^n & \end{cases}$$

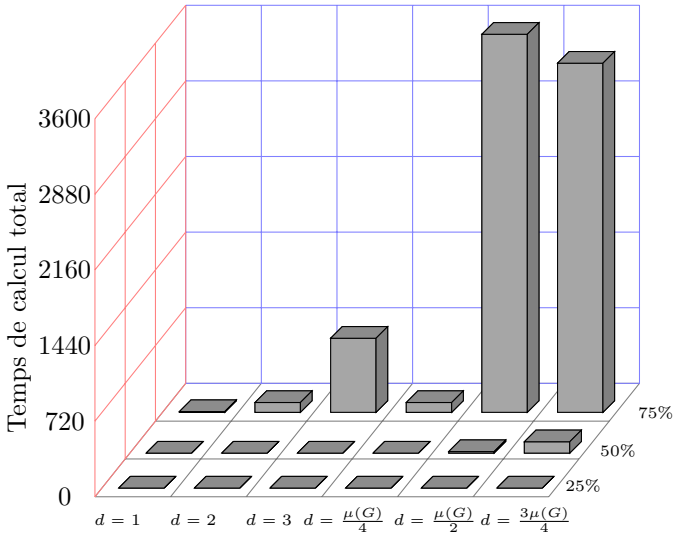
La première contrainte assure toujours que l'intersection entre le d-transversal et les solutions optimales déjà trouvées est bien de d éléments. La deuxième contrainte assure que y est un couplage et la troisième contrainte assure que y est un couplage optimal.

Tests variante 2

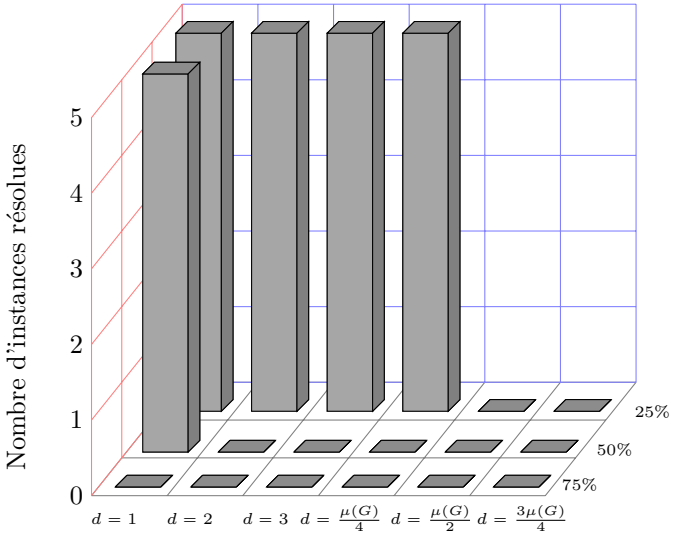
Instances de 20 sommets

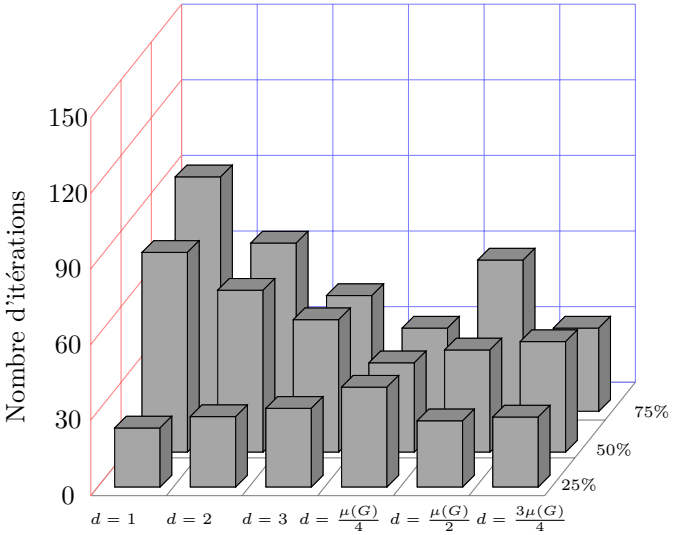


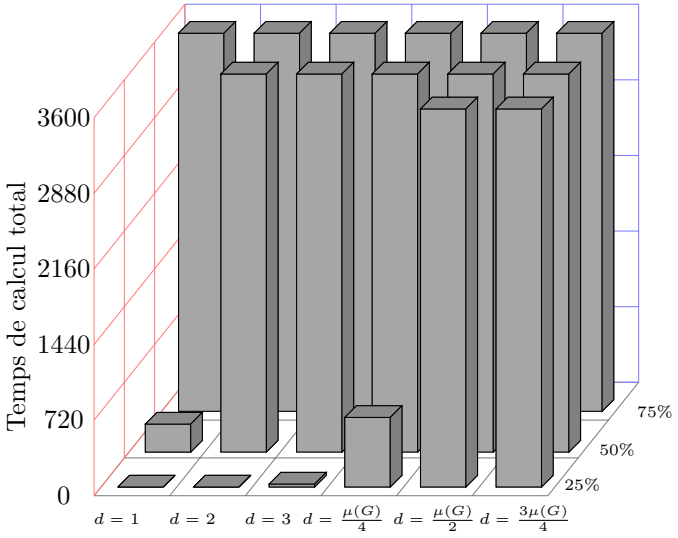




Instances de 40 sommets







Résultats variante 2

On peut constater que pour toutes les instances, la deuxième variante de l'approche est moins performante que la première. Les temps de calculs sont plus importants et moins d'instances sont résolues. Le nombre d'itérations est important pour la résolution de toutes les instances. On pourrait donc observer que la première variante, comme dans le cas des d-transversaux optimaux de stables, est plus efficace que la deuxième si la résolution nécessite beaucoup d'itérations, mais il faudrait pouvoir effectuer des comparaisons sur des instances où la résolution peut s'effectuer en peu d'itérations.

Notons qu'il n'existe aucune arête forcée dans les instances générées. On ne peut donc pas observer de lien entre l'efficacité de la deuxième formulation et le nombre d'arêtes forcées comme dans les d-transversaux optimaux de stables. Des tests supplémentaires sur des instances avec un nombre fixe d'arêtes forcées seraient nécessaires.

5.5 Perspectives

Dans ce chapitre, nous avons étudié quelques cas particuliers de d-transversaux pondérés dans les graphes bipartis. Puis nous avons proposé une approche pour déterminer des d-transversaux de problèmes modélisé par un programme en variables binaires par la génération de contraintes. Nous avons implémenté deux variantes de cette approche que nous avons testées pour déterminer des d-transversaux optimaux de stables et des d-transversaux optimaux de couplages. La première variante semble plus performante que la deuxième dans la plupart des cas. Des tests supplémentaires sur des instances dont le nombre d'éléments forcés est fixe pourraient être envisagés pour le confirmer. Si la majorité des instances ont été résolues pour les d-transversaux optimaux des stables, ce n'est pas le cas pour les d-transversaux optimaux de couplages où très peu d'instances ont été résolues. L'approche de résolution montre donc rapidement des limites.

Il reste plusieurs pistes d'amélioration de l'approche de résolution. Lors de la génération de contraintes, plusieurs séquences de contraintes peuvent aboutir à un d-transversal. Il serait donc intéressant de générer le minimum de contraintes et ainsi de réduire le nombre d'itérations de l'algorithme. Une autre piste pourrait être de résoudre le programme biniveau en améliorant l'algorithme proposé dans le chapitre 2.

Conclusion

Dans cette thèse, nous avons étudiés trois catégories de problèmes : les d-extensibles, les d-bloqueurs et les d-transversaux.

Dans un premier temps, après avoir rappelé des notions générales de théorie des graphes et une caractérisation des stables optimaux dans les graphes bipartis, nous avons donné les définitions des problèmes étudiés. Puis nous avons évoqué des généralités de la programmation mathématique biniveau.

Le chapitre 3, portait sur l'étude des d-extensibles optimaux de stables dans les graphes bipartis. Nous avons tout d'abord donné des propriétés structurelles des stables extensibles par rapport à un ensemble de sommets et des d-extensibles de stables dans les graphes bipartis. Puis nous avons proposé une modélisation de la recherche d'un d-extensible de cardinal maximal de stables par un programme mathématique à trois niveaux en variables binaires. Nous avons ensuite déterminé une borne inférieure du cardinal maximal d'un d-extensible maximum de stables. Nous avons également montré que déterminer un 1-extensible optimal dans un graphe biparti dont tous les sommets sont libres dont le graphe des composantes est une arborescence et que déterminer un d-extensible optimal dans une grille, un cycle peut se faire en temps polynomial. En ce qui concerne les graphes bipartis qui contiennent des sommets forcés, nous avons montré que pour $d > \frac{\Phi(G)}{2}$, déterminer un d-extensible optimal pouvait se faire en temps polynomial. Nous nous sommes ensuite intéressés aux d-extensibles optimaux de stables lorsque le graphe est un arbre dont tous les sommets sont libres pour lesquels avons démontré d'autres propriétés structurelles et déterminé une borne du cardinal maximal. Nous avons enfin démontré que déterminer un d-extensible optimal lorsque le graphe est un arbre dont le graphe des composante est une arborescence, une chenille, ou un homard pouvait être fait en temps polynomial. Il reste

encore de nombreuses pistes à explorer. En effet, si nous avons démontré que déterminer un 1-extensible optimal lorsque le graphe est un arbre se fait en temps polynomial, le cas des d-extensibles optimaux pour $d > 1$ reste à traiter. Pour ce qui est des graphes bipartis arbitraires, déterminer s'il existe un d-extensible de cardinal $2d$ ou un d-extensible de cardinal strictement supérieur à $2d$ est encore un problème dont le statut de la complexité est ouvert.

Dans le chapitre 4, nous avons étudié les d-bloqueurs de coût minimal de stables de cardinal maximal lorsque le graphe est un arbre. Après avoir caractérisé les d-bloqueurs, nous avons démontré que déterminer un d-bloqueur de coût minimal pouvait être réalisé en temps polynomial pour les arbres qui ne contiennent pas de sommets exclus. Mais la recherche de d-bloqueurs de coût minimal de stables lorsque le graphe est un arbre qui contient des sommets exclus reste un problème de complexité ouverte.

Dans le chapitre 5, nous nous sommes intéressés aux d-transversaux de cardinal maximal de problèmes modélisés par des programmes mathématiques en variables binaires. Nous avons étudié quelques cas particuliers des d-transversaux pondérés de stables optimaux dans les graphes bipartis dont tous les sommets sont libres. Le statu de la complexité du problème reste ouvert dans le cas d'un graphe biparti quelconque. De futurs travaux pourront s'intéresser à d'autres classes de graphes particulières comme les arbres. Puis, nous avons considéré la modélisation de la recherche des d-transversaux par un programme mathématique biniveau. Pour le résoudre, dans le chapitre 2, nous avons modifié un algorithme de la littérature et constaté que cette approche de résolution n'est pas performante. Nous avons ensuite proposé une approche pour déterminer des d-transversaux optimaux de problèmes modélisés par des programmes en variables binaires, par une méthode de génération de contraintes. Deux variantes de cette approche ont été proposées. Leur efficacité a été comparée sur des instances générées aléatoirement en cherchant des d-transversaux optimaux de stables de cardinal maximal et des d-transversaux optimaux de couplages de cardinal maximal. L'approche de génération de contraintes pour déterminer des d-transversaux de problèmes modélisés par les programmes mathématiques en variables binaires montre ses limites, particulièrement dans le cas des d-transversaux optimaux de couplages. Des perspectives d'amélioration comme une résolution plus efficace

du problème maître, ou une résolution du programme biniveau par une autre méthode que celle présentée, restent à explorer. Il serait aussi d'intéressant d'appliquer cette approche pour la recherche de d-transversaux pondérés optimaux de problèmes faisant intervenir des variables binaires.

En conclusion, dans cette thèse, nous avons démontré quelques propriétés structurelles et étudié des cas particuliers pour des problèmes de d-extensibles de cardinal maximal, de d-bloqueurs de coût minimal et de d-transversaux pondérés de cardinal maximal. Toutefois, il reste encore des problèmes pour lesquels le statut de la complexité est ouvert. Nous avons également proposé et testé une approche de génération de contraintes pour déterminer des d-transversaux de problèmes modélisés par des programmes mathématiques en variables binaires. Cette approche fonctionne jusqu'à une certaine taille d'instance, qui dépend des problèmes. A nouveau, il reste des pistes d'améliorations qui pourraient être étudiés dans de futurs travaux.

Bibliographie

[1] Eitaro Aiyoshi and Kiyotaka Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *Automatic Control, IEEE Transactions on*, 29(12) :1111–1114, 1984.

[2] İbrahim Akgün, Barbaros Ç Tansel, and R Kevin Wood. The multi-terminal maximum-flow network-interdiction problem. *European Journal of Operational Research*, 211(2) :241–251, 2011.

[3] Faiz A Al-Khayyal, Reiner Horst, and Panos M Pardalos. Global optimization of concave functions subject to quadratic constraints : an application in nonlinear bilevel programming. *Annals of Operations Research*, 34(1) :125–147, 1992.

[4] Robert EL Aldred and Michael D Plummer. On matching extensions with prescribed and proscribed edge sets ii. *Discrete mathematics*, 197 :29–40, 1999.

[5] Charles Audet, Gilles Savard, and Walid Zghal. New branch-and-cut algorithm for bilevel linear programming. *Journal of Optimization Theory and Applications*, 134(2) :353–370, 2007.

[6] Michael O Ball, Bruce L Golden, and Rakesh V Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2) :73–76, 1989.

[7] Amotz Bar-Noy, Samir Khuller, and Baruch Schieber. The complexity of finding most vital arcs and nodes. Technical report, University of Maryland, 1998.

[8] Jonathan F Bard. An investigation of the linear three level programming problem. *Systems, Man and Cybernetics, IEEE Transactions on*, (5) :711–717, 1984.

[9] Jonathan F Bard and James T Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*,

11(2) :281–292, 1990.

[10] Jonathan F Bard and James T Moore. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3) :419–435, 1992.

[11] C. Bazgan, S. Toubaline, and Z. Tuza. Complexity of the most vital nodes for independant set on tree structures. *Lecture Notes in Computer Science*, 6460 :154–166, 2010.

[12] Cristina Bazgan, Cédric Bentz, Christophe Picouleau, and Bernard Ries. Blockers for the stability number and the chromatic number. *Graphs and Combinatorics*, 31(1) :73–90, 2015.

[13] Cristina Bazgan, Sonia Toubaline, and Zsolt Tuza. The most vital nodes with respect to independent set and vertex cover. *Discrete Applied Mathematics*, 159(17) :1933–1946, 2011.

[14] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Complexity of determining the most vital elements for the p-median and p-center location problems. *Journal of Combinatorial Optimization*, 25(2) :191–207, 2013.

[15] B.Colson, P.Marcotte, and G.Savard. Bilevel programming : A survey. *4or*, 3 :87–107, 2005.

[16] Omar Ben-Ayed and Charles E Blair. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3) :556–560, 1990.

[17] Cédric Bentz, M-C Costa, Christophe Picouleau, Bernard Ries, and Dominique De Werra. d-transversals of stable sets and vertex covers in weighted bipartite graphs. *Journal of Discrete Algorithms*, 17 :95–102, 2012.

[18] Wayne F Bialas and Mark H Karwan. Two-level linear programming. *Management science*, 30(8) :1004–1020, 1984.

[19] Pierre Le Bodic. *Variantes non standards de problèmes d'optimisation combinatoire*. PhD thesis, 2012.

[20] Endre Boros, Martin C Golumbic, and Vadim E Levit. On the number of vertices belonging to all maximum stable sets of a graph. *Discrete Applied Mathematics*, 124(1) :17–25, 2002.

[21] Luce Brotcorne, Saïd Hanafi, and Raïd Mansi. One-level reformulation of the bilevel knapsack problem using dynamic programming. *Discrete Optimization*, 10(1) :1–10, 2013.

[22] Luce Brotcorne, Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4) :345–358, 2001.

[23] Luce Brotcorne, Patrice Marcotte, and Gilles Savard. Bilevel programming : The montreal school. *INFOR : Information Systems and Operational Research*, 46(4) :231–246, 2008.

[24] Gerald G Brown, W Matthew Carlyle, Robert C Harney, Eric M Skroch, and R Kevin Wood. Interdicting a nuclear-weapons project. *Operations Research*, 57(4) :866–877, 2009.

[25] Manoel Campelo, Simone Dantas, and Susana Scheimberg. A note on a penalty function approach for solving bilevel linear programs. *Journal of global optimization*, 16(3) :245–255, 2000.

[26] Wilfred Candler and Robert Townsley. A linear two-level programming problem. *Computers & Operations Research*, 9(1) :59–76, 1982.

[27] Sebastian M Cioabă and Weiqiang Li. The extendability of matchings in strongly regular graphs. *the electronic journal of combinatorics*, 21(2) :P2–34, 2014.

[28] Benoît Colson, Patrice Marcotte, and Gilles Savard. A trust-region method for nonlinear bilevel programming : algorithm and computational experience. *Computational Optimization and Applications*, 30(3) :211–227, 2005.

[29] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1) :235–256, 2007.

[30] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*, volume 1. Siam, 2000.

[31] M.-C. Costa, D. de Werra, C. Picouleau, B. Ries, and C. Bentz. *Progress in Combinatorial Optimization*, chapter Weighted Transversals and Blockers for Some Optimization Problems in Graphs, pages 203–222. Wiley, 2011.

[32] M.-C. Costa, C. Picouleau, and D. de Werra. Minimum d-blockers and d-transversals in graphs. *Journal of Combinatorial Optimization*, 22(4) :857–872, 2011.

[33] M.-C. Costa, C. Picouleau, and D. de Werra. Minimal graphs for matching extensions. *Journal of Combinatorial Optimization*, 2015.

[34] Jean-Philippe Côté, Patrice Marcotte, and Gilles Savard. A bilevel modelling approach to pricing and fare optimisation in the airline industry. *Journal of Revenue and Pricing Management*, 2(1) :23–36, 2003.

[35] Stephan Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

[36] Stephan Dempe and Jonathan F Bard. Bundle trust-region algorithm for bilinear bilevel programming. *Journal of Optimization Theory and Applications*, 110(2) :265–288, 2001.

[37] Stephan Dempe, Vyacheslav Kalashnikov, Gerardo A Pérez-Valdés, and Nataliya Kalashnykova. Bilevel programming problems, 2015.

[38] Scott DeNegre. *Interdiction and discrete bilevel linear programming*. PhD thesis, 2011.

[39] ST DeNegre and Ted K Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. pages 65–78, 2009.

[40] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3) :449–467, 1965.

[41] E Erkut, C ReVelle, and Y Ülküsal. Integer-friendly formulations for the r-separation problem. *European Journal of Operational Research*, 92(2) :342–351, 1996.

[42] Robert Faure, Bernard Lemaire, and Christophe Picouleau. *Précis de recherche opérationnelle-7e éd. : Méthodes et exercices d'application*. Dunod, 2014.

[43] José Fortuny-Amat and Bruce McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the operational Research Society*, pages 783–792, 1981.

[44] Greg N Frederickson and Roberto Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2) :244–266, 1999.

[45] D Ray Fulkerson and Gary C Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13(1) :116–118, 1977.

[46] Michael R Garey and David S Johnson. *Computers and intractability : a guide to NP-completeness*. WH Freeman New York, 1979.

[47] Bruce Golden. A problem in network interdiction. *Naval Research Logistics Quarterly*, 25(4) :711–713, 1978.

[48] Jialin Han, Jie Lu, Yaoguang Hu, and Guangquan Zhang. Tri-level decision-making with multiple followers : Model, algorithm and case study. *Information Sciences*, 311 :182–204, 2015.

[49] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5) :1194–1217, 1992.

[50] Donald W Hearn and Motakuri V Ramana. *Solving congestion toll pricing models*. Springer, 1998.

[51] Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1) :73–88, 1992.

[52] Eitan Israeli and R Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2) :97–111, 2002.

[53] B Jaumard, G Savard, and J Xiong. An exact algorithm for convex quadratic bilevel programming. *Draft paper, Ecole Polytechnique de Montréal*, 2000.

[54] Robert G Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2) :146–164, 1985.

[55] Bahar Y Kara and Vedat Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, 38(2) :188–196, 2004.

[56] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems : total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2) :204–233, 2008.

[57] Matthias Köppe, Maurice Queyranne, and Christopher Thomas Ryan. Parametric integer programming algorithm for bilevel mixed integer programs. *Journal of optimization theory and applications*, 146(1) :137–150, 2010.

[58] Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management science*, 44(12-part-1) :1608–1622, 1998.

[59] Chuan-Min Lee. Variations of maximum-clique transversal sets on graphs. *Annals of Operations Research*, 181(1) :21–66, 2010.

[60] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4) :538–548, 1983.

[61] Kenneth Levenberg. A method for the solution of certain non–linear problems in least squares. 1944.

[62] Guoshan Liu, Jiye Han, and Shouyang Wang. A trust region algorithm for bilevel programing problems. *Chinese science bulletin*, 43(10) :820–824, 1998.

[63] Pierre Loridan and Jacqueline Morgan. Weak via strong stackelberg problem : New results. *Journal of global Optimization*, 8(3) :263–287, 1996.

[64] Foad Mahdavi Pajouh, Vladimir Boginski, and Eduardo L Pasiliao. Minimum vertex blocker clique problem. *Networks*, 64(1) :48–64, 2014.

[65] Patrice Marcotte. Network design problem with congestion effects : A case of bilevel programming. *Mathematical Programming*, 34(2) :142–162, 1986.

[66] James T Moore and Jonathan F Bard. The mixed integer linear bilevel programming problem. *Operations research*, 38(5) :911–921, 1990.

[67] Michael D Plummer. Extending matchings in graphs : a survey. *Discrete Mathematics*, 127(1) :277–292, 1994.

[68] Bernard Ries, Cédric Bentz, Christophe Picouleau, Dominique de Werra, M-C Costa, and Rico Zenklusen. Blockers and transversals in some subclasses of bipartite graphs : When caterpillars are dancing on a grid. *Discrete Mathematics*, 310(1) :132–146, 2010.

[69] Johannes O Royset and R Kevin Wood. Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing*, 19(2) :175–184, 2007.

[70] Gilles Savard and Jacques Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15(5) :265–272, 1994.

[71] Alexander Schrijver. *Combinatorial optimization : polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[72] S.Dempe. *Bilevel programming : A survey.* Dekan der Fak. für Mathematik und Informatik, 2003.

[73] A.G.Mershaand S.Dempe. Linear bilevel programming with upper level constraints depending on the lower level solution. *Applied mathematics and computation*, 180(1) :247–254, 2006.

[74] Kiyotaka Shimizu and Eitaro Aiyoshi. A new computational method for stackelberg and min-max problems by use of a penalty method. *Automatic Control, IEEE Transactions on*, 26(2) :460–466, 1981.

[75] Luis Vicente, Gilles Savard, and J Judice. Discrete linear bilevel programming problem. *Journal of optimization theory and applications*, 89(3) :597–614, 1996.

[76] Luis Vicente, Gilles Savard, and Joaquim Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2) :379–399, 1994.

[77] DJ White. Penalty function approach to linear trilevel programming. *Journal of Optimization Theory and Applications*, 93(1) :183–197, 1997.

[78] Douglas J White and G Anandalingam. A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4) :397–419, 1993.

[79] Richard Wollmer. Removing arcs from a network. *Operations Research*, 12(6) :934–940, 1964.

[80] R Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2) :1–18, 1993.

[81] Rico Zenklusen, Bernard Ries, Christophe Picouleau, Dominique De Werra, M-C Costa, and Cédric Bentz. Blockers and transversals. *Discrete Mathematics*, 309(13) :4306–4314, 2009.

le cnam

Grégoire COTTÉ

d-extensibles, d-bloqueurs et d-transversaux de problèmes d'optimisation combinatoire

le cnam

Résumé :

Dans cette thèse, nous étudions trois catégories de problèmes : les d-extensibles, les d-bloqueurs et les d-transversaux.
Les d-extensibles de stables optimaux sont des ensembles de sommets d'un graphe G tels que tout stable de cardinal d du sous-graphe induit par un d-extensible peut être étendu à un stable optimal de G à l'aide de sommets qui n'appartiennent pas au d-extensible. Nous étudions les d-extensibles de cardinal maximal de stables dans les graphes bipartis. Nous démontrons quelques propriétés structurelles puis nous déterminons une borne inférieure du cardinal maximal d'un d-extensible. Nous étudions quelques classes de graphes dans lesquelles déterminer un d-extensible optimal de stables est un problème polynomial. Nous nous intéressons ensuite aux d-extensibles de stables dans les arbres. Nous prouvons plusieurs propriétés structurelles, déterminons une autre borne inférieure du cardinal maximal d'un d-extensible et étudions quelques classes d'arbres dans lesquelles déterminer un d-extensible optimal de stables est un problème polynomial.
Les d-bloqueurs de stables sont des ensembles de sommets d'un graphe G tels que, si on retire les sommets d'un d-bloqueur, le cardinal maximal d'un stable du graphe induit par les sommets restants est inférieur d'au moins d au cardinal maximal d'un stable du graphe initial. Nous nous intéressons ici aux d-bloqueurs de coût minimal de stables dans les arbres. Après avoir prouvé une caractérisation des d-bloqueurs de stables dans les arbres, nous démontrons que déterminer un d-bloqueur de coût minimal de stable est un problème polynomial dans une classe d'arbres particulière.
Soit Π un problème d'optimisation sur un ensemble d'éléments fini. Un d-transversal de Π est un ensembles d'éléments tel que l'intersection entre le d-transversal et toute solution optimale au problème Π est de cardinal supérieur égal à d. Nous proposons ici une approche de génération de contraintes pour déterminer des d-transversaux de cardinal maximal de problèmes modélisés par des programmes mathématiques en variables binaires. Nous étudions deux variantes de cette approche que nous testons sur des instances de graphes générés aléatoirement pour déterminer des d-transversaux de stables optimaux et des d-transversaux de couplages optimaux

Mots clés :

d-extensibles, d-bloqueurs, d-transversaux, graphe, biparti, stable maximum

Abstract :

In this thesis, we study three types of problems : the d-extensibles sets, the d-blockers and the d-transversals.
In a graph G, a d-extensible set of maximum independent sets is a subset of vertices of G such that every stable set of cardinality d in the subgraph restricted to the d-extensible set can be extented to a maximum stable set of G using only vertices that do not belong to the d-extensible set. We study d-extensible sets of mxaimum cardinality of stable sets in bipartite graphs. We show some structural properties and we determine a lower bound of the maximum cardinality of a d-extensible set. We consider some classes of graph where finding an optimum d-extensible set can be done in polynomial time. Then, we study the d-extensibles sets of stable sets in trees. We prove some properties on the structures of the d-extensibles sets and we determine another lower bound of the maximum cardinality of a d-extensible set. Finaly, we study somme classes of tree where a d-extensible sets of maximum cardinality can be done in polynomial time.
In a graph G, a d-blocker is a subset of vertices such that, if removed, a maximum stable set of the resulting subgraph is of cardinality at most the cardinality of a maximum stable set of G minus d. We study d-blocker of minimal cost of stable sets in tree.We prove a caracterisation of d-blockers in tree and we study a particular classe of trees where computing a d-blocker of minimal cost of stable sets can be done in polynomial time.
Let Π be an optimisation problem on a finite set of elements. A d-transversal of Π is a subset of elements such that the intersection between the d-transversal and every optimal solution of Π contains at lest d elements. We propose an approach to compute d-transversal of any optimisation problem modelised by mathematical program with binary variables. We use a contraints generation approach. We compare two variations of this approach on randomly generated graph by computing d-transversals of stables sets and d-transversals of matching.

Keywords :

d-extensibles, d-blockers, d-transversal, graph, bipartite, maximum stable set