Comparaison de réseaux biologiques

Hafedh Mohamed Babou

UNIVERSITÉ DE NANTES
FACULTÉ DES SCIENCES ET DES TECHNIQUES

_____

**SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET MATHÉMATIQUES**

Année 2012

Comparaison de réseaux biologiques

_____

THÈSE DE DOCTORAT
Discipline : INFORMATIQUE
Spécialité : INFORMATIQUE

*Présentée
et soutenue publiquement par*

Hafedh MOHAMED BABOU

Le 06 Novembre 2012, devant le jury ci-dessous

Président	Thierry LECROQ	Prof. des universités - Université de Rouen
Rapporteurs	Dominique BARTH	Prof. des universités - Université de Versailles
	Ioan TODINCA	Prof. des universités - Université d'Orléans
Examinateurs	Thierry LECROQ	Prof. des universités - Université de Rouen
	Stéphane VIALETTE	Directeur de recherche - CNRS, Université Paris-Est

Directeur de thèse :	Irena RUSU	Prof. des universités - Université de Nantes
Co-encadrant de thèse :	Guillaume FERTIN	Prof. des universités - Université de Nantes

Remerciements

Je voudrais tout d'abord exprimer mes plus profonds remerciements à Irena et Guillaume qui m'ont encadré pendant ces trois années de thèse. Je salue leur disponibilité, leur rigueur, leur intelligence et leur patience.

Merci à mes rapporteurs Dominique Barth et Ioan Todinca d'avoir accepté de lire et évaluer ce manuscrit.

Merci à Thierry Lecroq et Stéphane Vialette qui m'ont honoré par leur participation au jury de ma thèse.

Je tiens aussi à remercier tous les membres de mon équipe ComBi. Je remercie mes amis au LINA : Mohamed, Walid, Laurent, Marie, Ramdan, Toufik , Thomas V, Thomas C, Khaled …

Je remercie mes parents pour leur soutient permanent. Enfin, un grand merci à ma petite famille Aziza et Ayoub.

Table des matières

Liste des figures

Introduction

Un réseau biologique est une représentation abstraite d'un système biologique. L'objectif principal de la modélisation par des réseaux est d'étudier les relations entre les composants de ces réseaux pour analyser ou prédire des fonctions biologiques. Les études récentes montrent qu'un système biologique ne peut être compris en se basant uniquement sur la fonction de chacun de ses composants [BW07, TGH09, Han08, OFGK00], beaucoup des fonctions étant réalisées par un groupe de composants (par exemple les complexes des protéines, les opérons, etc.)

En bio-informatique, les réseaux biologiques sont souvent modélisés par des graphes dont les sommets sont les composants biologiques et les arêtes représentent leurs interactions. En fonction de la nature de ces réseaux, leurs graphes correspondants peuvent être orientés (par exemple réseaux métaboliques), non orientés (par exemple réseaux d'interaction des protéines) ou mixtes (par exemple réseaux d'interaction protéine-protéine et protéine-ADN).

La comparaison de réseaux biologiques est actuellement l'une des approches les plus prometteuses pour aider à la compréhension du fonctionnement des organismes vivants. Elle apparaît comme la suite attendue de la comparaison de séquences biologiques [OFGK00, KSK+03, SI06], dont l'étude a permis le développement de concepts nouveaux et une réelle avancée scientifique, mais qui ne représente en réalité qu'un aspect (l'aspect dit génomique) des informations manipulées par les biologistes [GK00].

L'objectif de cette thèse est de proposer un cadre formel, algorithmique et expérimental pour comparer efficacement (algorithmiquement et biologiquement) des réseaux biologiques.

Nous avons réalisé une étude bibliographique qui nous a permis de constater les aboutissements mais aussi les insuffisances des travaux actuels. Les principaux travaux de recherche dans ce domaine s'inspirent de l'alignement des séquences pour faire l'alignement de deux réseaux (c'est-à-dire la comparaison qui met en évidence les parties communes) et progresser vers l'alignement d'un nombre arbitraire de réseaux [KSK+03, SIK+04, SSK+05, FAC+08]. Cependant, l'alignement de deux réseaux nécessite que les deux réseaux soient de même type, et donc représentés par des graphes de même type (ce que nous appellerons des *réseaux homogènes*). D'autres travaux ont essayé de contourner cette difficulté [OFGK00, BML+05, DBVS09] en comparant des réseaux de types différents, mais ces comparaisons portent uniquement sur les réseaux non-orientés qui sont aussi les moins riches en informations.

Dans cette thèse, nous avons choisi de nous intéresser plus particulièrement à la comparaison de deux réseaux de types différents (par exemple

un réseau métabolique et un réseau d'interaction des protéines) et modélisés par des graphes de types différents (ce que nous appellerons des *réseaux hétérogènes*). Plus précisément, les deux réseaux sont modélisés respectivement par un graphe orienté D (appelé graphe *principal*) et un graphe non orienté G (appelé graphe *guide*), et seront dotés d'une fonction f établissant la correspondance entre les sommets de D et ceux de G. Notre objectif est d'extraire automatiquement des chemins dans D qui induisent (par la fonction f) des sous-graphes connexes dans G. Nous appellerons ce problème Skew SubGraph Mining (abrégé SkewGraM). Un exemple d'application de SkewGraM est la recherche de chaînes de réactions successives (des chemins) dans un réseau métabolique d'un organisme donné, qui sont catalysées par des protéines adjacentes dans le réseau d'interaction protéine-protéine du même organisme. Cette approche innovante nous permettra donc de confronter des vues différentes d'un même phénomène biologique, et donc d'appuyer notre analyse des systèmes biologiques sur des observations complémentaires et riches d'informations.

Ce manuscrit est composé de quatre chapitres. Le premier chapitre est un chapitre introductif dans lequel nous présentons des concepts de base (biologiques et informatiques) qui sont nécessaires à la compréhension de ce manuscrit.

Dans le deuxième chapitre nous formulons le problème SkewGraM et nous étudions d'abord la complexité de sa variante (appelée One-to-One SkewGraM) dans laquelle (i) la fonction de correspondance f est bijective, et (ii) le graphe D et un DAG. Nous identifions les instances faciles et les instances difficiles pour One-to-One SkewGraM. Pour résoudre les instances difficiles de ce problème, nous présentons deux algorithmes : une heuristique et un algorithme exact basé sur la méthode branch-and-bound. Ensuite, nous proposons une méthode permettant de créer une instance de One-to-One SkewGraM à partir de toute instance du problème SkewGraM dans laquelle D est un DAG et f est une fonction arbitraire. Enfin, pour montrer l'efficacité de notre méthode, nous l'appliquons sur des données simulées et sur des données biologiques.

Pour traiter le problème SkewGraM quand le graphe D contient des cycles, nous présentons dans le chapitre 3, une méthode de prétraitement qui consiste à décomposer le graphe D en DAGs induisant des sous-graphes connexes dans G. Nous considérons deux types de décomposition : un partitionnement de D en DAGs et une couverture de D par des DAGs. Nous étudions la complexité des problèmes issus de notre décomposition en proposant des algorithmes polynomiaux pour certains cas et des résultats de difficultés pour d'autres cas.

Le dernier chapitre (chapitre 4) est consacré à l'étude des réseaux physiques. Ces réseaux sont les supports physiques des signaux transmis entre la cellule et son environnement extérieur. Nous étudions dans ce chapitre les méthodes utilisées pour inférer les sens de circulation de ces signaux. Les méthodes en question sont basées sur des problèmes d'orientation "unidirectionnelle" des graphes (mixtes) : toute arête est remplacée par un seul arc. Nous complétons l'étude de complexité de certains de ces problèmes, et nous proposons une nouvelle approche dans laquelle nous

autorisons certaines arêtes à être doublement orientées (c'est-à-dire remplacées par deux arcs de directions opposées). Ensuite, nous étudions la complexité du problème résultant de notre approche, en identifiant ses cas faciles et ses cas difficiles.

Enfin, nous achevons ce manuscrit par un rappel des résultats obtenus et une présentation de nos travaux en cours de réalisation.

Concepts de base

1

Sommaire

Ce chapitre introductif est composé de trois sections. Dans la première section, nous présentons les réseaux biologiques, leurs modélisations et les méthodes expérimentales utilisées pour les construire. Nous rappelons les notions de base en théorie des graphes dans la deuxième section. La dernière section est consacrée à la présentation des notions nécessaires en théorie de complexité qui seront utilisées tout au long de ce manuscrit.

1.1 Modélisation de réseaux biologiques

Les réseaux biologiques représentent chacun une vue partielle de l'activité moléculaire à l'intérieur de la cellule. Dans cette section nous allons décrire les principaux réseaux couramment étudiés en biologie. Pour chaque type de réseau nous présentons les différentes modélisations proposées en littérature.

1.1.1 Réseaux d'interaction protéine-protéine

Les réseaux d'interaction protéine-protéine (ou PPI, pour protein-protein interaction) représentent les interactions physiques entre protéines

au sein d'un organisme (voir la Figure 1.1). Il existe des méthodes expérimentales, dites à *haut débit*, pour mettre en évidence ces interactions. Les méthodes à haut débits principalement utilisées sont d'une part la méthode *double hybride* [FS89, ICO+01], qui est est utilisée pour identifier les interactions entre deux protéines, et d'autre part la *spectrométrie de masse* [TSB+08] pour identifier les interactions entre plusieurs protéines. Il est important de noter que ces méthodes ne sont pas exactes, elles donnent parfois des interactions qui n'existent pas réellement, appelées faux positifs ; elles peuvent aussi manquer des interactions réelles (faux négatifs).

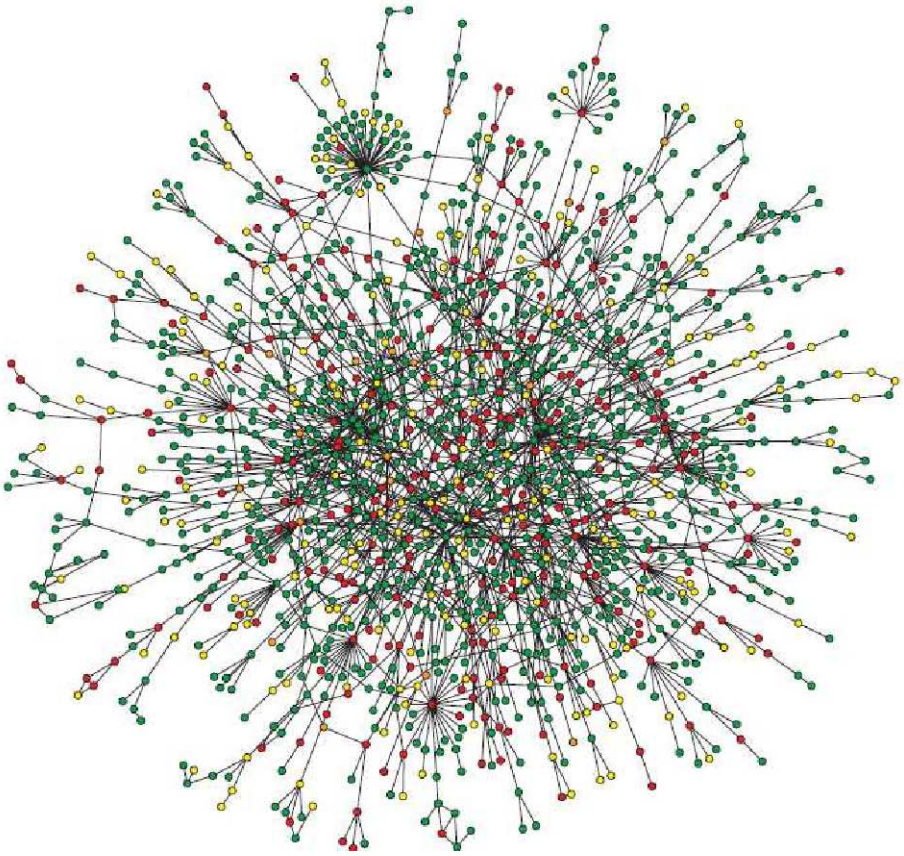


FIGURE 1.1 – *Le réseau PPI chez la levure Saccharomyces cerevisiae. Source : [JMBO01].*

Modélisation. Les interactions entre paires de protéines identifiées par la méthode expérimentale double hybride sont modélisées par un graphe non-orienté dans lequel les sommets sont les protéines et les arêtes représentent leurs interactions [KGS04, Kla09, TGH09, SUS07, LLB+09, SLF06, KBS09, CG08, BML+05, DBVS09, NK07, ZZW+07, KYL+04, HMD06, GH09, ESU08, KGS05, BHMK+09, ZRBV09]. Étant donné que les méthodes expérimentales ne sont pas exactes, il est possible d'ajouter des pondérations sur les arêtes pour indiquer le degré de confiance en l'existence d'une interaction, la force d'interaction ou la probabilité d'avoir une telle interaction [FNS+06, TF09, SSRS06, SXB08b, JHM08, DSG+08]. Les interactions entre plusieurs protéines, comme celles cherchées par la méthode de spectrométrie de masse, sont modélisées par un hypergraphe ;

une hyper-arête représente alors une interaction entre deux protéines ou plus [KGS04].

1.1.2 Réseaux métaboliques

Le métabolisme est généralement défini comme étant le processus à travers lequel les organismes vivants acquièrent et utilisent de l'énergie pour accomplir différentes tâches [LCTS08]. L'acquisition de l'énergie est un processus de synthèse organique appelé *anabolisme*, tandis que l'utilisation d'énergie est un processus de dégradation organique appelé *catabolisme*. L'anabolisme et le catabolisme sont réalisés par une succession des réactions transformant des substrats en produits. Certaines réactions sont spontanées (c'est-à-dire les substrats se transforment en produits sans catalyseur), mais la plupart nécessite la présence d'enzymes pour les catalyser. Une réaction est dite *irréversible* si elle s'effectue dans un seul sens. Par contre, une réaction *réversible* peut s'effectuer dans les deux sens : les substrats deviennent alors produits et inversement.

Étant donnée une réaction R, on note par $sub(R)$ (resp. $prod(R)$) l'ensemble des substrats (resp. produits) de R. L'ensemble des composants de R est noté $comp(R)$ ($comp(R) = sub(R) \cup prod(R)$).

Une réaction réversible R dont les substrats sont A et B et les produits sont C et D est représentée par l'équation suivante.

$$A + B \leftrightarrow C + D \tag{1.1}$$

Une réaction irréversible R, transformant des substrats A et B en produits C et D, est notée par l'équation suivante.

$$A + B \rightarrow C + D \tag{1.2}$$

Si, de plus, la réaction R est catalysée par une enzyme E, l'équation est écrite comme suit.

$$A + B \rightarrow^{E} C + D \tag{1.3}$$

Les réactions du métabolisme forment le réseau métabolique. Ce réseau est décomposé en ensemble des modules fonctionnels (des sous-réseaux) appelés voies métaboliques (en anglais : *metabolic pathways*). Voir la Figure 1.2 qui montre la voie métabolique de dégradation d'éthylbenzène chez la bactérie *Escherichia coli*.

Modélisation. Il existe essentiellement quatre types de représentations [LCTS08] : graphe des composants (substrats et produits), graphe des réactions, graphe des enzymes et graphe biparti (composants *vs* réactions). Ces graphes peuvent être orientés ou non-orientés, l'orientation permettant de différencier les réactions irréversibles de celles qui sont réversibles.

Le graphe des composants [ELJ06, WR07]

Le graphe des composants est un graphe non-orienté dont les sommets correspondent aux composants (c'est-à-dire substrats et produits) et

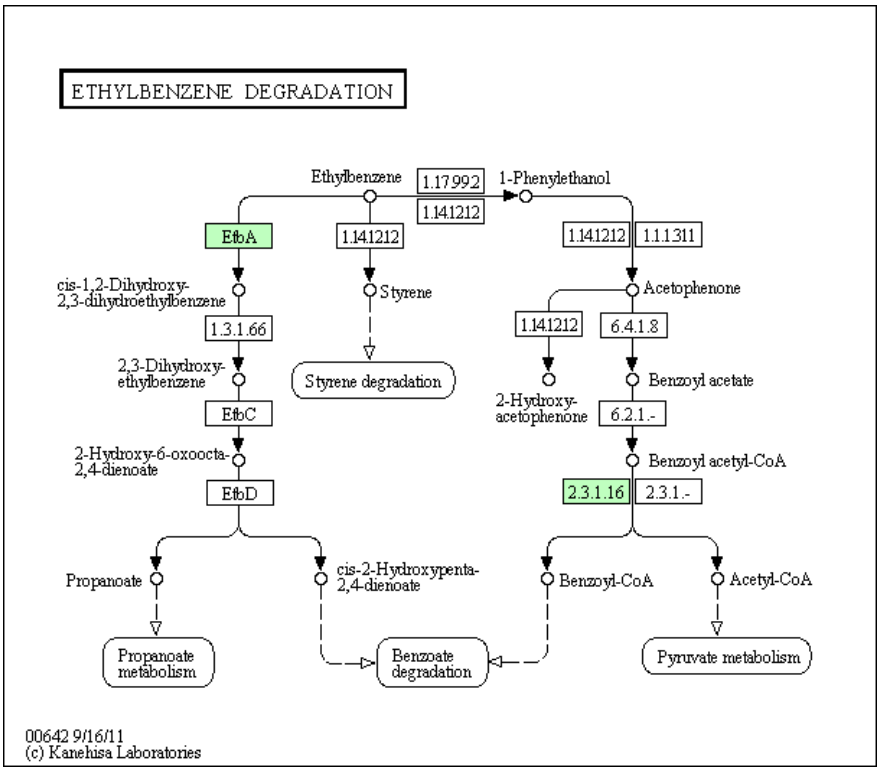


FIGURE 1.2 – *La voie métabolique de dégradation d'éthylbenzène chez la bactérie Escherichia coli. Source : KEGG, pathway eco*00642.

on met une arête entre deux composants A et B s'il existe une réaction dont l'un est substrat et l'autre est produit. Un exemple de graphe des composants est illustré dans la Figure 1.3.

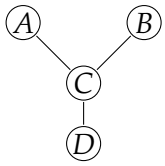


FIGURE 1.3 – *Le graphe des composants associé à chacun des deux ensembles de réactions :* $\{A \leftrightarrow C, B \leftrightarrow C, C \leftrightarrow D\}$ *et* $\{A + B \leftrightarrow C, C \leftrightarrow D\}$.

Le graphe des réactions [Kla09, BML+05, Alb05]

Dans cette représentation, les sommets correspondent aux réactions. Le graphe peut être orienté ou non. Dans le cas d'un graphe non-orienté, il existe une arête entre deux réactions si elles possèdent au moins un composant (substrat ou produit) en commun. Dans le cas d'un graphe orienté, on considère deux réactions R_i et R_j. Il y a alors trois cas :

1. R_i et R_j sont réversibles.
   Si $comp(R_i) \cap comp(R_j) \neq \emptyset$ on ajoute un arc de R_i vers R_j et un arc de R_j vers R_i.
2. R_i est réversible et R_j est irréversible.
   Si $comp(R_i) \cap sub(R_j) \neq \emptyset$ on ajoute un arc de R_i vers R_j.
   Si $comp(R_i) \cap prod(R_j) \neq \emptyset$ on ajoute un arc de R_j vers R_i.

3. R_i et R_j sont irréversibles.
   Si $prod(R_i) \cap sub(R_j) \neq \emptyset$ on ajoute un arc de R_i vers R_j.
   Si $prod(R_j) \cap sub(R_i) \neq \emptyset$ on ajoute un arc de R_j vers R_i.

Un exemple de graphe des réactions est donné dans la Figure 1.4(*a*).

Le graphe des enzymes [KGS04, ZZW+07, BML+05, Alb05, Kla09, PRYLZU05]

Dans cette représentation, les sommets correspondent aux enzymes et il existe une arête entre deux enzymes si elles catalysent deux réactions ayant des composants en commun. Ce graphe est construit à partir du graphe des réactions et il est orienté si le graphe des réactions est orienté. Dans le cas où le graphe des réactions est orienté, on met un arc allant d'une enzyme E_i vers une autre enzyme E_j s'il existe un arc, dans le graphe des réactions, entre deux réactions : R_i catalysée par E_i et R_j catalysée par Ej (voir une illustration dans la Figure 1.4(*b*)). Dans le cas où le graphe des réactions est non-orienté, les arcs décrits ci-dessus deviennent alors des arêtes.

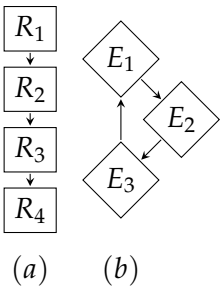


FIGURE 1.4 – *(a) Le graphe des réactions pour l'ensemble de réactions* $\{R_1, R_2, R_3, R_4\}$ *où* $R_1 : A + B \rightarrow^{E_1} C + D$, $R_2 : C \rightarrow^{E_2} F$, $R_3 : F \rightarrow^{E_3} G$. *et* $R_4 : G + K \rightarrow^{E_1} L$. *(b) Le graphe des enzymes correspondant aux réactions* R_1, R_2, R_3, *et* R_4.

Le graphe biparti [BL04, Alb05, BLC+07]

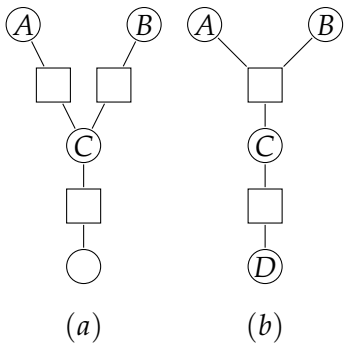


FIGURE 1.5 – *(a) Le graphe biparti associé au réseau* $\{A \leftrightarrow C, B \leftrightarrow C, C \leftrightarrow D\}$. *(b) Le graphe biparti associé au réseau* $\{A + B \leftrightarrow C, C \leftrightarrow D\}$. *Comme le montre la Figure 1.3, les deux réseaux ont le même graphe des composants.*

Les graphes des composants peuvent parfois être ambigus. En effet, si on considère les deux réseaux qui correspondent aux deux ensembles suivants :

Ensemble 1 : $\{A \leftrightarrow C, B \leftrightarrow C, C \leftrightarrow D\}$.

Ensemble 2 : $\{A + B \leftrightarrow C, C \leftrightarrow D\}$.

Dans ce cas, les deux réseaux auront le même graphe des composants, comme le montre la Figure 1.3.

On peut résoudre cette ambiguïté en ajoutant des étiquettes sur les arêtes [WR07], ou en se dotant d'un modèle de graphe plus expressif : un graphe biparti où il y a deux types de sommets, les composants et les réactions. Une arête existe entre une réaction et un composant si ce composant est un produit ou substrat de la réaction (voir une illustration à la Figure 1.5).

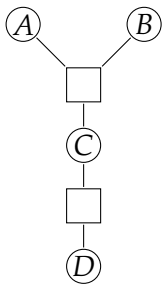


FIGURE 1.6 – *Le graphe biparti associé à chacun des deux ensembles de réactions* $\{A + B \leftrightarrow C, C \leftrightarrow D\}$ *et* $\{A \leftrightarrow B + C, C \leftrightarrow D\}$.

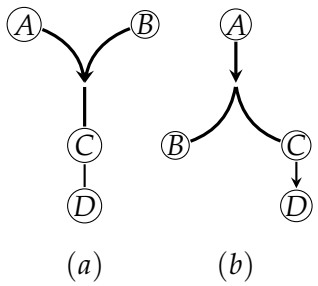


FIGURE 1.7 – *(a) L'hypergraphe orienté associé à l'ensemble de réactions* $\{A + B \rightarrow C, C \rightarrow D\}$. *(b) L'hypergraphe orienté associé à l'ensemble de réactions* $\{A \rightarrow B + C, C \rightarrow D\}$.

Remarque 1.1 *Les graphes bipartis sont aussi ambigus dans certains cas. En effet, les deux réseaux qui correspondent aux deux ensembles de réactions* $\{A + B \rightarrow C, C \rightarrow D\}$ *et* $\{A \rightarrow B + C, C \rightarrow D\}$ *ont le même graphe biparti (voir la Figure 1.6). Pour résoudre cette ambiguïté on peut modéliser les réseaux métaboliques par un hypergraphe orienté dans lequel les sommets correspondent aux composants, et à chaque réaction correspond un hyper arc reliant ses substrats et ses produits [BL04, Alb05, BLC+07]. Voir une illustration dans la Figure 1.7.*

1.1.3 Réseaux de régulation de la transcription des gènes

Les gènes sont transcrits puis traduits pour produire des protéines qui exécutent les fonctions cellulaires. La transcription dès gènes est contrôlée

par des protéines appelées *facteurs de transcription* qui se lient sur une partie (appelée *cis-régulateur*) des gènes cibles et cette liaison induit une stimulation ou inhibition de la transcription. La liaison entre le facteur de transcription et son gène cible est appelée *interaction protéine-ADN*.

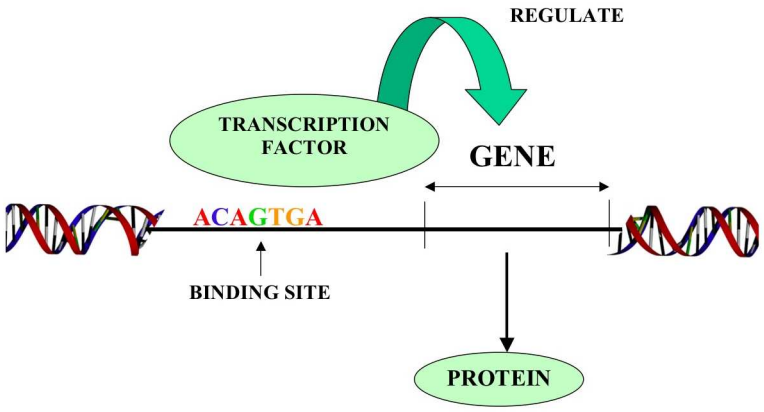


FIGURE 1.8 – *Régulation des gènes. Le facteur de transcription est une protéine qui se lie sur une zone de l'ADN (cis-régulateur) pour réguler la transcription du gène cible [SLW+06].*

Les interactions protéines-ADN peuvent être identifiées, expérimentalement, par la méthode *Chip-Chip* [AGS04] ou la méthode *Microarray* [LDM+97]. L'ensemble des interactions protéine-ADN d'un organisme donné forme ce qu'on appelle le *réseau de régulation de la transcription des gènes* (ou *réseau de régulation des gènes*).

Modélisation. Un réseau de régulation des gènes peut être modélisé par un graphe qui contient deux types de sommets, les facteurs de transcription et les gènes, et deux types d'arcs correspondant respectivement à la stimulation et l'inhibition de la transcription [Alo07a].

Ce modèle est souvent simplifié en ne considérant qu'un seul type d'arcs reliant les facteurs de transcription et leurs gènes cibles, sans préciser s'il s'agit d'une stimulation ou inhibition. Par ailleurs, les facteurs de transcription eux-mêmes sont des produits des gènes, donc ce dernier modèle peut être encore raffiné sous la forme d'un graphe orienté dont les sommets sont les gènes et dans lequel il y a un arc d'un gène g_1 vers un gène g_2, si g_1 produit une protéine qui régule la transcription de g_2 [BL04, FWC+06, Bro08]. Voir une illustration dans la Figure 1.9.

Il existe d'autres modélisations plus expressives qui prennent en considération l'état des gènes cibles (actives ou inactives). Voir [KS08] pour plus de détails.

1.1.4 Réseaux physiques

Le réseau physique [YIJ04] est obtenu à partir du réseau d'interaction protéine-protéine en rajoutant les interactions protéine-ADN.

Modélisation. Un réseau physique est modélisé par un graphe mixte dont les sommets sont les protéines et les arêtes représentent leurs inter-

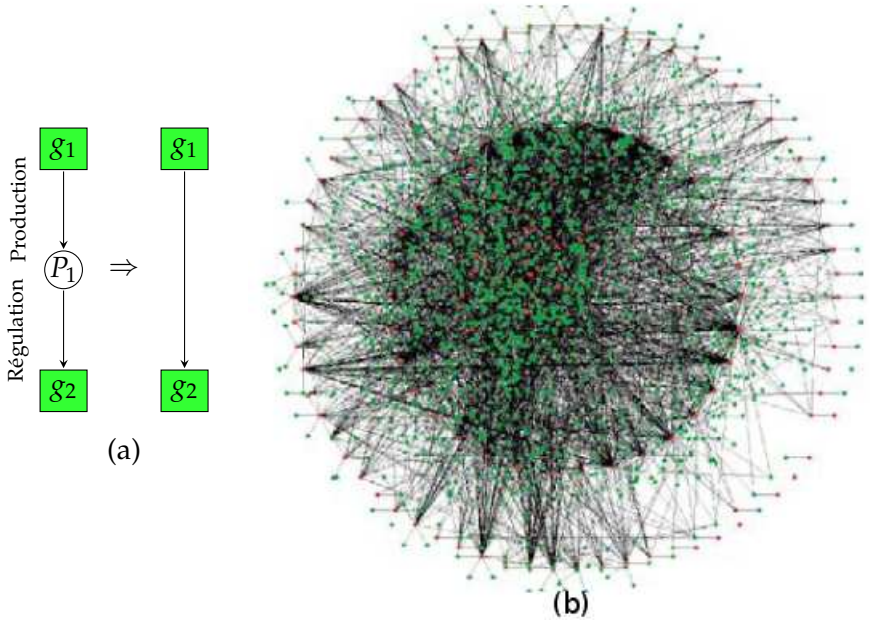


FIGURE 1.9 – *(a) Le gène g_1 produit la protéine P_1 et la protéine P_1 est un facteur de régulation de la transcription du gène g_2. Dans un modèle simplifié, nous mettons dans le graphe représentant le réseau de régulation, un arc de g_1 vers g_2. (b) Le réseau de régulation des gènes (modèle simplifié) chez la bactérie E. coli. Source : [BLA+04].*

actions physiques. On met également un arc d'une protéine P_1 vers une protéine P_2 si P_1 régule la transcription d'un gène produisant la protéine P_2 [YIJ04, MBZS08, SES11]. Nous étudierons les réseaux physiques de manière plus détaillée dans le Chapitre 4.

1.1.5 Réseaux de transduction de signal

La communication entre la cellule et son environnement extérieur est assurée par une cascade de modifications (activations/désactivations) des protéines. Les cascades correspondent à des chemins dans le réseau physique (tel que défini dans la Section 1.1.4) entre une protéine initiale et une protéine terminale. La protéine initiale est celle qui déclenche la transduction du signal suite à l'arrivée d'un événement de l'extérieur, et la modification de la protéine terminale représente la réponse de la cellule suite à cet événement (voir également le Chapitre 4 pour plus de détails).

Modélisation. Les réseaux de transduction de signal sont modélisés par des graphes orientés où les sommets sont des protéines et les arcs représentent les modifications [Alb05, Alo07a].

1.2 Éléments de théorie des graphes

Dans la section précédente, nous avons vu différents types de réseaux biologiques, chacun étant représenté par un graphe. L'étude de ces réseaux nécessite donc la connaissance d'un certain nombre de notions en théorie des graphes.

Les premières notions en théorie des graphes ont été introduites en 1736 quand Euler posa le problème des sept ponts de la ville Königsberg (voir une illustration la Figure 1.10) : est il possible, en partant d'un quartier quelconque de la ville, de traverser chaque pont sans passer deux fois par le même pont, et de revenir au quartier de départ ?

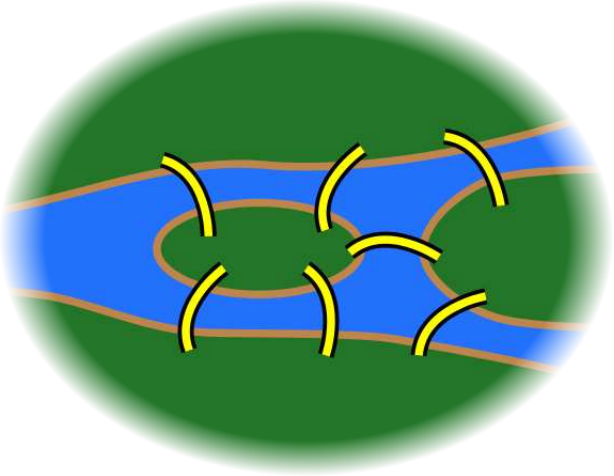


Figure 1.10 – *Illustration des sept ponts de Königsberg. Source : Wikipédia.*

La théorie des graphes s'est développée depuis et aujourd'hui elle a des applications dans diverses disciplines telles que la chimie (modélisation des structures), les sciences sociales (modélisation des relations humaines) et la biologie (modélisation des réseaux biologiques, comme nous l'avons vu précédemment).

Dans cette section, nous allons donner les notions de base, en théorie des graphes, qui seront utilisées dans les prochains chapitres de ce manuscrit. Pour plus de détails sur la théorie des graphes, nous invitons le lecteur à consulter les références suivantes : [Die05], [Ber76] et [BJG01].

1.2.1 Graphes non-orientés

Définitions et premiers exemples

Un graphe non-orienté G est un couple formé de deux ensembles : un ensemble V (noté $V(G)$) dont les éléments sont appelés *sommets* (*vertices*, en anglais) et un ensemble E de paires non-orientées (noté $E(G)$), $E \subseteq V \times V$, dont les éléments sont appelés *arêtes* (*edges*, en anglais). Le graphe non-orienté G est noté $G = (V, E)$. Un exemple d'un graphe non-orienté est illustré dans la Figure 1.11.

On dit que deux sommets $u, v \in V$ sont reliés par une arête dans G si $(u, v) \in E$, et dans ce cas, les sommets u et v sont appelés *extrémités* de l'arête $e = (u, v)$. Une *boucle* est une arête dont les extrémités coïncident.

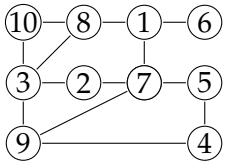


FIGURE 1.11 – *Exemple d'un graphe non-orienté d'ordre 10.*

Un graphe G est dit *simple* s'il ne contient aucune boucle et que pour tout $u, v \in V$, les sommets u et v sont reliés par au plus une arête. En général, quand on parle d'un graphe non-orienté on considère qu'il est simple, sauf si on précise explicitement la non simplicité du graphe.

L'ordre d'un graphe G est le nombre de sommets présents dans G. Deux sommets sont dits *adjacents* s'ils sont reliés par une arête. Le *degré* d'un sommet u, noté $deg(u)$, est le nombre d'arêtes dont ce sommet est une extrémité (une boucle est cependant comptée deux fois). Voir un exemple d'illustration dans la Figure 1.12.

Une *clique* est un ensemble de sommets deux-à-deux adjacents. Le graphe complet K_n est une clique d'ordre n (c'est-à-dire un graphe simple possédant n sommets tous reliés deux à deux par une arête). Le graphe *complémentaire* d'un graphe non-orienté G est un graphe non-orienté, noté $\overline{G}$, ayant les mêmes sommets que G et tel que deux sommets distincts de $\overline{G}$ sont adjacents si et seulement s'ils ne sont pas adjacents dans G.

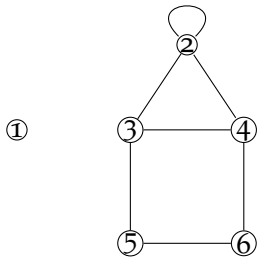


FIGURE 1.12 – *Exemple de calcul de degré.* $deg(1) = 0$*;* $deg(5) = deg(6) = 2$ *et* $deg(3) = deg(4) = 3$*. Le sommet* 2 *est relié à trois arêtes dont une est une boucle, donc* $deg(2) = 4$*.*

Le *voisinage* d'un sommet u (noté $N(u)$) est l'ensemble des sommets adjacents à u. Le degré maximum d'un graphe non-orienté $G = (V, E)$, noté $\Delta(G)$, et le degré minimum de ce graphe, noté $\delta(G)$, sont respectivement le maximum et le minimum des degrés de ses sommets, c'est-à-dire, $\Delta(G) = max_{u \in V} deg(u)$, et $\delta(G) = min_{u \in V} deg(u)$. Le graphe G est dit *k-régulier* si chacun des sommets de G est de degré k. Un graphe 3-régulier est appelé graphe *cubique*.

Un graphe $G' = (V', E')$ est dit *sous-graphe* de $G = (V, E)$ si $V' \subseteq V$ et $E' \subseteq E$. Un graphe *induit* dans G par un ensemble X, $X \subseteq V$, est un sous-graphe de G, noté $G[X]$, dont l'ensemble des sommets est X et l'ensemble des arêtes est $E \cap \{(u, v) \ : \ u \in X \text{ et } v \in X\}$.

Chemins et cycles

Soit $G = (V, E)$ un graphe non-orienté. Un *chemin* dans G est un sous-graphe $P = v_1 v_2 \dots v_m$ tel que pour tout $i \in \{1, 2, \dots, m-1\}$,

$(v_i, v_{i+1}) \in E$. Si $v_1 = v_m$, le chemin P est appelé *cycle*. La *longueur* d'un chemin ou d'un cycle est le nombre d'arêtes qui le composent. Un chemin $P = v_1 v_2 \dots v_m$ est *élémentaire* si tous les sommets v_i, $i \in \{1, \dots, m\}$ sont distincts. Si toutes arêtes (v_i, v_{i+1}), $i \in \{1, 2, \dots, m-1\}$, sont distinctes, alors le chemin est dit *simple*. Un cycle $P = v_1 v_2 \dots v_{m-1} v_1$ est *élémentaire* si tous les sommets v_i, $i \in \{1, \dots, m-1\}$, sont distincts. Le cycle P est dit *simple* si toutes ses arêtes sont distinctes. Voir une illustration dans la Figure 1.13.

La *distance* entre deux sommets u et v (notée $d(u, v)$) est la longueur du plus court chemin qui les relie. Le *diamètre* d'un graphe est la plus grande distance entre deux sommets.

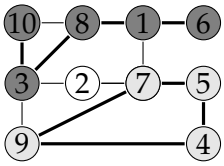


FIGURE 1.13 – *Le chemin* 10 3 8 1 6 *est un chemin élémentaire et simple de longueur* 4. *Le cycle* 9 4 5 7 9 *est un cycle élémentaire. Le chemin* 1 7 9 4 5 7 2 *est chemin simple mais il n'est pas élémentaire : le sommet* 7 *est présent deux fois.*

Connexité

Un graphe non-orienté $G = (V, E)$ est dit *connexe* si pour tous sommets u et v, il existe un chemin reliant u et v. Une *composante connexe* de G est un sous-graphe $G[X]$ induit par un ensemble X, $X \subseteq V$, tels que (i) $G[X]$ est connexe et (ii) $G[X]$ est maximum pour l'inclusion (c'est-à-dire pour tout $u \in V \backslash X$, $G[X \cup \{u\}]$ n'est pas connexe). Un sommet u est appelé *point d'articulation* d'un graphe connexe G si $G[V - \{u\}]$ n'est pas connexe (voir un exemple dans la Figure 1.14).

Propriété 1.1 *Soit $G = (V, E)$ un graphe non-orienté. Tout sommet de G appartient à une seule composante connexe.*

Définition 1.1 (Stable) *Soit G un graphe non-orienté. Un* stable *(ou en anglais* independent set*) dans G est un ensemble de sommets deux à deux non-adjacents.*

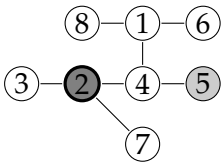


FIGURE 1.14 – *Le sommet* 2 *est un point d'articulation de G. En effet, le graphe G' obtenu à partir de G en retirant le sommet* 2 *et ses arêtes incidentes n'est pas connexe : il a trois composantes connexes induites par les ensembles de sommets* $\{3\}$, $\{7\}$ *et* $\{1, 4, 5, 6, 8\}$.

Classes particulières de graphes non-orientés

Une *forêt* est un graphe non-orienté acyclique. Un *arbre* est une forêt connexe. On distingue deux types de sommets dans un arbre : les *feuilles*

dont le degré est 1 et les sommets internes dont le degré est supérieur à 1. Une *étoile* est un graphe connexe dont tous les sommets, sauf un, sont de degré 1. On peut aussi définir une étoile d'ordre n, $n > 1$, comme étant un arbre avec $n - 1$ feuilles. Une étoile d'ordre n est notée S_n.

Un graphe est dit *planaire* si on peut le représenter dans le plan sans qu'aucune arête n'en croise une autre [Che81]. Le fait de représenter un graphe sous forme planaire le rend plus facile à interpréter, comme illustre l'exemple du problème du cavalier aux échecs (voir la Figure 1.15). Dans ce problème, nous voulons déplacer un cavalier d'échecs en touchant toutes les cases une et une seul fois. L'échiquier est représenté par un graphe dans lequel les sommets correspondent aux cases de l'échiquier et les arêtes figurent les mouvements possibles.

D'un point de vue algorithmique, de nombreux problèmes difficiles (NP-complets) se sont avérés faciles (polynomiaux) pour les graphes planaires. Par exemple, considérons le problème de 4-coloration : étant donné un graphe non-orienté G sans boucles, est-il possible de colorer les sommets de G en utilisant au plus 4 couleurs de sorte que deux sommets adjacents n'aient pas la même couleur ? Ce problème est NP-complet pour un graphe général [GJ79] mais il peut être résolu en temps polynomial quand G est un graphe planaire [AHK77].

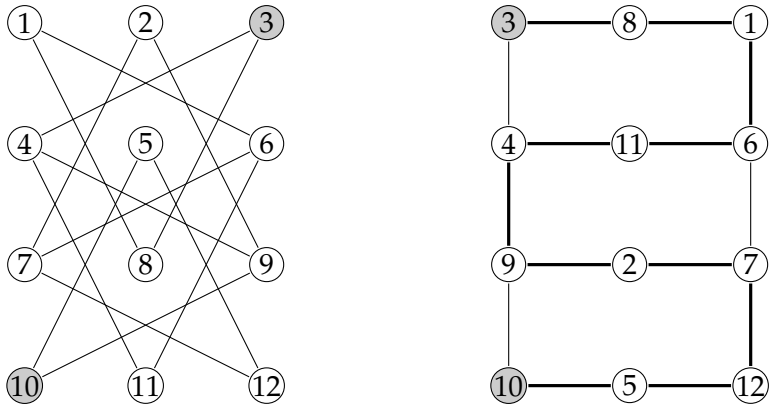


FIGURE 1.15 – *Le graphe de gauche (resp. de droite) est une représentation non planaire (resp. planaire) d'un échiquier de taille* 3×4. *Dans le premier graphe, le problème d'échiquier semble difficile. Par contre, on peut facilement extraire de la représentation planaire une solution (voir les arêtes solides) partant de la case 3 et arrivant à la case 10.*

Un graphe non-orienté $G = (V, E)$ est dit *planaire extérieur* (ou en anglais, *outerplanar*) si on peut le dessiner sur le plan de sorte que les sommets soient sur un cercle et que toutes les arêtes se trouvent à l'intérieur du cercle et ne se croisent qu'à leurs extrémités (voir une illustration dans la Figure 1.16).

Propriété 1.2 *[Felo4] Un graphe G est planaire extérieur si et seulement si le graphe formé en ajoutant à G un nouveau sommet et toutes les arêtes le reliant aux sommets de G est un graphe planaire.*

D'autres caractérisations des graphes planaires extérieurs sont fournies dans [LDM+97].

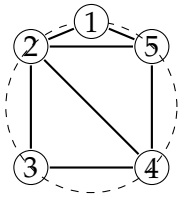


FIGURE 1.16 – *Exemple d'un graphe planaire extérieur. Les sommets du graphe sont dessinés sur un cercle (en pointillé) de sorte que toutes les arêtes soient à l'intérieur du cercle et ne se croisent qu'aux extrémités. Si on ajoute une arête entre les deux sommets 3 et 5, le graphe reste planaire mais par contre il n'est plus planaire extérieur.*

1.2.2 Graphes orientés

Un graphe orienté G est un couple formé de deux ensembles : un ensemble V (noté $V(G)$) dont les éléments sont appelés *sommets* et un ensemble de paires orientées A (noté $A(G)$), $A \subseteq \{uv \ : \ u,v \in V\}$, dont les éléments sont appelés *arcs*. Le graphe orienté G est noté $G = (V, A)$. Soit $uv \in A$ un arc dans G. Le sommet u (resp. v) est appelé *source* (resp. *cible*) de l'arc uv. Un graphe orienté $H = (V_H, A_H)$ est dit *sous-graphe* de G si $V_H \subseteq V$ et $A_H \subseteq A \cap \{uv \ : \ u,v \in V_H\}$. Si $A_H = A \cap \{uv \ : \ u,v \in V_H\}$, le graphe H est dit le sous-graphe de G *induit* par l'ensemble V_H.

Le *degré entrant* d'un sommet u (noté $deg^-(u)$) est le nombre d'arcs dont u est une cible. Le *degré sortant* d'un sommet u (noté $deg^+(u)$) est le nombre d'arcs dont u est une source. Le degré sortant (resp. entrant) maximum d'un graphe orienté $G = (V, A)$, noté $\Delta^+(G)$ (resp. $\Delta^-(G)$), et le degré sortant (resp. entrant) minimum de ce graphe, noté $\delta^+(G)$ (resp. $\delta^-(G)$), sont respectivement le maximum et le minimum des degrés sortant (resp. entrant) de ses sommets, c'est-à-dire, $\Delta(G)^+ = max_{u\in V} deg^+(u)$, $\Delta(G)^- = max_{u\in V} deg^-(u)$, $\delta^+(G) = min_{u\in V} deg^+(u)$ et $\delta^-(G) = min_{u\in V} deg^-(u)$.

Chemins et cycles

Un *chemin* dans un graphe orienté $G = (V, A)$ est un sous-graphe $P = v_1 v_2 \dots v_m$ tel que pour tout $i \in \{1, 2, \dots, m-1\}$, $v_i v_{i+1} \in A$. Si $v_1 = v_m$, le chemin P est appelé *cycle* ou *circuit*. Comme dans les graphes non-orientés, un chemin $P = v_1 v_2 \dots v_m$ est dit *élémentaire* si tous les sommets v_i sont distincts. Le chemin P est dit *simple* si tous ses arcs sont distincts. Un graphe orienté acyclique (ou DAG, pour Directed Acyclic Graph) est un graphe orienté sans cycles.

Soit $G = (V, A)$ un graphe orienté. On appelle graphe *support* de G (noté G^*), le graphe non-orienté obtenu à partir de G en supprimant l'orientation des arcs.

Connexité

Un graphe orienté $G = (V, A)$ est dit *fortement connexe* si pour tout $uv \in V \times V$, il existe dans G un chemin de u vers v et un chemin de v vers u. Le graphe G est dit *faiblement connexe* si G^* est connexe.

Une composante *fortement connexe* de G est un sous-graphe $G[X]$ induit par un ensemble X, $X \subseteq V$, tel que (i) $G[X]$ est fortement connexe et

(ii) $G[X]$ est maximum pour l'inclusion (c'est-à-dire pour tout $u \in V \backslash X$, $G[X \cup \{u\}]$ n'est pas fortement connexe). Une illustration est donnée dans la Figure 1.17.

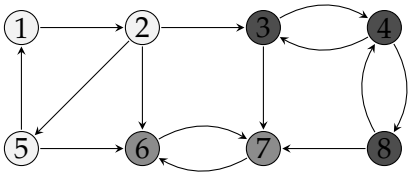


FIGURE 1.17 – *Ce graphe possède trois composantes fortement connexes induites par les ensembles de sommets* $\{1,2,5\}$, $\{6,7\}$ *et* $\{3,4,8\}$.

1.3 NOTIONS DE COMPLEXITÉ

Soit $\mathcal{P}$ un problème dont l'ensemble des instances possibles est noté $I(\mathcal{P})$. La *complexité en temps d'exécution* d'un algorithme $\mathcal{A}$, résolvant le problème $\mathcal{P}$, est le nombre d'opérations atomiques (c'est-à-dire indivisibles) nécessaires à l'accomplissement de l'algorithme. Cette complexité est par conséquent exprimée en fonction de la taille des instances du problème $\mathcal{P}$.

Généralement, pour analyser la complexité f d'un algorithme $\mathcal{A}$, on cherche un ordre de grandeur de f. Pour cette raison, nous étudions le comportement asymptotique de la fonction f en utilisant les notations dites *notations de Landau*.

Définition 1.2 (notations de Landau) *Soient f et g deux fonctions.*

- $f(n) = \mathcal{O}(g(n))$ *si* $\exists c > 0$, $\exists n_0 : \forall n > n_0$, $f(n) \leq c \cdot g(n)$ *(f est dominée asymptotiquement par g) ;*
- $f(n) = \Omega(g(n))$ *si* $g = \mathcal{O}(f(n))$ *(f domine g asymptotiquement) ;*
- $f(n) = \Theta(g(n))$ *si* $\exists c_1, c_2 > 0$, $\exists n_0 : \forall n > n_0$, $|c_1 \cdot g(n)| < |f(n)| < c_2 \cdot |g(n)|$ *(f et g sont du même ordre).*

1.3.1 Classes P et NP

Un *problème de décision* est un problème dont la réponse est "oui" ou "non".

Définition 1.3 (Classe P) *Soit $\mathcal{P}$ un problème de décision. Le problème $\mathcal{P}$ est dans la classe* P *(Polynomial), si et seulement si il existe un algorithme déterministe $\mathcal{A}$ (c'est-à-dire une suite d'étapes élémentaires sans choix aléatoire) qui résout $\mathcal{P}$ en un temps polynomial en la taille des instances du problème $\mathcal{P}$.*

On définit la classe NP des problèmes dits *Non-déterministes Polynomiaux* comme suit.

Définition 1.4 (Classe NP) *La classe* NP *contient les problèmes de décision qui peuvent être décidés sur une machine non déterministe en temps polynomial.*

De façon équivalente, NP est la classe des problèmes qui admettent

un algorithme polynomial déterministe (dit *vérifieur*) capable de tester la validité d'une réponse au problème.

L'un des problèmes les plus importants dans le domaine de l'informatique théorique est la question "est-ce que P est égal à NP ? ". Une autre manière de formuler cette question est : est-ce que si on arrive à vérifier efficacement l'exactitude d'une solution d'un problème donné, cela implique que nous puissions le résoudre efficacement ? Ce problème est toujours ouvert, mais de nombreux scientifiques pensent que $\mathsf{P} \neq \mathsf{NP}$.

Les problèmes les plus difficiles dans NP sont appelés NP-*complets*. Avant de donner une définition formelle de la NP-complétude, on définit une transformation polynomiale comme suit.

Définition 1.5 *Soient $\mathcal{P}$ et $\mathcal{P}'$ deux problèmes dans* NP. *Il existe une* transformation polynomiale *de $\mathcal{P}$ vers $\mathcal{P}'$ (et on note $\mathcal{P} \leq_{\mathsf{P}} \mathcal{P}'$) s'il existe un algorithme déterministe qui transforme toute instance $x \in I(\mathcal{P})$, en un temps polynomial (en fonction de la taille de x), vers une instance $x' \in I(\mathcal{P}')$, de sorte que la réponse au problème $\mathcal{P}$ avec l'instance x est "oui" si et seulement si la réponse au problème $\mathcal{P}'$ avec l'instance x' est "oui".*

Les problèmes NP-complets sont définis comme suit.

Définition 1.6 *[GJ79] Un problème $\mathcal{P}$ est dit* NP-complet *s'il vérifie les deux conditions suivantes.*

1. $\mathcal{P} \in \mathsf{NP}$;
2. $\mathcal{P}' \leq_{\mathsf{P}} \mathcal{P}$, *pour tout* $\mathcal{P}' \in \mathsf{NP}$.

Un problème $\mathcal{P}$ est dit NP-*difficile* si et seulement si il vérifie la condition 2. de la définition 1.6. Donc, la classe des problèmes NP-difficiles vérifie les propriétés suivantes.

Propriété 1.3 *[GJ79] Si* $\mathsf{P} = \mathsf{NP}$ *alors* NP-*complet*$= \mathsf{P}$. *Voir une illustration dans la Figure 1.18.*

Preuve. Évidemment, si $\mathsf{P} = \mathsf{NP}$ alors NP-complet $\subseteq \mathsf{P}$. Montrons maintenant que $\mathsf{P} \subseteq$ NP-complet. Soit $\mathcal{P} \in \mathsf{P}$. Nous rappelons que la classe P est une classe d'équivalence par rapport à la relation $\leq_{\mathsf{P}}$ de transformations polynomiales. Donc, $\mathcal{P}' \leq_{\mathsf{P}} \mathcal{P}$, pour tout $\mathcal{P}' \in \mathsf{P}$. Cela implique que dans le cas où $\mathsf{P} = \mathsf{NP}$, $\mathcal{P}' \leq_{\mathsf{P}} \mathcal{P}$, pour tout $\mathcal{P}' \in \mathsf{NP}$. Par conséquent $\mathcal{P} \in$ NP-difficile (et donc $\mathcal{P} \in$ NP-complet). Donc, $\mathsf{P} \subseteq$ NP-complet. □

Propriété 1.4 NP-*difficile* $\cap$ NP $=$ NP-*complet.*

Le premier problème qui a été prouvé NP-complet est le problème SAT de satisfiabilité booléenne. Étant donné un ensemble de variables booléennes $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, une *clause* C construite sur $\mathcal{X}$ est une proposition de la forme $l_1 \vee l_2 \vee \ldots \vee l_r$ où chaque l_i est une variable ou la négation d'une variable dans $\mathcal{X}$. Les l_i sont appelés *littéraux*. Une *affectation* des variables dans $\mathcal{X}$ consiste à attribuer à tout $x_i \in \mathcal{X}$ une valeur de l'ensemble $\{vrai, faux\}$. Une affectation des variables dans $\mathcal{X}$ satisfait une clause $C = l_1 \vee l_2 \vee \ldots \vee l_r$ s'il existe au moins un $i \in \{1, \ldots, r\}$ tel que $l_i = vrai$.

Le problème SAT est défini comme suit.

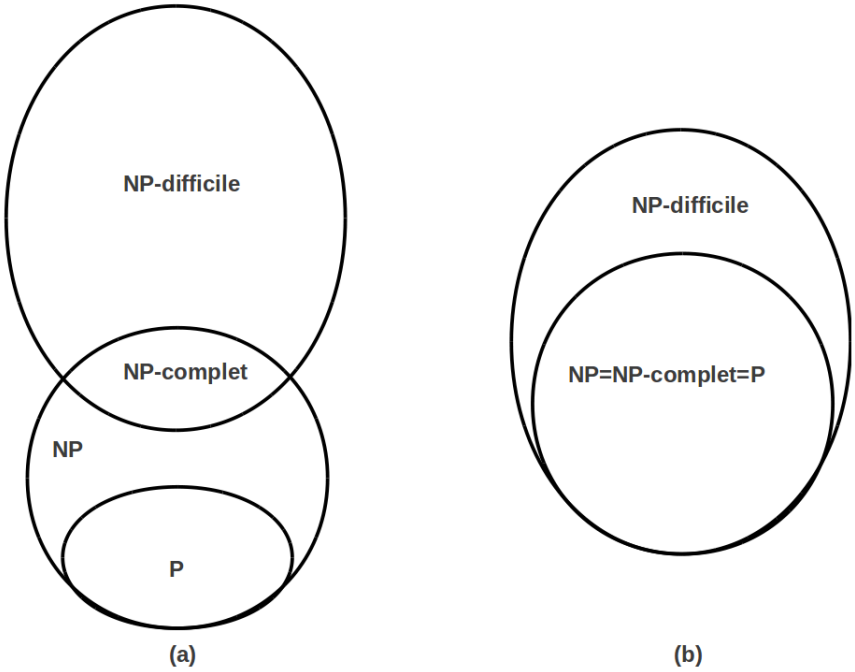


FIGURE 1.18 – *Représentation des classes* P, NP, NP*-difficile et* NP*-complet. (a) dans le cas où* P $\neq$ NP. *(b) dans le cas où* P $=$ NP.

BOOLEAN SATISFIABILITY (SAT)
Instance : Un ensemble de variables $\mathcal{X}$ et un ensemble $\mathcal{C}$ de clauses construites sur $\mathcal{X}$.
Question : Existe-t-il une affectation des variables de $\mathcal{X}$ qui satisfait toutes les clauses de $\mathcal{C}$?

C'est Stephen Cook-Levin [Coo71] qui a démontré, en 1971, la NP-complétude du problème SAT. Ce résultat est fondamental en théorie de la complexité. En effet, puisque tous les problèmes dans NP sont polynomialement réductibles à tout problème dans la classe NP-complet, pour démontrer qu'un problème $\mathcal{P}$ est NP-complet, il suffit de démontrer que (1) $\mathcal{P} \in$ NP et (2) SAT $\leq_{\mathsf{P}} \mathcal{P}$.

De manière générale, nous avons la propriété suivante.

Propriété 1.5 *La condition* 2. *de la Définition 1.6, peut être remplacée par la vérification s'il existe au moins un problème* NP*-difficile qui puisse être transformé, en un temps polynomial, vers* $\mathcal{P}$.

Les problèmes NP-complets les plus connus sont répertoriés dans le livre de Garey et Johnson [GJ79].

1.3.2 Classe NPO

Dans la section précédente, nous avons présenté les problèmes dans lesquels on se pose une question dont la réponse est "oui" ou "non". Cependant, les problèmes réels sont en général des problèmes d'optimisa-

tion : on cherche à trouver la meilleure solution parmi toutes les solutions possibles (dites solutions *réalisables*).

On s'intéresse particulièrement aux problèmes d'optimisation dont les instances sont reconnaissables (c'est-à-dire on peut vérifier si une entrée donnée est bien une instance du problème) en un temps polynomial et les solutions réalisables sont vérifiables en un temps polynomial. Cette classe des problèmes d'optimisation est appelée NPO.

Définition 1.7 (**Classe** NPO [PY91]) *Un problème $\mathcal{P} \in$ NPO est un quadruplet (I, sol, m, but) tels que*

1. *I est l'ensemble des instances de $\mathcal{P}$ et elles sont reconnaissables en temps polynomial.*
2. *Pour tout $x \in I$, $sol(x)$ est l'ensemble des solutions réalisables pour $\mathcal{P}$ avec l'instance x. Il existe un polynôme p tel que (i) pour tout $y \in sol(x)$, $|y| \leq p(|x|)$ ($|x|$ et $|y|$ sont respectivement les tailles de x et y) et (ii) pour tout $x, y \in I$ tel que $|y| \leq p(|x|)$, on peut vérifier en temps polynomial si $y \in sol(x)$.*
3. *Étant données une instance x et une solution réalisable y pour $\mathcal{P}$ avec l'instance x, $m(x, y)$ est un entier positif appelé* mesure *de la solution y. La fonction m est aussi calculable en un temps polynomial.*
4. *$but \in \{min, max\}$.*

Soit $\mathcal{P} \in$ NPO. Une solution *optimale* pour $\mathcal{P}$ avec l'instance x est une solution réalisable y vérifiant la condition suivante :

$$m(x, y) = but\{m(x, y') : y' \in sol(x)\} \tag{1.4}$$

Si $but = max$ (resp. $but = min$), le problème $\mathcal{P}$ est un problème de *maximisation* (resp. de *minimisation*).

Un exemple d'un problème de la classe NPO est le problème Maximum Satisfiability (MAX SAT) . Ce problème est défini comme suit.

Maximum Satisfiability (MAX SAT)
Instance : Un ensemble de variables $\mathcal{X}$ et un ensemble $\mathcal{C}$ de clauses construites sur $\mathcal{X}$.
Solution : Une affectation des variables de $\mathcal{X}$.
Mesure : Le nombre de clauses satisfaites.

Propriété 1.6 *[PY91]* MAX SAT $\in$ NPO.

Pour tout problème $\mathcal{P} = (I, sol, m, but)$ dans NPO, il existe une version de décision dite *version de décision naturelle*. Si $\mathcal{P}$ est un problème de maximisation (resp. de minimisation), la version de décision naturelle de $\mathcal{P}$ est définie comme suit : étant donnés une instance x et un entier $k \geq 0$, existe-t-il une solution réalisable y dont la mesure $m(x, y)$ est supérieure (resp. inférieure) ou égale à k ?

Par exemple, la version de décision qui correspond au problème MAX SAT est le problème MAX SAT-Dec défini comme suit.

MAX SAT-DEC
Instance : Un ensemble de variables $\mathcal{X}$, un ensemble $\mathcal{C}$ de clauses construites sur $\mathcal{X}$ et un entier $k \geq 0$.
Question : Existe-il une affectation des variables de $\mathcal{X}$ qui satisfait au moins k clauses dans $\mathcal{C}$?

Remarque 1.2 *Souvent on parle de la* NP*-complétude et la* NP*-difficulté des problèmes d'optimisation dans* NPO*, mais on considère implicitement leurs versions de décision naturelles.*

1.3.3 Comment contourner la NP-difficulté ?

Soit $\mathcal{P}$ un problème dans NPO dont la version de décision naturelle est NP-difficile. A moins que P = NP, il n'existe aucun algorithme déterministe qui peut résoudre $\mathcal{P}$ en un temps polynomial. Il existe essentiellement trois approches pour contourner la NP-difficulté : (1) les algorithmes d'approximation, (2) les heuristiques et (3) les algorithmes exacts de complexité exponentielle, mais aussi efficaces que possible.

Algorithmes d'approximation des problèmes dans NPO

Pour toute instance x d'un problème $\mathcal{P} \in$ NPO, on note par $opt(x)$ la mesure d'une solution optimale pour $\mathcal{P}$ avec l'instance x.

Définition 1.8 (**Ratio de performance**) *Soit $\mathcal{P} = (I, sol, m, but)$ un problème dans* NPO*. Étant donnée une solution réalisable pour $\mathcal{P}$ avec l'instance x, notée y, le ratio de performance de y par rapport à x est défini comme suit :*

$$R(x,y) = max\ \{\frac{m(x,y)}{opt(x)}, \frac{opt(x)}{m(x,y)}\} \tag{1.5}$$

Le ratio de performance est toujours supérieur ou égal à 1 ; s'il est proche de 1, alors la solution réalisable y est proche de la solution optimale.

Définition 1.9 (α**-approximation**) *Soit $\mathcal{P} = (I, sol, m, but)$ un problème dans* NPO*. Soit $\mathcal{A}$ un algorithme qui, pour toute instance $x \in I$, calcule une solution réalisable $\mathcal{A}(x)$. L'algorithme $\mathcal{A}$ est de α-approximation pour $\mathcal{P}$ si pour tout $x \in I$, $R(x, \mathcal{A}(x)) \leq \alpha$.*

En utilisant cette définition, on définit certaines classes dans NPO comme suit.

Définition 1.10 (**Classe** APX) *Un problème $\mathcal{P}$, $\mathcal{P} \in$* NPO*, est dans la classe* APX *s'il existe, pour $\mathcal{P}$, un algorithme polynomial de α-approximation pour une constante $\alpha > 1$.*

Définition 1.11 (**Classe** PTAS) *Un problème $\mathcal{P}$, $\mathcal{P} \in$* NPO*, est dans la classe* PTAS*, s'il existe pour tout $\epsilon > 0$, un algorithme polynomial de $(1+\epsilon)$-approximation pour $\mathcal{P}$.*

Pour définir les classes des problèmes APX-difficiles et les problèmes APX-complets, nous utilisons les réductions (dites *L*-réductions) introduites par Papadimitriou et Yannakakis [PY91].

Définition 1.12 (*L*-**réduction**) *Soient $\mathcal{P}_1 = (I_1, sol_1, m_1, but_1)$ et $\mathcal{P}_2 = (I_2, sol_2, m_2, but_2)$ deux problèmes dans* NPO. *On note par opt_1 (resp. opt_2) la mesure d'une solution optimale pour $\mathcal{P}_1$ (resp. $\mathcal{P}_2$). Une* L-réduction *est une paire (f, g) de fonctions polynomiales satisfaisant toutes les conditions suivantes.*

1. *Si $x \in I_1$, alors $f(x) \in I_2$;*
2. *Si $y \in sol_2(f(x))$, alors $g(y) \in sol_1(x)$;*
3. *Il existe une constante positive α telle que $opt_2(f(x)) \leq \alpha \cdot opt_1(x)$;*
4. *Il existe une constante positive β telle que $|opt_1(x) - m_1(x, g(y))| \leq \beta \cdot |opt_2(f(x)) - m_2(f(x), y)|$.*

Maintenant, nous pouvons définir les notions d'APX-complétude et d'APX-difficulté.

Définition 1.13 (APX-**complet**) *Soit $\mathcal{P} \in$ NPO. Le problème $\mathcal{P}$ est* APX*-complet s'il vérifie les deux conditions suivantes.*

1. *$\mathcal{P} \in$ APX ;*
2. *Pour tout problème $\mathcal{P}' \in$ APX, il existe une L-réduction de $\mathcal{P}'$ vers $\mathcal{P}$.*

La classe des problèmes APX-difficiles contiennent tous les problèmes qui vérifient la condition 2. de la définition 1.13.

Propriété 1.7 APX*-complet* = APX*-difficile* ∩APX.

Comme nous l'avons déjà évoqué pour les problèmes NP-complets et les problèmes NP-difficiles, pour prouver l'APX-complétude et l'APX-difficulté d'un problème $\mathcal{P}$, nous pouvons remplacer la condition 2. de la Définition 1.13 par la vérification de l'existence d'au moins un problème APX-difficile qui peut se transformer par une L-réduction vers $\mathcal{P}$.

Heuristiques

Une heuristique est un algorithme qui fournit rapidement (c'est-à-dire en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation. Contrairement aux algorithmes d'approximation, les heuristiques ne garantissent pas la qualité de la solution obtenue.

Algorithmes exacts

Une manière de résoudre un problème NP-difficile est de proposer un algorithme exact de complexité exponentielle, mais aussi efficace que possible. Nous présentons ici un exemple de ces types d'algorithmes. Il s'agit de la méthode de *séparation et évaluation* (*Branch-and-bound*) [LD60]. Un algorithme Branch-and-bound est un algorithme exact qui consiste à énumérer les solutions réalisables de manière intelligente : en utilisant certaines propriétés du problème en question, l'algorithme peut éliminer des solutions partielles qui ne mènent pas à la solution optimale. Dans un bon algorithme Branch-and-bound, seules les solutions potentiellement bonnes sont donc énumérées. Cependant, dans le pire des cas, l'algorithme Branch-and-bound peut énumérer toutes les solutions réalisables.

1.3.4 Complexité paramétrée

C'est une branche récente de la théorie de complexité. Elle a été introduite en 1992 par R. Downey et M. Fellows [DF92]. Le but est de chercher un algorithme exact pour résoudre un problème dans NPO dont la version de décision naturelle est NP-difficile. L'algorithme en question est donc exponentiel, mais la partie exponentielle porte seulement sur des paramètres (un sous-ensemble des paramètres du problème) fixés et considérés petits dans la pratique. Si un tel algorithme existe, le problème est dans la classe FPT (*fixed-parameter tractable*). Par exemple, supposons que nous avons un problème $\mathcal{P}$ qui possède deux paramètres n et k où k est considéré petit par rapport à n. Le problème est dans FPT, paramétré par k, s'il peut être résolu par un algorithme dont la complexité est de la forme $\mathcal{O}(f(k) \cdot p(n))$, où $f(k)$ est une fonction qui ne dépend que de k, et $p(n)$ est un polynôme qui ne dépend que de n (c'est-à-dire $p(n) = n^{\mathcal{O}(1)}$).

Définition 1.14 (**Problème paramétré**) *Un problème paramétré est un ensemble $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}^+$ où Σ est un alphabet fini, et une question de la forme : étant donné $(x,k) \in \Sigma^* \times \mathbb{N}^+$, est-ce que $(x,k) \in \mathcal{P}$? L'entier k est appelé* paramètre *du problème et x est un objet appelé* description *du problème.*

De façon plus formelle, on définit la classe FPT comme suit.

Définition 1.15 (**Classe** FPT) *Soit $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}^+$ un problème paramétré par k. Le problème $\mathcal{P}$ est dans la classe* FPT *s'il existe un algorithme déterministe $\mathcal{A}$ qui, étant donné $(x,k) \in \Sigma^* \times \mathbb{N}^+$, vérifie en temps de $f(k) \cdot n^{\mathcal{O}(1)}$ si $(x,k) \in \mathcal{P}$, où f est une fonction arbitraire et n est la taille de l'objet x.*

Les problèmes dans la classe FPT sont appelés également problèmes à *exponentielle contrôlée*.

Il existe des problèmes paramétrés à *exponentielle incontrôlée*. Ces problèmes sont dits *Fixed Parameter Intractable* [ADF93].

Pour définir les problèmes NP-complets nous avons utilisé la notion de transformations polynomiales, et pour définir les problèmes APX-complets nous avons utilisé les L-réductions. Par analogie, pour définir les problèmes à exponentielle incontrôlée nous avons besoin d'introduire une autre transformation appelée FPT-*transformation*.

Définition 1.16 (FPT-**transformation**) *Soient $\mathcal{P}_1 \subseteq \Sigma_1^* \times \mathbb{N}^+$, et $\mathcal{P}_2 \subseteq \Sigma_2^* \times \mathbb{N}^+$, deux problèmes paramétrés. Une* FPT-*transformation de $\mathcal{P}_1$ en $\mathcal{P}_2$ est une fonction $\mathcal{F} : \Sigma_1^* \times \mathbb{N}^+ \mapsto \Sigma_2^* \times \mathbb{N}^+$ telle que :*

- *Pour tout $(x_1,k_1) \in \Sigma_1^* \times \mathbb{N}^+$, $(x_1,k_1) \in \mathcal{P}_1$ si et seulement si $\mathcal{F}(x_1,k_1) \in \mathcal{P}_2$;*
- *Pour tout $(x_1,k_1) \in \Sigma_1^* \times \mathbb{N}^+$, $\mathcal{F}(x_1,k_1)$ peut se calculer en temps de $f(k_1) \cdot |x_1|^c$, tels que f est une fonction arbitraire et c est une constante ;*
- *Il existe une fonction $g : \mathbb{N}^+ \mapsto \mathbb{N}^+$ tel que, pour tout $(x_1,k_1) \in \Sigma_1^* \times \mathbb{N}^+$, avec $\mathcal{F}(x_1,k_1) = (x_2,k_2)$, on a $k_2 \leq g(k_1)$.*

Les FPT-transformations ont été utilisées pour classer les problèmes à exponentielle incontrôlée en hiérarchies dites W-hiérarchies [DFR98].

La classe W[t], le t-ième niveau de la W-hiérarchie, est définie comme étant la classe des problèmes paramétrés réductibles (par une FPT-

transformation) à une variante, notée ici *t-PDC*, du problème Parameterized Decision Circuit (*PDC*) [DFR98]. Pour présenter le problème *PDC* et sa variante *t-PDC*, nous avons besoin de quelques définitions.

Un *circuit de décision* produit un résultat (appelé *sortie*) en appliquant des opérations logiques (appelées *portes*) sur une entrée. Une porte elle même est composée d'entrées et de sorties. Un exemple de portes sont les ET/OU logiques. Une porte est dite *bornée* (resp. *non-bornée*) si elle possède un nombre limité (resp. illimité) d'entrées.

Un *chemin* dans un circuit $\mathcal{C}$ est une séquence de portes $S = p_1 p_2 \dots p_{|S|}$ telle que : l'entrée (resp. la sortie) du circuit est reliée à l'entrée (resp. à la sortie) de la porte p_1 (resp. $p_{|S|}$), et pour tout $i \in \{1, 2, \dots |S| - 1\}$, la sortie de p_i est reliée à l'entrée de p_{i+1}. La *trame* (en anglais *weft*) est le nombre maximum de portes non-bornées situées sur un chemin dans $\mathcal{C}$. On note par $t(\mathcal{C})$ la trame du circuit $\mathcal{C}$.

On dit qu'un vecteur v est *accepté* par un circuit de décision $\mathcal{C}$ si la sortie de $\mathcal{C}$ vaut 1 quand les entrées de $\mathcal{C}$ sont les éléments de v. Le *poids* du vecteur v est égal aux nombre d'éléments de v qui valent 1.

Nous pouvons maintenant définir les problèmes *PDC* et *t-PDC*.

Parameterized Decision Circuit (*PDC*)
Instance : Un circuit de décision $\mathcal{C}$ et un entier $k \geq 0$.
Question : Existe-il un vecteur v de poids k accepté par $\mathcal{C}$?

Pour tout $t \geq 1$, on note par *t-PDC* la version du problème *PDC* dans laquelle les circuits en entrée sont des circuits de trame égale à t. On définit alors les classes $\mathsf{W}[\mathrm{t}]$ comme suit.

Définition 1.17 (**Classe** $\mathsf{W}[\mathrm{t}]$) *Pour tout $t \geq 1$, les problèmes de la classe $\mathsf{W}[\mathrm{t}]$ sont les problèmes paramétrés réductibles (par une FPT-transformation) au problème t-PDC.*

R. Downey et M. Fellows [DFR98] ont hiérarchisé les problèmes paramétrés comme suit :

$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \dots$

Propriété 1.8 *Pour tout $t \geq 1$, $\mathsf{W}[\mathrm{t}]$-complet=$\mathsf{W}[\mathrm{t}] \cap \mathsf{W}[\mathrm{t}]$-difficile.*

Conclusion du chapitre

Dans ce chapitre, nous avons présenté les notions de base nécessaires pour comprendre notre manuscrit. Nous passons maintenant au premier chapitre de contribution dans lequel nous proposons une nouvelle approche de comparaison de réseaux biologiques hétérogènes.

Comparaison de réseaux biologiques Hétérogènes

2

Les travaux que nous allons présenter dans ce chapitre ont été publiés dans les actes de la conférence internationale SEA 2012 (11^{th} Symposium on Experimental Algorithms) [FMBR12a].

Sommaire

L'accroissement de la qualité et de la quantité de données disponibles crée le besoin en méthodes d'analyse automatique, afin de mieux comprendre les processus cellulaires, l'organisation des systèmes biologiques et l'évolution à l'échelle des espèces. Sur la base de l'hypothèse que la conservation des structures topologiques (par exemple chemin, sous-graphe dense) et de leurs étiquetages via l'évolution à l'échelle des espèces implique une signification fonctionnelle, des approches comparatives peuvent aider à améliorer l'exactitude des données, à interpréter des chemins métaboliques, des complexes de protéines ou encore à générer, à analyser et à valider des hypothèses sur les réseaux sous-jacents.

Nous avons montré dans le chapitre précédent (Chapitre 1) qu'il existe plusieurs types de réseaux biologiques (par exemple réseaux d'interaction protéine-protéine, réseaux métaboliques, réseaux physiques) et que,

en fonction de la nature de ces réseaux, ils peuvent être modélisés par différents types de graphes (graphe non-orienté, graphe orienté, graphe mixte). Cette diversité de réseaux et de leurs modélisations nous mène à définir l'homogénéité et l'hétérogénéité dans ces réseaux comme suit.

Définition 2.1 *Deux réseaux biologiques $\mathcal{R}_1$ et $\mathcal{R}_2$ sont dits* homogènes *si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont de même type et sont donc modélisés par des graphes de même type.*

Définition 2.2 *Deux réseaux biologiques $\mathcal{R}_1$ et $\mathcal{R}_2$ sont dits* hétérogènes *si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont de type différent.*

Quand on parle de comparaison de réseaux homogènes, on considère que les réseaux représentent des données d'espèces différentes, et la comparaison est donc *inter-espèces*. En revanche, dans la comparaison de réseaux hétérogènes, on considère que les réseaux représentent des données provenant de la même espèce, et la comparaison est alors *intra-espèces*. Dans le premier type de comparaison on s'intéresse à l'évolution à l'échelle des espèces, tandis que dans le deuxième type de comparaison, dit aussi *intégration des réseaux* [ZKW+05], on s'intéresse à l'analyse des données en observant la même information biologique dans des contextes différents.

Exemple 2.1 (**Réseau homogène vs réseau hétérogène**) *Soient $\mathcal{R}_1$ et $\mathcal{R}_2$ deux réseaux biologiques. Si $\mathcal{R}_1$ est le réseau d'interaction protéine-protéine (modélisé par un graphe non-orienté) de la bactérie E. coli, et $\mathcal{R}_2$ est également le réseau d'interaction protéine-protéine (modélisé par un graphe non-orienté) de la levure S. cerevisiae, alors $\mathcal{R}_1$ et $\mathcal{R}_2$ sont homogènes. En revanche, si $\mathcal{R}_2$ est le réseau métabolique (modélisé par un graphe orienté, par exemple graphe des réactions) de la bactérie E. coli, alors $\mathcal{R}_1$ et $\mathcal{R}_2$ sont hétérogènes.*

Un *module biologique* est une collection de composants biologiques (molécules, gènes, protéines etc.) qui coopèrent entre eux pour réaliser une fonction biologique [HHLM99]. Les modules biologiques d'un réseau $\mathcal{R}_1$ modélisé par un graphe G_1 sont des sous-graphes de G_1 (que nous appellerons structures) dont la topologie dépend de la nature du réseau $\mathcal{R}_1$. Ces structures sont notamment des sous-graphes connexes ou sous-graphes denses (complexes des protéines) [BML+05, DBVS09, NK07, KMM+10], des chemins (par exemple voies de signalisations) [YIJ04], des arborescences orientées ou plus généralement des DAGs (voies métaboliques) [PRYLZU05]. Nous présentons en annexe les structures présentées comme biologiquement significatives, sur la base d'une étude bibliographique portant sur plus de 70 articles parmi les plus cités dans le domaine de la comparaison de réseaux biologiques.

Dans ce chapitre, nous allons d'abord présenter l'état de l'art de la comparaison de réseaux biologiques homogènes et hétérogènes. Ensuite, nous allons présenter nos travaux lesquels situent dans le cadre de la comparaison de réseaux hétérogènes.

2.1 Comparaison de réseaux homogènes (alignement de réseaux)

La comparaison de réseaux homogènes a pour objectif de chercher des structures conservées à l'échelle de plusieurs espèces. L'hypothèse de base est que cette conservation ne peut pas être due au hasard, mais plutôt elle implique que ces structures représentent des modules biologiques. Les principales méthodes de comparaison de réseaux homogènes sont basées sur l'alignement des réseaux. Kelley et al. [KSK+03, SIK+04] sont parmi les premiers à avoir formalisé ce concept.

Nous allons d'abord définir l'alignement et ensuite formuler le problème d'alignement des réseaux homogènes. Pour simplifier, nous considérons l'alignement de deux réseaux modélisés par des graphes non-orientés. La modélisation que nous allons présenter peut être généralisée pour aligner plus de deux réseaux et elle est applicable également aux graphes orientés.

Définition 2.3 (Alignement) *Soient $\mathcal{R}_1$ et $\mathcal{R}_2$ deux réseaux biologiques homogènes modélisés respectivement par deux graphes non-orientés $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$. Un* alignement *de G_1 et G_2 est une fonction $a : V_1 \mapsto V_2 \cup \{-\}$. Le symbole "-" est appelé* gap. *Pour tout $u \in V_1$, si $a(u) = -$, alors le sommet u n'a pas de correspondant dans V_2.*

Le score d'un alignement est défini comme suit.

Définition 2.4 (Score) *Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes non-orientés. Soit a un alignement de G_1 et G_2. Le* score *de a, noté $s(a)$, est la valeur suivante :*

$$s(a) = \sum_{\substack{u \in V_1 \\ a(u) \neq -}} \alpha(u, a(u)) + \sum_{\substack{u \in V_1 \\ a(u) \neq -}} \sum_{\substack{v \in V_1 \\ a(v) \neq -}} \beta(u, a(u), v, a(v))$$

tels que :

- $\alpha : V_1 \times V_2 \mapsto \mathbb{R}$ *où $\alpha(u, v)$ est le score associé à une correspondance entre u et v.*
- $\beta : V_1 \times V_2 \times V_1 \times V_2 \mapsto \mathbb{R}$ *où $\beta(u, v, u', v')$ est le score associé à une correspondance entre les paires de sommets (u, v) et (u', v').*

Si G_1 et G_2 représentent deux réseaux d'interaction protéine-protéine, alors la fonction α mesure la similarité entre les protéines, et la fonction β mesure la conservation des interactions entre protéines.

Le *problème d'alignement des graphes* est le problème qui consiste à rechercher un alignement de score maximum. Ce problème est NP-complet, donc algorithmiquement difficile [KSK+03, SSK+05] car il contient, comme sous-problème, le problème de l'isomorphisme de sous-graphes [Bun00, CFSV04]. La plupart des méthodes proposées pour résoudre ce problème sont des heuristiques. Kelley et al. [KSK+03, SIK+04] ont proposé une heuristique qui (i) construit un graphe dit *graphe d'alignement* et ensuite (ii) décompose le graphe d'alignement en clusters. Les clusters de ce graphe représentent des structures conservées à l'échelle de plusieurs espèces.

Construction du graphe d'alignement. Nous expliquons ici la construction du graphe d'alignement de deux réseaux. Cette construction peut être généralisée pour construire le graphe d'alignement de plus de deux réseaux (alors appelé *graphe d'alignement multiple*).

Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2$ deux graphes non-orientés modélisant deux réseaux biologiques homogènes. Soit $f : V_1 \mapsto 2^{V_2}$ une fonction qui établit la correspondance entre les sommets de G_1 et ceux de G_2. Le graphe d'alignement de G_1 et G_2, noté $\mathcal{G}$, est construit comme suit. Les sommets de $\mathcal{G}$ sont les couples (A, a) tels que A (resp. a) est un sommet dans G_1 (resp. dans G_2) et $a \in f(A)$. Par exemple, dans le cas où G_1 et G_2 sont des réseaux d'interaction protéine-protéine, $a \in f(A)$ si les séquences des protéines A et a sont similaires, c'est-à-dire la valeur $\gamma(A, a)$ est supérieure à un seuil donné, où γ est une fonction qui mesure la similarité entre protéines.

On met dans $\mathcal{G}$ une arête entre deux sommets (A, a) et (B, b) dans les cas suivants.

1. $(A, B) \in E(G_1)$ et $(a, b) \in E(G_2)$;
2. $(A, B) \notin E(G_1)$, $(a, b) \in E(G_2)$ et il existe un chemin dans G_1 entre A et B ;
3. $(A, B) \in E(G_1)$, $(a, b) \notin E(G_2)$ et il existe un chemin dans G_2 entre a et b ;
4. $(A, B) \notin E(G_1)$, $(a, b) \notin E(G_2)$ et il existe un chemin dans G_1 (resp. dans G_2) entre A (resp. a) et B (resp. b) ;

Les arêtes créées dans le premier cas sont appelées arêtes *directes*. Les arêtes créées dans le deuxième et le troisième cas sont appelées *gap*. Enfin, les arêtes créées dans le dernier cas sont appelées *mismatch* (voir la Figure 2.1). Les *gaps* et les *mismatchs* sont des interactions indirectes qui permettent de prendre en compte l'évolution des réseaux et aussi les erreurs dues à leurs constructions.

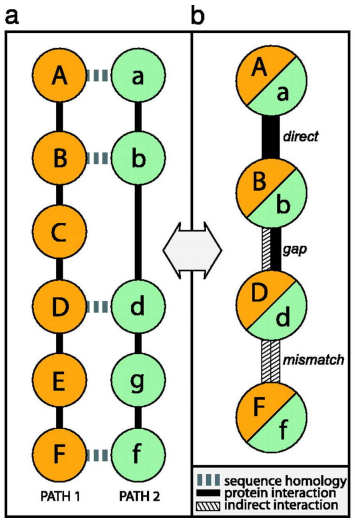


FIGURE 2.1 – *(a) le graphe G_1 (resp. G_2) est un chemin de A (resp. a) vers F (resp. f). Les lignes verticales représentent les interactions entre protéines. Les lignes horizontales (pointillées) représentent la similarité entre les séquences des protéines du graphe G_1 et celles du graphe G_2. (b) Le graphe $\mathcal{G}$ d'alignement de G_1 et G_2. Source : [KSK+03].*

En 2003, Kelley et al. ont présenté, dans [KSK+03], une méthode d'alignement de deux réseaux *PPI* pour identifier des chemins de plus hauts scores dans le graphe d'alignement. En 2004 ils ont amélioré cette méthode en présentant l'algorithme *PathBLAST* [SIK+04] qui cherche des sous-graphes conservés dans deux réseaux *PPI*. Biologiquement, les chemins conservés représentent des voies de signalisations et les sous-graphes conservés représentent des complexes de protéines. En 2005, Kelley et al. [SSK+05] ont généralisé *PathBLAST* en proposant l'algorithme *NetworkBLAST* qui permet l'alignement de trois réseaux. La limitation de cette méthode à l'alignement des trois réseaux est due à la taille du graphe d'alignement qui croit de manière exponentielle avec le nombre de réseaux à aligner. Pour contourner cette difficulté, Kalaev et al. [KBS09] ont présenté une méthode d'alignement appelée *NetworkBLAST-M* basée sur la construction d'un graphe dit *graphe à k-couches*. Avec cette méthode, ils ont pu aligner 10 réseaux *PPI* dont chacun contient plus de dix mille protéines.

La détection de complexes de protéines a été également étudiée par Sharan et al. [HMD06]. Ces auteurs ont présenté un modèle probabiliste pour chercher un sous-graphe dense dans le graphe d'alignement de plusieurs réseaux *PPI*.

Flannick et al. [FNS+06, FAC+08] ont proposé un algorithme appelé *Graemlin* utilisant un alignement progressif de deux réseaux pour construire le graphe d'alignement multiple.

Singh et al. [SXB08b, LLB+09] ont présenté l'algorithme *IsoRank* pour aligner des réseaux *PPI* en se basant sur l'intuition suivante : s'il y a une similarité entre deux protéines P_1 (du premier réseau) et P_2 (du deuxième réseau) alors les voisins de P_1 sont également similaires aux voisins du P_2.

Dans la section suivante, nous présentons l'état de l'art de la comparaison de réseaux hétérogènes.

2.2 Comparaison de réseaux hétérogènes

La comparaison de deux réseaux hétérogènes $\mathcal{R}_1$ et $\mathcal{R}_2$ consiste à chercher la même information biologique représentée dans $\mathcal{R}_1$ et $\mathcal{R}_2$ de façon différente, contrairement à la comparaison de réseaux homogènes où l'information est représentée de la même manière dans les deux réseaux.

Nous allons maintenant formuler le problème de comparaison de deux réseaux hétérogènes. Soient $\mathcal{R}_1$ et $\mathcal{R}_2$ deux réseaux hétérogènes modélisés respectivement par des graphes G_1 et G_2 et soit f, $f : V(G_1) \mapsto 2^{V(G_2)}$, une fonction qui établit la correspondance entre les sommets de G_1 et ceux de G_2. La comparaison de $\mathcal{R}_1$ et $\mathcal{R}_2$ consiste à chercher une structure H_1 (un sous-graphe de G_1) biologiquement significative dont les sommets induisent dans G_2, par la fonction f, une structure H_2 (un sous-graphe de G_2) qui soit aussi biologiquement significative.

M. Galperin et al. [GK00] ont montré, en se basant sur le concept d'adjacence, l'intérêt d'effectuer une telle étude multi-échelle dans les systèmes biologiques pour prédire des modules biologiques.

La plupart des méthodes actuelles pour comparer les réseaux biologiques hétérogènes sont généralistes (basées sur des principes communs), mais manuelles, ou alors sont des études au cas par cas (et donc spécifiques au cas étudié). La plupart de ces méthodes consiste à chercher des réactions successives dans les réseaux métaboliques, en étudiant la relation entre les gènes et/ou entre les protéines impliquées dans la catalyse de ces réactions. Ces relations sont notamment : la proximité génomique des gènes dans le génome [BML+05, LDAM04, OFGK00, PH04, RTT02, ZSF+02], la co-expression des gènes [ILB04, KVC04, WPM+06, WB04], les interactions des protéines [DW08, HGH08] et même, simultanément, la proximité génomique et la co-expression des gènes [WB04]. Pour étudier la proximité génomique des gènes qui catalysent des chaînes de réactions, Boyer et al. [BML+05] ont restreint la modélisation du graphe des réactions à un graphe non-orienté, en considérant que toutes les réactions sont réversibles. Ils ont modélisé également le génome par un graphe non-orienté dont les sommets sont les gènes et les arêtes représentent l'adjacence des gènes dans le génome. Leur méthode est basée sur la construction d'un multigraphe dit *multigraphe de correspondance*.

Construction du multigraphe de correspondance. Étant donnés deux graphes non-orientés $H_1 = (V_1, E_1)$ et $H_2 = (V_2, E_2)$ modélisant respectivement le réseau métabolique (représenté par le graphe des réactions) et le génome d'un organisme donné, le multigraphe de correspondance $H = (V, E'_1, E'_2)$ est construit comme suit.

- Les sommets de H sont des couples $(R_i, G_i) \in V_1 \times V_2$ tels que la réaction R_i est catalysée par un produit du gène G_i.
- On ajoute une arête dans E'_1 (resp. dans E'_2) entre (R_i, G_j) et (R_l, G_k) si $(R_i, R_l) \in E_1$ (resp. $(G_j, G_k) \in E_2$).

Une fois que le multigraphe de correspondance H est construit, les auteurs partitionnent H en composantes connexes communes, c'est-à-dire des sous-ensemble maximaux X, $X \subseteq V$, tels que pour tout $u, v \in X$ il existe dans H un chemin entre u et v ne contenant que des arêtes dans E'_1, et un chemin entre u et v ne contenant que des arêtes dans E'_2. Un exemple de construction d'un multigraphe de correspondance est illustré dans la Figure 2.2. Boyer al. [DBVS09] ont proposé une généralisation de l'approche présentée dans [BML+05] pour comparer plus de deux réseaux.

Les méthodes manuelles ou au cas par cas que nous avons mentionnées traitent généralement les problèmes de la comparaison d'un réseau métabolique, que nous appellerons *réseau principal*, avec un autre réseau, que nous appellerons *réseau guide*, modélisé par un graphe non-orienté. Par conséquent, ces types de problèmes se réduisent à une formalisation unique : trouver dans le réseau métabolique une chaîne de réactions impliquant des éléments, par exemple enzymes qui catalysent une réaction, ou des gènes produisant ces enzymes, qui sont proches dans le réseau guide. Le réseau métabolique doit être modélisé par un graphe orienté (par exemple le graphe des réactions présenté dans le Chapitre 1) pour pouvoir différencier les réactions irréversibles de celles qui sont réversibles. Une chaîne de réactions dans le réseau métabolique d'origine est donc un chemin orienté dans son graphe correspondant.

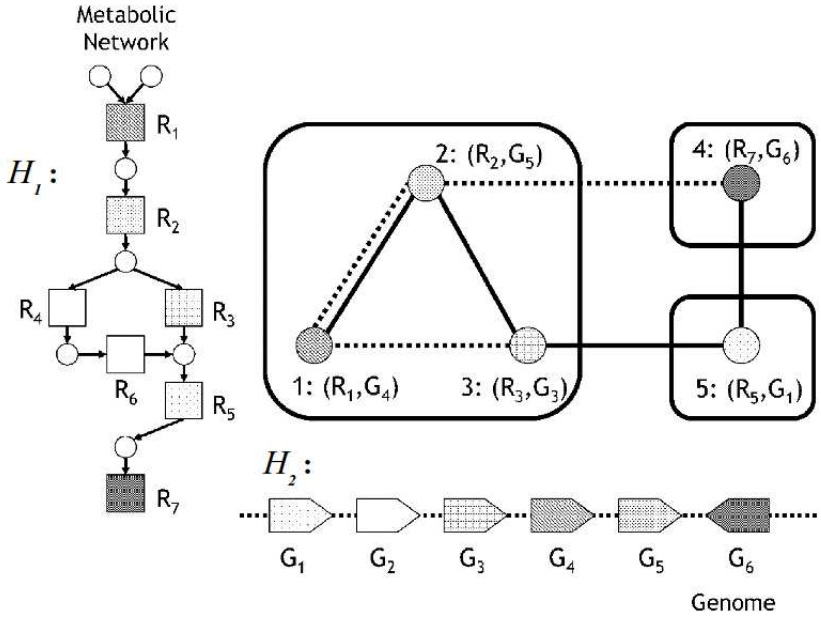


FIGURE 2.2 – **Exemple de construction d'un multigraphe de correspondance**. *Les sommets du graphe sont les paires (R_i, G_j) telles que R_i (un sommet du graphe des réactions H_1) est catalysée par le gène G_i (un sommet du graphe linéaire H_2 représentant le génome). Les arêtes en gras dans le multigraphe sont des arêtes dans H_1. Les arêtes pointillées sont des arêtes dans H_2. Les composantes connexes communes du multigraphe sont les trois ensembles $\{(R_2, G_5), (R_1, G_4), (R_3, G_3)\}$, $\{(R_7, G_6)\}$, $\{(R_5, G_1)\}$. Le fait que $\{(R_2, G_5), (R_1, G_4), (R_3, G_3)\}$ est une composante connexe commune, implique que les réactions R_1, R_2 et R_3 sont adjacentes dans le réseau métabolique, et sont catalysées par des gènes (G_3, G_4 et G_5) successifs dans le génome. Source : [BML+05].*

Dans ce chapitre nous allons proposer une modélisation unifiée, une analyse algorithmique unifiée, et des méthodes automatiques pour résoudre ce type de problèmes.

2.3 Notre modèle

Nous voulons comparer deux réseaux hétérogènes : un réseau principal et un réseau guide. Le premier réseau est représenté par un graphe orienté D (appelé *graphe principal*) et le deuxième est représenté par un graphe non-orienté G (appelé graphe *guide*). On suppose qu'il y a une correspondance entre les sommets de D et ceux de G. Cette correspondance est représentée par la fonction suivante.

$$f \ : \ V(D) \mapsto 2^{V(G)}$$

Intuitivement, la fonction f permet de représenter une similarité entre les sommets selon un critère donné (par exemple similarité entre les séquences de protéines).

Exemple 2.2 *Si D est le graphe des réactions (défini dans la Section 1.1.2 du Chapitre 1), et G est le graphe d'interaction protéine-protéine (défini dans la Section 1.1.1 du Chapitre 1) pour le même organisme, alors pour tout $R \in V(D)$, $f(R)$ désigne l'ensemble des protéines dans $V(G)$ qui catalysent la réaction R.*

Étant donné un graphe H, nous rappelons que, si H est non-orienté (resp. orienté), on note par $E(H)$ (resp. $A(H)$) l'ensemble d'arêtes (resp. d'arcs) de H. Pour tout $S \subseteq V(H)$, on note par $H[S]$ le sous-graphe de H induit par l'ensemble S.

Définition 2.5 (**Chemin** (D, G, f)-***consistant***) *Soient D un graphe orienté, G un graphe non-orienté et une fonction de correspondance $f : V(D) \mapsto 2^{V(G)}$. Un chemin P dans D est dit* (D, G, f)-consistant *s'il existe X, $X \subseteq V(G)$, tel que :*

1. $X \subseteq \bigcup_{v \in V(P)} f(v)$;
2. $f(v) \cap X \neq \emptyset$ *pour tout* $v \in V(P)$;
3. $G[X]$ *est connexe.*

Notre problème de comparaison est un problème de maximisation appelé Skew SubGraph Mining (abrégé SkewGraM). Le problème SkewGraM est défini comme suit.

SkewGraM
Instance : Un graphe orienté D, un graphe non-orienté G,
une fonction $f : V(D) \mapsto 2^{V(G)}$.
Solution : Un chemin (D, G, f)-consistant.
Mesure : La longueur du chemin (D, G, f)-consistant.

Le problème SkewGraM peut être vu comme étant un problème de recherche des motifs multidimensionnels [CSH+05, MHT+06, VM09]. Ici le motif est un chemin dans D et un sous-graphe connexe dans G tels que chaque sommet du chemin possède au moins un sommet correspondant (par la fonction f) dans le sous-graphe connexe. Donc ce problème peut être généralisé en modifiant les motifs de manière appropriée et/ou en introduisant une fonction objectif qui mesure l'intérêt des motifs recherchés. Ici, la fonction objectif est la longueur du chemin (D, G, f)-consistant recherché.

2.4 Raffinement du modèle

Dans ce chapitre, nous considérons le cas particulier où D est acyclique (DAG). Pour traiter le cas général (quand D contient des cycles) nous préconisons une décomposition préalable (sous forme de partition ou de couverture) de D en DAGs. Cette décomposition sera étudiée dans le prochain chapitre (Chapitre 3).

Nous avons des motivations algorithmiques et biologiques pour considérer l'acyclicité du graphe D. Du point de vue algorithmique, si le graphe D est un graphe orienté arbitraire, le problème SkewGraM contient, comme sous-problème, le problème du Plus Long Chemin qui est connu comme étant un problème NP-complet [GJ79]. Du point de vue biologique, nous allons appliquer ce modèle (en Section 2.8) pour comparer des voies métaboliques avec des réseaux d'interactions protéine-protéine. Or, les voies métaboliques peuvent être modélisées par des DAGs [GHM+02, PRYLZU05].

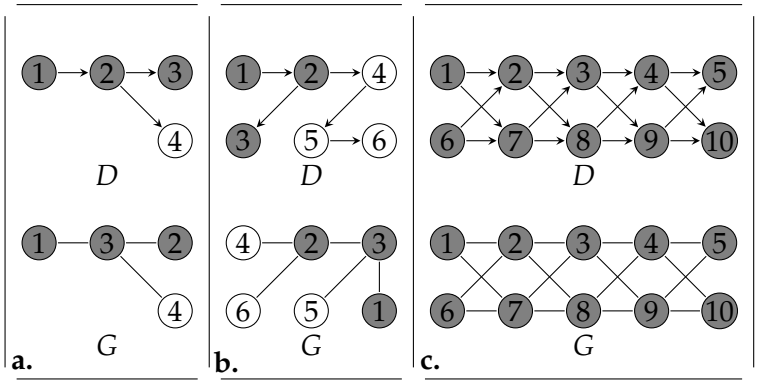


FIGURE 2.3 – ***Exemples des chemins*** (D, G)***-consistants***. *(a) Le plus long chemin* (D, G)*-consistant est le chemin* $1 \to 2 \to 3$*. (b) Le plus long chemin* (D, G)*-consistant est aussi le chemin* $1 \to 2 \to 3$*. En revanche, le plus long chemin dans* D *est le chemin* $1 \to 2 \to 4 \to 5 \to 6$*. Ce chemin n'est pas* (D, G)*-consistant, parce que le sous-graphe induit dans* G *par l'ensemble de sommets* $\{1, 2, 4, 5, 6\}$ *n'est pas connexe. (c) Il y a* 32 *chemins* (D, G)*-consistants de longueur* 4*.*

Pour étudier le problème SKEWGRAM, nous allons commencer par sa version la plus simple, à savoir le cas où la fonction de correspondance f est une bijection de l'ensemble $V(D)$ vers l'ensemble $\{\{v\} : v \in V(G)\}$. Cette version est appelée ONE-TO-ONE SKEWGRAM. Dans cette version, nous pouvons, sans perte de généralité, supposer que $V(D) = V(G) = \{1, 2, \ldots, n\}$ et que f vérifie : $f(v) = \{v\}$, pour tout $v \in V(D)$. Par convention, nous considérons que f est la fonction identité, c'est-à-dire nous écrivons $f(v) = v$, au lieu de $f(v) = \{v\}$. Dans ce cas, un chemin (D, G, f)-consistant est dit (D, G)*-consistant*. Par conséquent, un chemin (D, G)-consistant est un chemin dans D dont l'ensemble de sommets induit dans G un sous-graphe connexe. Donc, ONE-TO-ONE SKEWGRAM est le problème de maximisation formulé comme suit.

ONE-TO-ONE SKEWGRAM
Instance : Un DAG D, un graphe non-orienté G ayant le même ensemble de sommets $\{1, 2, \ldots, n\}$.
Solution : Un chemin (D, G)-consistant.
Mesure : La longueur du chemin (D, G)-consistant.

Pour mieux comprendre le problème, nous montrons quelques exemples dans la Figure 2.3.

Remarque 2.1 *Le plus long chemin* (D, G)*-consistant peut être différent du plus long chemin dans* D*. Dans certains cas, il n'est même pas un sous-chemin du plus long chemin dans* D*, comme le montre la Figure 2.3 (b).*

Remarque 2.2 *Nous ne pouvons pas nous contenter de calculer (par un algorithme exhaustif) tous les chemins* (D, G)*-consistants et ensuite de produire le chemin* (D, G)*-consistant le plus long. Dans certains cas, le nombre des chemins (et même le nombre de plus longs chemins)* (D, G)*-consistants est exponentiel. En effet, on peut généraliser l'exemple de la Figure 2.3(c) aux graphes à n sommets. Dans ce cas, le nombre de plus longs chemins* (D, G)*-consistants est* $2^{\frac{n}{2}}$*.*

Tout au long du reste de ce chapitre nous supposons que D est un DAG et G est un graphe non-orienté.

2.5 Étude de complexité du problème One-to-One SkewGraM

Nous étudions la complexité du problème One-to-One SkewGraM en considérant des contraintes topologiques sur les graphes D et G. Les résultats de complexité que nous avons obtenus sont résumés dans le Tableau 2.1. Nous rappelons que D^* est le graphe orienté obtenu à partir de D en supprimant l'orientation des arcs. Le graphe D^* est planaire extérieur si on peut mettre ses sommets sur un cercle de sorte que toutes les arêtes soient à l'intérieur du cercle et ne se rencontrent qu'aux extrémités (voir un exemple dans la Figure 1.16 du Chapitre 1). Nous rappelons également que le graphe non-orienté G est une étoile (resp. bi-étoile) s'il n'y a qu'un seul (resp. deux) sommet(s) dont le degré est au moins 2.

Avant d'expliquer les contraintes topologiques présentées dans le Tableau 2.1, nous définissions la décomposition arborescente d'un graphe non-orienté.

Définition 2.6 *Étant donné un graphe non-orienté $G = (V, E)$, une* décomposition arborescente *de G est un couple (X, T), où $X = \{X_1, \ldots, X_r\}$ est une famille de sous-ensembles de sommets de V et T est un arbre dont chaque nœud est étiqueté par un seul X_i, et chaque $X_i \in X$ est l'étiquette d'un seul nœud de T tels que :*

- $\bigcup_{1 \leq i \leq r} X_i = V$;
- *Pour tout $(u, v) \in E$, il existe $i, 1 \leq i \leq r$, tel que $u, v \in X_i$;*
- *Si X_i et X_j contiennent un même sommet $v \in V$, alors tous les nœuds X_k de T sur le chemin entre X_i et X_j contiennent v. De façon équivalente, pour tout $v \in V$, $T[\bigcup\{X_i : v \in X_i\}]$ est un sous-arbre de T.*

La largeur *d'une décomposition est la valeur $max_{1 \leq i \leq r}|X_i| - 1$. La largeur arborescente de G (en anglais :* treewidth*), notée $tw(G)$, est la largeur minimale parmi les largeurs de toutes les décompositions arborescentes de G [RS86]. Un exemple d'une décomposition arborescente est donné dans la Figure 2.4.*

Propriété 2.1 *[Bod88] Soit G un graphe non-orienté connexe. Le graphe G est un arbre si et seulement si $tw(G) = 1$.*

Propriété 2.2 *[Bod88] Soit G un graphe non-orienté connexe. Si G est un graphe planaire extérieur et G n'est pas un arbre alors $tw(G) = 2$.*

Le Tableau 2.1 résume les résultats de complexité du problème One-to-One SkewGraM en considérant différentes valeurs de la largeur arborescente du graphe D^*. On remarque que même si le graphe G est un arbre de diamètre 4, la valeur $tw(D^*)$ définit la frontière entre les instances faciles ($tw(D^*) = 1$) et les instances difficiles ($tw(D^*) = 2$) du problème One-to-One SkewGraM. Dans le cas général le problème est APX-difficile.

Nous commençons d'abord par étudier les cas où le problème One-to-One SkewGraM est facile.

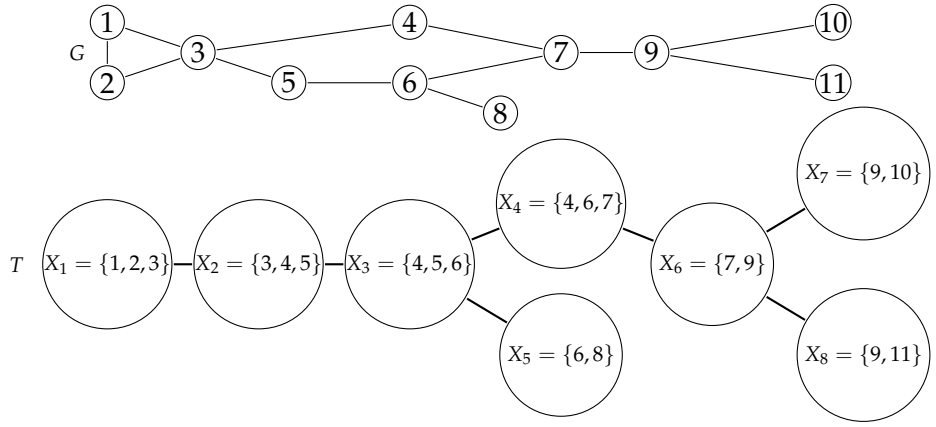


FIGURE 2.4 – *Une décomposition arborescente du graphe G. La largeur de la décomposition est égale à* 2 *et par conséquent* $tw(G) = 2$*, parce que G n'est pas un arbre (voir la Propriété 2.1).*

Théorème 2.1 *Le problème* ONE-TO-ONE SKEWGRAM *peut se résoudre en un temps polynomial quand au moins l'une des conditions suivantes est satisfaite.*

a) D^* *est un arbre.*
b) G est un chemin ou cycle élémentaire.
c) G est une bi-étoile.
d) G est une étoile.

Preuve. Tout d'abord nous rappelons que le problème du plus long chemin ainsi que le problème du plus long chemin entre deux sommets donnés, dans un DAG D, peuvent se résoudre en temps polynomial (par programmation dynamique).

a) Quand D^* est un arbre, l'ensemble $\mathcal{P}$ des chemins (orientés) maximaux dans D peut être calculé en temps polynomial. En effet, il suffit de commencer par tous les sommets sources (degré entrant 0) et d'effectuer un parcours dans le graphe jusqu'à ce que l'on arrive à tous les sommets cibles possibles (degré sortant 0).

Étant donnés deux graphes non-orientés $G_1 = (V, E_1)$ et $G_2 = (V, E_2)$, les composantes connexes communes entre G_1 et G_2 peuvent se calculer en temps polynomial [GHPR03]. Pour finir la preuve, il suffit de calculer, pour tout $P \in \mathcal{P}$, les composantes connexes communes entre P^* et $G[V(P)]$. Ces composantes connexes communes forment l'ensemble des chemins (D, G)-consistants maximaux. Cela est dû au fait que tout chemin dans D est un sous-chemin d'un chemin $P \in \mathcal{P}$. Le plus long chemin

G \ D^*	Arbre ($tw(D^*) = 1$)	Planaire extérieur ($tw(D^*) = 2$)	Graphe général
Chemin élémentaire, Cycle élémentaire, (bi-)étoile	P [Lem. 2.1]		
Arbre de diamètre 4	P [Lem. 2.1]	NPC [Th. 2.2]	NPC [Th. 2.2]
Graphe général	P [Lem. 2.1]	NPC [Th. 2.2]	NPC [Th. 2.2] APX-diff. [Th. 2.3]

Tableau 2.1 – *Complexité du problème* ONE-TO-ONE SKEWGRAM.

(D, G)-consistant est donc le chemin induit dans D par la composante connexe commune de cardinalité maximum.

b) Dans ce cas, un sous-graphe H de G est connexe si et seulement si $H = G$ ou H est un chemin (un sous-chemin de G). Le nombre des sous-chemins de G est en $\mathcal{O}(n^2)$ (où $n = |V|$), et ces chemins peuvent être calculés en temps polynomial. Par ailleurs, pour chacun de ces chemins, nous pouvons vérifier en temps polynomial si son ensemble de sommets induit un chemin dans D (c'est-à-dire on vérifie si le chemin est (D, G)-consistant). Enfin, parmi tous les chemins (D, G)-consistants, on choisit celui ayant la plus grande longueur.

c) Soient s_1 et s_2 les deux sommets de G (nécessairement adjacents) dont le degré est au moins 2. On note par T_1 (resp. T_2) l'étoile induite dans G par s_1 (resp. s_2) et les sommets adjacents à s_1 (resp. s_2) à l'exception de s_2 (resp. s_1). Soit P_1 (resp. P_2) le plus long chemin dans $D[V(T_1)]$ (resp. $D[V(T_2)]$) et soit P_3 le plus long chemin dans D passant par s_1 et s_2. Tous les chemins P_1, P_2 et P_3 sont (D, G)-consistants et peuvent se calculer en temps polynomial. Le plus long chemin parmi eux est le plus long chemin (D, G)-consistant. En effet, soit P un chemin (D, G)-consistant qui contient au moins deux sommets. Alors il y a trois possibilités :

(i) $s_1 \in V(P)$ et $s_2 \notin V(P)$. Dans ce cas $|V(P)| \leq |V(P_1)|$.

(ii) $s_2 \in V(P)$ et $s_1 \notin V(P)$. Dans ce cas $|V(P)| \leq |V(P_2)|$.

(iii) $s_1 \in V(P)$ et $s_2 \in V(P)$. Dans ce cas $|V(P)| \leq |V(P_3)|$.

Donc, on en déduit que le plus long chemin parmi P_1, P_2 et P_3 est le plus long chemin (D, G)-consistant.

d) Ce cas est similaire au cas c).

□

Maintenant, nous passons à l'étude des cas où le problème One-to-One SkewGraM devient difficile.

Théorème 2.2 *Le problème* One-to-One SkewGraM *(en sa version de décision naturelle) est* NP*-complet même si D^* est planaire extérieur et G est un arbre de diamètre* 4*.*

Preuve. La version de décision du problème One-to-One SkewGraM est le problème suivant : étant donnés un graphe non-orienté $G = (V, E)$, un DAG $D = (V, A)$ et un entier $k > 0$, existe il un chemin (D, G)-consistant de longueur au moins k ?

Étant donné un ensemble de sommets $X \subseteq V$, nous pouvons vérifier en temps polynomial si les conditions suivantes sont satisfaites : $D[X]$ est un chemin, $G[X]$ est un sous-graphe connexe et la longueur du chemin $D[X]$ est au moins k (c'est-à-dire $|X| \geq k + 1$). Donc le problème One-to-One SkewGraM est dans NP. Pour montrer sa NP-complétude, nous proposons une réduction du problème NP-complet MAX 2Sat [GJ79] : étant donnés un ensemble de clauses $\mathcal{C} = \{C_1, \ldots, C_p\}$ et un ensemble de variables $\mathcal{X} = \{x_1, \ldots x_n\}$, tels que chaque clause est une disjonction de deux littéraux et chaque littéral est une variable ou le complément d'une variable dans $\mathcal{X}$, le but est de trouver une affectation des variables dans $\mathcal{X}$ qui satisfait un nombre maximum de clauses dans $\mathcal{C}$.

Nous allons construire une instance (D, G) du problème One-to-One SkewGraM telle que D est un DAG, D^* est un graphe

planaire extérieur, G est un arbre de diamètre 4 et il existe une affectation des variables dans $\mathcal{X}$ satisfaisant k clauses dans $\mathcal{C}$ si et seulement si il existe un chemin (D, G)-consistant de longueur au moins $p + k + 1 + 2n$.

Construction du graphe orienté D. Le graphe D est construit sur $2p + 2n + 2$ niveaux divisés en deux catégories : niveaux optionnels et niveaux obligatoires. Les niveaux optionnels sont marqués par une étoile.

- niveau 0 : un sommet s ;
- niveau* $2i - 1$, $1 \leq i \leq p$: deux sommets $v_{i,1}$ et $v_{i,2}$ correspondant aux littéraux de la clause C_i ;
- niveau $2i$, $1 \leq i \leq p$: un sommet c_i correspondant à la clause C_i ;
- niveau $2p + 1$: deux sommets $v_{p+1,1}$ et $v_{p+1,2}$ correspondant, respectivement, aux variables x_n et $\overline{x_n}$;
- niveau $2p + 2$: un sommet c_{p+1} ;
- niveau $2p + 2 + 2i - 1$, $1 \leq i \leq n$: deux sommets a_i et b_i ;
- niveau $2p + 2 + 2i$, $1 \leq i < n$: un sommet A_i.

Après la création des sommets, on ajoute (a) tous les arcs possibles entre deux niveaux consécutifs, (b) l'arc sc_1 et (c) les arcs $c_i c_{i+1}$, $1 \leq i < p$. Un exemple de construction est montré dans la Figure 2.5.

Il est clair que le graphe orienté D est un DAG. Le graphe D^* est planaire extérieur. En effet, il suffit de dessiner les sommets de D^* sur un cercle en suivant cet ordre : $s, v_{1,1}, c_1, v_{2,1}, c_2, \ldots, v_{p+1,1}, c_{p+1}, a_1, A_1, \ldots, a_{n-1}, A_{n-1}, a_n, b_n, \ldots, b_1, v_{p+1,2}, \ldots, v_{1,2}$.

Construction du graphe non-orienté G. Le graphe G est construit sur le même ensemble de sommets que D. Il est défini comme étant un arbre de racine s et est composé des arêtes suivantes :

- Il y a une arête entre s et tous les sommets de l'ensemble $\{a_i, b_i : 1 \leq i \leq n\} \cup \{A_i : 1 \leq i < n\} \cup \{c_i : 1 \leq i \leq p + 1\}$;
- Il y a une arête entre chaque sommet a_i (resp. b_i) et tous les sommets $v_{l,m}$ avec $1 \leq l \leq p + 1, 1 \leq m \leq 2$, tels que $v_{l,m}$ correspond au littéral x_i (resp. $\overline{x_i}$). On rappelle que les sommets $v_{p+1,1}$ et $v_{p+1,2}$ correspondent respectivement aux littéraux x_n et $\overline{x_n}$.

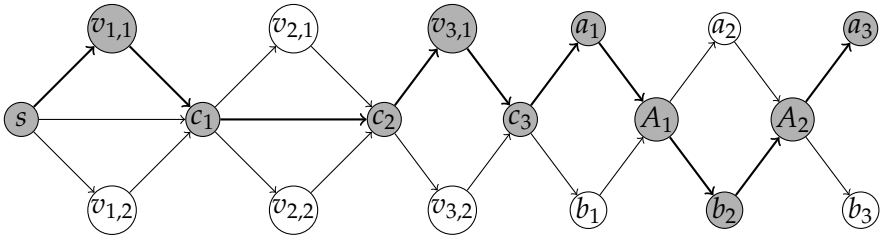
Évidemment le diamètre de l'arbre G est égal à 4 (voir une illustration dans la Figure 2.5).

Nous allons montrer la propriété suivante.

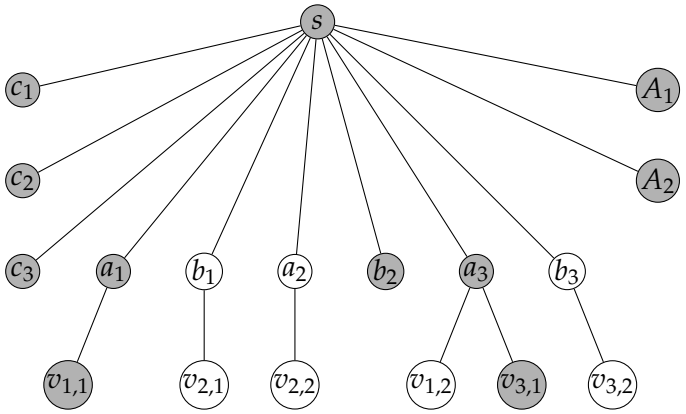
Propriété 2.3 *Il existe une affectation des variables dans $\mathcal{X}$ satisfaisant au moins k clauses dans $\mathcal{C}$ si, et seulement si, il existe un chemin (D, G)-consistant de longueur au moins $p + k + 1 + 2n$.*

Avant de prouver cette propriété, nous exhibons quelques propriétés élémentaires.

1. Tout chemin (D, G)-consistant de longueur au moins 1 doit contenir le sommet s. En effet, aucune composante connexe de $G - \{s\}$ n'induit un chemin de longueur au moins 1 dans D.
2. Pour tout i, $1 \leq i \leq n$, il n'y a aucun chemin (D, G)-consistant qui contient en même temps deux sommets correspondant respectivement aux littéraux x_i et $\overline{x_i}$. En effet, par construction du graphe D, un chemin dans D ne peut pas contenir les deux sommets a_i et b_i



D



G

FIGURE 2.5 – *Construction d'une instance* (D, G) *du problème* ONE-TO-ONE SKEWGRAM *à partir d'une instance* $(\mathcal{C}, \mathcal{X})$ *du problème* MAX 2SAT. *Ici,* $\mathcal{C} = \{(x_1 \vee x_3) \wedge (\overline{x_1} \vee x_2)\}$ *et* $X = \{x_1, x_2, x_3\}$. *Donc* $p = 2$ *et* $n = 3$. *Les sommets pleins forment un chemin* (D, G)*-consistant correspondant à l'affectation :* $\{x_1 = vrai, x_2 = faux, x_3 = vrai\}$.

en même temps. Par ailleurs, un sous-graphe connexe dans G qui contient au moins deux sommets ne peut pas contenir un sommet correspondant à x_i (resp. $\overline{x_i}$) sans qu'il contienne a_i (resp. b_i).

3. Tout chemin (D, G)-consistant P de longueur au moins $p+1$ contient nécessairement un sommet de chaque niveau obligatoire. En effet, la propriété 1 ci-dessus implique que P contient le sommet s. La longueur de P est au moins $p+1$, donc P contient au moins un sommet $v_{i,j}$, $1 \leq i \leq p+1$ et $1 \leq j \leq 2$. La connexité du sous-graphe $G[V(P)]$ implique qu'il existe au moins un r, $1 \leq r \leq n$ tel que $a_r \in V(P)$ ou $b_r \in V(P)$, et par construction de D, le chemin P doit donc contenir un sommet $v_{p+1,1}$ ou un sommet $v_{p+1,2}$ qui correspondent respectivement à x_n et $\overline{x_n}$. Encore une fois, la connexité de $G[V(P)]$ implique que l'un des sommets a_n ou b_n doit appartenir à P. Donc P contient un sommet de chaque niveau obligatoire.

Maintenant nous pouvons prouver la Propriété 2.3.
$\Rightarrow$: Soit $\mathcal{A}$ une affectation des variables dans $\mathcal{X}_n$ qui satisfait k' clauses de $\mathcal{C}$, tel que $k' \geq k$. On note $\mathcal{B}(i) = a_i$ si x_i est $vrai$, et on note $\mathcal{B}(i) = b_i$ sinon, pour tout $x_i \in \mathcal{X}$. On considère, sans perte de généralité, que les variables $v_{i_1,1}, \ldots, v_{i_{k'},1}$, $v_{p+1,1}$, avec $i_1 < i_2 < \ldots < i_{k'} < p+1$, correspondent aux littéraux avec l'affectation $vrai$.

Le chemin P dont l'ensemble de sommets est $\{s, c_1, \ldots, c_{i_1-1}, v_{i_1,1}, c_{i_1}, c_{i_1+1}, \ldots, c_{i_2-1}, v_{i_2,1}, c_{i_2}, \ldots, c_{i_{k'}-1}, v_{i_{k'},1}, c_{i_{k'}}, c_{i_{k'}+1}, \ldots c_p, v_{p+1,1}, c_{p+1}, \mathcal{B}(1), A_1, \mathcal{B}(2), A_2, \ldots, \mathcal{B}(n-1), A_{n-1}, \mathcal{B}(n)\}$ est un chemin (D, G)-consistant de longueur $p + k' + 1 + 2n$. En effet, les sommets $v_{i_1,1}, \ldots, v_{i_{k'},1}, v_{p+1,1}$ sont connectés, dans $G[V(P)]$, au sommet s via $\mathcal{B}(l_{i_j})$ (tel que $v_{i_j,1}$ correspond à $x_{l_{i_j}}$ ou à $\overline{x_{l_{i_j}}}$), $1 \leq j \leq k'$, et $\mathcal{B}(n)$ respectivement. Tous les autres sommets du P sont connectés directement à s.

$\Leftarrow$: Soit P un chemin (D, G)-consistant tel que $|V(P)| \geq p + k + 2 + 2n$. On note par k' le nombre de sommets de P qui appartiennent aux niveaux optionnels. En utilisant la propriété 2, on déduit que si on met à *vrai* les k' littéraux associés aux sommets de P qui se situent aux niveaux optionnels, on obtient une affectation satisfaisant k' clauses dans $\mathcal{C}$. Par ailleurs, les remarques 1 et 3 impliquent que P contient $p + 2 + 2n$ sommets qui se situent aux niveaux obligatoires. Donc $|V(P)| = p + 2 + k' + 2n$. Par conséquent, $k' \geq k$.

□

Nous avons montré la NP-complétude du problème ONE-TO-ONE SKEWGRAM pour une configuration particulière : G est un arbre de diamètre 4 et D^* est planaire extérieur. Maintenant, nous allons montrer que le problème est APX-difficile dans le cas général.

Théorème 2.3 *Le problème* ONE-TO-ONE SKEWGRAM *est* APX*-difficile.*

Preuve. Nous rappelons qu'un stable (en anglais : *independent set*) dans un graphe non-orienté $H = (V_H, E_H)$ est un ensemble $X \subseteq V_H$ tel que pour tout $u, v \in X$, nous avons $(u, v) \notin E_H$. Le problème de la recherche d'un stable de cardinalité maximum dans H (appelé MAXIMUM INDEPENDENT SET) est un problème APX-difficile, même si le graphe H est cubique (c'est-à-dire tous les sommets de H sont de degré 3) [AK00]. Nous notons par 3-MIS le problème de la recherche d'un stable de cardinalité maximum dans un graphe non-orienté cubique. Pour montrer l'APX-difficulté du problème ONE-TO-ONE SKEWGRAM, nous proposons une L-réduction à partir du problème 3-MIS.

Soit $\Gamma = (I, F)$ un graphe cubique dont l'ensemble de sommets est $I = \{1, 2, \ldots, n\}$. Nous posons $V = I \cup \{a_u^i \mid u \in I, 1 \leq i \leq 3\} \cup \{t\} \cup \{b_u^i \mid u \in I, 1 \leq i \leq 3\}$. Nous allons construire une instance (D, G) du problème ONE-TO-ONE SKEWGRAM telle que D et G ont le même ensemble de sommets V.

Construction du graphe orienté D. Soit $\overline{\Gamma}$ le graphe complémentaire de Γ. Soit D_1 le DAG obtenu de $\overline{\Gamma}$ en remplaçant toute arête (u, v) dans $\overline{\Gamma}$, telle que $u < v$, par un arc uv. Soit D_2 le graphe orienté défini comme suit.

- Pour tout $(u, v) \in F$, on crée un graphe orienté, noté H_{uv}, dont l'ensemble de sommets est $\{a_u^i, b_u^i, a_v^j, b_v^j\}$, et l'ensemble des arcs est $\{a_u^i b_u^i, a_v^j b_v^j\}$, où les entiers i, j sont choisis de sorte que (1) $1 \leq i, j \leq 3$ et que (2) pour tout $(u, v), (u', v') \in F$, si $u \neq u'$ ou $v \neq v'$ alors $V(H_{uv}) \cap V(H_{u'v'}) = \emptyset$. Autrement dit, les ensembles de sommets des graphes H_{uv} sont deux à deux disjoints, pour tout $(u, v) \in F$.

Les sommets a_u^i, a_v^j sont appelés *sources* et les sommets b_u^i, b_v^j sont appelés *puits* ;

- Soit $\prec$ un ordre arbitraire défini sur l'ensemble $\{H_{uv} : (u,v) \in F\}$. Pour toute paire des graphes $H_{uv}, H_{u'v'}$ tel que $H_{uv} \prec H_{u'v'}$, on ajoute des arcs dans D_2 partant de tout puits de H_{uv} vers toute source de $H_{u'v'}$.

Pour compléter la construction du graphe D, on ajoute tous les arcs possibles de $V(D_1)$ vers t, et de t vers toutes les sources de D_2.

Construction du graphe non-orienté G. Pour tout $u \in I$, nous construisons un chemin non-orienté $R_u : a_u^3 a_u^2 a_u^1 u b_u^1 b_u^2 b_u^3$. Ensuite, on ajoute une arête entre t et b_u^3, pour tout $u \in I$.

Nous allons montrer la propriété suivante.

Propriété 2.4 *Le graphe Γ a un stable à k sommets si et seulement si il existe un chemin (D,G)-consistant de longueur $7k$.*

La preuve de cette propriété est basée sur les propriétés élémentaires suivantes.

1. Tout chemin (D,G)-consistant de longueur au moins 1 contient le sommet t. En effet, soit P un chemin (D,G)-consistant de longueur au moins 1. Supposons, par contradiction, que $t \notin V(P)$. La construction du graphe D implique que $V(P) \subseteq I$ ou $V(P) \subseteq V(D_2)$. Dans un premier temps, supposons que $V(P) \subseteq I$. Soient $u, v \in V(P)$. La construction du graphe G implique qu'il n'y a aucun sous-graphe de $G - \{t\}$ connexe et contenant, à la fois, u et v. Cela induit une contradiction, car le chemin P est (D,G)-consistant. Maintenant, supposons que $V(P) \subseteq V(D_2)$. La connexité du sous-graphe $G[V(P)]$ et le fait que $V(P) \cap I = \emptyset$ impliquent que P ne contient aucun sommet de l'ensemble $\{a_u^i \mid u \in I, 1 \leq i \leq 3\}$. Donc, P contient au moins deux sommets b_u^i, b_v^j. En revanche, le degré entrant dans D des deux sommets b_u^i et b_v^j est égal à 1, et par conséquent le chemin P doit contenir a_u^i ou a_v^j. Cela induit une contradiction, car on vient de montrer que P ne peut contenir aucun sommet de l'ensemble $\{a_u^i \mid u \in I, 1 \leq i \leq 3\}$.
2. Soit P un chemin (D,G)-consistant de longueur au moins 1. Pour tout $u \in I$, si le chemin P contient u, alors P contient aussi tous les sommets du chemin R_u. En effet, la propriété 1 ci-dessus implique que $t \in V(P)$, et la construction du graphe G implique que P doit nécessairement contenir le sommet b_u^3. Par ailleurs, le degré entrant dans D du sommet b_u^3 est égal à 1, donc P doit aussi contenir le sommet a_u^3. Encore une fois, la construction du graphe G implique que P doit contenir tous les sommets de R_u pour assurer la connexité du sous-graphe $G[V(P)]$.
3. Soit P un chemin (D,G)-consistant de longueur au moins 1. Pour tout $u \in I$, si P ne contient pas u alors P ne contient non plus aucun sommet de R_u. En effet, supposons par contradiction que P contient un sommet x, $x \in V(R_u)$ et $x \neq u$. La propriété 1 implique que $t \in V(P)$. Le chemin P contient donc les deux sommets x et t. La

connexité de $G[V(P)]$ implique que P contient b_u^3. Donc P contient également a_u^3, car le degré entrant dans D du sommet b_u^3 est égal à 1. Par conséquent, la construction de G implique P contient le sommet u, c'est donc une contradiction.

4. Comme conséquence des propriétés 2 et 3, on en déduit que si P est un chemin (D, G)-consistant de longueur au moins 1, alors il existe un entier $p \geq 1$ tel que $|V(P)| = 7p + 1$.

Nous prouvons maintenant la Propriété 2.4.
$\Rightarrow$: Soit $X = \{x_1, x_2, \ldots, x_k\}$ un stable dans Γ tel que $x_i < x_{i+1}$ pour tout i, $1 \leq i \leq k-1$. Le graphe $\overline{\Gamma}[X]$ est complet, et donc $P = x_1 x_2 \ldots x_k$ est un chemin orienté dans D_1. Nous prolongeons ce chemin avec t et l'ensemble $\{a_u^i, b_u^i | \, u \in X, 1 \leq i \leq 3\}$, ce prolongement est bien possible quel que soit l'ordre $\prec$ défini sur l'ensemble de graphes $\{H_{uv} \ : \ uv \in F\}$. Le chemin P résultant du prolongement est (D, G)-consistant, parce que $G[V(P)]$ est le graphe formé par l'ensemble de chemins $\{R_u \ : \ u \in X\}$ connectés entre eux par le sommet t. Le nombre de sommets de P est égal à $7|X| + 1$. Mais $|X| = k$, donc la longueur du chemin P est $7k$.
$\Leftarrow$: Soit $Y \subseteq V$ tel que $|Y| = 7k + 1$ et Y induit un chemin (D, G)-consistant P. Soit $X = Y \cap I$. On va montrer que X est un stable dans Γ avec $|X| = k$. Le propriétés précédentes impliquent d'une part que $t \in V(P)$ et que $V(R_u) \subseteq V(P)$, pour tout $u \in X$. D'autre part, $V(R_{u'}) \cap V(P) = \emptyset$, pour tout $u' \notin X$. Donc $Y = X \cup \{t\} \cup \bigcup_{u \in X} \{a_u^3, a_u^2, a_u^1, b_u^1, b_u^2, b_u^3\}$. Donc $|X| = |Y| - 1 - 6|X|$ et par conséquent $|X| = k$.

Montrons maintenant que l'ensemble X est un stable dans Γ. Soient $u, v \in X$. Supposons par contradiction que $(u, v) \in F$. Alors, le graphe D contient un sous-graphe H_{uv}. Le chemin P ne peut donc pas contenir les deux sources de H_{uv}, car il n'y a pas de chemin, dans D, entre les sources de H_{uv}. Ceci est une contradiction, car nous avons déjà montré que $V(R_u) \subseteq V(P)$ et $V(R_v) \subseteq V(P)$.

Il est clair que cette réduction est bien une L-réduction car il y a un facteur constant "7" entre la cardinalité du stable et la longueur du chemin (D, G)-consistant.

□

2.6 Deux algorithmes pour résoudre One-to-One SkewGraM

Dans cette section, nous présentons deux algorithmes pour résoudre le problème One-to-One SkewGraM : une heuristique appelée AlgoH et un algorithme exact, appelé AlgoBB, basé sur la méthode branch-and-bound (présentée dans le Chapitre 1). Les deux algorithmes prennent en argument un DAG $D = (V, A)$, un graphe non-orienté $G = (V, E)$ et un arc $xy \in A$ et imposent que les deux graphes D^* et G soient connexes. Le but des deux algorithmes est de calculer le plus long chemin (D, G)-consistant passant par xy. Donc, pour résoudre le problème One-to-One SkewGraM, il suffit d'appliquer les deux algorithmes en considérant tous les arcs de D et ensuite de garder le chemin le plus long parmi tous les chemins obtenus. Nous considérons dans cette section que $V = \{1, \ldots, n\}$.

Avant de présenter les deux algorithmes, nous donnons quelques définitions et notations.

Nous rappelons que les composantes connexes communes sont définies comme suit.

Définition 2.7 *[GHPR03] Étant donnés deux graphes non-orientés $G_1 = (U, E_1)$ et $G_2 = (U, E_2)$, une composante connexe commune entre G_1 et G_2 est un ensemble maximal $X \subseteq U$ tel que $G_1[X]$ et $G_2[X]$ sont connexes.*

Notation 2.1 *On note par $i \rightsquigarrow^D j$ un chemin dans D partant du sommet i vers un sommet j. Quand ij est un arc de D, le chemin $i \rightsquigarrow^D j$ est noté $i \rightarrow^D j$.*

Notation 2.2 *On note par $CCC(D^*, G, i \rightsquigarrow^D j)$ la composante connexe commune entre D^* et G qui contient tous les sommets du chemin $i \rightsquigarrow^D j$, si une telle composante existe. Si une telle composante n'existe pas, on note $CCC(D^*, G, i \rightsquigarrow^D j) = \emptyset$.*

Notation 2.3 *On note par S_i^+ l'ensemble de sommets j dans D tel qu'il existe un chemin $i \rightsquigarrow^D j$. On note par S_i^- l'ensemble de sommets j dans D tel qu'il existe un chemin $j \rightsquigarrow^D i$.*

Définition 2.8 *Soit $r \in V$. Le sommet r est dit* pont *de $i \rightsquigarrow^D j$* par rapport *à G, s'il n'y a aucune composante connexe commune entre $D^*[V - \{r\}]$ et $G[V - \{r\}]$ contenant tous les sommets du chemin $i \rightsquigarrow^D j$, c'est-à-dire $CCC(D^*[V - \{r\}], G[V - \{r\}], i \rightsquigarrow^D j) = \emptyset$.*

2.6.1 L'heuristique ALGOH.

L'idée de cette heuristique est de construire un chemin (D, G)-consistant de manière progressive, en prolongeant l'arc xy donné en argument. Nous allons d'abord définir un *ensemble couvrant d'un chemin*. Ensuite, nous présentons l'algorithme ALGOH qui calcule les ensembles couvrants et les utilise pour réduire les graphes D et G en supprimant des sommets qui ne peuvent pas prolonger un chemin (D, G)-consistant.

Définition 2.9 L'ensemble couvrant *d'un chemin $i \rightsquigarrow^D j$, noté* COVERSET*(D,G, $i \rightsquigarrow^D j$), est l'ensemble X vérifiant les conditions suivantes :*

1. *$V(i \rightsquigarrow^D j) \subseteq X \subseteq S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^D j)$;*
2. *$D^*[X]$ et $G[X]$ sont connexes ;*
3. *Si r est un pont de $i \rightsquigarrow^{D[X]} j$ par rapport à $G[X]$ alors $X \subseteq S_r^- \cup S_r^+ \cup \{r\}$;*
4. *X est maximal (par rapport à l'inclusion) avec les conditions 1., 2. et 3.*

Si, pour un chemin $i \rightsquigarrow^D j$, il n'y a aucun ensemble X satisfaisant les conditions 1., 2. *et* 3., *alors par convention* COVERSET*(D,G, $i \rightsquigarrow^D j$)$= \emptyset$.

L'ensemble couvrant d'un chemin donné est unique, et peut se calculer en un temps polynomial.

Lemme 2.1 (Héritage des ponts) *Soit $i \rightsquigarrow^D j$ un chemin dans D. Soient $X, Y \subseteq V(D)$ deux ensembles de sommets tels que $V(i \rightsquigarrow^D j) \subseteq X \subseteq Y$ et $D^*[X], G[X], D^*[Y]$ et $G[Y]$ sont connexes. Si r est un pont de $i \rightsquigarrow^{D[Y]} j$ par rapport à $G[Y]$, alors r est aussi un pont de $i \rightsquigarrow^{D[X]} j$ par rapport à $G[X]$. On dit que $i \rightsquigarrow^{D[X]} j$* hérite *les ponts de $i \rightsquigarrow^{D[Y]} j$.*

Preuve. Supposons par contradiction que le sommet r n'est pas un pont de $i \rightsquigarrow^{D[X]} j$ par rapport à $G[X]$. Nous posons $S = CCC(D^*[X - \{r\}], G[X -$

$\{r\}], i \rightsquigarrow^{D[X]} j)$. Donc, S vérifie l'inégalité suivante : $S \supseteq V(i \rightsquigarrow^{D} j)$. Le fait que $X \subseteq Y$ implique que $S \subseteq CCC(D^*[Y - \{r\}], G[Y - \{r\}], i \rightsquigarrow^{D[Y]} j)$. Par conséquent, $CCC(D^*[Y - \{r\}], G[Y - \{r\}], i \rightsquigarrow^{D[Y]} j) \supseteq V(i \rightsquigarrow^{D} j)$. Mais cela contredit le fait que r est un pont de $i \rightsquigarrow^{D[Y]} j$ par rapport à $G[Y]$. □

Nous prouvons maintenant, en utilisant le Lemme 2.1, l'unicité de l'ensemble couvrant d'un chemin donné.

Lemme 2.2 *L'ensemble couvrant d'un chemin donné est bien défini.*

Preuve. Nous montrons que pour tout chemin $i \rightsquigarrow^{D} j$, si les conditions 1., 2. et 3 de la Définition 2.9 sont satisfaites, alors il existe un unique ensemble qui satisfait la condition 4.

Supposons par contradiction qu'il existe deux ensembles distincts U_1 et U_2 satisfaisant toutes les conditions de la Définition 2.9. Soit $S = U_1 \cup U_2$. On va montrer que S satisfait les trois premières conditions de la Définition 2.9.

1. On a $V(i \rightsquigarrow^{D} j) \subseteq U_k \subseteq S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^{D} j)$ pour tout $k \in \{1,2\}$. Donc, $V(i \rightsquigarrow^{D} j) \subseteq S \subseteq S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^{D} j)$.
2. Les sous-graphes $D^*[S]$ et $G[S]$ sont connexes, parce que : (i) $D^*[U_k]$ et $G[U_k]$ sont connexes pour tout $k \in \{1,2\}$ et (ii) $V(i \rightsquigarrow^{D} j) \subseteq U_1 \cap U_2$.
3. Si r est un pont de $i \rightsquigarrow^{D[S]} j$ par rapport à $G[S]$, alors le Lemme 2.1 implique que r est aussi un pont de $i \rightsquigarrow^{D[U_k]} j$ par rapport à $G[U_k]$, pour tout $k \in \{1,2\}$. Donc $U_k \subseteq S_r^- \cup S_r^+ \cup \{r\}$, pour tout $k \in \{1,2\}$. Par conséquent, $S \subseteq S_r^- \cup S_r^+ \cup \{r\}$.

Donc l'ensemble S satisfait les trois premières conditions de la Définition 2.9. Par ailleurs, $U_1 \neq U_2$, donc on a $S \supset U_1$. Cela contredit la maximalité de U_1. □

Nous proposons un algorithme, appelé GETCOVERSET (l'Algorithme 1), pour calculer l'ensemble couvrant d'un chemin donné. Cet algorithme utilise l'algorithme GENPARTREFINEMENT, présenté dans l'article [GHPR03], pour calculer les composantes connexes communes entre deux graphes. Un exemple de calcul d'un ensemble couvrant est présenté dans la Figure 2.6.

Complexité de l'algorithme GETCOVERSET. Soient $n = |V|$ et $m = |A(D)| + |E(G)|$. Nous montrons dans le lemme suivant que l'algorithme GETCOVERSET calcule correctement l'ensemble couvrant d'un chemin donné en un temps polynomial en n et m.

Lemme 2.3 *L'algorithme* GETCOVERSET *calcule correctement l'ensemble couvrant d'un chemin donné en un temps de* $\mathcal{O}(n^2 \log n + nm \log^2 n)$.

Preuve. Soit S l'ensemble calculé par l'algorithme GETCOVERSET pour un chemin donné $i \rightsquigarrow^{D} j$. On pose $C = \text{COVERSET}(D, G, i \rightsquigarrow^{D} j)$. Nous allons montrer que $S = C$.

Dans un premier temps, nous montrons que $S \subseteq C$. Nous avons deux possibilités pour l'ensemble S.

Algorithme 1 GETCOVERSET$(D, G, i \rightsquigarrow^D j)$

Entrées : Un DAG $D = (V, A(D))$, un graphe non-orienté $G = (V, E(G))$, un chemin $i \rightsquigarrow^D j$.
But : Calcul de l'ensemble couvrant de $i \rightsquigarrow^D j$.

1: $S := S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^D j)$;
2: $S := CCC(D^*[S], G[S], i \rightsquigarrow^{D[S]} j)$; $STOP := faux$;
3: **tantque** (($STOP = faux$) et ($S \neq \emptyset$)) **faire**
4: $S_{tmp} := S$; /* Les chemins $i \rightsquigarrow^{D[S]} j$ et $i \rightsquigarrow^D j$ sont identiques */
5: **pour tout** pont r de $i \rightsquigarrow^D j$ par rapport à G **faire**
6: $S_{tmp} := S_{tmp} \cap (\{r\} \cup S_r^- \cup S_r^+)$;
7: **fin pour**
8: **si** $(S = S_{tmp})$ **alors**
9: $STOP := vrai$;
10: **sinon**
11: $S := S_{tmp}$; $S := CCC(D^*[S], G[S], i \rightsquigarrow^{D[S]} j)$;
12: **finsi**
13: **fin tantque**
14: **return** S

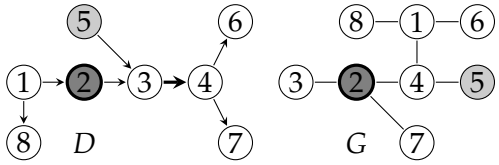


FIGURE 2.6 – ***Calcul de* CoverSet**$(D, G, 3 \rightarrow^D 4)$ ***par l'algorithme* GETCOVERSET**. *L'ensemble S est initialisé à $S_3^- \cup S_4^+ \cup V(3 \rightarrow^D 4) = \{1,2,5\} \cup \{6,7\} \cup \{3,4\} = \{1,2,3,4,5,6,7\}$. Le sommet 2 est un pont de $3 \rightarrow^D 4$ par rapport à $G[S]$. Donc dans l'algorithme* GETCOVERSET *(ligne 6) on retire de S tous les sommets qui ne peuvent pas appartenir à un chemin dans D passant par le sommet 2 (dans cet exemple, on retire le sommet 5). Donc $S = \{1,2,3,4,6,7\}$, et la prochaine exécution de la boucle "tantque" (ligne 3) ne va pas modifier l'ensemble S. La sortie de l'algorithme* GETCOVERSET *est donc l'ensemble $\{1,2,3,4,6,7\}$ qui est l'ensemble couvrant de $3 \rightarrow^D 4$.*

- $S = \emptyset$. Évidemment, pour ce cas nous avons $S \subseteq C$.
- $S \neq \emptyset$. Donc S est égal à un certain S_{tmp} calculé par la boucle "pour" (lignes 5-7). Le Lemme 2.1 implique que S satisfait la condition 3. de la Définition 2.9. Par ailleurs, les instructions dans les lignes 1, 2 et 11 impliquent que S satisfait également les deux premières conditions de la Définition 2.9. Donc, on en déduit, par maximalité de C, que $S \subseteq C$.

Passons maintenant à la deuxième partie de la preuve. On veut donc montrer que $C \subseteq S$. L'ensemble S est initialisé à la valeur $S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^D j)$. Donc, avant la boucle "pour", on a $C \subseteq S$. La deuxième condition de la Définition 2.9 implique qu'aucun sommet supprimé aux lignes 2 et 11 ne peut appartenir à C. En ce qui concerne la ligne 5, le Lemme 2.1 implique que tout pont r du chemin $i \rightsquigarrow^{D[S]} j$ par rapport à $G[S]$ est aussi un pont du chemin $i \rightsquigarrow^{D[C]} j$ par rapport à $G[C]$. Donc, les sommets supprimés à la ligne 6 ne peuvent pas non plus appartenir à C. Par consé-

quent, $C \subseteq S$. Donc l'algorithme GETCOVERSET calcule correctement l'ensemble couvrant d'un chemin donné.

Pour calculer $CCC(D[S]^*, G[S], i \rightsquigarrow^{D[S]} j)$ (lignes 2 et 11), nous utilisons l'algorithme GENPARTREFINEMENT présenté dans [GHPR03]. Cet algorithme calcule les composantes connexes communes entre $D^* = (V, E(D^*))$ et $G = (V, E(G))$ en un temps de $\mathcal{O}(n \log n + m \log^2 n)$, avec $n = |V|$ et $m = |E(D^*)| + |E(G)|$. Donc la complexité de l'algorithme GETCOVERSET est de $\mathcal{O}(n^2 \log n + nm \log^2 n)$. En effet, le calcul des ponts peut être fait une seule fois dès le début de la boucle "tantque". Le Lemme 2.1 implique que les ponts seront hérités par tous les sous-ensemble de S. Donc, la complexité en $\mathcal{O}(n^2 \log n + nm \log^2 n)$ provient des instructions à la ligne 11 et les instructions de la boucle "tantque".

□

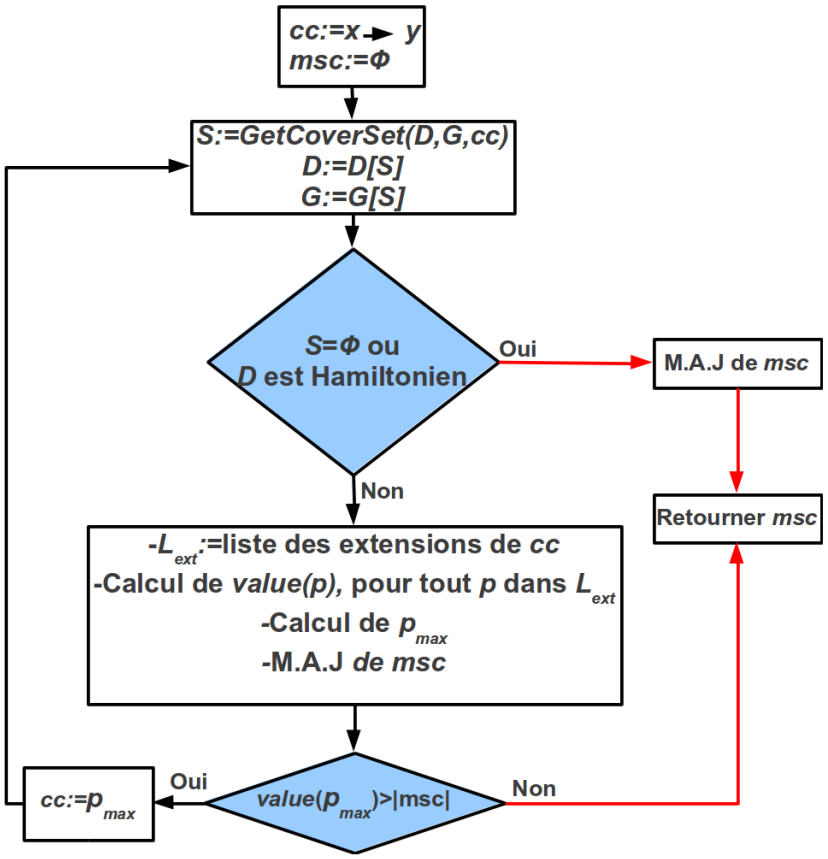


FIGURE 2.7 – *Diagramme illustrant l'heuristique* ALGOH. *Pour tout* $p \in L_{ext}$, $value(p)$ *est la longueur du plus long chemin dans le sous-graphe* $D[\text{COVERSET}(D, G, p)]$.

Nous voulons maintenant calculer un chemin (D, G)-consistant passant par un arc xy, et nous souhaitons que le chemin calculé soit le plus long possible. Pour atteindre cet objectif, nous proposons une heuristique appelée ALGOH (l'Algorithme 2). Un digramme illustrant cet algorithme est présenté dans la Figure 2.7. L'algorithme ALGOH initialise une variable cc (*chemin courant*) à un chemin de longueur 1 contenant l'arc xy ($cc := x \rightarrow^D y$). Le chemin cc est noté $s \rightsquigarrow^D t$ où s est le premier sommet du chemin (dit *source*) et t est son dernier sommet (dit *terminal*).

Dans la ligne 7 on réduit les graphes D et G aux sous-graphes induits

Algorithme 2 ALGOH(D, G, xy)

Entrées : Un DAG $D = (V, A(D))$, un graphe non-orienté $G = (V, E(G))$, un arc $xy \in A(D)$.

But : Calcul d'un ensemble $S \subseteq V$ tels que $D[S]$ est un chemin (D, G)-consistant passant par xy, ou $S = \emptyset$.

1: /* msc : meilleure solution courante ; cc : chemin courant */
2: /* L_{ext} : la liste des chemins dans D qui sont obtenus en prolongeant le chemin cc, dans D, par un seul sommet */
3: /* DEC : les sous-graphes de D induits par les ensembles couvrants des chemins dans L_{ext} */
4: /* HEC : les sous-graphes Hamiltoniens induits par les ensembles couvrants des chemins dans L_{ext} */
/* s (resp. t) est la source (resp. le terminal) du chemin courant cc */
5: $msc := \emptyset$; $cc := x \rightarrow^D y$; $s := x$; $t := y$; $STOP := faux$;
6: **tantque** ($STOP = faux$) **faire**
7: $S :=$ GETCOVERSET(D, G, cc) ; $D := D[S]$; $G := G[S]$;
8: **si** ($S = \emptyset$ ou D est Hamiltonien) **alors**
9: $STOP := vrai$;
10: **si** $|msc| > |S|$ **alors** $S := msc$ **finsi**
11: **sinon**
12: $L_{ext} := \emptyset$; /* extension du chemin courant $cc = s \rightsquigarrow^D t$ */
13: **pour** *tout v qui est un prédécesseur de s ou successeur de t* **faire**
14: $p =$ EXTEND(cc, v) ; /* extension de cc par v */
15: $L_{ext} := L_{ext} \cup \{p\}$;
16: **fin pour**
17: $DEC := \{D[$COVERSET$(D, G, p)] : p \in L_{ext}\}$;
18: $HEC := \{d \in DEC : d$ est Hamiltonien$\}$;
19: Soit $h_{max} \in HEC$ t.q. $|V(h_{max})| = max\{|V(h)| : h \in HEC\}$
20: **si** $|msc| < |V(h_{max})|$ **alors** $msc := V(h_{max})$ **finsi**
21: Soit $p_{max} = s_{max} \rightsquigarrow^D t_{max}$ t.q.
$value(p_{max}) = max\{value(p) : p \in L_{ext}\}$;
22: **si** $|msc| \geq value(p_{max})$ **alors**
23: /* il n'existe aucun chemin (D, G)-consistant passant par xy et de longueur supérieure à msc*/
24: $S := msc$; $STOP := vrai$;
25: **sinon**
26: $cc := p_{max}$; /* choix de l'extension la plus prometteuse */
27: $s := s_{max}$; $t := t_{max}$
28: **finsi**
29: **finsi**
30: **fin tantque**
31: **return** S

par l'ensemble couvrant du chemin cc. Après cette réduction, on distingue les cas suivants.

1. $V(D) = \emptyset$. Dans ce cas, l'algorithme ALGOH ne trouve aucun chemin (D, G)-consistant passant par xy.
2. D est Hamiltonien. Dans ce cas l'algorithme retourne la meilleure

solution courante (l'ensemble *msc*). Le chemin (D, G)-consistant correspondant est $D[msc]$.

3. $V(D) \neq \emptyset$ et D n'est pas Hamiltonien. Dans ce cas, on commence d'abord par calculer la liste L_{ext} (lignes 12-16) des chemins obtenus en prolongeant (à l'aide de la fonction EXTEND) le chemin courant $cc = s \rightsquigarrow^{D} t$ par un seul sommet (un prédécesseur de s ou un successeur de t). A la ligne 17, on calcule l'ensemble DEC des sous-graphes induits dans D par les chemins de la liste L_{ext}. Les graphes de l'ensemble HEC, $HEC \subseteq DEC$, sont des graphes Hamiltoniens et par conséquent sont des chemins (D, G)-consistants passant par xy ; ils sont donc utilisés pour mettre à jour la meilleure solution courante *msc*. Le calcul de ces graphes est effectué à la ligne 18, et la mise à jour de la solution courante est effectuée aux lignes 19-20. Maintenant, pour choisir une extension parmi celles dans L_{ext}, on utilise la fonction *value* définie comme suit : pour tout $p \in L_{ext}$, $value(p)$ est la longueur du plus long chemin dans le sous-graphe $D[\text{CoverSet}(D, G, p)]$. L'extension "la plus prometteuse", p_{max}, (calculée à la ligne 21) est celle qui correspond à un maximum pour la fonction *value*. Ici également on distingue deux sous-cas :

   3.1 La valeur de l'extension la plus prometteuse est supérieure à la longueur de la meilleure solution courante (lignes 25-28), alors $cc := p_{max}$. Ensuite, on passe à la prochaine itération de la boucle "tantque". C'est à ce moment là que l'heuristique risque de perdre la solution optimale. Car nous pouvons choisir une extension p_{max} qui a une très grande valeur, alors qu'il n' y a aucun chemin (D, G)-consistant qui contient p_{max} comme sous-chemin.

   3.2 La valeur de l'extension la plus prometteuse est inférieure à la longueur de la meilleure solution courante. Dans ce cas, il n' y aura aucun chemin (D, G)-consistant passant par xy dont la longueur est supérieure à longueur de la meilleure solution courante (lignes 22-24). Donc l'algorithme s'arrête et retourne la meilleure solution courante.

Complexité de l'algorithme ALGOH. Nous notons par Δ le degré maximum du graphe D, c'est-à-dire, $\Delta = max_{u \in V} \ (\Delta^+(u) + \Delta^-(u))$ tel que $\Delta^+(u)$ (resp. $\Delta^-(u)$) est le degré sortant (resp. entrant) de u. On note par L la longueur du plus long chemin (D, G)-consistant passant par xy. Nous rappelons que le plus long chemin dans le DAG $D = (V, A)$ (et aussi le plus long chemin dans D passant par l'arc xy) peut se calculer facilement en un temps de $\mathcal{O}(|V| + |A|)$, en utilisant la programmation dynamique. La boucle "tantque" (ligne 6) est exécutée au plus L fois. L'instruction la plus coûteuse, en temps d'exécution, est celle qui se trouve à la ligne 17. Cette instruction fait au plus 2Δ appels à l'algorithme GETCOVERSET. Donc, la complexité globale de l'algorithme ALGOH est de $\mathcal{O}(\Delta L(n^2 \log n + nm \log^2 n))$, avec $n = |V|$ et $m = |A(D)| + |E(G)|$.

2.6.2 L'algorithme exact AlgoBB.

Dans le but de mesurer la performance de notre heuristique, nous proposons dans cette section un algorithme exact pour résoudre le problème One-to-One SkewGraM. Ensuite, nous comparons l'heuristique avec l'algorithme exact dans la Section 2.8.

L'algorithme AlgoBB prend comme arguments un DAG $D = (V, A)$, un graphe non-orienté $G = (V, E)$ tels que D^* et G sont connexes, ainsi qu'un arc xy de D. La sortie de de l'algorithme est le plus long chemin (D, G)-consistant passant par xy. Donc pour résoudre le problème One-to-One SkewGraM, il suffit d'appliquer l'algorithme AlgoBB pour tous les arcs de D, et ensuite de garder le chemin le plus long parmi les chemins calculés.

L'algorithme AlgoBB est basé sur la méthode de séparation et évaluation (branch-and-bound), présentée dans le Chapitre 1.

L'arbre TS des sous-solutions est construit comme suit. La racine de l'arbre est associée à l'arc xy. Chaque sommet s est associé à un chemin, noté $p(s)$, prolongeant l'arc xy. A la fin de la construction de l'arbre TS, ses feuilles sont associées aux chemins (D, G)-consistants passant par xy. Donc le plus long chemin (D, G)-consistant passant par xy est le plus long chemin $i \rightsquigarrow^D j$ associé à une feuille.

Séparation. Nous explorons un sommet s, avec $p(s) = v_l \rightsquigarrow^D v_m$, comme suit. Pour tout v_k qui est un prédécesseur (resp. successeur) dans D de v_l (resp. v_m), on ajoute dans TS un fils de s associé au chemin $v_k.p(s)$ (resp. $p(s).v_k$).

Pour tout $s \in V(TS)$, on rappelle que $value(p(s))$ est la longueur du plus long chemin dans le graphe $D[\text{CoverSet}(D, G, p(s))]$. Nous utilisons la fonction notée $BBvalue$ pour évaluer les sommets de TS. La fonction $BBvalue$ est définie comme suit :

- Si s n'est pas encore exploré, alors $BBvalue(s) = value(p(s))$.
- Si s a été déjà exploré à un moment donné de la construction de TS, alors on distingue deux cas.
  - Le chemin $p(s)$ est (D, G)-consistant. Dans ce cas $BBvalue(s)$ est égale à la longueur du chemin $p(s)$.
  - Le chemin $p(s)$ n'est pas (D, G)-consistant. Dans ce cas $BBvalue(s) = 0$.

En utilisant la fonction $BBvalue$, on définit les opérations *évaluation* (*bounding*) et *élagage* (*pruning*) de la méthode branch-and-bound comme suit.

Évaluation (Règle 1). Soit $\{s_1, s_2, \ldots, s_k\}$ l'ensemble des sommets à explorer. Nous choisissons le sommet s^* tel que $BBvalue(s^*) = \max\{BBvalue(s_i) : 1 \leq i \leq k\}$. Dans le cas où il y a plusieurs sommets s_i dont $BBvalue(s_i)$ est maximum, on en choisit arbitrairement un parmi eux.

Élagage (Règle 2). Soit s_{max} un sommet de TS satisfaisant les conditions suivantes : (*i*) s_{max} a été déjà exploré, et (*ii*) pour tout sommet s de TS, si s a été déjà exploré, alors $BBvalue(s_{max}) \geq BBvalue(s)$. On supprime

dans TS toutes les feuilles s tel que $BBvalue(s) \leq BBvalue(s_{max})$. Cette suppression est appliquée récursivement sur les sommets (sauf s_{max}) qui deviennent feuilles après la suppression de leurs fils.

Par définition de la fonction $BBvalue$, on en déduit la propriété suivante.

Propriété 2.5 *Soit $\mathcal{P}_{exp}$ l'ensemble des chemins associés aux sommets de TS qui ont été déjà explorés. On note par $\mathcal{P}_{exp-c}$, $\mathcal{P}_{exp-c} \subseteq \mathcal{P}_{exp}$, l'ensemble des chemins dans $\mathcal{P}_{exp}$ qui sont (D, G)-consistants. Soit s_{max} le sommet choisi au moment de l'opération d'élagage ($p(s_{max}) \in \mathcal{P}_{exp}$). On distingue les deux cas suivants.*

- *$BBvalue(s_{max}) > 0$. Dans ce cas on a (i) $p(s_{max})$ est un chemin (D, G)-consistant et donc $BBvalue(s_{max})$ est égal la longueur du chemin $p(s_{max})$ et (ii) il n'y a aucun chemin dans $\mathcal{P}_{exp-c}$ plus long que $p(s_{max})$.*
- *$BBvalue(s_{max}) = 0$. Dans ce cas $\mathcal{P}_{exp-c} = \emptyset$.*

L'algorithme AlgoBB est défini comme suit.

1. Créer la racine de l'arbre TS associée à l'arc xy.
2. **Tant que** (TS contient un sommet s à explorer) **faire**
   (a) évaluer TS (**Règle 1**) ;
   (b) séparer TS par rapport au sommet s ;
   (c) élaguer TS (**Règle 2**) ;
3. Soit s_{max} une feuille de TS vérifiant : $BBvalue(s_{max}) \geq BBvalue(s)$, pour toute feuille s de TS. Soit $p_{max} = p(s_{max})$.
4. Si p_{max} est (D, G)-consistant, alors p_{max} est le plus long chemin (D, G)-consistant passant par xy. Sinon, il n'y a aucun chemin (D, G)-consistant passant par xy.

Nous montrons que l'algorithme AlgoBB calcule le plus long chemin (D, G)-consistant passant par un arc xy donné en argument. Par conséquent, l'algorithme AlgoBB (appliqué avec tous les arcs de D) calcule une solution exacte du problème One-to-One SkewGraM

Théorème 2.4 *L'algorithme AlgoBB résout le problème One-to-One SkewGraM.*

Preuve. Pour explorer un sommet s, nous considérons tous les sommets qui peuvent prolonger le chemin $p(s)$. Donc, pour montrer l'exactitude de l'algorithme, il suffit de montrer qu'on ne perd aucune solution optimale en appliquant la règle 2 à l'étape 2.(c) de l'algorithme. En effet, soit s un sommet supprimé par la règle 2. On distingue les cas suivants :

- Le sommet était une feuille de TS obtenue immédiatement après l'application de la règle 1 (s n'a pas été exploré). Donc $BBvalue(s) = value(p(s))$. Par conséquent, $value(p(s)) \leq BBvalue(s_{max})$ et donc en prolongeant le chemin $p(s)$ nous ne pourrons pas obtenir un chemin (D, G)-consistant plus long que le chemin courant $p(s_{max})$.
- Le sommet était un sommet interne dans TS. Donc par définition de la fonction $BBvalue$, $BBvalue(s)$ est la longueur du chemin $p(s)$ s'il est (D, G)-consistant, et sinon $BBvalue(s) = 0$. Le sommet s a été supprimé par la règle 2, donc on en déduit que tous les fils de s ont été supprimés avant. Par conséquent, on ne peut pas obtenir un chemin (D, G)-consistant, plus long que $p(s_{max})$, en prolongeant $p(s)$. Pour finir la preuve, $BBvalue(s) \leq BBvalue(s_{max})$, donc si le

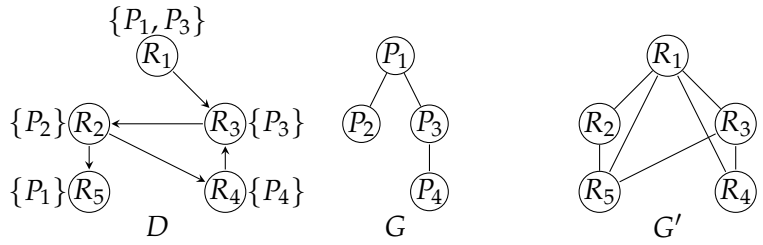


FIGURE 2.8 – **Exemple d'une bij-transformation**. *Nous transformons une instance du problème* SKEWGRAM *vers une instance du problème* ONE-TO-ONE SKEWGRAM. *Ici,* $f(R_1) = \{P_1, P_3\}$, $f(R_2) = \{P_2\}$, $f(R_3) = \{P_3\}$, $f(R_4) = \{P_4\}$ *et* $f(R_5) = \{P_1\}$.

chemin $p(s)$ lui même est (D, G)-consistant, alors il n'est pas plus long que $p(s_{max})$.

□

2.7 TRANSFORMATION D'UNE INSTANCE DE SKEWGRAM VERS UNE INSTANCE DE ONE-TO-ONE SKEWGRAM

Nous rappelons que le problème ONE-TO-ONE SKEWGRAM est le cas particulier du problème SKEWGRAM pour lequel la fonction de correspondance f est une bijection entre $V(D)$ et l'ensemble $\{\{v\} : v \in V(G)\}$ (voir la Section 2.3).

Dans cette section, nous présentons une méthode, appelée *bij-transformation*, permettant de transformer une instance du problème SKEWGRAM, dans son cas général, vers une instance du problème ONE-TO-ONE SKEWGRAM, pour lequel nous avons proposé deux algorithmes (voir la Section 2.6). La transformation que nous proposons n'est pas exacte, c'est-à-dire, une solution exacte de l'instance ONE-TO-ONE SKEWGRAM (obtenue par exemple par l'algorithme ALGOBB) ne donne pas nécessairement une solution exacte du problème SKEWGRAM. Théoriquement, dans certains cas, une solution exacte de ONE-TO-ONE SKEWGRAM ne correspond à aucune solution (même approchée) du problème SKEWGRAM. En revanche, nous avons appliqué efficacement les bij-transformations sur des données réelles (Section 2.8).

Définition 2.10 (bij-transformation) *Soient* $D = (V(D), A(D))$, $G = (V(G), E(G))$ *et* f *une instance du problème* SKEWGRAM. *Une* bij-transformation *de l'instance* (D, G, f) *est une instance* (D', G', f') *du problème* ONE-TO-ONE SKEWGRAM *construite comme suit.*

- $D' = D$;
- $G' = (V(D), E')$ *tel que* $(u', v') \in E'$ *si et seulement si il existe dans* G *deux sommets* $u \in f(u')$, $v \in f(v')$ *tels que* $(u, v) \in E(G)$.
- f' *est l'identité.*

Un exemple d'une bij-transformation est présenté dans la la Figure 2.8.

Le théorème suivant donne une condition suffisante pour obtenir une solution du problème SKEWGRAM à partir d'une solution d'une instance de ONE-TO-ONE SKEWGRAM construite à l'aide d'une bij-transformation.

Théorème 2.5 *Soient $D = (V, A)$, $G = (V, E)$ et $f : V(D) \to 2^{V(G)}$ une fonction qui établit une correspondance entre les sommets de D et ceux de G. Soit G' le graphe non-orienté par une bij-transformation de l'instance (D, G, f) du problème* SkewGraM. *Si $G[f(v)]$ est connexe pour tout $v \in V(D)$, alors tout chemin (D, G')-consistant est aussi un chemin (D, G, f)-consistant.*

Preuve. Soit $i \rightsquigarrow^D j$ un chemin (D, G')-consistant. On pose $X = \bigcup\{f(u') : u' \in V(i \rightsquigarrow^D j)\}$. On va montrer que $G[X]$ est connexe. D'abord, pour tout $u' \in V(i \rightsquigarrow^D j)$, le sous-graphe $G[f(u')]$ est connexe (par hypothèse). De plus, les sous-graphes $G[f(u')]$ sont connectés les uns aux autres. En effet, soit (u', v') une arête de $G'[V(i \rightsquigarrow^D j)]$, alors par construction de G', on en déduit qu'il existe une arête (u, v) de G telle que $u \in f(u')$ et $v \in f(v')$. Cette arête assure la connexité du graphe $G[f(u') \cup f(v')]$ parce que les deux sous-graphes $G[f(u')]$ et $G[f(v')]$ sont connexes. En appliquant cette règle pour toutes les arêtes de $G'[V(i \rightsquigarrow^D j)]$, on en déduit que $G[X]$ est connexe. Donc le chemin $i \rightsquigarrow^D j$ est (D, G, f)-consistant. □

Il est important de noter que l'hypothèse "$G[f(v)]$ est connexe pour tout $v \in V(D)$" est une condition suffisante, mais elle n'est pas nécessaire en pratique. En effet, en appliquant notre méthode sur des données réelles (données biologiques), nous avons obtenu des solutions efficaces pour le problème SkewGraM, malgré le fait que cette hypothèse de connexité n'était pas toujours vérifiée (voir la Section 2.8).

2.8 Résultats expérimentaux

Étant donné un graphe non-orienté, la distribution des degrés (notée $P(k)$) est la probabilité qu'un sommet ait le degré k. Un graphe non-orienté est dit *aléatoire* si les arêtes sont distribuées de manière aléatoire sur les sommets (c'est-à-dire chaque paire de sommets présente autant de chances que toute autre paire de sommets d'être reliée par une arête). Ce type de graphes sont dits graphes d'Erdös-Rényi [ER59]. Dans un graphe aléatoire, la distribution $P(k)$ suit une loi de *Poisson*. Dans ce cas $P(k)$ est maximum pour une valeur moyenne k_{moyen}, et décroît quand $k < k_{moyen}$ ou $k > k_{moyen}$.

A. Barabási et al. [Bar09] ont montré que de nombreux réseaux réels (par exemple réseaux biologiques, réseaux sociaux, réseaux d'internet) ne sont pas aléatoires. La distribution de ces réseaux diminue, en suivant une loi de *puissance*, lorsque le degré augmente, c'est-à-dire $P(k) = k^{-\gamma}$. Ces réseaux sont dits *invariants d'échelle* ou *scale-free*. Cela veut dire que la distribution observée reste de forme identique quelle que soit l'échelle considérée, par exemple s'il y a 5 fois plus de sommets de degré 1 que de degré 5, alors il y aura à peu près 5 fois plus de sommets degré 10 que de degré 20 [Bro08]. Au contraire, dans les réseaux aléatoires, la distribution des degrés n'est pas uniforme, il existe très peu de sommets qui sont hautement connectés et un grand nombre de sommets de degré faible. Les sommets hautement connectés sont dénommés *hubs* et maintiennent la connexité du réseau.

Dans le but de montrer l'efficacité de notre heuristique AlgoH, nous l'avons appliquée d'abord sur les graphes aléatoires (Erdös-Rényi) et les

graphes scale-free. Nous avons comparé les solutions obtenues avec celles obtenues par l'algorithme exact AlgoBB. Ensuite, nous avons appliqué AlgoH pour comparer des réseaux biologiques hétérogènes.

2.8.1 Données simulées

Soit $|\text{AlgoBB}|$ (resp. $|\text{AlgoH}|$) le nombre de sommets d'une solution (c'est-à-dire un chemin (D,G)-consistant) trouvée par l'algorithme exact AlgoBB (resp. trouvée par l'heuristique AlgoH). Nous avons mesuré la performance de AlgoH en calculant le ratio $\rho = \frac{|\text{AlgoH}|}{|\text{AlgoBB}|}$ pour toute instance (D,G,xy). Par convention, $\rho = 1$ quand l'algorithme exact ne trouve aucun chemin (D,G)-consistant passant par un arc donné.

Génération des graphes aléatoires

Nous avons choisi de varier les deux paramètres suivants : le nombre n de sommets de D et G (dans la plage 20, 30, 40, 50, 60), et la probabilité p d'avoir une arête entre deux sommets donnés (dans la plage : 0.05, 0.1, 0.15, 0.2). Nous avons considéré toutes les combinaisons possibles des paramètres n et p, ce qui conduit donc à 20 configurations. Nous avons généré les graphes non-orientés G en utilisant la méthode d'Erdös-Rényi. Nous avons adapté cette méthode pour construire les DAG aléatoires D, en effectuant une orientation aléatoire des arêtes. Pour un n et un p fixés, nous avons généré 100 couples (D,G). Pour chacun de ces couples, nous avons appliqué AlgoH et AlgoBB pour 5 arcs xy choisis aléatoirement, et ensuite nous avons calculé le ratio ρ pour chacune des 5 instances (D,G,xy). Nous avons calculé le nombre, noté $N_i^{(D,G,p,n)}$, $0 \leq N_i^{(D,G,p,n)} \leq 5$, d'arcs dont $\rho \times 100$ appartient à l'intervalle I_i, avec $1 \leq i \leq 10$ et $I_1 = [0,10[, I_2 = [10,20[, \ldots, I_{10} = [90,100]$. Nous avons obtenu un résultat global en calculant, pour chaque I_i, la valeur $m_i^{(n,p)} = \sum_{(D,G)} N_i^{(D,G,n,p)}$ c'est-à-dire, la somme des $N_i^{(D,G,n,p)}$ pour l'ensemble des 100 couples (D,G) générés aléatoirement, avec n et p fixés.

Génération des graphes scale-free

Dans notre expérimentation, nous avons généré 100 couples (D,G) de 100 sommets, en utilisant l'outil public *Network Graphs for Computer Epidemiologists (NGCE)* [VVCS05]. L'outil *NGCE* construit les graphes scale-free en se basant sur le concept dit *d'attachement préférentiel* : les nouveaux sommets à ajouter à un graphe ont tendance à être connectés aux sommets hautement connectés appelés *hubs*. Le processus de construction d'un graphe scale-free commence par la création d'un graphe aléatoire connexe H_0. Ensuite, des nouveaux sommets sont ajoutés au graphe H_0, sommet par sommet. Un nouveau sommet est connecté avec un sommet i de H_0 avec la probabilité

$$p_i = \frac{k_i}{\sum_j k_j},$$

où k_i est le degré du sommet i dans le graphe H_0.

Nous avons remarqué que les chemins consistants ne sont pas abondants dans les graphes scale-free aléatoires (D,G). C'est donc pour cette

raison que nous avons choisi aléatoirement, pour chaque graphe D, 10 arcs au lieu de 5 comme dans l'expérience précédente. Ensuite, nous avons calculé pour chaque couple (D, G) et pour chaque intervalle I_i, le nombre $N_i^{(D,G,n)}$, $0 \leq N_i^{(D,G,n)} \leq 10$, d'arcs pour lesquels la valeur $\rho \times 100$ appartient à l'intervalle I_i, ainsi que la valeur globale $m_i^n = \sum_{(D,G)} N_i^{(D,G,n)}$.

Performances de l'heuristique AlgoH

Nous avons observé une très bonne performance de notre heuristique AlgoH. En effet, dans le cas des graphes aléatoires nous avons obtenu, pour 16 combinaisons de paramètres n et p, un ratio $\rho = 1$ pour toutes les instances générées. Pour les 4 combinaisons restantes, ainsi que pour les graphes scale-free, $\rho \times 100$ appartient à l'intervalle $[90, 100]$, pour plus de 90% des instances générées (une illustration de ces résultats est proposée dans les Figures 2.9 et 2.10). Cela implique que les solutions trouvées par l'heuristique sont dans la majorité des cas très proches des solutions optimales. En terme de temps d'exécution, nous avons observé que l'heuristique est beaucoup plus rapide que l'algorithme exact AlgoBB. Par exemple, pour les 500 instances du problème générées pour les paramètres $n = 60$ et $p = 0.2$ (la Figure 2.9), l'heuristique AlgoH était 11 fois plus rapide que AlgoBB.

2.8.2 Données réelles biologiques

Dans cette section, nous présentons les résultats que nous avons obtenus en appliquant notre heuristique pour comparer des réseaux biologiques hétérogènes.

Voie métabolique vs réseau PPI de la levure *S. cerevisiae*

Nous rappelons que les voies métaboliques peuvent se modéliser par des graphes orientés (appelé *graphes de réactions*) dont les sommets sont les réactions et il existe un arc d'une réaction R_1 vers une réaction R_2 si R_1 utilise, comme produit, un substrat de R_2 (pour plus de détails, voir la Section 1.1.2 du Chapitre 1). Les graphes de réactions modélisant les voies métaboliques peuvent être considérés acycliques (DAGs) [GHM+02, PRYLZU05]. Nous rappelons également que les réseaux d'interaction protéine-protéine (PPI) peuvent se modéliser par un graphe non-orienté dont les sommets sont les protéines et les arêtes représentent leurs interactions physiques (voir la Section 1.1.1 du Chapitre 1).

Nous avons appliqué notre heuristique pour chercher d'une manière automatique, dans une voie métabolique, une chaîne de réactions (c'est-à-dire un chemin) qui sont catalysées par des protéines en interaction, c'est-à-dire des protéines qui forment un sous-graphe connexe dans le réseau PPI. Ces chemins sont biologiquement significatifs [DW08]. En effet, P. Durek et D. Walther [DW08] ont décomposé le réseau PPI de l'espèce *S. cerevisiae* en clusters fonctionnels et ont observé que les paires de protéines qui catalysent des réactions successives sont en général plus susceptibles d'interagir que toute autre paire de protéines. Ils ont fourni un

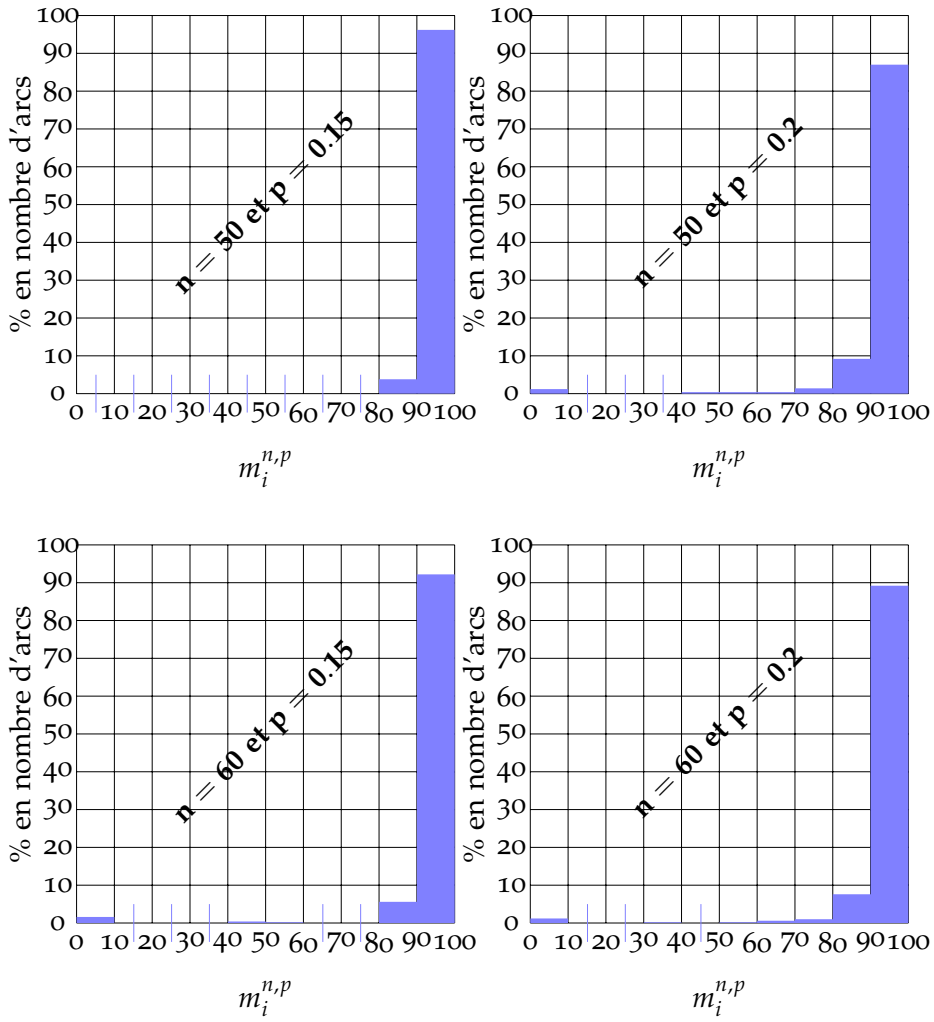


FIGURE 2.9 – ***Performances de l'heuristique* ALGOH*, appliquée sur les graphes aléatoires***. *Nous montrons le pourcentage des arcs pour lesquels la valeur* $\frac{|\text{ALGOH}|}{|\text{ALGOBB}|} \times 100$ *appartient à un intervalle* I_i*, avec* $1 \leq i \leq 10$ *et* $I_1 = [0, 10[, I_2 = [10, 20[, \ldots, I_{10} = [90, 100]$ *(voir la Section 2.8.1). Le paramètre n est le nombre de sommets de D et G. Le paramètre p est la probabilité d'avoir un arc (resp. arête) entre deux sommets de D (resp. G). Pour n et p fixés nous avons généré 100 couples* (D, G) *de graphes aléatoires et* 5 *arcs par couple de graphes.*

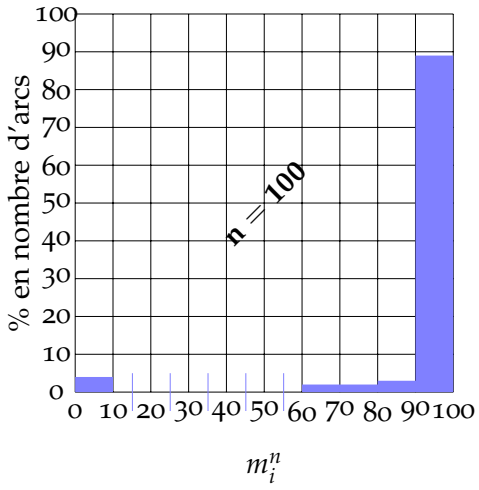


FIGURE 2.10 – ***Performances de l'heuristique* ALGOH*, appliquée sur les graphes aléatoires scale-free.*** *Nous montrons le pourcentage des arcs pour lesquels la valeur* $\frac{|\text{ALGOH}|}{|\text{ALGOBB}|} \times 100$ *appartient à l'intervalle* I_i*, avec* $1 \leq i \leq 10$ *et* $I_1 = [0,10[$, $I_2 = [10,20[, \ldots, I_{10} = [90,100]$ *(voir la Section 2.8.1). Le paramètre* n *est le nombre de sommets de* D *et* G*. Pour* $n = 100$ *nous avons généré 100 couples* (D,G) *de graphes aléatoires scale-free.*

exemple de chemin court (de longueur 6) dans la voie métabolique *"Glycolyse/Gluconéogenèse"* de *S. cerevisiae*, correspondant à un cluster fonctionnel dans le réseau PPI pour la même espèce. Dans le but de comparer nos résultats avec les leurs, nous avons construit le graphe PPI G pour la même espèce *S. cerevisiae*, à partir de la base de données BioGRID [SBR+06] (http ://thebiogrid.org/, version *v*2.0.63). Nous avons également construit la voie métabolique *"Glycolyse/Gluconéogenèse"* de *S. cerevisiae* (graphe D) à partir de la base de données *KEGG* (*pathway sce00010.xml*). Nous avons établi la correspondance entre les sommets de D et G en utilisant les noms des gènes produisant les enzymes (protéines) qui d'une part catalysent les réactions dans D et d'autre part interagissent dans G. Donc la fonction de correspondance f est définie comme suit : $f(i)$ est l'ensemble des protéines qui catalysent la réaction i (voir le Tableau 2.2). Le triplet (D,G,f) est une instance du problème SKEWGRAM. A partir de l'instance (D,G,f), nous avons construit une instance (D,G') du problème ONE-TO-ONE SKEWGRAM, en utilisant une bij-transformation (voir la Définition 2.10). En appliquant notre heuristique ALGOH avec D, G' et l'arc entre les sommets 1 et 23, nous avons calculé automatiquement un chemin (D,G')-consistant de 12 sommets, induisant un sous-graphe connexe de 20 protéines (une illustration est fournie dans les Figures 2.11 et 2.12). Le chemin obtenu inclut le chemin observé dans [DW08].

Voie métabolique vs génome linéaire de la bactérie *E. coli*

Le génome d'un organisme est constitué par l'ensemble de ses gènes. Le génome est modélisé par un graphe non-orienté dont les sommets sont les gènes et il existe une arête entre un gène g_1 et un gène g_2, si g_1 et g_2 sont adjacents dans le génome.

Réaction i	**Enzymes** $(f(i))$
30	YCL040W, YFR053C, YGL253W
28	YKL127W, YMR105C, YMR278W
32	YBR196C, YCL040W, YFR053C, YGL253W
27	YBR196C, YMR169C, YMR170C, YER073W, YOR374W, YPL061W
25	YLR377C
24	YGR240C, YMR205C
23	YDR050C
29	YBR196C
1	YKL060C
21	YGR192C, YJL052W, YJR009C
42	YCR012W
20	YDL021W, YKL152C
19	YGR254W, YHR174W, YMR323W, YOR393W
79	YKR097W
17	YAL038W, YOR347C
12	YBR221C, YER178W, YDL080C, YGR087C, YLR044C
8	YDL080C, YGR087C, YLR044C
13	YBR221C, YER178W
10	YBR145W, YDL168W, YGL256W, YMR083W, YMR303C, YOL086C
4	YMR169C, YMR170C, YER073W, YOR374W, YPL061W
91	YAL054C, YLR153C
14	YNL071W
41	YFL018C, YPL017C
36	YCL040W, YFR053C, YGL253W

Tableau 2.2 – *La correspondance entre les réactions et les enzymes (protéines) dans la voie métabolique "Glycolyse/Gluconéogenèse" de* S. cerevisiae. *Ici,* $f(i)$ *est l'ensemble des protéines qui catalysent la réaction i. Nous avons considéré seulement les réactions catalysées par au moins une protéine. Source : KEGG sceoo010.xml*

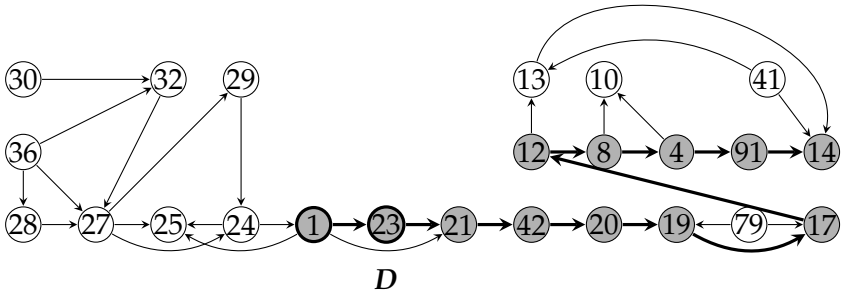


FIGURE 2.11 – *Le graphe D est le graphe des réactions de la voie métabolique "Glycolyse/Gluconéogenèse" pour l'espèce* S. cerevisiae. *Nous avons gardé la numérotation des réactions dans le fichier sceoo010.xml correspondant à cette voie dans KEGG. Les sommets pleins dans D correspondent à un chemin consistant calculé par l'heuristique* ALGOH *avec l'arc entre entre les deux sommets* 1 *et* 23 *passé en argument. Le graphe G est montré dans la Figure 2.12.*

Nous avons appliqué ALGOH pour calculer automatiquement, dans une voie métabolique, une chaîne de réactions qui sont catalysées par des pro-

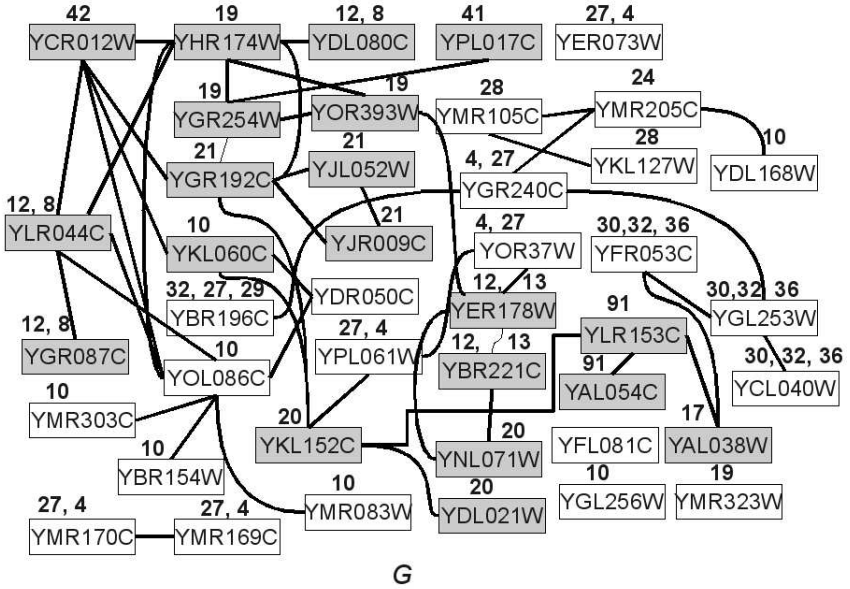


FIGURE 2.12 – *Le graphe G est le graphe d'interactions protéine-protéine dans lequel nous avons considéré seulement les protéines qui catalysent des réactions dans la voie "Glycolyse/Gluconéogenèse" (Figure 2.11). Nous mettons, au dessus de chaque protéine, les numéros des réactions qu'elle catalyse (par exemple la protéine YDL080C catalyse les réactions* 12 *et* 8*). Les sommets pleins dans G (et aussi dans le graphe D montré dans la Figure 2.11) correspondent au résultat de l'heuristique* ALGOH*, quand l'arc entre les sommets* 1 *et* 23 *est passé en argument.*

duits (protéines) de gènes adjacents dans le génome. Nous avons construit la séquence linéaire (graphe G) des gènes de la bactérie *E. coli* à partir de la base de données *NCBI* (http ://www.ncbi.nlm.nih.gov/genome). Nous avons également construit la voie métabolique (graphe D) à partir de *KEGG* (*pathway ecooo550.xml*). Comme dans l'exemple précédent (Section 2.8.2), nous avons établi la correspondance entre les sommets de D et G en utilisant les noms de gènes (voir le Tableau 2.3). La fonction de correspondance f est définie comme suit : $f(i)$ est l'ensemble des gènes qui produisent des protéines catalysant la réaction i. Nous avons construit une instance (D, G') du problème ONE-TO-ONE SKEWGRAM en utilisant une bij-transformation de l'instance (D, G, f) du problème SKEWGRAM. Ensuite, nous avons calculé le plus long chemin (D, G')-consistant en appliquant ALGOH pour tous les arcs de D (voir le chemin orienté dans la Figure 2.13). En supprimant l'orientation du chemin calculé on obtient exactement le chemin trouvé par Boyer et al. [BML$^+$05] (voir la Figure 3.b dans [BML$^+$05]), avec une petite différence représentée par la ligne pointillée dans la Figure 2.13 (génome d'*E. coli*). On note que cette différence est due à un changement de version par rapport aux données utilisées dans [BML$^+$05]. Il est important de rappeler que la méthode présentée par Boyer et al. [BML$^+$05] ne tient pas compte de l'orientation des arcs du graphe de réactions (voir plus de détails sur cette méthode dans la Section 2.2). Donc même si le chemin calculé par notre méthode a la même longueur que celui calculé dans [BML$^+$05], notre chemin est beaucoup plus riche d'informations. En effet, le fait que notre chemin soit orienté, implique qu'on différencie les réactions réversibles de celles qui sont irréversibles.

Réaction i	Gènes ($f(i)$)	Réaction i	Gènes ($f(i)$)	Réaction i	Gènes ($f(i)$)
2	murF	10	ddlB	15	mraY
6	bbjG	12	murD	16	murG
7	murG	13	murC	48	murA
8	mraY	14	murE	49	murB
9	murF				

Tableau 2.3 – *La correspondance entre les gènes et réactions de la voie métabolique "biosynthèse du peptidoglycane" pour la bactérie* E. coli*. Source : KEGG ecooo550.xml*

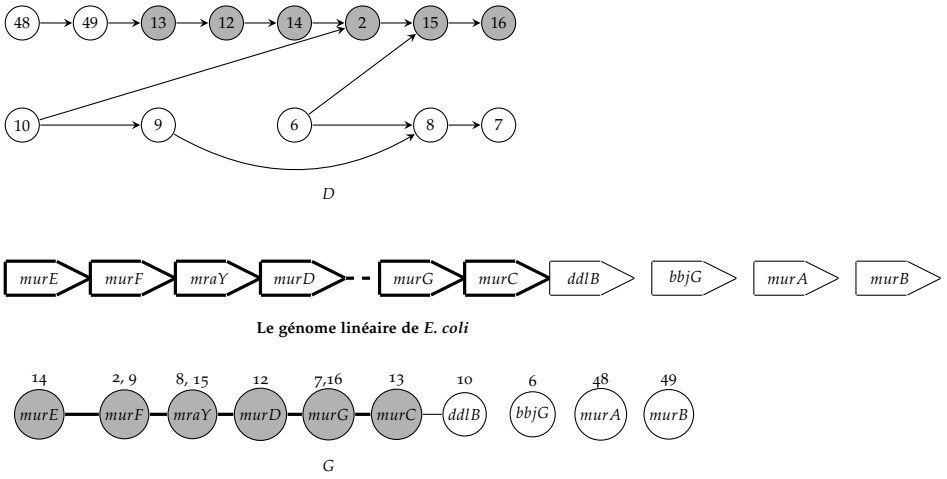


FIGURE 2.13 – *Le graphe D est le graphe des réactions de la voie métabolique "biosynthèse du peptidoglycane" pour E. coli. Nous avons construit le graphe non-orienté G à partir du génome linéaire de la bactérie E. coli. Les sommets de G sont les gènes de E. coli et il existe une arête entre deux gènes dans G si, et seulement si, ils sont adjacents dans le génome. Nous mettons, au dessus de chaque gène, les numéros des réactions qui sont catalysées par un produit du gène (par exemple les réactions* 2 *et* 9 *sont catalysées par un produit du gène murF). Les sommets pleins dans D et G sont les sommets du plus long chemin consistant obtenu en appliquant* ALGOH *pour tous les arcs de D.*

CONCLUSION DU CHAPITRE

Dans ce chapitre nous avons proposé une nouvelle approche pour comparer les réseaux biologiques hétérogènes. Le problème SKEWGRAM résultant de notre modélisation est un problème de recherche de motifs multi-dimensionnels. Dans notre problème, les motifs sont un chemin dans un graphe orienté D et un sous-graphe connexe dans un graphe non-orienté G. Nous avons étudié la complexité du problème ONE-TO-ONE SKEWGRAM qui est un cas particulier de SKEWGRAM dans lequel il y a une bijection entre les sommets de D et ceux de G. Nous avons identifié les instances faciles et les instances difficiles de ce problème. Pour les instances difficiles, nous avons proposé deux algorithmes : une heuristique et un algorithme exact basé sur la méthode branch-and-bound. Ensuite, nous avons proposé une méthode permettant de transformer une instance du problème SKEWGRAM en une instance du problème ONE-TO-ONE SKEWGRAM. Nous avons appliqué nos algorithmes sur des données simulées et sur des données biologiques. Nos résultats expérimentaux ont montré l'efficacité de notre heuristique. Dans ce chapitre nous avons restreint le graphe orienté D à un DAG. Pour traiter le cas général, quand D est arbitraire, nous étu-

dions dans le prochain chapitre une méthode de décomposition du graphe D en DAGs.

Comme perspectives de ce travail, il serait intéressant d'étudier la complexité du problème One-to-One SkewGraM en termes d'approximabilité (pour certaines classes de graphes spécifiques) et en termes de complexité paramétrée. Il serait également intéressant de considérer des versions "inexactes" du problème One-to-One SkewGraM, par exemple, la version où on permet de petites différences entre l'ensemble de sommets de D et ceux de G. Ces différences permettront de prendre en compte l'évolution des réseaux et aussi les erreurs dues à leurs modélisations, comme les *gaps* et les *mismatchs* considérés par les méthodes d'alignement de réseaux. Enfin, nous voulons généraliser notre approche pour pouvoir comparer plus de deux réseaux hétérogènes.

Décomposition d'un graphe orienté en DAGs

3

Les travaux que nous allons présenter dans ce chapitre ont été réalisés en collaboration avec l'équipe AlgoB du Laboratoire d'Informatique Gaspard Monge (LIGM) de l'Université Paris-Est Marne-la-Vallée. Ces travaux ont été publiés dans les actes de la conférence internationale COCOA 2011 (5^{th} Conference on Combinatorial Optimization and Applications) [BFMB+11].

Sommaire

Dans le chapitre précédent, nous avons supposé que le graphe D en entrée du problème One-to-One SkewGraM est acyclique. Nous avons expliqué que cette supposition est due à des raisons algorithmiques et aussi au fait que nous nous sommes intéressés aux voies métaboliques qui sont traitées comme étant des sous-réseaux du réseau métabolique, et qui peuvent être modélisées par des DAGs. En revanche, en général les réseaux métaboliques contiennent des cycles. Donc, pour que notre approche soit applicable sur ces réseaux (et non pas seulement sur des voies métaboliques), nous devons prendre en compte la présence de cycles. C'est pour cette raison que nous proposons dans ce chapitre un prétraitement des graphes D et G. Le prétraitement consiste à chercher une décomposition (soit sous forme de partition, soit sous forme de couverture) du graphe D en DAGs, de sorte que les sommets de chaque DAG induisent dans G un sous-graphe connexe. Si le graphe D (resp. G) représente le réseau métabolique (resp. le réseau PPI pour le même organisme), alors la décomposition proposée nous ramène au cadre biologique du chapitre précédent : on cherche la même information biologique observée dans deux contextes différents. En effet, comme nous l'avons déjà mentionné, les voies métaboliques correspondent à des DAGs [GHM+02,

PRYLZU05], et les complexes de protéines correspondent à des sous-graphes connexes [BML+05, DBVS09, NK07, TF09, KMM+10].

Dans ce chapitre nous considérons le problème de décision appelé k-DAGCC défini comme suit : étant donnés un graphe orienté D, un graphe non-orienté G ayant le même ensemble de sommets V, et un entier k, existe-t-il des ensembles de sommets $V_1, V_2, \dots V_{k'}$, $k' \leq k$, tels que, pour tout $1 \leq i \leq k'$, (i) $D[V_i]$ est un DAG et (ii) $G[V_i]$ est un sous-graphe connexe ?

Nous allons étudier des variantes de ce problème appelées respectivement k-DAGCC-Partition et k-DAGCC-Cover. Le problème k-DAGCC-Partition (resp. k-DAGCC-Cover) est la variante du problème k-DAGCC dans laquelle les ensembles V_i, $1 \leq i \leq k'$, forment une partition (resp. couverture, c'est-à-dire une décomposition possiblement chevauchante) de V.

Ce chapitre est organisé comme suit. Dans la première section, nous présentons brièvement quelques travaux portant sur la décomposition de graphes. Dans la deuxième section, nous formulons nos problèmes. Nous consacrons la Section 3.3 à l'étude de complexité des problèmes k-DAGCC-Partition et sa version de minimisation appelée Min-DAGCC-Partition. Dans la Section 3.4, nous étudions la complexité du problème k-DAGCC-Cover, ainsi que celle de sa version de minimisation appelée Min-DAGCC-Cover.

3.1 Décomposition de graphes

Dans le cas où le graphe G est une clique (tous les sommets sont adjacents), Min-DAGCC-Partition contient (comme sous-problème) le problème *Acyclic k-coloring* [Debo8a, Debo8b, NCDR$^+$10, NHLS10] : étant donnés un graphe orienté $D = (V, A)$ et un entier k, $k \leq |V|$, est-il possible de colorer les sommets de D en utilisant *exactement* k couleurs de sorte qu'il n'y ait aucun cycle *monochromatique* (c'est-à-dire un cycle dont les sommets ont tous la même couleur) ? En effet, les sommets de tout sous-graphe de D induisent un sous-graphe connexe dans G. Donc une solution du problème Min-DAGCC-Partition est une partition minimale de D en DAGs. Par conséquent, la réponse au problème *Acyclic k-coloring* est "oui" si, et seulement si, le nombre de DAGs dans une partition minimale est inférieur ou égal à k. Donc le problème Min-DAGCC-Partition contient, comme sous-problème, le problème *Acyclic k-coloring*.

Le problème *Acyclic k-coloring* a des applications en micro-économie, en particulier dans l'analyse du comportement rationnel des consommateurs. Deb [Debo8a, Debo8b] a montré que le problème *Acyclic* 2*-coloring* est NP-complet dans le cas général. Nobibon et al. [NCDR$^+$10] ont montré que le problème reste NP-complet même si le graphe D ne contient aucun cycle de taille 2 et ils ont proposé trois algorithmes exacts pour le résoudre. Nobibon et al. [NCDR$^+$10] ont également considéré une variante d'optimisation appelée *Max Acyclic* 2*-coloring* (abrégée *Max-A2C*) : le but est de colorer un nombre maximum de sommets de D, en utilisant seulement deux couleurs, de sorte qu'il n'y ait aucun cycle *monochromatique*. Ils ont prouvé que, à moins que $\mathsf{P} = \mathsf{NP}$, il existe $\epsilon > 0$, tel que *Max-A2C* est non-approximable avec un facteur de n^ϵ. Une heuristique a été proposée dans [NHLS10], pour résoudre le problème d'optimisation *Max-A2C*.

Le problème *Acyclic k-coloring* a été également bien étudié pour les graphes non-orientés [CH69, GJ79, WYZ96, RSK95, AJ07]. Dans, ce cas le problème consiste à la recherche d'une partition d'un graphe non-orienté H en k forêts. Clairement, à partir d'une partition de $H = D^*$ (le graphe support de D) en k forêts, nous pouvons obtenir facilement une partition de D en k DAGs et par conséquent nous obtenons une solution de k-DAGCC-Partition dans le cas où le graphe G est une clique.

Étant donné un graphe non-orienté H, l'entier minium k, noté $a(H)$, pour lequel la réponse au problème *Acyclic k-coloring* est "oui", est appelé *arboricité de sommets* de H (ou en anglais *vertex arboricity*). Chartrand et al. [CH69] ont démontré que $a(H) \leq 3$, pour tout graphe non-orienté planaire. Cette borne est la meilleure possible pour les graphes planaires, parce que les auteurs ont fournit, dans le même article, un exemple de graphe non-orienté planaire dont l'arboricité de sommets est égale à 3. La partition d'un graphe non-orienté planaire en trois forêts peut être construite en temps polynomial [RSK95]. Pour les graphes planaires extérieurs, nous avons la propriété suivante : $a(H) \leq 2$ [CH69].

Le problème *Acyclic k-coloring* est NP-complet quand le graphe en entrée H est un graphe non-orienté général et $k \geq 3$ [GJ79]. Wu. et al. [WYZ96] ont montré que le problème reste NP-complet même si $\Delta(H) = 5$, où $\Delta(H)$ est le degré maximum de H. En revanche, ils ont

montré que le problème peut se résoudre en temps polynomial quand $\Delta(H) = 4$.

Il a été démontré dans [AJ07] que si H est un graphe non-orienté planaire de diamètre inférieur ou égal à 2, alors $a(H) \leq 2$. Dans le même article, les auteurs ont montré que le problème *Acyclic 2-coloring* est NP-complet, même si H est un graphe non-orienté planaire de diamètre 2.

3.2 Formulation des problèmes

Tout au long du chapitre, $D = (V, A)$ et $G = (V, E)$ sont respectivement un graphe orienté et un graphe non-orienté ayant le même ensemble de sommets V. On pose $n = |V|$, $m_D = |A|$ et $m_G = |E|$.

Définition 3.1 *Soit un ensemble X quelconque. Un ensemble $\mathcal{C} = \{S_1, S_2, \ldots, S_k\}$ avec $S_i \subseteq X$, pour $1 \leq i \leq k$, est dit* couverture *de X si les conditions suivantes sont vérifiées :*

- $S_i \neq \emptyset$, *pour* $1 \leq i \leq k$;
- $\bigcup_{1 \leq i \leq k} S_i = X$.

Une *partition* d'un ensemble X est une couverture $\mathcal{C} = \{S_1, S_2, \ldots, S_k\}$ qui vérifie la condition suivante : pour tout $i, j \in \{1, 2, \ldots, k\}$ si $i \neq j$ alors $S_i \cap S_j = \emptyset$.

Définition 3.2 *Étant donnés $D = (V, A)$ et $G = (V, E)$, une partition (resp. couverture) $\{V_1, V_2 \ldots V_k\}$ de V est dite* valide *si, pour tout $1 \leq i \leq k$, $D[V_i]$ est un DAG et $G[V_i]$ est connexe.*

Les problèmes k-DAGCC-Partition et k-DAGCC-Cover sont définis comme suit.

> k-DAGCC-Partition
> **Instance** : Un graphe orienté $D = (V, A)$, un graphe non-orienté $G = (V, E)$ et entier k.
> **Question** : Existe-t-il une partition valide $\mathcal{P} = \{V_1, V_2, \ldots, V_{k'}\}$ de V telle que $k' \leq k$?

> k-DAGCC-Cover
> **Instance** : Un graphe orienté $D = (V, A)$, un graphe non-orienté $G = (V, E)$ et un entier k.
> **Question** : Existe-t-il une couverture valide $\mathcal{P} = \{V_1, V_2, \ldots, V_{k'}\}$ de V telle que $k' \leq k$?

Le problème Min-DAGCC-Partition (resp. Min-DAGCC-Cover) est la version de minimisation de k-DAGCC-Partition (resp. k-DAGCC-Cover) dans laquelle on cherche une partition (resp. couverture) de cardinalité minimale.

3.3 Partitionnement d'un graphe en DAGs

Dans cette section, nous étudions la complexité du problème k-DAGCC-Partition et de sa version de minimisation

MIN-DAGCC-PARTITION, en considérant différentes classes de graphes D et G. Les résultats de complexité que nous avons obtenus sont résumés dans le Tableau 3.1.

k \ G	Graphe général	Planaire extérieur	Arbre	Étoile	Chemin
$n-k=\mathcal{O}(1)$	P [Th. 3.1]				
$k=\mathcal{O}(1)$	NPC [Th. 3.4]	P [Th. 3.2]			
k non borné	Non-approx. avec $n^{1-\epsilon}$ [Th. 3.6]	APX-difficile [Th. 3.7]		NPC [Th. 3.5]	P [Th. 3.3]

Tableau 3.1 – *Complexité des problèmes* k-DAGCC-PARTITION *et* MIN-DAGCC-PARTITION.

Nous commençons par présenter des algorithmes polynomiaux pour chacun des cas suivants :

- $n-k$ est une constante ;
- G est planaire extérieur et k est une constante ;
- G est un chemin.

Nous rappelons que nous pouvons vérifier si D est un DAG (resp. si G est connexe) en un temps de $\mathcal{O}(n+m_D)$ (resp. $\mathcal{O}(n+m_G)$) en effectuant un parcours en profondeur de D (resp. de G).

Théorème 3.1 *Le problème* k-DAGCC-PARTITION *peut se résoudre en temps polynomial quand* $n-k$ *est une constante.*

Preuve. Nous proposons une procédure exhaustive. Nous calculons, pour tout $k \in \{1,2,\ldots,n\}$, toutes les k-partitions de V (c'est-à-dire les partitions de V en k parties). Nous commençons par l'unique n-partition de V et nous calculons, de manière itérative, toutes les k-partitions de V obtenues à partir des $(k+1)$-partitions en fusionnant deux parties. Nous appelons *Part* la fonction qui prend en argument un ensemble V et un entier k, et qui renvoie, sous forme de liste toutes les partitions de V en k parties. La fonction *Part* est définie comme suit.

Fonction $Part(V,k)$:

1. **si** $k=n$ **alors return** $\{\{1\},\{2\},\ldots,\{n\}\}$;
2. **sinon**
   3. $L_1 := Part(V,k+1)$;
   4. Soit L_2 la liste qui contient toutes les k-partitions de V obtenues en fusionnant deux parties d'une partition dans L_1.
   5. **return** L_2
6. **finsi**

Nous notons par $f(n,k)$ la complexité en temps d'exécution pour la fonction *Part*. Le calcul de la liste L_1 se fait en temps de $f(n,k+1)$. Pour chaque partition dans L_1, il existe $\binom{k+1}{2}$ possibilités pour choisir les deux parties à fusionner (ligne 4). Donc la fonction $f(n,k)$ vérifie la relation de récursion suivante :

$$f(n,k) = f(n,k+1) + |L_1| \cdot \binom{k+1}{2}$$

Évidemment, $|L_1| = \mathcal{O}(f(n,k+1))$. Donc, $f(n,k) = \mathcal{O}(f(n,k+1) \cdot \binom{k+1}{2})$. Cela implique que $f(n,k) = \mathcal{O}(f(n,n) \cdot \prod_{i=k+1}^{n} \binom{i}{2})$. Or, $f(n,n) = \mathcal{O}(n)$. Donc, $f(n,k) = \mathcal{O}(n^{2(n-k)+1})$. Par ailleurs, nous pouvons tester en un temps polynomial la validité de chacune de ces $\mathcal{O}(n^{2(n-k)+1})$ partitions. On en déduit donc que k-DAGCC-Partition peut se résoudre en temps polynomial quand $n-k = \mathcal{O}(1)$. □

Théorème 3.2 *Le problème k-DAGCC-Partition peut se résoudre en temps polynomial quand G est un graphe planaire extérieur et k est une constante.*

Preuve. Le graphe G est planaire extérieur, donc on peut dessiner les sommets de G sur un cercle C de sorte que les arêtes de G ne se rencontrent qu'à leurs extrémités (voir la Section 1.2.1 du Chapitre 1). Supposons que les sommets de V sont dessinés sur le cercle C, dans le sens trigonométrique, et dans l'ordre $v_1, v_2, \ldots, v_n$.

Soit $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ une partition valide de V. On associe à chaque partie V_i, $1 \leq i \leq k$, le plus petit arc du cercle C couvrant tous les sommets de V_i. Un tel arc est noté a_i. Le fait que $\mathcal{P}$ est une partition de V implique que chaque sommet de V est couvert par au moins un arc a_i et que pour tout $i, j \in \{1, 2, \ldots, k\}$, si $i \neq j$ alors a_i et a_j n'ont aucune extrémité en commun.

Pour tout $i, j \in \{1, 2, \ldots, k\}$, $G[V_i]$ et $G[V_j]$ sont connexes. Donc si $i \neq j$, par le fait que G est planaire extérieur on en déduit qu'il n'y a pas de chevauchement propre entre les deux arcs a_i et a_j : (i) a_i est inclus dans a_j ou (ii) a_j est inclus dans a_i ou (iii) a_i et a_j n'ont aucun sommet en commun.

Pour résoudre le problème k-DAGCC-Partition, nous proposons un algorithme exhaustif qui énumère tous les ensembles possibles associés aux k arcs $\{a_1, a_2, \ldots, a_k\}$ du cercle C tels que : pour tout $i, j \in \{1, 2, \ldots, k\}$, $i \neq j$, il n'y a pas de chevauchement propre entre a_i et a_j.

Pour tout $1 \leq i \leq n$, on considère tous les sous-ensembles de $V \setminus \{v_i, v_{i-1}\}$ de taille $2(k-1)$, dits les $2(k-1)$-sous-ensembles de V. Par convention, on note $v_0 = v_n$. Soit $\{v_{j_1}, v_{j_2}, \ldots, v_{j_{2(k-1)}}\}$ un $2(k-1)$-sous-ensemble de V. Supposons que les sommets de l'ensemble $\{v_{j_1}, v_{j_2}, \ldots, v_{j_{2(k-1)}}\}$ sont dessinés sur le cercle C, dans le sens trigonométrique, et dans l'ordre $v_{j_1}, v_{j_2}, \ldots, v_{j_{2(k-1)}}$.

Nous voulons maintenant construire l'ensemble $\mathcal{A}_k$ des k arcs du cercle C tel que pour toutes paires d'arcs a_i, a_j, $i \neq j$, les deux conditions suivantes sont vérifiées :

- Les extrémités de a_i et a_j sont des sommets de l'ensemble $\{v_{j_1}, v_{j_2}, \ldots, v_{j_{2(k-1)}}\}$;
- Il n 'y a pas de chevauchement propre entre a_i et a_j.

On notera que la construction de ces arcs revient à la construction de tous les parenthésages corrects d'une chaîne de taille $2k$.

Soit $\{b_1, b_2, \ldots, b_k\} \in \mathcal{A}_k$. A partir des k arcs $\{b_1, b_2, \ldots, b_k\}$, nous pouvons construire une partition $\{V_1^b, V_2^b, \ldots, V_k^b\}$ de V comme suit : V_i contient tous les sommets couverts par l'arc b_i à l'exception des sommets qui sont couverts par un arc b_j strictement inclus dans b_i. Ensuite, on teste la validité de la partition calculée.

Cet algorithme peut être réalisé en un temps de $\mathcal{O}(n^2 \binom{n}{2(k-1)} \binom{2(k-1)}{k-1} (n+m))$, où $m = \max\{m_D, m_G\}$. En effet, nous

avons n possibilités pour choisir le sommet de référence v_i. En commençant par un sommet v_i, on construit l'ensemble $\mathcal{A}_k$ des k arcs du cercle C, cet ensemble contient donc $\binom{n}{2(k-1)} \frac{1}{k} \binom{2(k-1)}{k-1}$ éléments, où $\frac{1}{k} \binom{2(k-1)}{k-1}$ est le nombre de parenthésages corrects d'une chaîne de taille $2k$. Enfin, partant d'un ensemble de k arcs $\{b_1, b_2, \ldots, b_k\} \in \mathcal{A}_k$, on peut construire la partition $\{V_1^b, V_2^b, \ldots, V_k^b\}$ en un temps de $\mathcal{O}(n)$. La validité de cette partition peut être testée en un temps de $\mathcal{O}(n+m)$. Par conséquent, la complexité globale de l'algorithme est de $\mathcal{O}(n^2 \binom{n}{2(k-1)} \binom{2(k-1)}{k-1} (n+m))$. Donc, c'est une complexité polynomiale quand k est une constante.

□

Théorème 3.3 *Le problème* Min-DAGCC-Partition *peut se résoudre en temps polynomial quand G est un chemin.*

Preuve. Supposons que G est le chemin $v_1 v_2 \ldots v_n$. Nous proposons un simple algorithme glouton.

1. $S := \emptyset; \mathcal{P} := \emptyset$
2. **pour** i de 1 à n **faire**
   3. **si** $D[S \cup \{v_i\}]$ est un DAG **alors** $S := S \cup \{v_i\}$;
   4. **sinon** $\mathcal{P} := \mathcal{P} \cup \{S\}; S := \emptyset$; /* Création d'une nouvelle partie */
   5. **finsi**
6. **fin pour**
7. $\mathcal{P} := \mathcal{P} \cup \{S\}$; /* Création de la dernière partie */
8. **return** $\mathcal{P}$

Dans cet algorithme nous parcourons le chemin G en visitant les sommets dans l'ordre $v_1, v_2, \ldots, v_n$ (la boucle "pour"). Pour un sommet v_i, si $D[S \cup \{v_i\}]$ est un DAG, alors on ajoute v_i à S (ligne 3), sinon S est une partie de la partition recherchée (ligne 4) et on commence la recherche d'une nouvelle partie. A la fin de l'algorithme (ligne 7), l'ensemble S correspond à la dernière partie.

Il est facile de vérifier que cet algorithme glouton produit une partition valide de V. Nous allons maintenant montrer la minimalité de cette partition.

Soit $\mathcal{P} = \{V_1, V_2, \ldots V_k\}$ la partition obtenue par l'algorithme. Nous supposons qu'il existe une partition valide $\mathcal{P}' = \{V_1', V_2', \ldots, V_\ell'\}$ telle que $\ell < k$, et nous voulons obtenir une contradiction. D'abord, par construction de $\mathcal{P}$, l'ensemble V_1 ne peut pas être un sous-ensemble propre de V_1'. Autrement, $D[V_1']$ serait cyclique (c'est-à-dire n'est pas un DAG) parce qu'il contiendrait un sous-graphe cyclique induit par les sommets de l'ensemble $V_1 \cup \{v\}$ où v est le sommet le plus à gauche de V_2. Cela contredit la validité de la partition $\mathcal{P}'$. En plus, le fait que tout sous-graphe connexe de G est un chemin implique qu'il existe deux entiers i et j, $2 \leq i < k$ et $2 \leq j \leq \ell$, tels que (1) V_i est un sous-ensemble propre de V_j', et (2) V_j' contient le sommet le plus à gauche de V_{i+1}, noté x. Ceci est une contradiction, parce que $D[V_i \cup \{x\}]$ n'est pas un DAG. □

Dans le reste de cette section, nous identifions des instances difficiles pour les problèmes k-DAGCC-Partition et Min-DAGCC-Partition.

Théorème 3.4 *Pour tout $k \geq 2$, le problème k-DAGCC-Partition est* NP*-complet même si G est un graphe complet.*

Preuve. Le problème k-DAGCC-Partition est clairement dans NP. Pour prouver la NP-complétude de k-DAGCC-Partition, nous proposons une réduction du problème NP-complet Not-All-Equal 3SAT (NAE 3Sat) [GJ79] : étant donnés un ensemble de variables $\mathcal{X}_n = \{x_1, \dots x_n\}$ et un ensemble $\mathcal{C}_q = \{c_1, \dots c_q\}$ de clauses construites sur $\mathcal{X}_n$ où chaque clause est une conjonction de trois littéraux, existe-t-il une affectation des variables de $\mathcal{X}_n$ de sorte que toute clause contient au moins un littéral avec la valeur `vrai` et au moins un littéral avec la valeur `faux` ?

Pour tout $1 \leq j \leq 3$, on note par c_i^j le j-ième littéral de la clause c_i. Étant donnée une instance $(\mathcal{X}_n, \mathcal{C}_q)$ de NAE 3Sat, on construit les graphes $D = (V, A)$ et $G = (V, E)$ comme suit :

- $V = \{v_i^j : 1 \leq i \leq q, 1 \leq j \leq 3\} \cup \{v_i : 3 \leq i \leq k\}$;
- $A = \{v_i^j v_{i'}^{j'},\ v_{i'}^{j'} v_i^j \ :\ c_i^j = \overline{c_{i'}^{j'}}\} \cup \{v_i^1 v_i^2,\ v_i^2 v_i^3,\ v_i^3 v_i^1 \ :\ 1 \leq i \leq q\} \cup \{v_i v,\ v v_i \ :\ 3 \leq i \leq k, v \in V \setminus \{v_i\}\}$;
- $E = \{(v, v') : v, v' \in V\}$

Le sommet v_i^j correspond au j-ième littéral de la clause c_i. Les sommets v_i, $3 \leq i \leq k$, sont des sommets fictifs que nous avons ajouté pour adapter la preuve pour tout $k \geq 2$. Il existe un cycle de taille deux entre toute paire de sommets qui représentent respectivement une variable et son complément (par exemple x_i et $\overline{x_i}$). Il existe également un cycle de taille deux entre tout sommet v_i, $3 \leq i \leq k$, et chacun des sommets de V. En outre, il existe un cycle de taille trois entre le triplet de sommets (v_i^1, v_i^2, v_i^3) correspondant aux trois littéraux d'une clause c_i. Enfin, le graphe G est un graphe complet dont l'ensemble de sommets est V. Un exemple de construction du graphe D est illustré dans la Figure 3.1.

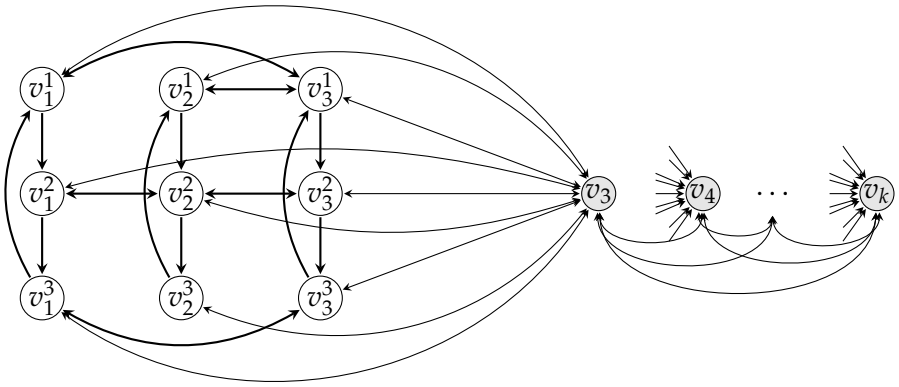


Figure 3.1 – *Illustration de la construction du graphe orienté D à partir d'une instance du problème* NAE 3Sat. *Ici, $\mathcal{C}_q = \{(x_1 \vee x_2 \vee x_3), (x_1 \vee \overline{x_2} \vee x_4), (\overline{x_1} \vee x_2 \vee \overline{x_3})\}$. Afin d'assurer la lisibilité de la figure, nous n'avons pas mis les arcs $\{(v_i^j, v_l), (v_l, v_i^j) : 1 \leq i \leq q, 1 \leq j \leq 3,\ 4 \leq l \leq k\}$. Nous avons également représenté les cycles de taille deux par une seule arête orientée dans deux directions opposées.*

Nous allons prouver la propriété suivante : il existe une solution de NAE 3SAT si et seulement il existe une partition valide $\{V_1, V_2, \ldots, V_k\}$ de V.
$\Rightarrow$: Soit $\mathcal{A}$ une affectation des variables de $\mathcal{X}_n$ telle que toute clause contient au moins un littéral avec la valeur `vrai` et au moins un littéral avec la valeur `faux`. On construit une partition $\mathcal{P} = \{V_1, V_2 \ldots V_k\}$ de V comme suit :

- $V_1 = \{v_i^j : c_i^j = \texttt{vrai}\}$;
- $V_2 = \{v_i^j : 1 \leq i \leq q, 1 \leq j \leq 3\} \setminus V_1$;
- $V_i = \{v_i\}$ pour tout i, $3 \leq i \leq k$.

Par définition, il y a un littéral ou deux avec la valeur `faux`. Donc, parmi les trois sommets qui correspondent aux littéraux de c_i, au moins un sommet et au plus deux appartiennent à l'ensemble V_1 (resp. V_2). Par ailleurs, dans l'affectation $\mathcal{A}$, aucune variable ne peut être affectée à `vrai` et `faux` simultanément. Par conséquent, deux sommets représentant deux littéraux complémentaires ne peuvent pas appartenir simultanément à V_1 (resp. à V_2). Donc, $D[V_i]$, $1 \leq i \leq k$, est un DAG, parce que tous les cycles ont été supprimés. Le graphe G est un graphe complet, donc la partition $\mathcal{P}$ est valide.
$\Leftarrow$: Soit $\mathcal{P} = \{V_1, V_2 \ldots V_k\}$ une partition valide de V. Nous supposons, sans perte de généralité, que $V_3 = \{v_3\}, \ldots, V_k = \{v_k\}$. Nous voulons maintenant construire une affectation $\mathcal{A}$ qui résout le problème NAE 3SAT. L'affectation $\mathcal{A}$ est définie comme suit : pour tout $1 \leq i \leq q$ et $1 \leq j \leq 3$, on attribue la valeur `vrai` au littéral c_i^j si, et seulement si, $v_i^j \in V_1$. Le sous-graphe $D[V_1]$ est un DAG, donc V_1 ne contient pas deux littéraux complémentaires et par conséquent l'affectation $\mathcal{A}$ est correcte : une variable et son complémentaire ne peuvent pas avoir la même valeur. Par construction de V_1, pour toute clause $c_i \in \mathcal{C}_q$, V_1 contient au moins un sommet et au plus deux sommets de l'ensemble $\{v_i^1, v_i^2, v_i^3\}$ correspondant aux littéraux de la clause c_i. Donc, l'affectation $\mathcal{A}$ résout NAE 3SAT. □

Dans le Théorème 3.5, nous allons montrer que le problème k-DAGCC-PARTITION reste également difficile même si le graphe est une étoile. Soit $G_I = (V_I, E_I)$ un graphe non-orienté. Nous rappelons qu'un stable (ou en anglais *independent set*) dans G_I est un ensemble de sommets deux à deux non-adjacents.

Théorème 3.5 *Le problème k-*DAGCC-PARTITION *est* NP*-complet même si le graphe G est une étoile.*

Preuve. Nous proposons une réduction du problème NP-complet MAXIMUM INDEPENDENT SET (MIS) [GJ79] dont la version de décision naturelle est définie comme suit : étant donnés un graphe non-orienté $G_I = (V_I, E_I)$ et un entier $k' \geq 0$, existe-t-il dans G_I un stable $V_I' \subseteq V_I$ tel que $|V_I| \geq k'$?

Étant donnée une instance G_I du problème MIS, on construit les graphes $D = (V, A)$ et $G = (V, E)$ comme suit :

- $V = V_I \cup \{v_r\}$;
- $A = \{vv',\ v'v\ :\ (v, v') \in E_I\}$;
- $E = \{(v_r, v) : v \in V_I\}$.

Le graphe D est obtenu de G_I en remplaçant toute arête par deux arcs dans deux directions opposées, et en ajoutant un sommet isolé v_r. Le graphe G est une étoile dont le centre v_r est connecté par une arête à tous les sommets de l'ensemble V_I. Un exemple de construction de D et G est montré dans la Figure 3.2.

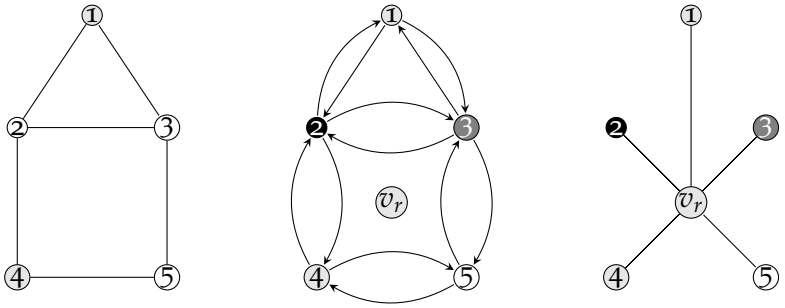


FIGURE 3.2 – *Exemple de construction des graphes D et G à partir d'une instance G_I du problème* MAXIMUM INDEPENDENT SET. *Les sommets de l'ensemble $\{1,4\}$ forment un stable dans G_I. A partir de cet ensemble, on obtient une partition valide de V composée des quatre parties : $\{1,4,v_r\}$, $\{2\}$, $\{3\}$ et $\{5\}$.*

Nous allons montrer qu'il existe un stable à k' sommets dans $G_I = (V_I, E_I)$ si, et seulement si, il y a une solution pour k-DAGCC-PARTITION avec $k = |V_I| - k' + 1$.
$\Rightarrow$: Soit V'_I un stable dans G_I tel que $|V'_I| = k'$. Soit $k = |V_I| - k' + 1$. On construit une partition $\mathcal{P} = \{V_1, V_2 \dots V_k\}$ de V comme suit. On pose $V_1 = V'_I \cup \{v_r\}$. On considère un ordre arbitraire des sommets dans $V_I \setminus V'_I$, et pour tout $2 \leq i \leq k$, on pose $V_i = \{v\}$, tel que v est le $(i-1)$-ième sommet dans cet ordre.

Pour tout i, $2 \leq i \leq k$, le sous-graphe $D[V_i]$ est un DAG, parce que V_i contient un seul sommet. Par définition, $\nexists\{v, v'\} \subseteq V'_I$, tel que $(v, v') \in E_I$. Donc, $D[V_1]$ est composé de sommets isolés et par conséquent $D[V_1]$ est aussi un DAG. Évidemment, pour tout i, $2 \leq i \leq k$, le sous-graphe $D[V_i]$ est connexe ; de plus, la connexité du sous-graphe $G[V_1]$ est assurée par le sommet v_r. Par conséquent, la partition $\mathcal{P} = \{V_1, V_2 \dots V_k\}$ est valide.
$\Leftarrow$: Soit $\mathcal{P} = \{V_1, V_2 \dots V_k\}$ une solution du problème k-DAGCC-PARTITION (c'est-à-dire $\mathcal{P}$ est une partition valide), avec $k = |V_I| - k' + 1$. Nous allons construire un stable V'_I de $G_I = (V_I, E_I)$ tel que $|V'_I| = k'$. On pose $V'_I = V_j \setminus \{v_r\}$ tel que $D[V_j]$ est le DAG contenant v_r. Évidemment V'_I est un stable dans G_I, parce que $D[V_j]$ est acyclique. Montrons maintenant que $|V'_I| = k'$.

Le graphe G_I est une étoile de racine v_r, donc tout sous-graphe connexe de G_I d'ordre supérieur ou égal à 2 doit contenir le sommet v_r. D'autre part, $\mathcal{P}$ est une partition, donc v_r appartient à un unique V_i, $i \in \{1, 2, \dots, k\}$. Par conséquent, v_r appartient uniquement à l'ensemble V_j, et donc les ensembles V_i, $i \in \{1, 2, \dots, k\} \setminus \{j\}$, sont des singletons. Donc, $|V_j| = (|V_I| + 1) - (k - 1) = |V_I| - k + 2$. Cela implique que $|V_j| = |V_I| - k + 2$. Or $k = |V_I| - k' + 1$, donc $|V_j| = k' + 1$. Par ailleurs, $V'_I = V_j \setminus \{v_r\}$. Par conséquent, $|V'_I| = k'$.

□

Dans le reste cette section, nous présentons quelques résultats de non-approximabilité du problème MIN-DAGCC-PARTITION. Avant de présenter ces résultats, nous avons besoin de la définition suivante.

Définition 3.3 (Coloration propre) *Soit $G_C = (V_C, E_C)$ un graphe non-orienté. Une* coloration propre *de G_C est une affectation de couleurs aux sommets de G_C de sorte que n'importe quels deux sommets adjacents dans G_C aient des couleurs différentes.*

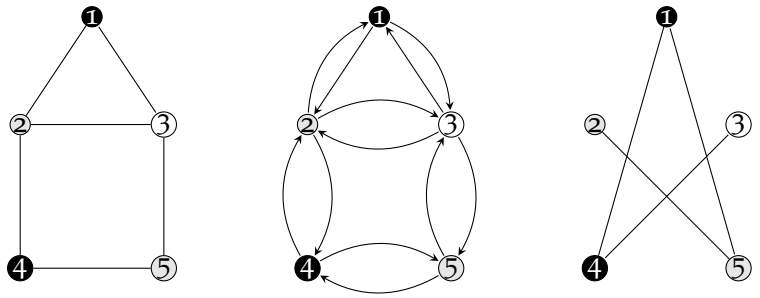
Théorème 3.6 *Le problème* MIN-DAGCC-PARTITION *est non-approximable avec un facteur de $n^{1-\epsilon}$, pour tout $\epsilon > 0$.*

Preuve. Nous proposons une L-réduction du problème MINIMUM CHROMATIC NUMBER (MIN-CN) défini comme suit : étant donné un graphe non-orienté $G_C = (V_C, E_C)$, déterminer une coloration propre de G_C qui utilise un nombre minimum de couleurs.

Soit G_C une instance de MIN-CN. On construit les graphes D et G comme suit :

- $V = V_C$;
- $A = \{vv',\ v'v\ :\ (v, v') \in E_C\}$;
- $E = \{(v, v')\ :\ v, v' \in V_C\} \setminus E_C$

Les graphes D et G sont construits sur le même ensemble de sommets que G_C. Le graphe D est obtenu de $G_C = (V_C, E_C)$ en remplaçant toute arête dans E_C par deux arcs dans des directions opposées. Le graphe G est le graphe complémentaire de G_C. Un exemple de construction est donné dans la Figure 3.3.



G_C D G

FIGURE 3.3 – *Exemple de construction des graphes D et G à partir d'une instance G_C du problème* MIN-CN. *Le graphe G_C est coloré en utilisant trois couleurs. A partir de cette coloration propre, nous avons construit une partition valide $\mathcal{P} = \{\{1,4\},\{3\},\{2,5\}\}$.*

Nous allons prouver la propriété suivante : il existe une coloration propre de G_C utilisant k couleurs si, et seulement si, il existe une partition valide de V composée de k parties.

$\Rightarrow$: Considérons une coloration propre de G_C utilisant k couleurs $\{1, 2, \ldots, k\}$. On note par V_i l'ensemble des sommets dont la couleur est i. On pose $\mathcal{P} = \{V_i : 1 \leq i \leq k\}$. Nous voulons montrer que $\mathcal{P}$ est une partition valide de V c'est-à-dire, pour tout i, $1 \leq i \leq k$, $D[V_i]$ est un DAG et $G[V_i]$ connexe.

Par définition d'une coloration propre, chaque V_i, $1 \leq i \leq k$, est un stable dans G_C. Donc, tout sous-graphe $D[V_i]$, $1 \leq i \leq k$, est composé de sommets isolés et donc $D[V_i]$ est un DAG. D'autre part, pour tout $1 \leq i \leq$

k, $G[V_i]$ est une clique (et par conséquent $G[V_i]$ est connexe) parce que V_i est un stable dans G_C. On en déduit que la partition $\mathcal{P}$ est une partition valide de V.
$\Leftarrow$: Étant donnée une partition valide $\mathcal{P} = \{V_1, V_2 \dots V_k\}$, on attribue à tout sommet $v \in V_i$ la couleur i, pour tout $1 \leq i \leq k$.

Pour tout i, $1 \leq i \leq k$, $D[V_i]$ est un DAG, donc $D[V_i]$ est composé de sommets isolés, et par conséquence V_i est un stable dans G_C. On en déduit que cette coloration est une coloration propre de G_C utilisant k couleurs.

La réduction proposée est une réduction préservant le paramètre k, qui est d'une part le nombre de couleurs nécessaires pour colorer G_C et d'autre part le nombre de parties d'une partition valide de V. Donc, en partant d'une approximation de Min-DAGCC-Partition, nous pouvons obtenir une approximation, avec le même ratio, du problème Min-CN. Or, Min-CN est non-approximable avec un ratio de $n^{1-\epsilon}$ pour tout $\epsilon > 0$ [Zuc07], donc on en déduit que le même résultat de non-approximabilité est aussi valide pour Min-DAGCC-Partition. □

Nous étudions maintenant la complexité du problème Min-DAGCC-Partition en considérant le cas où le graphe G est un arbre. Pour présenter le prochain résultat, nous utilisons la définition suivante.

Définition 3.4 (Ensemble couvrant) *Soit $\mathcal{X} = \{x_1, \dots x_n\}$ un ensemble et soit $\mathcal{C} = \{S_1, \dots S_q\}$ une collection des sous-ensembles de $\mathcal{X}$. Un ensemble $\mathcal{C}' \subseteq \mathcal{C}$ est dit* couvrant *de $\mathcal{X}$ si $\mathcal{X} = \bigcup_{S_i \in \mathcal{C}'} S_i$.*

Théorème 3.7 *Le problème* Min-DAGCC-Partition *est* APX*-difficile même si G est un arbre.*

Preuve. Nous proposons une L-réduction du problème APX-difficile Set Cover-2 [PY91] défini comme suit : étant donné un ensemble $\mathcal{X} = \{x_1, \dots x_n\}$ et une collection $\mathcal{C} = \{S_1, \dots S_q\}$ de sous-ensembles de $\mathcal{X}$ tel que tout élément $x_j \in \mathcal{X}$ apparaît au plus dans deux sous-ensembles $S_i \in \mathcal{C}$, trouver un ensemble couvrant minimal de $\mathcal{X}$.

Soit $(\mathcal{X}, \mathcal{C})$ une instance du problème Set Cover-2. Pour tout $S_i \in \mathcal{C}$, et pour tout l, $1 \leq l \leq |S_i|$, on note par s_i^l le l-ième élément de l'ensemble S_i.

Nous allons construire les graphes $D = (V, A)$ et $G = (V, E)$ comme suit :

- $V = \{v_r\} \cup \{v_i : S_i \in \mathcal{C}\} \cup \{v_i^\alpha : x_\alpha = s_i^j, S_i \in \mathcal{C}, 1 \leq j \leq |S_i|\}$
- $A = \{v_i^\alpha v_j^\alpha,\ v_j^\alpha v_i^\alpha\ :\ x_\alpha \in S_i \cap S_j\} \cup \{v_r v_i^\alpha,\ v_i^\alpha v_r : \nexists S_j \text{ s.t. } x_\alpha \in S_i \cap S_j\}$
- $E = \{(v_r, v_i)\ :\ S_i \in \mathcal{C}\} \cup \{(v_i, v_i^\alpha) : x_\alpha \in S_i, S_i \in \mathcal{C}\}$.

Le graphe G est un arbre de racine v_r. Les fils de v_r sont des sommets $v_i, 1 \leq i \leq q$, qui représentent les ensembles S_i, $1 \leq i \leq q$. Chaque sommet v_i a un fils pour chaque élément de S_i (v_i^α pour $x_\alpha \in S_i$). Les sommets v_i^α sont les feuilles de l'arbre G.

Le graphe D a le même ensemble de sommets que G. Dans D, les sommets $v_i, 1 \leq i \leq q$, sont des sommets isolés. Il y a un cycle de taille deux entre les sommets v_i^α et v_j^α qui représentent un élément x_α qui apparaît deux fois dans $\mathcal{C}$. Enfin, chacun des sommets restants forme dans D un

cycle de taille deux avec le sommet v_r. Un exemple de construction est montré dans la Figure 3.4.

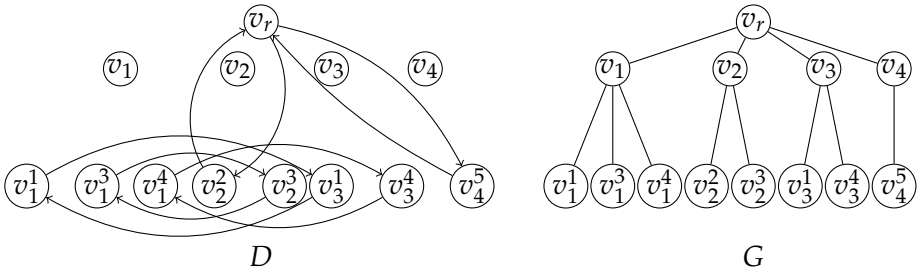


FIGURE 3.4 – *Exemple de construction d'une instance* (D, G) *du problème* MIN-DAGCC-PARTITION *à partir d'une instance* $(\mathcal{X}, \mathcal{C})$ *du problème* SET COVER-2. *Ici,* $\mathcal{X} = \{x_1, x_2, x_3, x_4, x_5\}$ *et* $\mathcal{C} = \{\{x_1, x_3, x_4\}, \{x_2, x_3\}, \{x_1, x_4\}, \{x_5\}\}$.

Nous allons montrer la propriété suivante : il existe un ensemble couvrant $\mathcal{C}'$ de $\mathcal{X}$ tel que $|\mathcal{C}'| = k$ si, et seulement si, il existe une partition valide de V composée de $k+1$ parties $V_1, V_2 \dots V_{k+1}$.
$\Rightarrow$: Soit $\mathcal{C}'$ un ensemble couvrant tel que $|\mathcal{C}'| = k$. On suppose sans perte de généralité que $\mathcal{C}'$ est formé par les k premiers ensembles de $\mathcal{C}$, c'est-à-dire, $\mathcal{C}' = \{S_1, S_2, \dots, S_k\}$. On construit une partition $\mathcal{P} = \{V_1, V_2 \dots, V_k, V_{k+1}\}$ de V comme suit :

- $V_i = \{v_i\} \cup \{v_i^\alpha : x_\alpha = s_i^j\}$, pour tout $1 \leq i \leq k$;
- $V_{k+1} = \{v_r\} \cup \{v_i, v_i^\alpha : x_\alpha = s_i^j, S_i \in \mathcal{C} \setminus \mathcal{C}'\}$.

Autrement dit, pour tout $i \in \{1, 2, \dots, k\}$, l'ensemble V_i contient les sommets du sous-arbre de G induit par v_i et ses fils ; tandis que V_{k+1} contient les sommets restants (y compris la racine v_r). Donc par construction, pour tout i, $1 \leq i \leq k+1$, le sous-graphe $G[V_i]$ est un sous-arbre de G et donc $G[V_i]$ est connexe. Par ailleurs, pour tout i, $1 \leq i \leq k$, $D[V_i]$ est un DAG parce que $v_r \notin V_i$ et tout élément de $\mathcal{X}$ apparaît au plus une fois dans l'ensemble S_i (et donc il n'y a pas de cycle de taille deux). Enfin, $D[V_{k+1}]$ est un DAG. En effet, $\mathcal{C}'$ est un ensemble couvrant et donc par définition de SET COVER-2, aucun élément de $\mathcal{X}$ ne peut apparaître plus d'une fois dans $\mathcal{C} \setminus \mathcal{C}'$. Donc $D[V_{k+1}]$ ne contient aucun cycle de taille deux et par conséquent $D[V_{k+1}]$ est un DAG. On en déduit la validité de la partition $\mathcal{P} = \{V_1, V_2 \dots, V_k, V_{k+1}\}$.
$\Leftarrow$: Soit $\mathcal{P} = \{V_1, V_2 \dots V_{k+1}\}$ une partition valide de V. On suppose, sans perte de généralité, que $v_r \in V_1$. On construit un ensemble couvrant $\mathcal{C}' = \{S_1, \dots, S_k\}$ comme suit. On pose $\mathcal{C}' = \{S_i : \exists v_i^\alpha \notin V_1\}$ c'est-à-dire, on ajoute à $\mathcal{C}'$ les ensembles S_i contenant au moins un élément dont le sommet correspondant n'appartient pas à V_1.

Montrons que $|\mathcal{C}'| \leq k$. Soient $S_i, S_j \in \mathcal{C}'$ avec $i \neq j$. Donc il existe $\alpha, \beta \in \{1, 2, \dots, |\mathcal{X}|\}$ tels que $v_i^\alpha \notin V_1$ et $v_j^\beta \notin V_1$. Or, $\mathcal{P} = \{V_1, V_2 \dots V_{k+1}\}$ est une partition de V, donc $\exists\ a, b \in \{2, 3, \dots, k+1\}$ tels que $v_i^\alpha \in V_a$ et $v_j^\beta \in V_b$. Dans l'arbre G, tout chemin reliant v_i^α à v_j^β doit passer par v_r. Mais v_r est un sommet de V_1, donc la connexité des sous-graphes $G[V_a]$ et $G[V_b]$ implique que $a \neq b$. Par conséquent, on en déduit que $|\mathcal{C}'| \leq k$. Il reste donc à montrer que $\mathcal{C}'$ est un ensemble couvrant de $\mathcal{X}$. Soit $x_\alpha \in \mathcal{X}$. Nous distinguons deux cas.

1. x_α apparaît exactement une seule fois dans $\mathcal{C}$. Soit S_j l'ensemble contenant x_α. Le sommet v_j^α correspondant à x_α ne peut pas appartenir à V_1, parce que dans ce cas $D[V_1]$ contiendra un cycle de taille deux entre v_r et v_j^α. Donc, $S_j \in \mathcal{C}'$.
2. x_α apparaît deux fois dans $\mathcal{C}$. Soient S_i et S_j les deux ensembles contenant x_α. Les deux sommets v_i^α et v_j^α qui correspondent à x_α ne peuvent pas appartenir simultanément à V_1, sinon il y aura dans $D[V_1]$ un cycle de taille deux entre v_j^i et $v_{j'}^i$. Donc, on a $v_i^\alpha \notin V_1$ ou $v_j^\alpha \notin V_1$ et par conséquence $S_i \in \mathcal{C}'$ ou $S_j \in \mathcal{C}'$

Nous avons donc montré que tout $x_i \in \mathcal{X}$ apparaît au moins une fois dans $\mathcal{C}'$. Donc $\mathcal{C}'$ est un ensemble couvrant de $\mathcal{X}$ avec $|\mathcal{C}'| \leq k$. Enfin, l'ensemble $\mathcal{C}'$ peut être complété en ajoutant des ensembles de $\mathcal{C} \setminus \mathcal{C}'$, choisis arbitrairement, pour assurer que $\mathcal{C}' = k$, et dans ce cas $\mathcal{C}'$ reste toujours un ensemble couvrant de $\mathcal{X}$.

Nous avons montré qu'il existe un ensemble couvrant de k éléments si, et seulement si, il existe une partition valide de $k+1$ éléments, donc la réduction que nous avons proposée est une L-réduction de Set Cover-2 vers Min-DAGCC-Partition. Or, Set Cover-2 est APX-difficile, donc on en déduit le problème Min-DAGCC-Partition est APX-difficile même si le graphe G est un arbre.

□

3.4 Couverture d'un graphe par DAGs

Dans cette section, nous étudions la complexité des problèmes k-DAGCC-Cover et Min-DAGCC-Cover en considérant les mêmes classes de graphes D et G vues dans la section précédente. Le Tableau 3.1 montre les résultats de complexité que nous avons obtenus.

k \ G	Graphe	Planaire extérieur	Arbre	Étoile	Chemin
$k = \mathcal{O}(1)$	NPC [Th. 3.9]				P [Th. 3.8]
k non borné	Non-approx. avec $n^{1-\epsilon}$ [Th. 3.10]				P [Th. 3.8]

Tableau 3.2 – *Complexité des problèmes k-*DAGCC-Cover *et* Min-DAGCC-Cover.

Nous commençons d'abord par démontrer que, comme c'était le cas pour le problème Min-DAGCC-Partition (Théorème 3.3), la version de couverture (c'est-à-dire Min-DAGCC-Cover) peut se résoudre en un temps polynomial quand G est un chemin. Pour prouver ce résultat, nous utilisons le lemme suivant.

Lemme 3.1 *Étant donnés un graphe orienté $D = (V, A)$ et un chemin $G = (V, E)$, la réponse au problème k-*DAGCC-Partition *avec l'instance (D, G) est OUI si et seulement la réponse au problème k-*DAGCC-Cover *avec la même instance est OUI.*

Preuve. $\Rightarrow$: Ce sens de la preuve est trivial, parce que toute partition est une couverture.

$\Leftarrow$: Soit $\mathcal{C}_1 = \{V_1, V_2, \dots V_{k_1}\}$, $k_1 \leq k$, une couverture valide de V, c'est-à-dire pour tout $1 \leq i \leq k_1$, $G[V_i]$ est connexe et $D[V_i]$ est un DAG. Le graphe G est un chemin, donc $G[V_i]$ est aussi un chemin, pour tout $1 \leq i \leq k_1$. Supposons qu'il existe un sommet $v \in V$ qui appartient à trois ensembles distincts $V_p, V_q, V_r \in \mathcal{C}_1$. Comme G est un chemin, l'un de ces trois ensembles, par exemple V_p (sans perte de généralité), vérifie : $V_p \subseteq V_q \cup V_r$. Donc, nous pouvons supprimer V_p de l'ensemble $\mathcal{C}_1$ pour obtenir une couverture de V valide et de cardinalité plus petite. Nous pouvons appliquer cette suppression récursivement jusqu'à l'obtention d'une couverture $\mathcal{C}'_1 = \{V'_1, \dots V'_{k'_1}\}$, $k'_1 \leq k_1$, dans laquelle tout sommet $v \in V$ appartient au plus à deux ensembles. Supposons que les ensembles V'_i sont ordonnés en fonction de leurs sommets les plus à gauche dans le chemin G. Nous construisons une partition $\mathcal{P} = \{V''_i \ : \ 1 \leq i \leq k'_1\}$ comme suit :

- pour tout $1 \leq i < k'_1$, $V''_i = V'_i \setminus V'_{i+1}$;
- $V''_{k'_1} = V'_{k'_1}$.

L'ensemble $\mathcal{P}$ est donc une partition de V en k'_1 parties avec $k'_1 \leq k_1$ (et donc $k'_1 \leq k$). Pour tout i, $1 \leq i < k'_1$, nous avons (i) $D[V''_i]$ est un DAG parce qu'il est un sous-graphe du DAG $D[V'_i]$, et (ii) $G[V''_i]$ est un sous-chemin de G et il est donc un sous-graphe connexe. Enfin, $D[V''_{k'_1}]$ est un DAG et $G[V''_{k'_1}]$ est connexe, parce que $V''_{k'_1} = V'_{k'_1}$ et la couverture $\mathcal{C}'$ est valide. Par conséquent, la partition $\mathcal{P}$ est valide et donc la réponse au problème Min-DAGCC-Partition avec l'instance (D, G) est OUI.

□

Théorème 3.8 *Le problème* Min-DAGCC-Cover *peut se résoudre en temps polynomial quand G est un chemin.*

Preuve. Le graphe G est un chemin, donc le lemme précédent (Lemme 3.1) implique que la réponse au problème k-DAGCC-Partition avec une instance (D, G) est OUI si et seulement si la réponse au problème k-DAGCC-Cover avec l'instance (D, G) est OUI. Donc, le nombre d'éléments (parties) dans une partition minimale est égal au nombre d'éléments d'une couverture minimale. Or, toute partition est une couverture, donc quand G est un chemin, toute solution de Min-DAGCC-Partition est une solution de Min-DAGCC-Partition. Par ailleurs, le problème Min-DAGCC-Cover peut également se résoudre en temps polynomial quand G est un chemin (voir le Théorème 3.3), donc dans ce cas le problème Min-DAGCC-Cover peut se résoudre en temps polynomial.

□

Maintenant nous allons montrer que, contrairement au problème k-DAGCC-Partition, le problème k-DAGCC-Cover est NP-complet même si le graphe G est une étoile et k est une constante.

Théorème 3.9 *Pour tout $k \geq 3$, k-*DAGCC-Cover *est* NP*-complet, même si G est une étoile.*

Preuve. Évidemment le problème k-DAGCC-Cover est dans NP. Pour prouver la NP-complétude, nous proposons une réduction du problème Min-CN en considérant sa version de décision naturelle, définie comme suit : étant donnés un graphe non-orienté $G_C = (V_C, E_C)$ et un entier

k, existe-t-il une coloration propre de G_C utilisant au plus k couleurs ? Il a été démontré que ce problème est NP-difficile pour toute constante $k \geq 3$ [GJ79].

Soit $G_C = (V_C, E_C)$ une instance du problème MIN-CN. Nous allons construire $D = (V, A)$ et $G = (V, E)$ comme suit :

- $V = V_C \cup \{v_r\}$;
- $A = \{vv',\ v'v\ :\ (v, v') \in E_C\} \cup \{vv_r\ :\ v \in V_C\}$;
- $E = \{v_r v\ :\ v \in V_C\}$.

Le graphe D est obtenu à partir de G_C en remplaçant toute arête par deux arcs dans des directions opposées, et en ajoutant un arc partant de v_r vers tout sommet de $V \setminus \{v_r\}$. Le graphe G est une étoile de racine v_r. Une illustration pour cette construction est montrée dans la Figure 3.5.

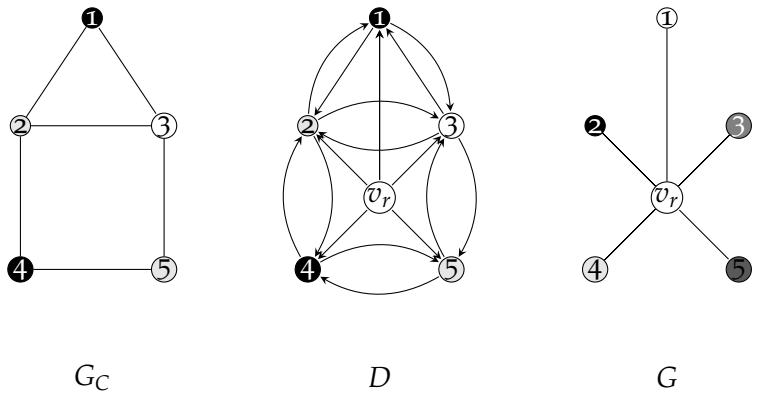


FIGURE 3.5 – *Exemple de construction des graphes D et G à partir d'une instance G_C du problème* MIN-CN. *Le graphe G_C est coloré en utilisant trois couleurs. A partir de cette coloration propre, on obtient une couverture valide de V composée des trois sous-ensembles : $\{1, 4, v_r\}$, $\{2, 5, v_r\}$ et $\{3, v_r\}$.*

Nous allons prouver la propriété suivante : il existe une coloration propre de G_C utilisant k couleurs si, et seulement si, il existe une couverture $\mathcal{C} = \{V_1, V_2 \dots V_k\}$ de V valide et composée de k éléments.
$\Rightarrow$: Considérons une coloration propre de G_C utilisant k couleurs. On note par S_i, $1 \leq i \leq k$, l'ensemble de sommets dont la couleur est égale à i. A partir des ensembles S_i, on calcule une couverture $\mathcal{C} = \{V_1, V_2, \dots, V_k\}$ de V comme suit : pour tout $1 \leq i \leq k$, $V_i = S_i \cup \{v_r\}$. Par définition d'une coloration propre, chaque S_i est un stable dans G_C (et donc dans D). Le degré entrant de v_r dans D est égal à zéro, donc pour tout $1 \leq i \leq k$, $D[V_i]$ est un DAG. Par ailleurs, pour tout $1 \leq i \leq k$, $G[V_i]$ est connexe parce que (i) G est une étoile de centre v_r, et (ii) V_i contient le sommet v_r. Par conséquent, la couverture $\mathcal{C}$ est valide.
$\Leftarrow$: Soit $\mathcal{C} = \{V_1, V_2 \dots V_k\}$ une couverture valide de V. On attribue la couleur i à tout sommet $v \in V_i \setminus \{v_r\}$, pour tout $1 \leq i \leq k$. Cette coloration est une coloration propre parce que, pour tout $1 \leq i \leq k$, $D[V_i]$ est un DAG et donc V_i est un stable dans G_C.

□

Dans la section précédente (Théorème 3.6), nous avons montré un ré-

sultat de non-approximabilité pour le problème Min-DAGCC-Partition quand G est un graphe général. Nous allons montrer le même résultat pour Min-DAGCC-Cover, mais pour des instances plus restreintes, en l'occurrence quand G est une étoile.

Théorème 3.10 *Le problème* Min-DAGCC-Cover *est non-approximable avec un facteur* $n^{1-\epsilon}$, *pour tout* $\epsilon > 0$, *même si* G *est une étoile.*

Preuve. La réduction proposée dans la preuve du Théorème 3.9 est une L-réduction. En effet, la taille des solutions est identique dans les deux problèmes, car nous avons montré que la propriété suivante est satisfaite : il existe une coloration propre de G_C utilisant k couleurs si, et seulement si, il existe une couverture $\mathcal{C} = \{V_1, V_2 \dots V_k\}$ valide de V. Donc, étant donné un algorithme d'approximation pour Min-DAGCC-Partition, on peut obtenir une approximation pour Min-DAGCC-Cover avec le même ratio d'approximation. Or, pour tout $\epsilon > 0$, le problème Min-DAGCC-Partition est non-approximable avec un facteur de $n^{1-\epsilon}$ [Zuco7]. Par conséquent, le même résultat de non-approximabilté est aussi valide pour Min-DAGCC-Cover, même si G est une étoile.

□

Conclusion du chapitre

Dans ce chapitre, nous avons étudié un problème de décomposition des graphes. Étant donné un graphe orienté $D = (V, A)$ et un graphe non-orienté $G = (V, E)$ ayant le même ensemble de sommets, le problème étudié consiste à chercher une décomposition de V en k sous-ensembles V_1, V_2, …, V_k tels que pour tout i (i) $D[V_i]$ est un DAG et $G[V_i]$ est connexe. La décomposition proposée peut être considérée comme étant un prétraitement pour notre approche présentée dans le chapitre précédent (Chapitre 2) pour comparer deux réseaux biologiques hétérogènes.

Nous avons étudié deux versions de décomposition : une version de partition et une version de couverture. Nous avons identifié des instances difficiles et des instances faciles pour ces problèmes en considérant certaines classes de graphes D et G. Cependant, reste encore des questions ouvertes. Premièrement, nous voulons étudier la complexité paramétrée du problème k-DAGCC-Partition quand G est un arbre et quand le paramètre est par exemple le nombre de parties. On sait que ce problème n'est pas FPT quand G est un graphe général, parce que nous avons montré qu'il est NP-complet pour $k = 2$. Il est également intéressant de chercher des classes de graphes spécifiques pour lesquelles les problème Min-DAGCC-Cover ou Min-DAGCC-Partition deviennent approximables avec un facteur constant.

Enfin, nous avons imposé la connexité des sous-graphes $G[V_i]$, pour retrouver des protéines en interaction, dans le cas où G représente un réseau PPI. Pour la même raison, dans le cas où D représente un réseau métabolique, il serait aussi intéressant d'ajouter la contrainte de connexité des sous-graphes $D^*[V_i]$, où D^* est le graphe support de D, pour retrouver des voies métaboliques "connexes". Le problème résultant de l'ajout de

cette contrainte est donc défini comme suit : étant donné un entier k, est il possible de décomposer V en k' sous-ensembles $V_1, V_2, \ldots V_{k'}$, $k' \leq k$ tels que, pour tout $1 \leq i \leq k'$, (i) $D[V_i]$ est un DAG, (ii) $G[V_i]$ est connexe, et (iii) $D^*[V_i]$ est connexe ? A notre connaissance, ce dernier problème n'a pas été étudié.

Orientation de graphes

4

Une partie des travaux présentés dans ce chapitre a été publiée dans les actes de la 13e conférence nationale JOBIM 2012 (Journées Ouvertes en Biologie, Informatique et Mathématiques) [FMBR12b]. La version complète a été soumise au journal Theoretical Computer Science (TCS).

Sommaire

Ce chapitre est consacré à l'étude des réseaux physiques que nous avons introduits brièvement dans le premier chapitre.

Nous rappelons que les facteurs de transcription sont des protéines qui régulent la transcription des gènes. La régulation est réalisée par une fixation du facteur de transcription sur une partie du gène cible, appelée cis-régulateur, et cette liaison est appelée interaction protéine-ADN (voir la Section 1.1.3 du Chapitre 1).

Un *réseau physique* [YIJ04] représente un ensemble d'interactions protéine-protéine et d'interactions protéine-ADN. La modélisation la plus expressive d'un tel réseau est un graphe mixte $G = (V_1 \cup V_2, E, A_1 \cup A_2)$ dans lequel :

- V_1 est un ensemble de protéines ;
- V_2 est un ensemble de gènes ;
- A_1 est l'ensemble des arcs représentant les interactions protéine-ADN, c'est-à-dire, un arc va d'une protéine P ($P \in V_1$) vers un gène g ($g \in V_2$) si P est un facteur de transcription du gène g ;

- A_2 est formé par l'ensemble des arcs gP ($P \in V_1$ et $g \in V_2$) tels que la protéine P est un produit du gène g ;
- E est l'ensemble des arêtes représentant les interactions protéine-protéine.

Ce modèle est souvent réduit [GSS10, SES11, ESD+11] à un graphe mixte $G = (V, E, A)$ qui contient un seul type de sommets qui sont les protéines. L'ensemble d'arêtes E représente les interactions protéine-protéine. Les arcs A de ce graphe sont des arcs "indirects" construits de la manière suivante : il y a un arc P_iP_k s'il existe un gène g_j tel que (i) P_i est un facteur de transcription de g_j et (ii) P_k est un produit du gène g_j. La Figure 4.1 illustre cette construction.

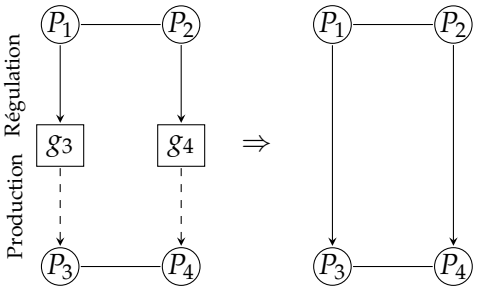


FIGURE 4.1 – *Modélisation des réseaux physiques par des graphes mixtes. Le graphe G_1 est une modélisation dans laquelle il y a deux types de sommets (les gènes et les protéines). Le graphe G_2 est obtenu à partir du graphe G_1 en supprimant les gènes et en remplaçant tout chemin orienté $P_ig_jP_k$ dans G_1 par un arc P_iP_k dans G_2. L'arc P_ig_j signifie que la protéine P_i est un facteur de transcription du gène g_j. L'arc g_jP_k signifie que le gène g_j produit la protéine P_k.*

Les réseaux physiques sont les supports de la communication entre une cellule et son environnement extérieur. En effet, les événements se propagent de l'extérieur vers l'intérieur de la cellule via une cascade d'activations/désactivations de protéines. Ces cascades sont appelées *voies de signalisation* et elles correspondent à des chemins, dans le réseau physique, partant d'une protéine *source* (dite *cause* de l'événement) vers une protéine *terminale* (dite *effet* de l'événement).

Dans ce chapitre, nous étudions le problème de l'inférence des voies de signalisation, en combinant les informations sur les causes et sur les effets [MBZS08]. L'inférence de ces voies permet d'expliquer les causes de certaines maladies. Par exemple, certains types de diabète sont causés par des erreurs de signalisation dans la voie de l'insuline (cette voie est illustrée dans la Figure 4.2).

Dans ce chapitre, nous allons d'abord présenter les différentes méthodes qui ont été proposées en littérature (Section 4.1), et qui sont basées sur l'orientation des graphes. Nous allons ensuite étudier certains de ces problèmes et nous proposerons également une nouvelle approche.

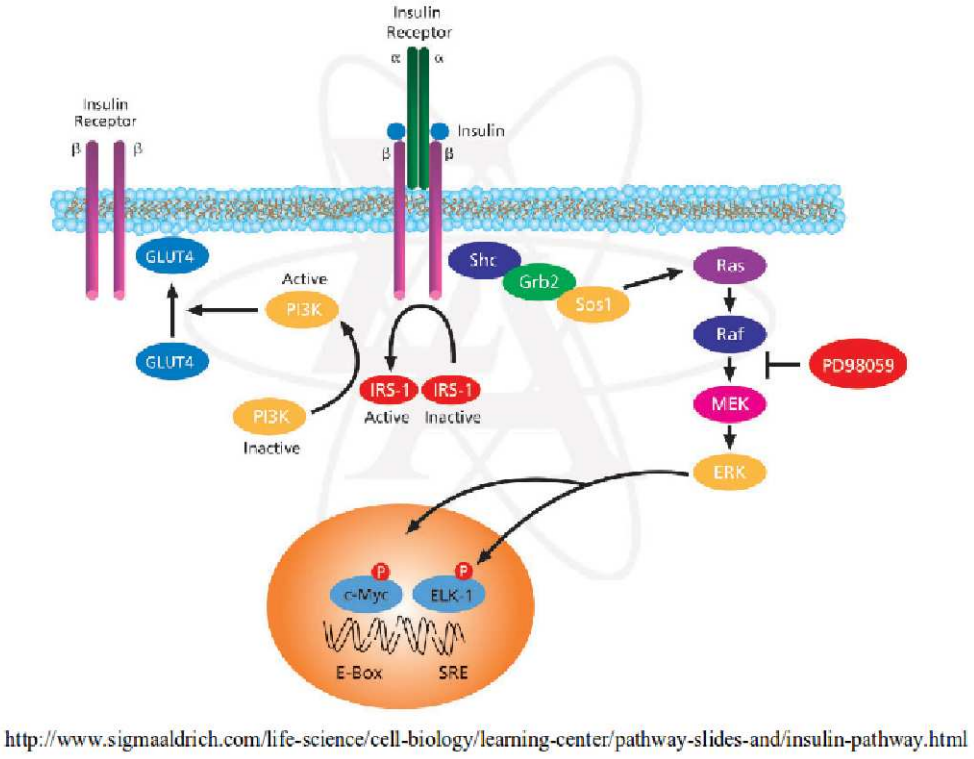


FIGURE 4.2 – *Voie de signalisation de l'insuline. La "cause" de cette voie est l'arrivée de l'insuline et l'"effet" est la régulation du niveau du sucre dans le sang.*

4.1 ÉTAT DE L'ART SUR L'ORIENTATION DE GRAPHES

Sharan et al. [MBZS08] sont parmi les premiers à avoir modélisé l'inférence des voies de signalisation par un problème d'orientation de graphes non-orientés. Le problème est appelé MAXIMUM GRAPH ORIENTATION (MGO). Gumzu et al. [GSS10] ont généralisé MGO pour les graphes mixtes.

Avant de présenter ce problème, nous donnons quelques définitions.

Définition 4.1 *Soit $G = (V, E, A)$ un graphe mixte. Une* orientation simple *de G est un graphe orienté G' construit à partir de G en remplaçant toute arête $(u, v) \in E$ par* un seul *arc $(uv$ ou $vu)$.*

Étant donnés un graphe mixte $G = (V, E, A)$ et une orientation simple G' du graphe G, une paire de sommets $(u, v) \in V \times V$ est dite *satisfaite* dans G' s'il existe un chemin (orienté) dans G' de u à v.

Le problème MGO est le problème de maximisation formulé comme suit.

MGO [GSS10]
Instance : Un graphe mixte $G = (V, E, A)$ et un ensemble $\mathcal{P} \subseteq V \times V$ de paires de sommets source-terminal.
Solution : Une orientation simple $G' = (V, A')$ de G.
Mesure : Le nombre de paires $(s, t) \in \mathcal{P}$ satisfaites dans G'.

D'un point de vue biologique, les sommets sources et terminaux dans la formulation du problème MGO correspondent respectivement aux causes et aux effets des événements extra-cellulaires.

Nous présentons maintenant les principaux articles portant sur l'orientation des graphes. Pour chaque article, nous présentons le problème étudié et quelques résultats sur sa complexité.

4.1.1 MGO et ses variantes

Le problème MGO

Sharan et al. [MBZS08] ont montré que le problème MGO peut se résoudre en un temps polynomial ($\mathcal{O}(n^3)$) quand G est un chemin non-orienté, où n est le nombre de sommets de G. Ils ont montré également que le problème est NP-complet même si G est une étoile non-orientée. En terme d'approximabilité, ils ont montré que MGO est non-approximable avec un ratio de $12/11$, même si le graphe G est un arbre non-orienté. Cependant, ils ont proposé un algorithme d'approximation dont le ratio est de $\mathcal{O}(\log\ n)$, lorsque G est un graphe non-orienté d'ordre n.

Gumzu et al. [GSS10] ont amélioré le ratio d'approximation $\mathcal{O}(\log\ n)$ (proposé dans [MBZS08]) en $\mathcal{O}(\log\ n\ /\log\log\ n)$. L'algorithme proposé a été adapté pour orienter les graphes mixtes avec certaines contraintes topologiques motivées par des hypothèses biologiques. Silverbush et al. [SES11] ont résolu le problème MGO à l'aide de la programmation linéaire en nombres entiers. Ils ont appliqué leur méthode pour orienter le réseau physique de la levure *S. cerevisiae*.

Elberfeld et al. [ESD+11] ont montré que MGO est non-approximable avec un ratio de $8/7$ quand le graphe G un graphe mixte général. Ils ont également proposé le premier algorithme d'approximation du problème MGO dans son cas général. L'algorithme proposé a un ratio de $\mathcal{O}(M^{1/\sqrt{2}}\log n)$, où $M = max\{n, |\mathcal{P}|\}$. Dorn et al. [DHK+11] ont proposé un algorithme de complexité quadratique ($\mathcal{O}(n^2)$) améliorant la complexité cubique de l'algorithme proposé par Sharan et al. [MBZS08] pour résoudre MGO quand le graphe G est un chemin non-orienté. Les auteurs ont également étudié la complexité paramétrée du problème MGO en considérant le cas particulier où G est un arbre non-orienté. Ils ont montré que dans ce cas le problème est FPT paramétré par le nombre de paires insatisfaites. Ils ont aussi proposé d'autres algorithmes FPT, pour les arbres non-orientés, avec d'autres paramètres, par exemple le paramètre m_v qui mesure le nombre des chemins P allant d'un sommet source vers un sommet terminal ($(s, t) \in \mathcal{P}$) et passant par un sommet donné $v \in V$. La complexité de l'algorithme proposé est de $\mathcal{O}(2^{m_v}.|\mathcal{P}| + n^3)$. D'un point de vue biologique, le paramètre m_v mesure le flux des signaux passant à travers une protéine donnée.

Variantes de MGO

Hakimi et al. [HSY97] ont étudié le problème d'orientation simple des graphes non-orientés dans le but de maximiser le nombre des paires $(s,t) \in V \times V$ pour lesquels il existe un chemin dans le graphe orienté allant de s à t. Ce problème est donc équivalent au problème MGO quand (i) G est un graphe non-orienté et (ii) $\mathcal{P} = V \times V$. Les auteurs ont proposé un algorithme polynomial, de complexité quadratique, pour résoudre cette variante.

Une autre variante du problème MGO, que nous appelons dans ce chapitre Simple-GraphOrientation (S-GO), a été étudiée en 1989 par Hassin et al. [HM89]. Le problème S-GO est la variante de MGO où toutes les paires de $\mathcal{P}$ doivent être satisfaites, c'est-à-dire on demande qu'il existe un chemin dans le graphe orienté G', de s à t, pour tout $(s,t) \in \mathcal{P}$. Les auteurs ont proposé un algorithme polynomial pour résoudre S-GO quand G est un graphe non-orienté. La complexité de l'algorithme proposé est de $\mathcal{O}(|\mathcal{P}| \times |E|)$. Arkin et al. [AH02] ont montré que le problème S-GO devient NP-complet quand G est un graphe mixte général. Pour le cas $|\mathcal{P}| = 2$, ils ont proposé un algorithme polynomial.

4.1.2 D'autres problèmes d'orientation de graphes

Robbins a étudié en 1939 [Rob39] un problème d'orientation de graphes non-orientés, motivé par des applications en contrôle de trafic routier. Dans ce contexte, un réseau routier est modélisé par un graphe non-orienté $G = (V,E)$ où les sommets sont les villes et les arêtes représentent les routes. La question qui a été étudiée par Robbins est la suivante : est-il possible de trouver une orientation simple de G telle que le graphe orienté obtenu G' soit fortement connexe, c'est-à-dire, pour tout $u,v \in V$, il existe dans G' un chemin de u à v et un chemin de v à u ? Robbins a démontré la propriété suivante.

Propriété 4.1 *[Rob39] Un graphe non-orienté et connexe $G = (V,E)$ admet une orientation simple G' telle que le graphe G' soit fortement connexe si, et seulement si, G ne contient aucun pont, c'est-à-dire, G ne contient aucune arête e dont la suppression augmente le nombre de composantes connexes de G.*

La question étudiée par Robbins [Rob39] a été généralisée par Boesch et Tindell [BT80] pour les multigraphes mixtes. Ils ont démontré qu'un multigraphe mixte M admet une orientation simple M' telle que M' est fortement connexe si, et seulement si, (i) pour toute paire de sommets (u,v), il y a dans M un chemin allant de u à v et un chemin allant de v à u et (ii) M^* ne contient aucun pont, où M^* est le graphe support de M.

Chartrand et al. [CHSW94] ont étudié le problème d'une orientation dite *unilatérale* : étant donné un graphe non-orienté $G = (V,E)$, existe-il une orientation simple de G, notée G', telle que pour tout $u,v \in V$, il existe un chemin orienté dans G' reliant u et v (allant de u vers v ou de v vers u) ? Ils ont prouvé qu'un graphe non-orienté et connexe G admet une orientation unilatérale si, et seulement si, il existe un chemin dans G qui contient tous les ponts de G.

Récemment, le problème de recherche d'une orientation unilatérale a

été étudié pour les graphes mixtes. Mchedlidze et al. [MS10] ont proposé un algorithme linéaire pour vérifier si un graphe mixte admet une orientation unilatérale. Si le graphe admet une orientation unilatérale, celle-ci peut être également construite en un temps linéaire.

4.2 Notre contribution

Dans ce chapitre, nous étudions la complexité du problème S-GO en considérant certaines restrictions sur les instances en entrée (par exemple le degré de G ou la cardinalité de $\mathcal{P}$). Nous introduisons également un problème de minimisation, appelé Min-Double-GraphOrientation (MIN-D-GO), qui peut être vu comme étant une variante de S-GO dans laquelle nous autorisons que certaines arêtes de G soient *doublement orientées* (c'est-à-dire remplacées par deux arcs dans deux directions opposées). Dans un contexte biologique, une arête doublement orientée reflète la présence d'une réaction réversible. En outre, dans un système biologique stable, la plupart des réactions tendent à être irréversibles [Luco8]. C'est pour cette raison que, dans le problème MIN-D-GO, nous cherchons à minimiser le nombre d'arêtes doublement orientées.

Le reste de ce chapitre est organisé comme suit. D'abord nous formulons les problèmes S-GO et MIN-D-GO dans la Section 4.3. Ensuite, nous montrons dans la Section 4.4 que pour les deux problèmes nous pouvons supposer, sans perte de généralité, que le graphe G est un graphe mixte acyclique (ou MAG, pour Mixed Acyclic Graph). Dans la Section 4.5, nous étudions la complexité du problème S-GO. Enfin, nous étudions la complexité du problème MIN-D-GO dans la Section 4.6.

4.3 Formulation des problèmes

Tout au long de chapitre, $G = (V, E, A)$ est un graphe mixte sans boucles et dont les arcs et les arêtes sont simples. Un *chemin* P, dans G, allant d'un sommet v_1 vers un sommet v_m est un sous-graphe $P = v_1v_2 \dots v_{m-1}v_m$ tel que, pour tout $1 \leq i \leq m-1$, $(v_i, v_{i+1}) \in E$ ou $v_iv_{i+1} \in A$. Un *cycle* dans G est un chemin $v_1v_2 \dots v_{m-1}v_m$ tel que $v_1 = v_m$. Un cycle $v_1v_2 \dots v_{m-1}v_m$ dans G est dit *circuit* si $v_iv_{i+1} \in A$ pour tout $1 \leq i \leq m-1$.

Définition 4.2 (MAG) *Un* MAG *(*Mixed Acyclic Graph*) est un graphe mixte qui ne contient aucun cycle (et par conséquent aucun circuit).*

Un exemple de MAG est montré dans la Figure 4.3.

Propriété 4.2 *Soit $G = (V, E, A)$ un MAG. Le graphe non-orienté (V, E) est une forêt, et le graphe orienté (V, A) est un DAG.*

Informellement, un MAG peut être vu comme étant un ensemble de super-nœuds N_i tels que chaque N_i contient un arbre T_i, et le graphe orienté obtenu en fusionnant les sommets de chaque super-nœud en unique sommet est un DAG. Voir les Figures 4.3 et 4.4.

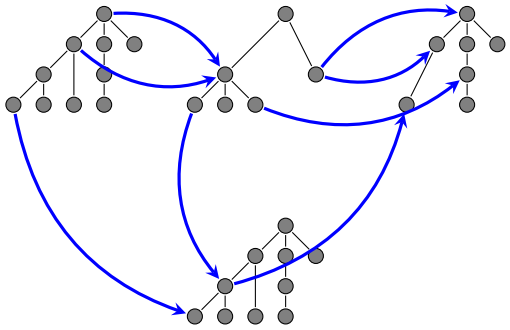


FIGURE 4.3 – *Exemple de MAG.*

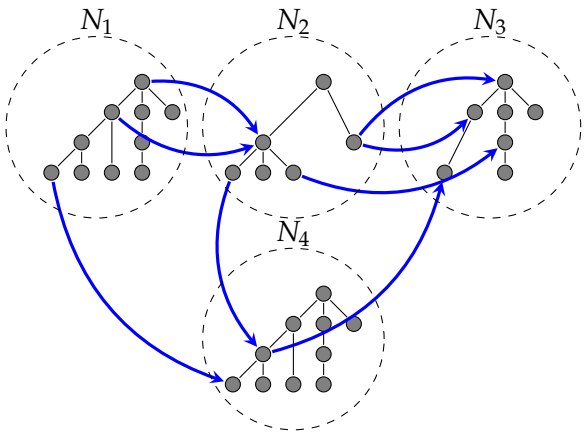


FIGURE 4.4 – *Le MAG (V, E, A) montré dans la Figure 4.3, peut être vu comme étant un ensemble de super-nœuds N_i contenant chacun un arbre, et reliés entre eux par des arcs tel que (V, A) est un DAG.*

Définition 4.3 *Une* orientation *d'un graphe mixte $G = (V, E, A)$ est un graphe orienté G' obtenu de G en remplaçant chaque arête $(u, v) \in E$ par (i) un arc uv, ou (ii) un arc vu, ou par (iii) uv et vu simultanément. Une arête (u, v) remplacée par les deux arcs uv et vu est appelée arête* doublement orientée.

Une orientation simple (voir la Définition 4.1) est donc une orientation qui ne contient aucune arête doublement orientée. Une paire de sommets $(u, v) \in V \times V$ est dite *satisfaite* dans une orientation G' s'il existe un chemin (orienté) dans G' de u à v.

Soit $P = v_1 v_2 \dots v_{m-1} v_m$ un chemin dans G. Tout au long de ce chapitre, quand nous écrivons *l'orientation de P de v_1 vers v_m*, nous faisons référence à l'orientation qui remplace toute arête dans P de la forme (v_i, v_{i+1}), $1 \leq i \leq m-1$, par l'arc $v_i v_{i+1}$.

Définition 4.4 ($\mathcal{P}$-connexité) *[AH02] Soit $G = (V, E, A)$ et soit $\mathcal{P} \subseteq V \times V$ un ensemble de paires de sommets. Le graphe G est dit* $\mathcal{P}$-connexe *si pour tout $(u, v) \in \mathcal{P}$, il existe un chemin dans G de u à v.*

Définition 4.5 ($\mathcal{P}$-orientation) *[AH02] Soient $G = (V, E, A)$ et $\mathcal{P} \subseteq V \times V$ tel que G est $\mathcal{P}$-connexe. Une orientation G' de G est dite* $\mathcal{P}$-orientation *si les deux conditions suivantes sont vérifiées : (i) G' est une orientation simple de G et (ii) G' satisfait toutes les paires de $\mathcal{P}$ (c'est-à-dire G' est $\mathcal{P}$-connexe).*

Nous appelons S-GO le problème de décision qui consiste à vérifier si un graphe mixte G donné possède une $\mathcal{P}$-orientation.

S-GO [HM89, AH02]
Instance : Un graphe mixte $G = (V, E, A)$ et un ensemble de paires $\mathcal{P} \subseteq V \times V$ tel que G est $\mathcal{P}$-connexe.
Question : G possède-t-il une $\mathcal{P}$-orientation ?

Par analogie avec les $\mathcal{P}$-orientations, nous définissons un autre type d'orientations, noté $(\mathcal{P}, k)$-*D-orientation*, autorisant la création d'arêtes doublement orientées.

Définition 4.6 ($(\mathcal{P}, k)$-*D-orientation*) *Soit $G = (V, E, A)$ et soit $\mathcal{P} \subseteq V \times V$ un ensemble de paires de sommets tel que G est $\mathcal{P}$-connexe. Soit k un entier, $k \geq 0$. Une* $(\mathcal{P}, k)$-D-orientation *de G est une orientation G' vérifiant les deux conditions suivantes :*

- *G' contient exactement k arêtes doublement orientées ;*
- *G' satisfait toutes les paires de $\mathcal{P}$.*

Nous pouvons maintenant formuler le problème de minimisation MIN-D-GO.

MIN-D-GO
Instance : Un graphe mixte $G = (V, E, A)$ et un ensemble de paires $\mathcal{P} \subseteq V \times V$ tel que G est $\mathcal{P}$-connexe.
Solution : Une $(\mathcal{P}, k)$-D-orientation de G.
Mesure : k.

Pour illustrer les $\mathcal{P}$-orientations et $(\mathcal{P}, k)$-D-orientations, nous montrons quelques exemples dans la Figure 4.5.

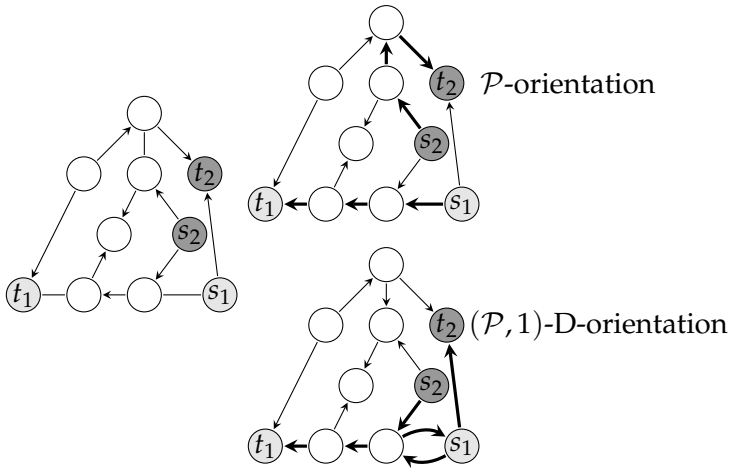


FIGURE 4.5 – *Exemple d'une $\mathcal{P}$-orientation et $(\mathcal{P}, 1)$-D-orientation pour le même graphe mixte à gauche. Ici, $\mathcal{P} = \{(s_1, t_1), (s_2, t_2)\}$.*

4.4 Réduction aux graphes mixtes acycliques (MAGs)

Il a été démontré dans [SES11] qu'en partant d'une instance $(G_1, \mathcal{P}_1)$ du problème MGO (défini dans l'introduction de ce chapitre), on peut construire une instance équivalente $(G_2, \mathcal{P}_2)$ telle que G_2 est un MAG.

Propriété 4.3 *[SES11] Soit $G_1 = (V_1, E_1, A_1)$ un graphe mixte et soit $\mathcal{P}_1 \subseteq V \times V$ un ensemble de paires. On peut construire un MAG $G_2 = (V_2, E_2, A_2)$ et un ensemble $\mathcal{P}_2 \subseteq V_2 \times V_2$, avec $|\mathcal{P}_2| = |\mathcal{P}_1|$, tel que, pour tout entier $\alpha \geq 0$, il existe une orientation simple de G_1 satisfaisant α paires dans $\mathcal{P}_1$ si, et seulement si, il existe une orientation simple de G_2 satisfaisant α paires dans $\mathcal{P}_2$.*

En appliquant la Propriété 4.3 avec $\alpha = |\mathcal{P}_1|$, on en déduit que G_1 admet une $\mathcal{P}_1$-orientation si et seulement si G_2 admet une $\mathcal{P}_2$-orientation. Par conséquent, dans le problème S-GO, nous pouvons toujours, sans perte de généralité, considérer que le graphe en entrée est un MAG.

Deux instances $(G_1, \mathcal{P}_1)$ et $(G_2, \mathcal{P}_2)$ de MIN-D-GO sont dites *équivalentes* dans le cas suivant : il existe une solution optimale pour MIN-D-GO avec l'instance $(G_1, \mathcal{P}_1)$ qui crée k arêtes doublement orientées si, et seulement si, il existe une solution optimale pour le même problème avec l'instance $(G_2, \mathcal{P}_2)$ qui crée également k arêtes doublement orientées.

Comme dans le problème S-GO, nous allons montrer que dans le problème MIN-D-GO nous pouvons aussi considérer, sans perte de généralité, que le graphe en entrée est un MAG. Dans un premier temps, nous montrons dans la Propriété 4.4 que, pour tout graphe mixte et $\mathcal{P}$-connexe G, et pour tout cycle C dans G, nous pouvons trouver une solution optimale de MIN-D-GO avec l'instance $(G, \mathcal{P})$ de sorte que, dans l'orientation, le cycle C devient un circuit. Ensuite, nous montrons dans la Propriété 4.6 que les sommets de chaque circuit peuvent être contractés en un seul sommet, selon un ordre maîtrisé sur les circuits, sans changer la nature du problème. La preuve que nous proposons pour cette propriété est proche à celle proposée dans [SES11].

Propriété 4.4 (Orientation des cycles) *Soient $G = (V, E, A)$ un graphe mixte et $\mathcal{P} \subseteq V \times V$ un ensemble de paires de sommets. Supposons que G est $\mathcal{P}$-connexe. Soit C un cycle dans G. Soit G' un graphe mixte obtenu à partir de G en orientant toutes les arêtes de C dans la même direction (une telle direction est choisie arbitrairement quand $A(C) = \emptyset$) pour obtenir un circuit C'. Alors, les instances $(G, \mathcal{P})$ et $(G', \mathcal{P})$ du problème* MIN-D-GO *sont équivalentes.*

Preuve. Nous voulons montrer qu'il existe une solution pour MIN-D-GO avec l'instance $(G, \mathcal{P})$ qui crée k arêtes doublement orientées si, et seulement si, il existe une solution pour le même problème avec l'instance $(G', \mathcal{P})$ qui crée également k arêtes doublement orientées.
$\Rightarrow$: D'abord, comme G est $\mathcal{P}$-connexe, il doit exister une solution optimale H pour l'instance $(G, \mathcal{P})$. Soit H' une orientation de G' telle que les arêtes dans $E(G') \backslash E(C)$ sont remplacées par les mêmes arcs que dans H (voir la Figure 4.6 pour une illustration).

Nous montrons maintenant que H' est une solution optimale, pour MIN-D-GO avec l'instance $(G', \mathcal{P})$, et donc ayant le même nombre d'arêtes

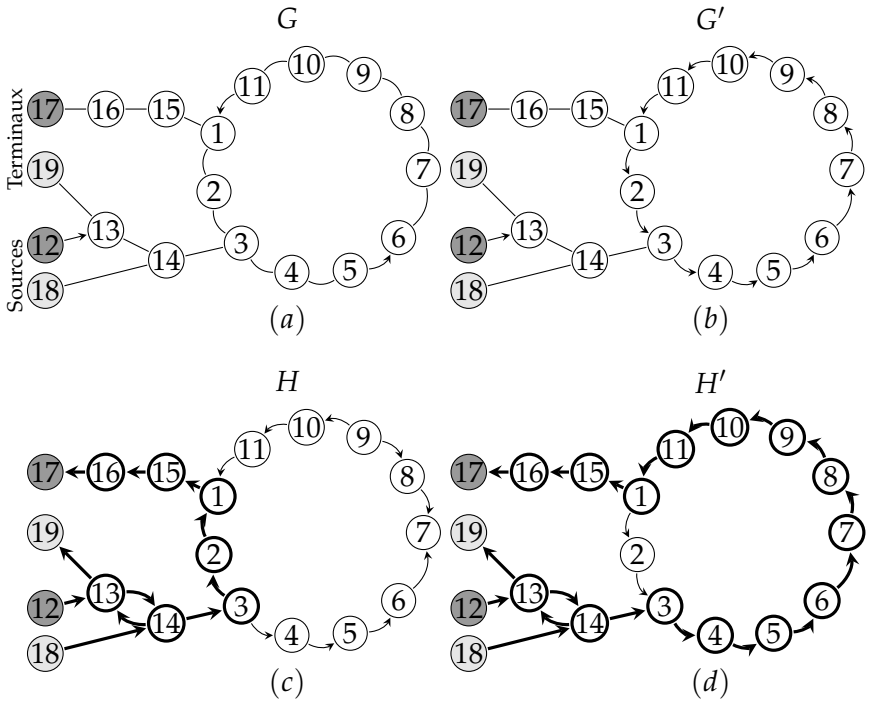


FIGURE 4.6 – *(a)* $(G, \mathcal{P})$ *est une instance du problème* MIN-D-GO *avec* $\mathcal{P} = \{(12, 17), (18, 19)\}$, *C est un cycle dans G dont l'ensemble de sommets est* $\{1, 2, \ldots, 11\}$. *(b)* G' *est un graphe mixte obtenu de G en orientant les arêtes du cycle C dans la même direction pour obtenir un circuit (c) H est une solution optimale de* MIN-D-GO, *avec l'instance* $(G, \mathcal{P})$, *qui crée une seule arête doublement orientée. (d)* H' *est une solution optimale pour le même problème avec l'instance* $(G', \mathcal{P})$ *qui crée aussi une seule arête doublement orientée. Les sommets remplis dans H (resp. dans* H'*) induisent un chemin satisfaisant la paire* $(12, 17)$.

doublement orientées que H. Soit $(u, v) \in \mathcal{P}$. Si $u, v \in V(C)$ alors évidemment la paire (u, v) est satisfaite dans H' par un chemin dans le circuit C'. Si $u \notin V(C)$ ou $v \notin V(C)$, alors on considère le chemin $P = a_1 a_2 \ldots a_m$ dans H, de u à v ($u = a_1$ et $v = a_m$), qui satisfait la paire (u, v). On pose $x = min\ \{i : a_i \in V(C)\}$ et $y = max\ \{i : a_i \in V(C)\}$. La paire (u, v) est satisfaite dans H' par le chemin $a_1 a_2 \ldots a_{x-1} Q a_{y+1} a_{y+1} \ldots a_m$ tel que Q est un sous-chemin de C' partant de a_x vers a_y (voir un exemple d'illustration dans la Figure 4.6, dans lequel $a_x = 3$ et $a_y = 1$).

Soit k (resp. k') le nombre d'arêtes doublement orientées dans H (resp. dans H'). Évidemment, on a $k' \leq k$. Par ailleurs, toute orientation de G' est aussi une orientation de G, donc l'optimalité de k implique que $k' = k$ et que H' est aussi une solution optimale pour MIN-D-GO avec l'instance $(G', \mathcal{P})$.

$\Leftarrow$: Le graphe G' est $\mathcal{P}$-connexe parce que G est $\mathcal{P}$-connexe et l'orientation du cycle n'a pas d'impact sur la $\mathcal{P}$-connexité. Donc, il doit exister une solution optimale G'_{opt} pour MIN-D-GO avec l'instance $(G', \mathcal{P})$. Soit k' le nombre d'arêtes doublement orientées dans G'_{opt}. Le graphe G'_{opt} est donc une $(\mathcal{P}, k')$-D-orientation de G. Montrons que G'_{opt} est une solution optimale pour MIN-D-GO avec l'instance $(G, \mathcal{P})$. Supposons, par contradiction, qu'il existe une $(\mathcal{P}, k)$-D-orientation de G, notée H, telle que $k < k'$. Comme nous l'avons déjà fait dans le sens " $\Rightarrow$", nous pouvons construire

à partir de H un graphe H' tel que H' est une $(\mathcal{P}, k)$-D-orientation de G'. Cela contredit la minimalité de k'.

□

Soit $G = (V, E, A)$ un graphe mixte et soit $\mathcal{P} \subseteq V \times V$ un ensemble de paires de sommets. Soit $G_1 = (V_1, E_1, A_1)$ le graphe mixte obtenu à partir de G en appliquant la procédure suivante appelée RÉDUIRE-INSTANCE :

1. $G_1 := G$;
2. **Tant que** (G_1 contient un cycle C) **faire**
   - Remplacer les arêtes de C par des arcs de sorte que C devienne un circuit ;
   - Appeler G_1 le graphe résultant.
3. Soit $\mathcal{P}_1$ l'ensemble des paires obtenues à partir de $\mathcal{P}$ en supprimant toutes les paires $(u, v) \in \mathcal{P}$ pour lesquelles il existe un chemin orienté dans G_1 allant de u à v.

La Propriété 4.4 implique que les deux instances $(G, \mathcal{P})$ et $(G_1, \mathcal{P})$ sont équivalentes pour tous les graphes G_1 (y compris le dernier) construits pendant la procédure. Il est clair également que $(G_1, \mathcal{P}_1)$ et $(G_1, \mathcal{P})$ sont équivalentes. L'instance $(G_1, \mathcal{P}_1)$ obtenue à partir de l'instance $(G, \mathcal{P})$ en appliquant la procédure RÉDUIRE-INSTANCE est dite instance *réduite*. Donc la propriété suivante est vérifiée.

Propriété 4.5 (Instances réduites) *Soit $(G_1, \mathcal{P}_1)$ une instance réduite, de* MIN-D-GO*, obtenue à partir de l'instance $(G, \mathcal{P})$. Les instances $(G, \mathcal{P})$ et $(G_1, \mathcal{P}_1)$ sont équivalentes.*

Propriété 4.6 (Contraction de circuits) *Soit $(G_1, \mathcal{P}_1)$ une instance réduite de* MIN-D-GO*, et soit C' un circuit dans G_1. Soit $(G_2, \mathcal{P}_2)$ l'instance de* MIN-D-GO *définie comme suit :*

- *$\mathcal{P}_2 = \mathcal{P}_1$;*
- *G_2 est le graphe obtenu de G_1 en contractant les sommets de C' en un seul sommet x_0.*

Les instances $(G_1, \mathcal{P}_1)$ et $(G_2, \mathcal{P}_2)$ sont équivalentes.

Preuve. Supposons que $G_1 = (V_1, E_1, A_1)$ et que $G_2 = (V_2, E_2, A_2)$. Montrons que les instances $(G_1, \mathcal{P}_1)$ et $(G_2, \mathcal{P}_2)$, du problème MIN-D-GO, sont équivalentes. Soit $G'_1 = (V_1, A'_1)$ une solution optimale pour MIN-D-GO avec l'instance $(G_1, \mathcal{P}_1)$ créant k_1 arêtes doublement orientées. Nous construisons une orientation G'_2, pour le graphe G_2, comme suit. Soit $(u, v) \in E_2$. Si $u \neq x_0$ et $v \neq x_0$, alors l'arête (u, v) est orientée dans G'_2 de la même façon que dans G'_1. Si $u = x_0$ (ou de façon similaire quand $v = x_0$), alors il existe un sommet $w \in V(C')$ tel que $(w, v) \in E_1$. Si $wv \in A'_1$ (resp. $vw \in A'_1$) on remplace l'arête (x_0, v), dans G_2, par l'arc x_0v (resp. vx_0).

On note par k_2 le nombre d'arêtes doublement orientées dans G'_2. Clairement, $k_1 = k_2$, parce que le circuit C' ne contient aucune arête doublement orientée.

Montrons maintenant que G'_2 satisfait toutes les paires de $\mathcal{P}_2$ (on rappelle que $\mathcal{P}_2 = \mathcal{P}_1$). Soit $(u, v) \in \mathcal{P}_1$ et soit $P'_1 = a_1a_2 \dots a_m$ un chemin orienté dans G'_1 allant de u à v ($u = a_1$ et $v = a_m$) satisfaisant la paire

(u, v). On pose $\alpha = min\ \{i : a_i \in V(C')\}$ et $\beta = max\ \{i : a_i \in V(C')\}$. Donc, la paire (u, v) est satisfaite dans G'_2 par le chemin $a_1 a_2 \dots a_{\alpha-1} x_0 a_{\beta+1} \dots a_m$.

Réciproquement, partant d'une solution optimale pour MIN-D-GO avec l'instance $(G_2, \mathcal{P}_2)$, nous pouvons, de façon similaire, construire une solution pour le même problème avec l'instance $(G_1, \mathcal{P}_1)$, en créant le même nombre d'arêtes doublement orientées. Par conséquent, la propriété est vérifiée. □

Maintenant, en utilisant les propriétés précédentes, nous sommes capables de montrer que dans le problème MIN-D-GO nous pouvons toujours et sans perte de généralité, supposer que le graphe en entrée est un MAG.

Propriété 4.7 (Réduction au MAG) *Soit $(G, \mathcal{P})$ une instance du problème* MIN-D-GO. *Il existe une instance $(G_M, \mathcal{P}_M)$ de* MIN-D-GO *équivalente à l'instance $(G, \mathcal{P})$ telle que G_M est un MAG.*

Preuve. On construit le graphe G_M et l'ensemble de paires $\mathcal{P}_M$ en utilisant la procédure suivante :

1. On construit l'instance réduite $(G_1, \mathcal{P}_1)$ obtenue à partir de $(G, \mathcal{P})$ en utilisant la procédure RÉDUIRE-INSTANCE ;
2. On construit le graphe G_2, obtenu en contractant dans G_1 chaque circuit en un seul sommet.
3. Si G_2 est un MAG alors on pose $G_M = G_2$ et $\mathcal{P}_M = \mathcal{P}_1$. Sinon, on pose $G = G_2$ et $\mathcal{P} = \mathcal{P}_1$, et on revient à l'étape 1.

L'équivalence des instances $(G_M, \mathcal{P}_M)$ et $(G, \mathcal{P})$ (et par conséquent la preuve de cette propriété) est assurée par les Propriétés 4.5 et 4.6 que nous avons prouvées précédemment. □

Dans le reste de ce chapitre, nous allons considérer que pour toute instance $(G, \mathcal{P})$ de MIN-D-GO (resp. S-GO), $G = (V, E, A)$ est un MAG et G est $\mathcal{P}$-connexe. On note également que nous pouvons supposer que le graphe G^* est connexe. En effet, si G^* n'est pas connexe, nous pouvons considérer séparément chacun des graphes $G_1, G_2, \dots, G_r$ induits dans G par les sommets des composantes connexes de G^*.

Soit $G = (V, E, A)$ un MAG et soit $\mathcal{P} = \{(s_i, t_i) \in V \times V : 1 \leq i \leq m\}$ un ensemble de paires de sommets. Pour tout entier i, $1 \leq i \leq m$, on note par n_i le nombre de chemins dans G allant de s_i à t_i. Tout au long du reste de ce chapitre, l'entier B est la valeur suivante :

$$B = \max\{n_i : 1 \leq i \leq m\}.$$

Dans les prochaines sections, nous étudions la complexité de S-GO (Section 4.5) et MIN-D-GO (Section 4.6), en considérant différentes contraintes sur les trois paramètres : $\Delta(G^*)$, B et $|\mathcal{P}|$.

Notons que, quand $B = 1$, nous pouvons résoudre facilement les problèmes S-GO et MIN-D-GO. En effet, G est $\mathcal{P}$-connexe et G^* est connexe, donc si en plus $B = 1$, alors pour toute paire $(s_i, t_i) \in \mathcal{P}$ il y a un unique chemin P_i dans G allant de s_i à t_i. Par conséquent, pour satisfaire la paire (s_i, t_i) nous devons orienter P_i de s_i vers t_i. Ensuite, on oriente chaque arête restante (u, v) en choisissant une direction arbitraire. Évidemment, dans

cette orientation nous créons un nombre minimum d'arêtes doublement orientées, et donc nous obtenons une solution optimale pour MIN-D-GO. Si la solution obtenue n'a aucune arête doublement orientée, alors on obtient une $\mathcal{P}$-orientation de G. Sinon, il n'existe aucune $\mathcal{P}$-orientation pour le graphe G. Remarquons également que le cas $\Delta(G^*) = 1$ est trivial.

Les résultats de complexité, quand $B \geq 2$ et $\Delta(G^*) \geq 2$, sont résumés dans les Tableaux 4.1 et 4.2.

Le Tableau 4.1 montre que le paramètre B définit la frontière entre les instances faciles ($B = 2$) et celles difficiles ($B = 3$) du problème de décision S-GO, même si G^* est un graphe dont le degré maximum est relativement petit ($\Delta(G^*) = 3$). D'autre part, le paramètre $\Delta(G^*)$ définit également la frontière entre les instances faciles ($\Delta(G^*) = 2$) et les instances difficiles ($\Delta(G^*) = 3$) pour le problème d'optimisation MIN-D-GO, même si $B = 2$. Dans le Tableau 4.2, nous montrons des résultats de complexité pour les deux problèmes en considérant différentes valeurs de $|\mathcal{P}|$. Les deux problèmes sont faciles quand (i) $|\mathcal{P}| \leq 2$ ou (ii) $|\mathcal{P}|$ et B sont constantes, et difficiles quand $|\mathcal{P}|$ est non borné, même si $B = 3$ (et plus généralement quand $B \geq 3$).

	$\Delta(G^*) = 2$	$\Delta(G^*) = 3$		
		$B = 2$	$B = 3$	B est non-borné
S-GO	P [Cor. 4.1]	P [Th. 4.1]	NPC [Th. 4.3]	NPC [Th. 4.3]
MIN-D-GO	P [Th. 4.4, 4.5]	APX-d [Th. 4.10]	NPC [Th. 4.8], Non-approx. [Th. 4.9]	NPC [Th. 4.8], Non-approx. [Th. 4.9], W[1]-d [Th. 4.11]

Tableau 4.1 – *Complexité de* S-GO *et* MIN-D-GO *quand G est un MAG et G^* est un graphe de degré borné. Rappelons que $B = \max\{n_i, 1 \leq i \leq |\mathcal{P}|\}$, où n_i est le nombre de chemins allant dans G de s_i à t_i. On note également que le résultat du Théorème 4.1 reste valide même si $\Delta(G^*)$ est non-borné.*

	$\|\mathcal{P}\| \leq 2$	$\|\mathcal{P}\| \geq 3$ (et $\|\mathcal{P}\| = \mathcal{O}(1)$)		$\|\mathcal{P}\|$ est borné
		$B = \mathcal{O}(1)$	B est non-borné	$B \geq 3$
S-GO	P [Arkin et al. [AH02]]	P [Th. 4.2]	Ouvert	NPC [Arkin et al. [AH02]]
MIN-D-GO	P [Th. 4.6]	P [Th. 4.7]	Ouvert	APX-d [Th. 4.10]

Tableau 4.2 – *Complexité de* S-GO *et* MIN-D-GO *pour différentes valeurs de $|\mathcal{P}|$ quand G est un MAG.*

4.5 Complexité du problème S-GO

Cette section est consacrée à l'étude de complexité du problème S-GO, en considérant les MAGs avec $\Delta(G^*)$ borné et/ou B borné (Tableau 4.1), et $|\mathcal{P}|$ borné (Tableau 4.2).

4.5.1 Algorithmes polynomiaux

Théorème 4.1 *Le problème* S-GO *peut se résoudre en temps polynomial quand G est un MAG et $B = 2$.*

Preuve. Pour toute paire $(s_i, t_i) \in \mathcal{P}$ il y a au plus deux chemins distincts, dans G, allant de s_i à t_i, et ces chemins peuvent être calculés en temps polynomial [GL09].

Si pour une paire $(s_i, t_i) \in \mathcal{P}$, il existe un seul chemin de s_i à t_i, alors nous orientons ce chemin de s_i vers t_i et ensuite nous retirons la paire (s_i, t_i) de l'ensemble $\mathcal{P}$. Nous continuons ce processus jusqu'à ce que l'un des trois cas suivants apparaisse :

1. G n'est plus $\mathcal{P}$-connexe ;
2. $\mathcal{P} = \emptyset$;
3. Pour toute paire $(s_i, t_i) \in \mathcal{P}$ il existe exactement deux chemins dans G allant de s_i à t_i.

Le premier cas implique que G n'a aucune $\mathcal{P}$-orientation. Dans le deuxième cas, pour obtenir une $\mathcal{P}$-orientation de G, il suffit d'orienter arbitrairement chaque arête, dans le graphe résultant, dans une direction unique. Enfin, dans le troisième cas, nous avons une instance du problème S-GO dans laquelle, pour tout $(s_i, t_i) \in \mathcal{P}$, il y a dans G exactement deux chemins de s_i à t_i.

On note par X_{i1} et X_{i2} les deux chemins dans G allant de s_i à t_i. Étant donnés $i, j \in \{1, 2, \ldots |\mathcal{P}|\}$, $i \neq j$, et $a, b \in \{1, 2\}$, on dit que les deux chemins X_{ia} et X_{jb} sont en *conflit* si l'orientation de X_{ia} de s_i vers t_i et de X_{jb} de s_j vers t_j crée au moins une arête doublement orientée.

Nous allons maintenant construire une instance $(\mathcal{X}, \mathcal{C})$ du problème 2-SAT, équivalente à l'instance $(G, \mathcal{P})$ du problème S-GO.

Soit $\mathcal{X} = \{x_{i1}, x_{i2} \ : \ 1 \leq i \leq |\mathcal{P}|\}$ l'ensemble des variables. Pour tout $i \in \{1, 2, \ldots, |\mathcal{P}|\}$, on ajoute à $\mathcal{C}$ la clause $c_i = (x_{i1} \vee x_{i2})$. Pour tout $i, j \in \{1, 2, \ldots, |\mathcal{P}|\}$, $i \neq j$, et $a, b \in \{1, 2\}$, on ajoute la clause $(\overline{x_{ia}} \vee \overline{x_{jb}})$ si les chemins X_{ia} et X_{jb} sont en conflit.

Nous allons montrer la propriété suivante : il existe une affectation des variables de $\mathcal{X}$ qui satisfait toutes les clauses de $\mathcal{C}$ si, et seulement si, il existe une $\mathcal{P}$-orientation pour G.

$\Rightarrow$: Considérons une affectation satisfaisant toutes les clauses de $\mathcal{C}$ et soit x_{ih_i}, $1 \leq h_i \leq 2$, un littéral de la clause c_i, $1 \leq i \leq |\mathcal{P}|$, avec la valeur `vrai`. On oriente dans G le chemin X_{ih_i}, de s_i vers t_i, pour tout i, $1 \leq i \leq |\mathcal{P}|$. Cette orientation ne peut pas créer une arête doublement orientée. Autrement, il y aurait des entiers $i, j \in \{1, 2, \ldots |\mathcal{P}|\}$, $i \neq j$, tels que les chemins X_{ih_i} et X_{jh_j} seraient en conflit, et cela impliquerait que la clause $(\overline{x_{ih_i}} \vee \overline{x_{jh_j}})$ serait insatisfaite. Pour finir l'orientation, on oriente arbitrairement, dans une unique direction, chaque arête restante.

$\Leftarrow$: Dans une $\mathcal{P}$-orientation de G, considérons l'ensemble $\{Y_{1h_1}, Y_{2h_2}, \ldots, Y_{|\mathcal{P}|h_{|\mathcal{P}|}}\}$ tel que Y_{ih_i} est un chemin (orienté) de s_i à t_i. Chaque chemin Y_{ih_i} est l'orientation (du sommet source vers le sommet terminal) d'un chemin mixte X_{ih_i}, $h_i \in \{1, 2\}$, pour tout $1 \leq i \leq |\mathcal{P}|$. On met à `vrai` les variables $\{x_{ih_i} \ : \ 1 \leq i \leq |\mathcal{P}|\}$ et on met à `faux` les variables restantes. Évidemment, cette affectation satisfait les clauses $\{c_i \ : \ 1 \leq i \leq |\mathcal{P}|\}$. Montrons maintenant que les clauses restantes sont aussi satisfaites. Supposons, par contradiction, qu'il existe une clause insatisfaite $(\overline{x_{ia}} \vee \overline{x_{jb}})$. Donc, $x_{ia} =$ `vrai` et $x_{jb} =$ `vrai`. Par conséquent, le

chemin X_{ia} (resp. X_{jb}) a été orienté dans G de s_i vers t_i (resp. de s_j vers t_j). C'est une contradiction, parce que dans une $\mathcal{P}$-orientation les arêtes doublement orientées ne sont pas autorisées et donc nous ne pouvons pas utiliser, simultanément, deux chemins en conflit.

Comme le problème 2-SAT se résout en temps polynomial [APT79], on en déduit que le problème S-GO se résout aussi en temps polynomial quand G est un MAG, et qu'il y a dans G au plus deux chemins de s_i à t_i, pour tout $(s_i, t_i) \in \mathcal{P}$. □

Corollaire 4.1 *Le problème* S-GO *peut se résoudre en temps polynomial quand G est un MAG et $\Delta(G^*) = 2$.*

Preuve. Le graphe G^* est connexe. Donc, quand $\Delta(G^*) = 2$, le graphe G^* doit être un chemin ou un cycle, et par conséquent $B \leq 2$. Si $B = 1$, le problème S-GO est trivial. Si $B = 2$, on en déduit à partir du résultat précédent (le Théorème 4.1) que le problème S-GO peut se résoudre en temps polynomial. □

Nous montrons maintenant que le problème S-GO est facile quand les deux paramètres B et $|\mathcal{P}|$ sont bornés.

Théorème 4.2 *Le problème* S-GO *peut se résoudre en temps polynomial quand G est un MAG, $|\mathcal{P}| = \mathcal{O}(1)$ et $B = \mathcal{O}(1)$.*

Preuve. Soient $G = (V, E, A)$ un MAG et $\mathcal{P} = \{(s_i, t_i), 1 \leq i \leq m\}$ un ensemble de paires de sommets source-terminal. On pose $n = |V|$. Rappelons que pour tout $(s_i, t_i) \in \mathcal{P}$ il y a dans G au plus B chemins allant de s_i à t_i. On note par χ_i l'ensemble de chemins dans G allant de s_i à t_i. Nous pouvons calculer l'ensemble χ_i en temps polynomial. En effet, on commence par la construction d'un graphe orienté G' obtenu de G en remplaçant toute arête $(u, v) \in E$ par les deux arcs uv et vu, et ensuite on calcule l'ensemble χ'_i des B plus courts chemins dans G' allant de s_i à t_i. Ce calcul peut se faire en temps de $\mathcal{O}(B(n(|E| + |A|) + n^2 \log\log n))$ [GL09]. Enfin, l'ensemble χ_i est obtenu de χ'_i en remplaçant, dans chaque chemin orienté $P' \in \chi'_i$, tout arc $uv \in A(P') \backslash A$ par l'arête (u, v).

Par ailleurs, pour satisfaire la paire (s_i, t_i) on doit choisir un chemin $P_i \in \chi_i$ et orienter ses arêtes pour créer un chemin orienté de s_i à t_i. L'orientation de P_i peut être effectuée en temps de $\mathcal{O}(n)$, parce que P_i est acyclique. Or $|\chi_i| \leq B$, donc nous pouvons considérer toutes les combinaisons possibles, en choisissant un chemin par paire, en temps de $\mathcal{O}(B^m)$. Ensuite, on oriente (du sommet source vers le sommet terminal) les m chemins obtenus, en un temps de $\mathcal{O}(mn)$. Donc, le choix des chemins et leurs orientations peuvent être effectués en temps de $\mathcal{O}(B^m \times mn))$. Maintenant, nous distinguons deux cas :

- Chacune des orientations obtenues contient des arêtes doublement orientées. Dans ce cas, le graphe G ne possède pas de $\mathcal{P}$-orientation.
- L'une des orientations obtenues ne contient aucune arête doublement orientée. Donc, une telle orientation peut être complétée en remplaçant toute arête par un seul arc, orienté dans un sens arbitraire, pour obtenir une $\mathcal{P}$-orientation de G.

□

Remarquons que la complexité de l'algorithme proposé dans la preuve du Théorème 4.2 peut s'écrire sous la forme $f(B, |\mathcal{P}|) \cdot n^{\mathcal{O}(1)}$, où f est une fonction exponentielle et indépendante de n. Donc, on en déduit que le problème S-GO est FPT paramétré par B et $|\mathcal{P}|$.

4.5.2 Résultats de difficulté

Nous avons montré dans le Théorème 4.1 que le problème S-GO est facile quand $B = 2$. En revanche, dans le théorème suivant, nous montrons que le problème devient difficile quand $B = 3$.

Théorème 4.3 *Le problème* S-GO *est* NP*-complet même si le graphe G est un MAG, $\Delta(G^*) = 3$ et $B = 3$.*

Preuve. Arkin et al. [AH02] ont présenté une preuve de NP-complexité pour S-GO quand G est un MAG général. Leur preuve est basée sur une réduction du problème de Satisfiabilité (SAT). Ici, nous modifions le MAG G construit dans leur réduction pour assurer que $\Delta(G^*) = 3$. Dans cet objectif, nous proposons une réduction du problème NP-complet 3-SAT-4 [Tov84] définie comme suit. Soit $\mathcal{X}_n = \{x_1, \dots x_n\}$ un ensemble de variables et soit $\mathcal{C}_m = \{c_1, \dots c_m\}$ un ensemble de clauses construites sur $\mathcal{X}$ telles que :

- chaque clause est une conjonction de trois littéraux ;
- chaque variable apparaît dans au plus quatre clauses.

Le but du problème 3-SAT-4 est de vérifier s'il existe une affectation des variables de $\mathcal{X}_n$ qui satisfait toutes les clauses dans $\mathcal{C}_m$.

Soit $(\mathcal{C}_m, \mathcal{X}_n)$ une instance du problème 3-SAT-4. Pour tout j, $1 \leq j \leq n$, la variable x_j satisfait la condition suivante : (1) x_j et $\overline{x_j}$ apparaissent au plus dans quatre clauses. De plus, nous pouvons considérer sans perte de généralité que (2) pour chaque variable x_j, il existe au moins une clause qui contient x_j et au moins une clause qui contient $\overline{x_j}$. Autrement, la variable x_j peut être fixée arbitrairement à `vrai` ou à `faux`.

Nous construisons maintenant une instance $(G, \mathcal{P})$ du problème S-GO (voir la Figure 4.7 pour une illustration). Pour chaque clause c_i, on crée deux sommets s_i et t_i, $1 \leq i \leq m$. Pour chaque variable x_j, on crée les 14 sommets suivants : $\{u_j, v_j\} \cup \{a_{jk}, b_{jk}, a'_{jk}, b'_{jk}\}_{1 \leq k \leq 3}$. Ensuite, on ajoute une arête (u_j, v_j) et les quatre chemins orientés : $a_{j1}a_{j2}a_{j3}u_j$, $v_j b_{j3} b_{j2} b_{j1}$, $a'_{j1}a'_{j2}a'_{j3}v_j$ et enfin $u_j b'_{j3} b'_{j2} b'_{j1}$, pour tout $1 \leq j \leq n$.

Pour chaque variable x_j, il y a k_j clauses contenant x_j et k'_j clauses contenant $\overline{x_j}$ telles que $1 \leq k_j \leq 3$, $1 \leq k'_j \leq 3$ et $k_j + k'_j \leq 4$. Soit $\{c_{i_1}, c_{i_2}, \dots, c_{i_{k_j}}\}$ (resp. $\{c_{i'_1}, c_{i'_2}, \dots, c_{i'_{k'_j}}\}$) l'ensemble des clauses qui contiennent x_j (resp. $\overline{x_j}$). On ajoute un arc $s_{i_\alpha} a_{j\alpha}$ et un arc $b_{j\alpha} t_{i_\alpha}$, pour tout $\alpha \in \{1, 2, \dots k_j\}$. On ajoute également un arc $s_{i'_\beta} a'_{j\beta}$ et un arc $b'_{j\beta} t'_{i_\beta}$, pour tout $\beta \in \{1, 2, \dots k'_j\}$. Pour finir la construction, on pose $\mathcal{P} = \{(s_i, t_i), 1 \leq i \leq m\}$.

A l'aide des conditions (1) et (2), nous pouvons facilement vérifier que $\Delta(G^*) = 3$. De plus, pour chaque paire (s_i, t_i) il y a exactement trois chemins dans G allant de s_i à t_i, parce que chaque clause de $\mathcal{C}_m$ contient exactement trois littéraux. Donc $B = 3$.

Nous allons montrer la propriété suivante : il existe une affectation qui satisfait toutes les clauses de $\mathcal{C}_m$ si, et seulement si, il existe une $\mathcal{P}$-orientation pour G.
$\Rightarrow$: Considérons une affectation qui satisfait toutes les clauses de $\mathcal{C}_m$. De façon similaire à la preuve présentée dans [AH02], si $x_j = \texttt{vrai}$ (resp. $x_j = \texttt{faux}$) alors on oriente l'arête (u_j, v_j) de u_j vers v_j (resp. de v_j vers u_j). Soit l_i un littéral de la clause c_i avec la valeur $\texttt{vrai}$. Donc, il existe une variable x_j telle que $l_i = x_j$ ou $l_i = \overline{x_j}$. Si $l_i = x_j$ (resp. $l_i = \overline{x_j}$), et par conséquent il existe un entier k_i, $1 \leq k_i \leq 3$, tel que $s_i a_{jk_i}, b_{jk_i} t_i \in A(G)$ (resp. $s_i a'_{jk_i}, b'_{jk_i} t_i \in A(G)$). Donc, la paire (s_i, t_i) est satisfaite par le chemin $s_i a_{jk_i} a_{j(k_i+1)} \dots u_j v_j b_{j3} \dots b_{jk_i} t_i$ (resp. $s_i a'_{jk_i} a'_{j(k_i+1)} \dots v_j u_j b'_{j3} \dots b'_{jk_i} t_i$).
$\Leftarrow$: Étant donnée une $\mathcal{P}$-orientation G' de G, on met la variable x_j à $\texttt{vrai}$ (resp. à $\texttt{faux}$) si l'arête $u_j v_j \in A(G')$ (resp. $v_j u_j \in A(G')$). Soit c_i une clause de $\mathcal{C}_m$. La paire (s_i, t_i) est satisfaite par un chemin (orienté) P dans G', allant de s_i à t_i et passant par un arc $u_j v_j$ ou un an arc $v_j u_j$. Si P contient l'arc $u_j v_j$, alors la clause c_i doit contenir le littéral x_j et donc c_i est satisfaite. Si P contient l'arc $v_j u_j$ (et par conséquent $x_j = \texttt{faux}$), la clause c_i doit contenir le littéral $\overline{x_j}$ et donc c_i est aussi satisfaite. □

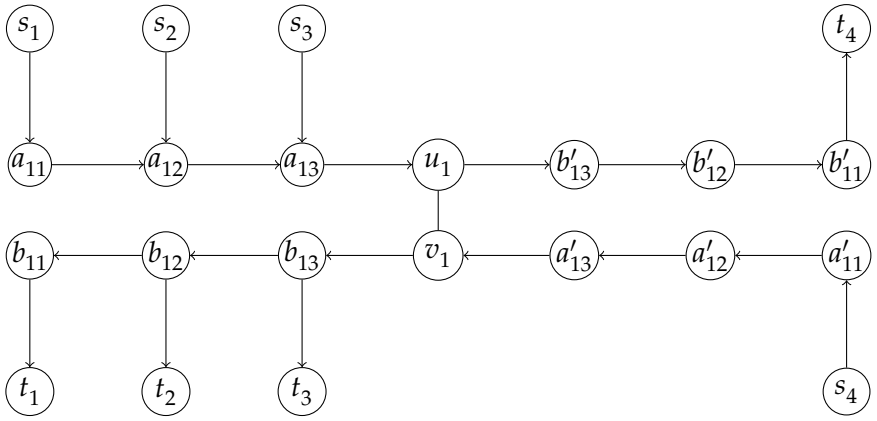


FIGURE 4.7 – *Construction d'une instance $(G, \mathcal{P})$ du problème* S-GO *à partir d'une instance de 3-SAT-4. L'ensemble de variables est $\mathcal{X} = \{x_j, 1 \leq j \leq 6\}$ et l'ensemble de clauses est $\mathcal{C} = \{c_i, 1 \leq i \leq 4\}$ avec $c_1 = (x_1 \vee \overline{x_2} \vee x_3)$, $c_2 = (x_1 \vee x_4 \vee x_5)$, $c_3 = (x_1 \vee \overline{x_4} \vee \overline{x_6})$ et $c_4 = (\overline{x_1} \vee \overline{x_5} \vee x_6)$. L'ensemble des paires à satisfaire est $\mathcal{P} = \{(s_i, t_i), 1 \leq i \leq 4\}$. Dans cette figure, nous ne montrons pas le graphe G mais seulement le sous-graphe de G correspondant à la variable x_1.*

4.6 Complexité du problème MIN-D-GO

Le problème d'optimisation MIN-D-GO peut être vu comme étant une variante du problème de décision, étudiée dans la section précédente (Section 4.5), dans laquelle nous autorisons certaines arêtes à être doublement orientées. Par conséquent, toute $\mathcal{P}$-orientation de G est une solution optimale de MIN-D-GO. Par contre, si le graphe G ne possède aucune $\mathcal{P}$-orientation, alors on en déduit seulement qu'il faut doubler l'orientation d'au moins une arête pour résoudre MIN-D-GO, mais en général cela ne

donne aucune information sur le nombre minimum d'arêtes doublement orientées nécessaires pour résoudre le problème.

Dans cette section, nous étudions la complexité du problème MIN-D-GO quand le graphe mixte en entrée est un MAG (voir les Tableaux 4.1 et 4.2). Comme dans la section précédente, nous supposons que G est $\mathcal{P}$-connexe et que G^* est connexe.

Dans le reste de cette section, on note par D-GO la version de décision naturelle associée au problème MIN-D-GO.

4.6.1 Algorithmes polynomiaux

Nous montrons d'abord que, quand $\Delta(G^*) = 2$, le problème MIN-D-GO peut se résoudre en temps polynomial. Le graphe G^* est connexe, donc G^* est un chemin ou un cycle. Si G^* est un chemin, alors pour tout $(s_i, t_i) \in \mathcal{P}$ il existe dans G un unique chemin P_i allant de s_i à t_i. Donc, pour résoudre MIN-D-GO il suffit d'orienter tous les chemins P_i de la source vers le terminal. Ensuite, on remplace chaque arête restante par un arc orienté dans un sens arbitraire.

Considérons maintenant les cas où G^* est un cycle et donc $B \leq 2$.

Soit $\mathcal{P}' \subseteq \mathcal{P}$ l'ensemble de paires pour lesquelles il existe un seul chemin dans G allant du sommet source vers le sommet terminal. Soit $\mathcal{P}''$ l'ensemble de paires pour lesquelles il existe deux chemins dans G allant du sommet source vers le sommet terminal. Le graphe G^* est un cycle, donc $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$.

Notation 4.1 *On note par $sr(\mathcal{P}'')$ (resp. $tr(\mathcal{P}'')$) l'ensemble des sommets sources (resp. terminaux) dans $\mathcal{P}''$, c'est-à-dire,*

$$sr(\mathcal{P}'') = \{s : \exists t \text{ tel que } (s,t) \in \mathcal{P}''\};$$

$$tr(\mathcal{P}'') = \{t : \exists s \text{ tel que } (s,t) \in \mathcal{P}''\}$$

On pose $m' = |\mathcal{P}'|$ et $m'' = |\mathcal{P}''|$. Quand $m'' = 1$, nous pouvons résoudre facilement le problème MIN-D-GO. En effet, pour toute paire $(s,t) \in \mathcal{P}'$ nous orientons (de s vers t) l'unique chemin dans G allant de s à t. Ensuite, pour satisfaire l'unique paire dans $\mathcal{P}''$, seulement deux orientations sont possibles. Donc, nous gardons celle qui crée le moins d'arêtes doublement orientées.

Dans le reste de cette section, nous supposons que $m'' \geq 2$.

Propriété 4.8 *Le graphe G est nécessairement composé des quatre sous-graphes suivants :*

1. *Un chemin non-orienté $P_1 = s_{i_1} X_1 s_{i_2} X_2 \dots X_{(m''-1)} s_{i_{m''}}$ t.q. $\{s_{i_1}, s_{i_2}, \dots, s_{i_{m''}}\} = sr(\mathcal{P}'')$ et pour tout k, $1 \leq k < m''$, X_k est un chemin non-orienté dans G ;*
2. *Un chemin non-orienté $P_2 = t_{j_1} Y_1 t_{j_2} Y_2 \dots Y_{(m''-1)} t_{j_{m''}}$ t.q. $\{t_{j_1}, t_{j_2}, \dots, t_{j_{m''}}\} = tr(\mathcal{P}'')$ et pour tout k, $1 \leq k < m''$, Y_k est un chemin non-orienté dans G ;*
3. *Un chemin P_3 allant de s_{i_1} à t_{j_1}, avec $A(P_3) \neq \emptyset$;*
4. *Un chemin P_4 allant de $s_{i_{m''}}$ à $t_{j_{m''}}$, avec $A(P_4) \neq \emptyset$.*

Nous pouvons aussi considérer, sans perte de généralité, que $j_1 = 1, j_2 = 2, \ldots, j_{m''} = m''$.

Une illustration de la Propriété 4.8 est donnée dans la Figure 4.8 (a) dans laquelle $m' = 2$, $m'' = 11$, $i_1 = 6$ et $i_{11} = 10$.

Preuve. Soit $(s,t) \in \mathcal{P}''$. On note par $p_1 = u_1u_2 \ldots u_{k_1}$ et $p_2 = v_1v_2 \ldots v_{k_2}$ les deux chemins distincts dans G allant de s à t. Le graphe G^* est un cycle, donc G est formé par les deux chemins p_1 et p_2, c'est-à-dire, $V(G) = V(p_1) \cup V(p_2)$, $E(G) = E(p_1) \cup E(p_2)$ et $A(G) = A(p_1) \cup A(p_2)$. Or, G est un MAG, donc il existe un entier $i \in \{1, \ldots, k_1 - 1\}$ et un entier $j \in \{1, \ldots, k_2 - 1\}$ tels que $u_iu_{i+1} \in A(p_1)$ et $v_jv_{j+1} \in A(p_2)$. On note par α_1 (resp. β_1) l'entier minium (resp. maximum) pour lequel $u_{\alpha_1}u_{\alpha_1+1} \in A(p_1)$ (resp. $u_{\beta_1}u_{\beta_1+1} \in A(p_1)$). On note également par α_2 (resp. β_2) l'entier minimum (resp. maximum) pour lequel $v_{\alpha_2}v_{\alpha_2+1} \in A(p_2)$ (resp. $v_{\beta_2}v_{\beta_2+1} \in A(p_2)$). Chaque sommet $s_i \in sr(\mathcal{P}'')$ doit appartenir au chemin non-orienté dans G, noté X, allant de u_{α_1} à v_{α_2}. Autrement, à cause des arcs $u_{\alpha_1}u_{\alpha_1+1}$ et $v_{\alpha_2}v_{\alpha_2+1}$, le graphe G ne pourrait pas contenir deux chemins allant de s_i à t_i. De façon similaire, chaque sommet $t_i \in tr(\mathcal{P}'')$ doit appartenir à un chemin non-orienté dans G, noté Y, allant de u_{β_1+1} à v_{β_2+1}.

Supposons que le chemin X (resp. Y) est le chemin $a_1a_2 \ldots a_{|X|}$ (resp. $b_1b_2 \ldots b_{|Y|}$) t.q. $a_1 = u_{\alpha_1}$ et $a_{|X|} = v_{\alpha_2}$ (resp. $b_1 = u_{\beta_1+1}$ et $b_{|Y|} = v_{\beta_2+1}$). Soit q_1 (resp. r_1) l'entier minimum (resp. maximum) t.q. $a_{q_1} \in V(X) \cap sr(\mathcal{P}'')$ (resp. $a_{r_1} \in V(X) \cap sr(\mathcal{P}'')$). De la même manière, soit q_2 (resp. r_2) l'entier minimum (resp. maximum) t.q. $b_{q_2} \in V(Y) \cap tr(\mathcal{P}'')$ (resp. $b_{r_2} \in V(Y) \cap tr(\mathcal{P}'')$).

On note par P_1 (resp. P_2) le sous-chemin de X (resp. Y) allant de a_{q_1} (resp. b_{q_2}) à a_{r_1} (resp. b_{r_2}). Le graphe G est donc composé (1) du chemin P_1 dont les sommets dans $sr(\mathcal{P}'')$ peuvent être numérotés de s_{i_1} à $s_{i_{m''}}$ t.q. $sr(\mathcal{P}'') = \{s_{i_1}, s_{i_2}, \ldots, s_{i_{m''}}\}$, (2) du chemin P_2 dont les sommets dans $tr(\mathcal{P}'')$ peuvent être numérotés de t_{j_1} à $t_{j_{m''}}$ t.q. $tr(\mathcal{P}'') = \{t_{j_1}, t_{j_2}, \ldots, t_{j_{m''}}\}$, (3) du chemin P_3 allant de s_{i_1} à t_{j_1} et (4) le chemin P_4 allant de $s_{i_{m''}}$ à $t_{j_{m''}}$. □

La Propriété 4.8 nous mène à la définition suivante :

Définition 4.7 *Soit $G = (V, E, A)$ un MAG tel que G^* est un cycle. Soit $\mathcal{P}$ un ensemble de paires de sommets source-terminal. Une* $\mathcal{P}''$-représentation *de G est une représentation planaire de G sous forme d'un rectangle $\mathcal{ABCD}$ tel que :*

- *(i) Les sommets de $sr(\mathcal{P}'')$ se trouvent, dans G^*, sur un chemin vertical $\mathcal{AB}$ dans l'ordre $\mathcal{A} = s_{i_1}, s_{i_2}, \ldots, s_{i_{m''}} = \mathcal{B}$.*
- *(ii) Les sommets de $tr(\mathcal{P}'')$ se trouvent, dans G^*, sur un chemin vertical $\mathcal{DC}$ dans l'ordre $\mathcal{D} = t_1, t_2, \ldots, t_{m''} = \mathcal{C}$.*
- *(iii) $\mathcal{AD}$ et $\mathcal{BC}$ sont des chemins horizontaux tels que $A(\mathcal{AD}) \neq \emptyset$ et $A(\mathcal{BC}) \neq \emptyset$.*

Pour tout $(u, v) \in sr(\mathcal{P}'') \times tr(\mathcal{P}'')$, le graphe G (et de façon similaire, toute orientation de G) peut être décomposé en deux sous-graphes. Le premier (resp. le deuxième) est composé des sommets qui se trouvent sur le segment vertical $u\mathcal{A}$, le segment horizontal $\mathcal{AD}$ et le segment vertical

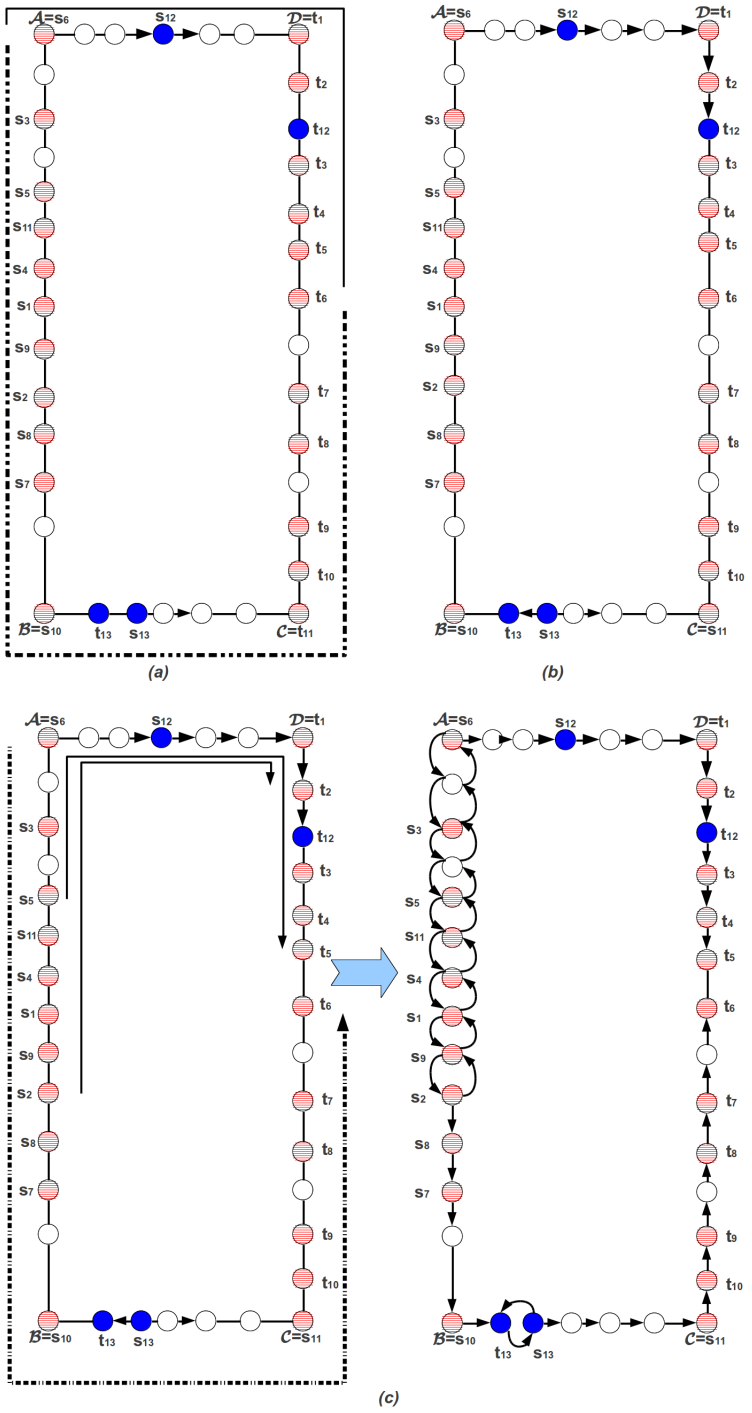


FIGURE 4.8 – **Le problème MIN-D-GO quand G^* est un cycle**. **(a)** *Les sommets remplis (resp. hachurés) correspondent à l'ensemble $\mathcal{P}'$ (resp. $\mathcal{P}''$) de paires (s,t) t.q. G contient exactement un (resp. deux) chemin(s) de s à t. Le rectangle $\mathcal{ABCD}$ est une $\mathcal{P}''$-représentation de G. Le chemin rempli (resp. en pointillés) est le chemin $G[s_6,t_6]^+$ (resp. $G[s_6,t_6]^-$)* **(b)** *Le graphe G' obtenu à partir de G après l'orientation de l'unique chemin allant de s à t pour tout $(s,t) \in \mathcal{P}'$.* **(c)** *L'orientation des trois sous-graphes $G'[s_6,t_6]^-$ (ligne en pointillés), $G'[s_5,t_5]^+$ et $G'[s_2,t_2]^+$, de la source vers le sommet terminal, donne (par le Lemme 4.1(c)) une orientation (partielle) réalisable.*

$\mathcal{D}v$ (resp. $u\mathcal{B}$, $\mathcal{BC}$ et $\mathcal{C}v$). On note le premier (resp. le second) sous-graphe par $G[u,v]^+$ (resp. $G[u,v]^-$). Une illustration est donnée dans la Figure 4.8.

Rappelons que pour tout $(s_i,t_i) \in \mathcal{P}'$, il existe dans G un unique chemin P_i de s_i à t_i. Donc, pour satisfaire les paires dans $\mathcal{P}'$, nous devons orienter les chemins P_i de s_i vers t_i .

Dans la suite, on note par G' le graphe mixte obtenu de G après l'orientation de tous les chemins P_i (voir l'exemple de construction de G' dans la Figure 4.8 (b)).

Considérons une $\mathcal{P}''$-représentation de G, et soit $(s_i,t_i) \in \mathcal{P}''$. Même si les sous-graphes $G[s_i,t_i]^+$ et $G[s_i,t_i]^-$ sont des chemins dans G, les sous-graphes $G'[s_i,t_i]^+$ et $G'[s_i,t_i]^-$ ne sont pas forcément des chemins. En revanche, dans le but de satisfaire la paire (s_i,t_i), nous devons orienter au moins l'un de ces sous-graphes pour obtenir un chemin orienté allant de s_i à t_i. Supposons que $G'[s_i,t_i]^+ = u_1u_2\dots u_{k_1}$ tel que $u_1 = s_i$, $u_{k_1} = t_i$ et $(u_j,u_{j+1}) \in E(G')$ ou $u_ju_{j+1} \in A(G')$ ou $u_{j+1}u_j \in A(G')$, pour tout $j \in \{1,2,\dots,k_1-1\}$. Dans la suite, nous allons souvent écrire *l'orientation de* $G'[s_i,t_i]^+$ (ou de façon similaire, l'orientation de $G'[s_i,t_i]^-$) *de* s_i *vers* t_i pour faire référence à l'orientation suivante :

- Si $(u_j,u_{j+1}) \in E(G')$ alors l'arête (u_j,u_{j+1}) est remplacée par l'arc u_ju_{j+1} ;
- Si $u_{j+1}u_j \in A(G')$ et $u_ju_{j+1} \notin A(G')$ alors nous ajoutons, dans G', l'arc u_ju_{j+1} (c'est-à-dire, nous créons une arête doublement orientée).

Cette orientation est également applicable à tout sous-graphe $G'[u,v]^+$ ou $G'[u,v]^-$, avec $(u,v) \in sr(\mathcal{P}'') \times tr(\mathcal{P}'')$.

Définition 4.8 (Orientation partielle) *Nous appelons orientation* partielle *de G' tout graphe mixte obtenu à partir de G' en orientant, pour certaines paires $(s_i,t_i) \in \mathcal{P}''$, le sous-graphe $G'[s_i,t_i]^-$ et/ou $G'[s_i,t_i]^+$ de s_i vers t_i.*

Pour simplifier la présentation, nous considérons que tous les sommets sources dans $\mathcal{P}''$ sont distincts, nous considérons aussi que tous les sommets terminaux dans $\mathcal{P}''$ sont distincts, c'est-à-dire, $|sr(\mathcal{P}'')| = |tr(\mathcal{P}'')| = m''$. Nous pouvons faire cette hypothèse sans aucune perte de généralité. En effet, si pour une instance donnée nous avons $(s,t),(s,t') \in \mathcal{P}''$, nous pouvons remplacer le sommet s, dans G', par deux sommets s et s' liés par une arête doublement orientée, et ensuite nous remplaçons, dans $\mathcal{P}''$, la paire (s,t') par la paire (s',t').

Définition 4.9 (Orientation réalisable vs. Orientation optimale) *Soit H une orientation partielle de G'. Le graphe H est dit* orientation réalisable *si pour tout $(s_i,t_i) \in \mathcal{P}$, il existe dans H un chemin orienté allant de s_i à t_i. Le graphe est dit* orientation optimale *s'il résout le problème* MIN-D-GO.

Un exemple d'une orientation (partielle) réalisable est donné dans la Figure 4.8 (c).

Pour résoudre le problème MIN-D-GO quand G^* est un cycle, nous proposons deux algorithmes polynomiaux appelés Min-Cycle-Orientation-1 et Min-Cycle-Orientation-2 (abrégés respectivement `MCO-1` et `MCO-2`).

L'algorithme MCO-1 (Algorithme 3)

L'algorithme est simple, nous orientons un chemin "+" et un chemin "-", du sommet source vers le sommet terminal, de sorte que les deux chemins couvrent tous les sommets sources et tous les sommets terminaux dans $\mathcal{P}''$. Nous faisons varier le début et la fin de chacun de ces chemins sur toutes les possibilités, et enfin nous gardons l'orientation réalisable créant un nombre minimum d'arête doublement orientées.

Algorithme 3 MCO-1$(G = (V, E, A), \mathcal{P} \subseteq V \times V)$

Entrées : Un MAG $G = (V, A, E)$ t.q. G^* est un cycle, un ensemble de paires $\mathcal{P} \subseteq V \times V$.

But : Trouver une orientation optimale de G.

1: /* $Update(G_{res}, G_i)$: Si le nombre d'arêtes doublement orientées d'une orientation réalisable G_i est strictement inférieur à celui de G_{res} alors $G_{res} := G_i$ */
2: /* $Represent(\mathcal{P}, G')$: Cette procédure construit la $\mathcal{P}''$-représentation de G', c'est-à-dire, un rectangle $\mathcal{ABCD}$ (définie page 103) */
3: Soit $\mathcal{P}'$ (resp. $\mathcal{P}''$) l'ensemble de paires (s, t) t.q. G contient exactement un (resp. deux) chemin(s) allant de s à t.
4: $G' := G$;
5: **pour** toute paire $(s, t) \in \mathcal{P}'$ **faire**
6: Orienter l'unique chemin dans G', entre s et t, de s vers t.
7: **fin pour**
8: $G_{res} := G'$; $Represent(\mathcal{P}, G')$; /* Construction du rectangle $\mathcal{ABCD}$ */
/* Orientation de $G_1[s_{i_1}, t_1]^-$ de la source vers le terminal */
9: $G_1 := G'$; $Orient(s_{i_1}, t_1, -, G_1)$; $Update(G_{res}, G_1)$;
/* Orientation de $G_1[s_{i_{m''}}, t_{m''}]^+$ de la source vers le terminal */
10: $G_2 := G'$; $Orient(s_{i_{m''}}, t_{m''}, +, G_2)$; $Update(G_{res}, G_2)$;
11: **pour tout** (g, j) avec $1 \leq g,j \leq m''$ et $g \geq j - 1$ **faire**
12: **pour tout** (k, l) avec $1 \leq k,l \leq m''$ et $k \geq l - 1$ **faire**
13: $G_3 := G'$; $Orient(s_{i_g}, t_k, +, G_3)$; $Orient(s_{i_j}, t_l, -, G_3)$;
14: **si** (G_3 est une orientation réalisable) **alors**
15: $Update(G_{res}, G_3)$;
16: **finsi**
17: **fin pour**
18: **fin pour**
/* Orientation arbitraire des arêtes restantes dans G_{res} */
19: Remplacer chaque arête $(u, v) \in E(G_{res})$ par l'arc uv.
20: **return** G_{res}

Théorème 4.4 *L'algorithme* MCO-1 *résout le problème* MIN-D-GO *quand G est un MAG et G^* est un cycle.*

Preuve. Soit G_{res} le graphe obtenu par l'algorithme MCO-1. Soit G_{opt} une orientation optimale de G. On note par n_{res} (resp. n_{opt}) le nombre d'arêtes doublement orientées dans G_{res} (resp. dans G_{opt}). Nous allons montrer que G_{res} est une orientation réalisable et que $n_{res} = n_{opt}$. Considérons une $\mathcal{P}''$-représentation de G_{res} et G_{opt}. Rappelons que dans cette représentation, les sommets sources se trouvent sur un segment vertical $\mathcal{AB}$ dans l'ordre

(de haut vers le bas) $\mathcal{A} = s_{i_1}, s_{i_2}, \ldots, s_{i_{m''}} = \mathcal{B}$. Les sommets terminaux se trouvent également sur un segment vertical $\mathcal{DC}$ dans l'ordre (de haut vers le bas) $\mathcal{D} = t_1, t_2, \ldots, t_{m''} = \mathcal{C}$, avec $m'' = |\mathcal{P}''|$.

Le graphe G_{res} est mis à jour, par la fonction *Update*, dans les lignes 9, 10 et 15. Dans la ligne 9 (resp. 10) le sous-graphe $G_{res}[s_{i_1}, t_1]^-$ (resp. $G_{res}[s_{i_{m''}}, t_{m''}]^+$) est un chemin orienté allant du sommet source vers le sommet terminal. Donc, dans ce cas G_{res} est une orientation réalisable. Évidemment, dans la ligne 15, le graphe G_{res} est aussi une orientation réalisable. Nous allons maintenant montrer que $n_{res} \leq n_{opt}$ (et par conséquent $n_{res} = n_{opt}$).

D'abord, nous identifions deux cas :

- $G_{opt}[s_{i_1}, t_1]^-$ est un chemin orienté allant de s_{i_1} à t_1. Dans ce cas, $A(G_1) \subseteq A(G_{opt})$, où G_1 est le graphe construit à la ligne 9.
- $G_{opt}[s_{i_{m''}}, t_{m''}]^+$ est un chemin orienté allant de $s_{i_{m''}}$ à $t_{m''}$. Dans ce cas, $A(G_2) \subseteq A(G_{opt})$, où G_2 est le graphe construit à la ligne 10.

Par ailleurs, G_1 et G_2 sont deux orientations réalisables, et l'orientation arbitraire des arêtes à la ligne 19 n'augmente pas le nombre d'arêtes doublement orientées. Par conséquent, si l'un des cas 1 ou 2 apparaît, nous avons $n_{res} \leq n_{opt}$.

Supposons maintenant qu'aucun des sous-graphes $G_{opt}[s_{i_1}, t_1]^-$ ou $G_{opt}[s_{i_{m''}}, t_{m''}]^+$ n'est un chemin orienté du sommet source vers le sommet terminal. Soit (w, y) la paire d'entiers tels que $G_{opt}[s_{i_w}, t_y]^+$ est le plus long chemin orienté, dans le sens "+", allant d'un sommet source vers un sommet terminal. Soit également (x, z) la paire d'entiers tels que $G_{opt}[s_{i_x}, t_z]^-$ est le plus long chemin orienté, dans le sens "-", allant d'un sommet source vers un sommet terminal. Nous avons nécessairement les inéquations suivantes : $w \geq x - 1$ et $y \geq z - 1$. En effet, supposons par exemple que $w \leq x - 2$. Dans ce cas, la paire $(s_{i_{(w+1)}}, t_{(w+1)})$ ne peut pas être satisfaite dans G_{opt} ni par un chemin "+" (à cause de la maximalité du chemin $G_{opt}[s_{i_w}, t_y]^+$), ni par un chemin "-" (à cause de la maximalité du chemin $G_{opt}[s_{i_x}, t_z]^-$). Cela est en contradiction avec le fait que G_{opt} est une orientation réalisable.

La maximalité des chemins $G_{opt}[s_{i_w}, t_y]^+$ et $G_{opt}[s_{i_x}, t_z]^-$ implique que le graphe H obtenu de G' (pour sa valeur à la ligne 8) en orientant $G'[s_{i_w}, t_y]^+$ et $G'[s_{i_x}, t_z]^-$ du sommet source vers le sommet terminal, est une orientation réalisable vérifiant : $A(H) \subseteq A(G_{opt})$. Le graphe H correspond au graphe G_3 calculé par l'algorithme à la ligne 13 pour $g = w$, $j = x$, $k = y$ et $l = z$. Comme l'orientation des arêtes à la ligne 19 n'augmente pas le nombre d'arêtes doublement orientées, on en déduit donc que $n_{res} \leq n_{opt}$. □

Complexité de l'algorithme MCO-1. On pose $n = |V(G)|$, $m' = |\mathcal{P}'|$, $m'' = |\mathcal{P}''|$ et $m = m' + m''$. Le calcul des ensembles $\mathcal{P}'$ et $\mathcal{P}''$ (ligne 3) et l'orientation des chemins, aux lignes 5-7, peut être effectué en temps de $\mathcal{O}(mn)$. La construction d'une $\mathcal{P}''$-représentation de G' peut se faire en temps de $\mathcal{O}(mn)$. Chacune des deux fonctions *Update* et *Orient* peut être effectuée en un temps de $\mathcal{O}(n)$ parce que G^* est un cycle (et donc $|A| + |E| = \mathcal{O}(n)$). La vérification du fait que G_3 est une orientation réalisable (ligne 14) peut être faite en un temps de $\mathcal{O}(m''n)$. Enfin, il y a

$\mathcal{O}(m''^4)$ possibilités pour choisir les paires d'entiers (g, j) et (k, l). Donc la complexité globale de l'algorithme est de $\mathcal{O}((m' + m''^5) \cdot n)$.

L'algorithme MCO-2 (Algorithme 4)

Cet algorithme est beaucoup plus élaboré que l'algorithme MCO-1 et sa complexité est de $\mathcal{O}((m' + m''^3) \cdot n)$. L'algorithme MCO-2 est donné page 111. Afin de présenter cet algorithme, nous avons besoin de quelques définitions supplémentaires. Nous généralisons les $\mathcal{P}''$-représentations comme suit.

Définition 4.10 *Soit $G = (V, E, A)$ un MAG tel que G^* est un cycle. Soit $\mathcal{P}$ un ensemble de paires de sommets et soit $\mathcal{R} \subseteq \mathcal{P}''$. Une* $\mathcal{R}$-représentation *de G est une représentation planaire de G sous forme d'un rectangle $\mathcal{ABCD}$ tels que :*

- *(i) Les sommets sources de $sr(\mathcal{R})$ se trouvent, dans G^*, sur un chemin vertical $\mathcal{AB}$ dans l'ordre $\mathcal{A} = s^{\mathcal{R}}_{i_1}, s^{\mathcal{R}}_{i_2}, \ldots, s^{\mathcal{R}}_{i_{|\mathcal{R}|}} = \mathcal{B}$.*
- *(ii) Les sommets de $tr(\mathcal{R})$ se trouvent, dans G^*, sur un chemin vertical $\mathcal{DC}$ dans l'ordre $\mathcal{D} = t^{\mathcal{R}}_1, t^{\mathcal{R}}_2, \ldots, t^{\mathcal{R}}_{i_{|\mathcal{R}|}} = \mathcal{C}$.*
- *(iii) $\mathcal{AD}$ et $\mathcal{BC}$ sont des chemins horizontaux tels que $A(\mathcal{AD}) \neq \emptyset$ et $A(\mathcal{BC}) \neq \emptyset$.*

Remarque 4.1 *La Propriété 4.8 implique que pour tout $\mathcal{R} \subseteq \mathcal{P}''$, il existe une $\mathcal{R}$-représentation pour G ainsi que pour toute orientation de G.*

Notons que quand $|\mathcal{R}| = 1$, la $\mathcal{R}$-représentation de G n'est pas vraiment un rectangle parce que $\mathcal{A} = \mathcal{B}$ et $\mathcal{D} = \mathcal{C}$.

Dans la suite, par convention, quand $\mathcal{R} = \mathcal{P}''$, les sommets $s^{\mathcal{R}}_{i_k}$ et $t^{\mathcal{R}}_{i_k}$ sont notés respectivement s_{i_k} et t_{i_k}.

Pour tout $(u, v) \in sr(\mathcal{R}) \times tr(\mathcal{R})$, on note par $G[u, v]^+_{\mathcal{R}}$ (resp. $G[u, v]^-_{\mathcal{R}}$) le sous-graphe formé par le segment vertical $u\mathcal{A}$, le segment horizontal $\mathcal{AD}$ et le segment vertical $\mathcal{D}v$ (resp. $u\mathcal{B}$, $\mathcal{BC}$ et $\mathcal{C}v$). Quand $|\mathcal{R}| = 1$, les deux chemins reliant u et v, avec $\mathcal{R} = \{(u, v)\}$, sont notés arbitrairement par $+$ et $-$.

Dans une $\mathcal{R}$-représentation, pour tout $1 \leq \alpha, \beta \leq |\mathcal{R}|$, nous notons $s^{\mathcal{R}}_{i_\alpha} \prec s^{\mathcal{R}}_{i_\beta}$ (resp. $s^{\mathcal{R}}_{i_\alpha} \succ s^{\mathcal{R}}_{i_\beta}$) si $s^{\mathcal{R}}_{i_\alpha}$ se situe en dessous (resp. au dessus) de $s^{\mathcal{R}}_{i_\beta}$ par rapport au segment vertical $\mathcal{AB}$. De façon similaire, pour tout $1 \leq \alpha, \beta \leq |\mathcal{R}|$, nous notons $t^{\mathcal{R}}_\alpha \prec t^{\mathcal{R}}_\beta$ (resp. $t^{\mathcal{R}}_\alpha \succ t^{\mathcal{R}}_\beta$) si $t^{\mathcal{R}}_\alpha$ se situe en dessous (resp. au dessus) de $t^{\mathcal{R}}_\beta$ par rapport au segment vertical $\mathcal{DC}$.

Rappelons que G' est le graphe obtenu de G en orientant (de la source vers le terminal) l'unique chemin allant de s à t, pour tout $(s, t) \in \mathcal{P}'$.

Lemme 4.1 *Soit G_1 une orientation partielle de G'. Soit $\mathcal{R} \subseteq \mathcal{P}''$. Soit $\mathcal{ABCD}$ une $\mathcal{R}$-représentation de G_1. Supposons qu'il existe un entier $x \in \{1, \ldots, |\mathcal{R}|\}$ tel que $G_1[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_x]^-_{\mathcal{R}}$ soit un chemin orienté. Un seul cas parmi les cas suivants peut alors exister.*

- *(a) $x = 1$ et toutes les paires de $\mathcal{R}$ sont satisfaites dans G_1.*
- *(b) $x \geq 2$ et toutes les paires dans $\mathcal{R}$ sont satisfaites dans le graphe G_2 obtenu à partir de G_1 en orientant $G_1[s^{\mathcal{R}}_{x-1}, t^{\mathcal{R}}_{x-1}]^+_{\mathcal{R}}$ de $s^{\mathcal{R}}_{x-1}$ vers $t^{\mathcal{R}}_{x-1}$.*

(c) $x \geq 2$ *et toutes les paires de* $\mathcal{R}$ *sont satisfaites dans le graphe* G_3 *obtenu à partir de* G_2 *en orientant* $G_2[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^+$ *de* $s_y^{\mathcal{R}}$ *vers* $t_y^{\mathcal{R}}$*, où* $s_y^{\mathcal{R}}$ *est le sommet le plus bas dans le segment* $\mathcal{AB}$ *t.q.* $(s_y^{\mathcal{R}}, t_y^{\mathcal{R}})$ *soit insatisfaite dans* G_2*.*

Pour une illustration du Lemme 4.1(c), voir la Figure 4.8 (c), dans laquelle $\mathcal{R} = \mathcal{P}''$, $i_1 = 6$, $x = 6$ et $y = 2$.

Preuve. Si $x = 1$, alors évidemment toutes les paires de $\mathcal{R}$ sont satisfaites par un sous-chemin du chemin orienté $G_1[s_{i_1}^{\mathcal{R}}, t_1^{\mathcal{R}}]_{\mathcal{R}}^-$. Considérons maintenant le cas $x \geq 2$. Soit G_2 le graphe obtenu de G_1 en orientant $G_1[s_{x-1}^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]_{\mathcal{R}}^+$ de $s_{x-1}^{\mathcal{R}}$ vers $t_{x-1}^{\mathcal{R}}$. Si toutes les paires de $\mathcal{R}$ sont satisfaites dans G_2, nous sommes dans le cas (b). Autrement, soit G_3 le graphe obtenu de G_2 en orientant $G_2[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^+$ de $s_y^{\mathcal{R}}$ vers $t_y^{\mathcal{R}}$, où $s_y^{\mathcal{R}}$ est le sommet le plus bas dans $\mathcal{AB}$ t.q. $(s_y^{\mathcal{R}}, t_y^{\mathcal{R}})$ est insatisfaite dans G_2.

Nous allons montrer que G_3 satisfait toutes les paires de $\mathcal{R}$.

La paire $(s_y^{\mathcal{R}}, t_y^{\mathcal{R}})$ est insatisfaite dans G_2, donc $s_y^{\mathcal{R}} \prec s_{x-1}^{\mathcal{R}}$ et $t_y^{\mathcal{R}} \succ t_{x-1}^{\mathcal{R}}$. En effet, si $s_y^{\mathcal{R}} \succ s_{x-1}^{\mathcal{R}}$ alors nous avons deux cas : $t_y^{\mathcal{R}} \succ t_{x-1}^{\mathcal{R}}$ ou $t_y^{\mathcal{R}} \prec t_x^{\mathcal{R}}$. Dans le premier (resp. le second) cas, le sous-graphe $G_2[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^+$ (resp. $G_2[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^-$) est un chemin orienté et cela conduit à une contradiction parce que $(s_y^{\mathcal{R}}, t_y^{\mathcal{R}})$ est insatisfaite dans G_2. Si $s_y^{\mathcal{R}} \prec s_{x-1}^{\mathcal{R}}$ et $t_y^{\mathcal{R}} \prec t_{x-1}^{\mathcal{R}}$ alors $G_2[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^-$ est un chemin orienté (un sous-chemin de $G_2[s_{i_1}^{\mathcal{R}}, t_x^{\mathcal{R}}]_{\mathcal{R}}^-$), cela conduit également à une contradiction. Donc, $s_y^{\mathcal{R}} \prec s_{x-1}^{\mathcal{R}}$ et $t_y^{\mathcal{R}} \succ t_{x-1}^{\mathcal{R}}$. Par conséquent, le sous-graphe $G_3[s_y^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]_{\mathcal{R}}^+$ est un chemin orienté allant de $s_y^{\mathcal{R}}$ à $t_{x-1}^{\mathcal{R}}$ formé par le chemin $G_3[s_y^{\mathcal{R}}, t_y^{\mathcal{R}}]_{\mathcal{R}}^+$ et le chemin $G_3[t_y^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]_{\mathcal{R}}^+$ (un sous-chemin de $G_3[s_{x-1}^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]_{\mathcal{R}}^+$). Le sous-graphe $G_3[s_y^{\mathcal{R}}, t_x^{\mathcal{R}}]_{\mathcal{R}}^-$ est aussi un chemin orienté allant de $s_y^{\mathcal{R}}$ à $t_x^{\mathcal{R}}$ (un sous-chemin de $G_1[s_{i_1}^{\mathcal{R}}, t_x^{\mathcal{R}}]_{\mathcal{R}}^-$).

Nous pouvons maintenant montrer que pour tout $k \in \{1, \ldots, |\mathcal{R}|\}$, il existe un chemin orienté dans G_3 allant de $s_{i_k}^{\mathcal{R}}$ à $t_{i_k}^{\mathcal{R}}$. Évidemment, si $k \in \{x-1, x\}$, la paire $(s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}})$ est satisfaite dans G_3 parce qu'elle était déjà satisfaite dans G_2. Soit $k \in \{1, 2, \ldots k\} \setminus \{x-1, x\}$. Nous distinguons les cas suivants :

1. $s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}} \in V(G_3[s_y^{\mathcal{R}}, t_x^{\mathcal{R}}]^-)$;
2. $s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}} \in V(G_3[s_y^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]^+)$;
3. $s_{i_k}^{\mathcal{R}} \succ s_y^{\mathcal{R}}$ et $t_{i_k}^{\mathcal{R}} \prec t_x^{\mathcal{R}}$;
4. $s_{i_k}^{\mathcal{R}} \prec s_y^{\mathcal{R}}$ et $t_{i_k}^{\mathcal{R}} \succ t_{x-1}^{\mathcal{R}}$.

Dans le premier (resp. le second) cas, $G_3[s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}}]_{\mathcal{R}}^-$ (resp. $G_3[s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}}]_{\mathcal{R}}^+$) est un sous-chemin du chemin orienté $G_3[s_y^{\mathcal{R}}, t_x^{\mathcal{R}}]_{\mathcal{R}}^-$ (resp. $G_3[s_y^{\mathcal{R}}, t_{x-1}^{\mathcal{R}}]_{\mathcal{R}}^+$). Dans le troisième cas, $G_3[s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}}]_{\mathcal{R}}^-$ est un sous-chemin du chemin orienté $G_3[s_{i_1}^{\mathcal{R}}, t_x^{\mathcal{R}}]_{\mathcal{R}}^-$. Dans le dernier cas, il doit exister dans G_2 (et donc dans G_3) un chemin orienté allant de $s_{i_k}^{\mathcal{R}}$ à $t_{i_k}^{\mathcal{R}}$. Autrement, la paire $(s_{i_k}^{\mathcal{R}}, t_{i_k}^{\mathcal{R}})$ devrait être choisie au lieu de $(s_y^{\mathcal{R}}, t_y^{\mathcal{R}})$. □

Le deuxième algorithme MCO-2 est basé sur le Lemme 4.1.

Idées intuitives. D'abord, nous considérons une $\mathcal{P}''$-représentation de G' et G_{opt}. Nous allons montrer (preuve du Théorème 4.5) que pour toute orientation optimale G_{opt} nous avons (i) pour tout $k \in \{1, \ldots, m''\}$, $G_{opt}[s_{i_k}, t_{i_k}]^+$ est un chemin orienté de s_{i_k} à t_{i_k}, ou sinon (ii) il doit exister un entier r vérifiant :

• toutes les paires (s_{i_k}, t_{i_k}), $1 \leq k \leq r-1$, sont satisfaites par le chemin $G_{opt}[s_{i_k}, t_{i_k}]^+$, et

• les paires restantes sont satisfaites selon l'un des cas (a), (b), (c) du Lemme 4.1 avec $\mathcal{R}$=$\mathcal{P}'' \setminus \{(s_{i_k}, t_{i_k}) : 1 \leq k \leq r-1\}$ (et donc $s^{\mathcal{R}}_{i_1} = s_{i_r}$) et $1 \leq x \leq i_1$.

Par conséquent, l'algorithme MCO-2 considère dans G' les paires (s_{i_k}, t_{i_k}) dans un ordre croissant de k (la boucle "tant que" à la ligne 9), calcule l'orientation de G' comme si k était identique à r et garde la meilleure orientation courante dans G_{res} (lignes 10 à 23 en appliquant le Lemme 4.1), ensuite prépare le prochain k (ligne 24). De cette manière, toutes les valeurs possibles pour r sont testées, et donc G_{opt} correspond au G_{res} final.

On note que la boucle "pour" à la ligne 10 permet d'obtenir la meilleure valeur de x pour le k courant, selon le Lemme 4.1.

Théorème 4.5 *L'algorithme* MCO-2 *résout le problème* MIN-D-GO *quand G est un MAG et G^* est un cycle.*

Preuve. Soit G_{res} le graphe obtenu par l'algorithme MCO-2. Soit G_{opt} une solution optimale pour MIN-D-GO. On note par n_{res} (resp. n_{opt}) le nombre d'arêtes doublement orientées dans G_{res} (resp. dans G_{opt}). Montrons que G_{res} est une orientation réalisable et que $n_{res} = n_{opt}$.

Toutes les paires de $\mathcal{P}'$ sont satisfaites dans G_{res} par l'orientation aux lignes 5-7.

Remarquons qu'à chaque étape de l'algorithme, toutes les paires dans $\mathcal{P}'' \setminus \mathcal{R}$ sont également satisfaites dans G' par l'orientation à la ligne 24.

Évidemment, en utilisant le Lemme 4.1 on en déduit que les graphes G_i, $1 \leq i \leq 3$, crées aux lignes 11, 15 et 20, satisfont toutes les paires de $\mathcal{R}$ (et par conséquent toutes les paires de $\mathcal{P}$). Enfin, à la ligne 26, pour tout $k \in \{1, 2, \ldots, m''\}$ le sous-graphe $G'[s_{i_k}, t_{i_k}]^+$ est un chemin orienté de s_{i_k} à t_{i_k}, et donc G' est une orientation réalisable. Donc, le graphe G_{res} est aussi une orientation réalisable.

Nous passons maintenant à la deuxième partie de la preuve, nous allons montrer que $n_{res} \leq n_{opt}$ (et par conséquent, $n_{res} = n_{opt}$). Considérons une $\mathcal{P}''$-représentation de G_{opt}. Si pour tout $k \in \{1, \ldots, m''\}$, $G_{opt}[s_{i_k}, t_{i_k}]^+$ est un chemin orienté de s_{i_k} à t_{i_k} alors $A(G') \subseteq A(G_{opt})$ t.q. G' est le graphe obtenu, à la ligne 24, après $|\mathcal{P}''|$ exécutions de la boucle "pour" (ligne 9). Donc, dans ce cas nous avons $n_{res} \leq n_{opt}$.

Supposons maintenant que pour certaines paires $(s_{i_k}, t_{i_k}) \in \mathcal{P}''$, $k \in \{1, \ldots, m''\}$, le sous-graphe $G_{opt}[s_{i_k}, t_{i_k}]^+$ n'est pas un chemin orienté de s_{i_k} à t_{i_k} (et donc $G_{opt}[s_{i_k}, t_{i_k}]^-$ est un chemin orienté s_{i_k} à t_{i_k}). On pose

$$r = min\{k \ : \ G_{opt}[s_{i_k}, t_{i_k}]^- \text{ est un chemin orienté de } s_{i_k} \text{ à } t_{i_k}\}.$$

Soit H le graphe mixte obtenu à partir de G' (pour sa valeur à la ligne 8) en orientant (1) $G'[s_{i_k}, t_{i_k}]^+$ de s_{i_k} vers t_{i_k} pour tout k, $1 \leq k \leq r-1$, et (2) $G'[s_{i_r}, t_{i_r}]^-$ de s_{i_r} vers t_{i_r}. Il est clair que $A(H) \subseteq A(G_{opt})$. Le graphe H

Algorithme 4 MCO-2($G = (V, E, A), P \subseteq V \times V$)

Entrées : Un MAG $G = (V, A, E)$ t.q. G^* est un cycle, un ensemble de paires $\mathcal{P} \subseteq V \times V$.

But : Trouver une orientation optimale de G.

1: /* $Update(G_{res}, G_i)$: Si le nombre d'arêtes doublement orientées d'une orientation réalisable G_i est strictement inférieur à celui de G_{res} alors $G_{res} := G_i$ */
2: /* $Represent(\mathcal{R}, G')$: Cette procédure construit une $\mathcal{R}$-représentation de G', c'est-à-dire, un rectangle $\mathcal{ABCD}$ (défini page 108) */
3: Soit $\mathcal{P}'$ (resp. $\mathcal{P}''$) l'ensemble de paires (s, t) t.q. il y a dans G un seul (resp. deux) chemin(s) allant de s à t.
4: $G' := G$;
5: **pour** toute paire $(s, t) \in \mathcal{P}'$ **faire**
6: Orienter l'unique chemin dans G', entre s et t, de s vers t.
7: **fin pour**
8: $G_{res} := G'$; $\mathcal{R} := \mathcal{P}''$; $Represent(\mathcal{R}, G')$; /* Construction de $\mathcal{ABCD}$ */
9: **tantque** ($|\mathcal{R}| \neq \emptyset$) **faire**
10: **pour** $x := i_1$ à 1 par pas de -1 **faire**
11: $G_1 := G'$; $Orient(s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_x, -, G_1)$;
/* Orientation de $G_1[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_x]^-_{\mathcal{R}}$ de $s^{\mathcal{R}}_{i_1}$ vers $t^{\mathcal{R}}_x$ */
12: **si** ($x = 1$) **alors**
13: $Update(G_{res}, G_1)$; /* cas (a) du Lemme 4.1 */
14: **sinon**
15: $G_2 := G_1$; $Orient(s^{\mathcal{R}}_{x-1}, t^{\mathcal{R}}_{x-1}, +, G_2)$;
/* Orientation de $G_2[s^{\mathcal{R}}_{x-1}, t^{\mathcal{R}}_{x-1}]^+_{\mathcal{R}}$ de $s^{\mathcal{R}}_{x-1}$ vers $t^{\mathcal{R}}_{x-1}$ */
16: **si** (G_2 satisfait toutes les paires de $\mathcal{R}$) **alors**
17: $Update(G_{res}, G_2)$; /* cas (b) du Lemme 4.1 */
18: **sinon**
19: Soit $s^{\mathcal{R}}_y \in sr(\mathcal{R})$ le plus bas sommet (par rapport au segment vertical $\mathcal{AB}$) t.q. $(s^{\mathcal{R}}_y, t^{\mathcal{R}}_y)$ est insatisfaite dans G_2.
/* cas (c) du Lemme 4.1 */
20: $G_3 := G_2$; $Orient(s^{\mathcal{R}}_y, t^{\mathcal{R}}_y, +, G_3)$; $Update(G_{res}, G_3)$;
21: **finsi**
22: **finsi**
23: **fin pour**
/* Reconstruction du rectangle $\mathcal{ABCD}$ */
24: $Orient(s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{i_1}, +, G')$; $\mathcal{R} := \mathcal{R} \backslash \{(s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{i_1})\}$; $Represent(\mathcal{R}, G')$;
25: **fin tantque**
26: $Update(G_{res}, G')$;
/* Orientation arbitraire des arêtes restantes dans G_{res} */
27: Remplacer chaque arête $(u, v) \in E(G_{res})$ par l'arc uv.
28: **return** G_{res}

correspond au graphe G_1 (à la ligne 11) obtenu à la r-ième exécution de la boucle "tant que" (ligne 9). A cette étape, $\mathcal{R} = \mathcal{P}'' \backslash \{(s_{i_k}, t_{i_k}) \ : \ 1 \leq k < r\}$.

Considérons une $\mathcal{R}$-représentation de H et G_{opt}. Dans une telle représentation, la paire $(s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{i_1})$ est identique à la paire (s_{i_r}, t_{i_r}). Donc, le

sous-graphe $H[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{i_1}]^-_{\mathcal{R}}$ (resp. $G_{opt}[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{i_1}]^-_{\mathcal{R}}$) est un chemin orienté de $s^{\mathcal{R}}_{i_1}$ à $t^{\mathcal{R}}_{i_1}$. On pose

$$\alpha = min\{k \ : \ G_{opt}[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{k}]^- \text{ est un chemin orienté de } s^{\mathcal{R}}_{i_1} \text{ à } t^{\mathcal{R}}_{k}\}.$$

Nous distinguons deux cas :

- $\alpha = 1$. Donc $A(H_1) \subseteq A(G_{opt})$, où H_1 est le graphe obtenu de H en orientant le sous-graphe $H[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{1}]^-_{\mathcal{R}}$ de $s^{\mathcal{R}}_{i_1}$ vers $t^{\mathcal{R}}_{1}$. Le graphe H_1 correspond au graphe G_1 créé à la ligne 11 avec $x = 1$.
- $\alpha \geq 2$. Soit H_2 le graphe obtenu de H en orientant $H[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{\alpha}]^-_{\mathcal{R}}$ et $H[s^{\mathcal{R}}_{\alpha-1}, t^{\mathcal{R}}_{\alpha-1}]^+_{\mathcal{R}}$ de la source vers le terminal. Clairement, $A(H_2) \subseteq A(G_{opt})$. Le graphe H_2 correspond au graphe G_2 (à la ligne 15) résultant de l'exécution de la boucle "pour" (ligne 10) avec $x = \alpha$. Si H_2 ne satisfait pas toutes les paires de $\mathcal{R}$, alors on note par H_3 le graphe mixte obtenu de H_2 en orientant $H_2[s^{\mathcal{R}}_{y}, t^{\mathcal{R}}_{y}]^+_{\mathcal{R}}$ de $s^{\mathcal{R}}_{y}$ vers $t^{\mathcal{R}}_{y}$ (y est calculé à la ligne 19). Notons que $y < \alpha$, autrement la paire $(s^{\mathcal{R}}_{y}, t^{\mathcal{R}}_{y})$ serait satisfaite dans H_2 par un sous-chemin de $H_2[s^{\mathcal{R}}_{i_1}, t^{\mathcal{R}}_{\alpha}]^-_{\mathcal{R}}$. Par conséquent, le sous-graphe $G_{opt}[s^{\mathcal{R}}_{y}, t^{\mathcal{R}}_{y}]^-_{\mathcal{R}}$ ne peut pas être un chemin orienté à cause de la minimalité de α. Donc, $G_{opt}[s^{\mathcal{R}}_{y}, t^{\mathcal{R}}_{y}]^+_{\mathcal{R}}$ est un chemin orienté allant de $s^{\mathcal{R}}_{y}$ à $t^{\mathcal{R}}_{y}$. Par conséquent, $A(H_3) \subseteq A(G_{opt})$. Le graphe H_3 correspond au graphe G_3 crée à la ligne 20.

Le graphe G_{res} est mis à jour, par la fonction *Update*, aux lignes 13, 17, 20 et 26. Rappelons que le Lemme 4.1 implique que tous les graphes G_i, $1 \leq i \leq 3$, créés aux lignes 13, 17 et 20 sont des orientations réalisables. Le graphe G' (à la ligne 26) est aussi une orientation réalisable parce que le sous-graphe $G'[s_{i_k}, t_{i_k}]^+$ est un chemin orienté de s_{i_k} à t_{i_k}, pour tout $k \in \{1, 2, \ldots, m''\}$. Enfin, l'orientation à la ligne 27 ne crée pas d'arêtes doublement orientées, nous déduisons donc que $n_{res} \leq n_{opt}$.

□

Complexité de MCO-2. Rappelons que $n = |V(G)|$, $m' = |\mathcal{P}'|$, $m'' = |\mathcal{P}''|$ et $m = m' + m''$. Rappelons également que le calcul de $\mathcal{P}'$ et $\mathcal{P}''$ (ligne 3) et l'orientation des chemins, aux lignes 5-7, peuvent être réalisés en un temps de $\mathcal{O}(mn)$. La boucle aux lignes 9-25 et la boucle aux lignes 10-23 sont exécutées au plus m'' fois. Pour tout $\mathcal{R} \subseteq \mathcal{P}''$, une $\mathcal{R}$-représentation de G' peut être construite en $\mathcal{O}(|\mathcal{R}|n)$. Le graphe G^* est un cycle, donc les fonctions *Update* et *Orient* et les affectations ($G_1 := G'$, $G_2 := G_1$ et $G_3 := G_2$) aux lignes $\{11, 15, 20\}$ peuvent être réalisées en un temps de $\mathcal{O}(n)$. Le calcul de la paire $(s^{\mathcal{R}}_{y}, t^{\mathcal{R}}_{y})$ (à la ligne 19) peut être effectué en $\mathcal{O}(m''n)$. Donc la complexité globale de l'algorithme MCO-2 est de $\mathcal{O}((m' + {m''}^3)n)$. Donc, nous avons réduit la complexité de l'algorithme MCO-1 par un facteur de ${m''}^2$.

Dans le théorème suivant, nous montrons que, de façon similaire au problème S-GO (preuve proposée par Arkin et al. [AH02]), le problème MIN-D-GO peut se résoudre en temps polynomial quand G est un MAG général et $|\mathcal{P}| \leq 2$. Avant de présenter ce résultat, nous avons besoin de la définition suivante.

Définition 4.11 *[AH02] Soit $G = (V, E, A)$ un MAG et soit $\mathcal{P} \subseteq V \times V$ tel que G est $\mathcal{P}$-connexe. Une arête $(u, v) \in E$, est dite* $\mathcal{P}$-essentielle *si toute orientation de (u, v), dans une direction unique, casse la $\mathcal{P}$-connexité du graphe G.*

Les arêtes $\mathcal{P}$-essentielles d'un graphe mixte peuvent être calculées en un temps polynomial [AH02].

Théorème 4.6 *Le problème* MIN-D-GO *peut se résoudre en temps polynomial quand G est un MAG et $|\mathcal{P}| \leq 2$.*

Preuve. Supposons que $\mathcal{P} = \{(s_1, t_1), (s_2, t_2) \in V \times V\}$. Soit E_{ess} l'ensemble d'arêtes $\mathcal{P}$-essentielles. Soit E_{min} l'ensemble d'arêtes doublement orientées dans une solution de MIN-D-GO. Nous allons montrer que $E_{min} = E_{ess}$.

Soit $e \in E_{ess}$, avec $e = (u, v)$. Par définition, si on oriente e dans une direction unique, il y aura un entier i, $1 \leq i \leq 2$, tel que il n'y a dans G aucun chemin de s_i à t_i. Donc, quelle que soit l'orientation des arêtes de $E - \{e\}$, la paire (s_i, t_i) ne sera pas satisfaite. Par conséquent, nous devons doubler l'orientation de toutes les arêtes $\mathcal{P}$-essentielles, ce qui implique que $E_{ess} \subseteq E_{min}$. Réciproquement, soit $G' = (V, E', A')$ le graphe mixte obtenu à partir de G en remplaçant chaque arête $\mathcal{P}$-essentielle (u, v) par les deux arcs uv et vu, c'est-à-dire, $V(G') = V(G)$, $E(G') = E(G) \backslash E_{ess}$ et $A' = A \cup \{uv, vu : (u, v) \in E_{ess}\}$. Arkin al. [AH02] ont démontré qu'un graphe mixte et $\mathcal{P}$-connexe possède une $\mathcal{P}$-orientation si et seulement si il n'a aucune arête $\mathcal{P}$-essentielle. Donc, il existe une $\mathcal{P}$-orientation G'' pour le graphe G'. Par conséquent, G'' est une orientation de G qui satisfait toutes les paires de $\mathcal{P}$ et qui crée $|E_{ess}|$ arêtes doublement orientées, ce qui implique que $|E_{ess}| \geq |E_{min}|$. Or, nous avons déjà montré que $E_{ess} \subseteq E_{min}$, donc on en déduit que $E_{min} = E_{ess}$.

Maintenant, pour résoudre MIN-D-GO quand $|\mathcal{P}| = 2$, nous appliquons la procédure suivante :

1. On calcule les arêtes $\mathcal{P}$-essentielles de G en utilisant l'algorithme polynomial proposé dans [AH02] ;
2. On construit un graphe mixte G' en remplaçant toute arête $\mathcal{P}$-essentielle (u, v) par les deux arcs uv et vu ;
3. On applique l'algorithme polynomial présenté dans [AH02] pour construire une $\mathcal{P}$-orientation de G'.

□

Comme c'était le cas pour S-GO, le problème MIN-D-GO est facile quand les deux paramètres $|\mathcal{P}|$ et $|B|$ sont bornés.

Théorème 4.7 *Le problème* MIN-D-GO *peut se résoudre en un temps polynomial quand G est un MAG, $|\mathcal{P}| = \mathcal{O}(1)$ et $B = \mathcal{O}(1)$.*

Preuve. Supposons que $G = (V, E, A)$ et $\mathcal{P} = \{(s_i, t_i), 1 \leq i \leq m\}$. La preuve est similaire à celle du Théorème 4.2. Nous considérons toutes les combinaisons possibles de chemins satisfaisant les paires de $\mathcal{P}$ (on choisit un chemin par paire et on l'oriente du sommet source vers le sommet terminal), ce qui induit au plus B^m combinaisons. Ensuite, nous gardons le graphe résultant de l'orientation qui crée un nombre minimum d'arêtes

doublement orientées. Nous complétons cette orientation par une orientation arbitraire des arêtes restantes sans créer de nouvelles arêtes doublement orientées.

□

Comme nous l'avons remarqué pour le problème S-GO, l'algorithme proposé dans la preuve du Théorème 4.7 est un algorithme FPT paramétré par B et $|\mathcal{P}|$.

4.6.2 Résultats de difficulté

Nous montrons dans le théorème suivant que D-GO (la version de décision naturelle de MIN-D-GO) est NP-complet même si $\Delta(G^*) = 3$ et $B = 3$.

Théorème 4.8 *Le problème* D-GO *est* NP*-complet quand G est un MAG et $|\mathcal{P}|$ est non bornée, même si $\Delta(G^*) = 3$ et $B = 3$.*

Preuve. Clairement, le problème D-GO est dans NP. Pour montrer sa NP-complétude, nous proposons une réduction du problème S-GO étudié dans la section précédente (Section 4.5). Soit $G = (V, E, A)$ un MAG et soit $\mathcal{P} \subseteq V \times V$ un ensemble de paires de sommets. Nous avons montré dans le Théorème 4.3 que le problème S-GO est NP-complet même si G est un MAG, $\Delta(G^*) = 3$ et $B = 3$. On pose $n = |V|$. Nous construisons un MAG $G' = (V', E', A')$ tel que $V' = V \cup \{u_i, v_i, 1 \leq i \leq n+1\}$, $A' = A$ et $E' = E \cup \{(u_i, v_i), (v_i, u_{i+1}), 1 \leq i \leq n\} \cup \{(v_{n+1}, u_1)\}$. On pose $\mathcal{P}' = \mathcal{P} \cup \{(u_i, v_i), 1 \leq i \leq n+1\}$. Clairement, $\Delta(G'^*) = 3$ et pour toute paire $(s', t') \in \mathcal{P}'$ il y a dans G' au plus trois chemins allant de s' à t'. Donc $B' \leq 3$, où B' est donné par :

$$B' = max\ \{n'_i\ :\ n'_i \text{ est le nombre des chemins dans } G' \text{ de } s_i \text{ à } t_i\}.$$

Nous allons montrer la propriété suivante : il existe une $\mathcal{P}$-orientation pour G si, et seulement si, il existe une $(\mathcal{P}', n)$-D-orientation pour G'.

$\Rightarrow$: Le graphe G' est composé de deux sous-graphes : le graphe G et le chemin non-orienté $R = v_{n+1}u_1v_1u_2v_2 \ldots u_nv_nu_{n+1}$. On garde pour G son orientation obtenue à partir d'une solution de S-GO et on oriente le chemin R de u_{n+1} vers v_{n+1}, ensuite on ajoute l'arc u_iv_i pour tout i, $1 \leq i \leq n$. Cette orientation satisfait toutes les paires de $\mathcal{P}'$ et le nombre d'arêtes doublement orientées est égal à n. Donc, on obtient une $(\mathcal{P}', n)$-D-orientation de G'.

$\Leftarrow$: Il existe un unique chemin dans G' entre u_i et v_i, pour tout $1 \leq i \leq n+1$. Donc, pour satisfaire toutes les paires de $\mathcal{P}'$, toute orientation de G' doit orienter le chemin $R = v_{n+1}u_1v_1u_2v_2 \ldots u_nv_nu_{n+1}$, de u_{n+1} vers v_{n+1} et doit aussi ajouter un arc u_iv_i pour tout i, $1 \leq i \leq n$. Donc cette orientation de R crée n arêtes doublement orientées. Or, le nombre d'arêtes doublement orientées dans une $(\mathcal{P}', n)$-D-orientation est égal à n. Par conséquent, nous obtenons une orientation de G qui satisfait toutes les paires de $\mathcal{P}$ et qui ne crée aucune arête doublement orientée. Donc, il s'agit d'une $\mathcal{P}$-orientation de G.

□

Le reste de cette section est consacré à l'étude du problème MIN-D-GO en terme d'approximabilité et de complexité paramétrée.

Théorème 4.9 *Sauf si* P = NP*, le problème* MIN-D-GO *est non-approximable quand* G *est un MAG et* $|\mathcal{P}|$ *est non-borné, même si* $\Delta(G^*) = 3$ *et* $B = 3$.

Preuve. Rappelons qu'une $(\mathcal{P}, k)$-D-orientation est une orientation qui satisfait toutes les paires dans $\mathcal{P}$ et qui crée exactement k arêtes doublement orientées. Donc, une $(\mathcal{P}, 0)$-D-orientation est une $\mathcal{P}$-orientation.

Supposons par l'absurde qu'il existe un algorithme d'approximation pour MIN-D-GO, noté $\mathcal{A}$, avec un facteur $\rho > 1$. Soit k_{min} le nombre d'arêtes doublement orientées dans une solution exacte du problème MIN-D-GO. Donc, pour toute instance $(G, \mathcal{P})$ de MIN-D-GO, l'algorithme $\mathcal{A}$ calcule, en temps polynomial, une $(\mathcal{P}, k)$-D-orientation tel que $k \leq \rho k_{min}$.

Si $k = 0$, alors la $(\mathcal{P}, k)$-D-orientation calculée par $\mathcal{A}$ est une $\mathcal{P}$-orientation de G. Si $k > 0$, alors $k_{min} > 0$ et donc il n'y a aucune $\mathcal{P}$-orientation pour G. Donc, G possède une $\mathcal{P}$-orientation si et seulement si la $(\mathcal{P}, k)$-D-orientation calculée par l'algorithme $\mathcal{A}$ vérifie $k = 0$. Donc, l'algorithme $\mathcal{A}$ résout en temps polynomial le problème S-GO, et cela conduit à une contradiction car nous avons déjà prouvé la NP-complétude du problème S-GO (voir le Théorème 4.3) quand G est un MAG, $B = 3$ et $\Delta(G^*) = 3$. □

Dans le résultat suivant nous montrons que, contrairement au problème S-GO, le problème MIN-D-GO reste difficile même si $B = 2$.

Théorème 4.10 *Le problème* MIN-D-GO *est* APX*-difficile quand* G *est un MAG et* $|\mathcal{P}|$ *est non-borné, même si* $\Delta(G^*) = 3$ *et* $B = 2$.

Preuve. Nous proposons une L-réduction du problème APX-difficile Minimum Set Cover-2 [PY91]. Nous rappelons que ce problème est défini comme suit : étant donné un ensemble $\mathcal{X} = \{X_1, \dots X_n\}$ et une collection $\mathcal{C} = \{S_1, \dots S_q\}$ des sous-ensembles de $\mathcal{X}$ tel que tout élément $X_j \in \mathcal{X}$ apparaît au plus dans deux sous-ensembles $S_i \in \mathcal{C}$, trouver un ensemble couvrant minimal de $\mathcal{X}$ (voir la Définition 3.4 du Chapitre 3). D'abord, on considère que tout élément $X_j \in \mathcal{X}$ apparaît dans au moins un sous-ensemble $S_i \in \mathcal{C}$. Autrement, il n'existe aucun ensemble couvrant de $\mathcal{X}$. De plus, si un élément $X_j \in \mathcal{X}$ apparaît dans un unique sous-ensemble $S_i \in \mathcal{C}$, alors tout ensemble couvrant de $\mathcal{X}$ doit contenir S_i. Donc, nous pouvons réduire cette instance en supprimant S_i et les éléments de S_i ($\mathcal{C} := \mathcal{C} - \{S_i\}$ et $\mathcal{X} := \mathcal{X} \setminus S_i$) pour obtenir une instance équivalente. Donc, sans perte de généralité, nous pouvons considérer que tout élément $X_j \in \mathcal{X}$ apparaît dans exactement deux sous-ensembles $S_i \in \mathcal{C}$.

Nous allons construire une instance $(G, \mathcal{P})$ du problème MIN-D-GO. Pour tout $X_j \in \mathcal{X}$, on crée les trois sommets $\{x_j, x_j^1, x_j^2\}$, ensuite on crée le chemin orienté $x_j^1 x_j^2 x_j$. Pour tout S_i, on crée les sommets $\{s_i, s_i'\} \cup \{s_i^j, 1 \leq j \leq |S_i|\}$, et nous ajoutons une arête (s_i', s_i) et le chemin orienté $s_i s_i^1 s_i^2 \dots s_i^{|S_i|}$.

Soit $X_l \in \mathcal{X}$ le $j-$ième élément de l'ensemble S_i. Nous mettons un arc allant de s_i^j vers un *seul* sommet de l'ensemble $\{x_l^1, x_l^2\}$. Un tel sommet est

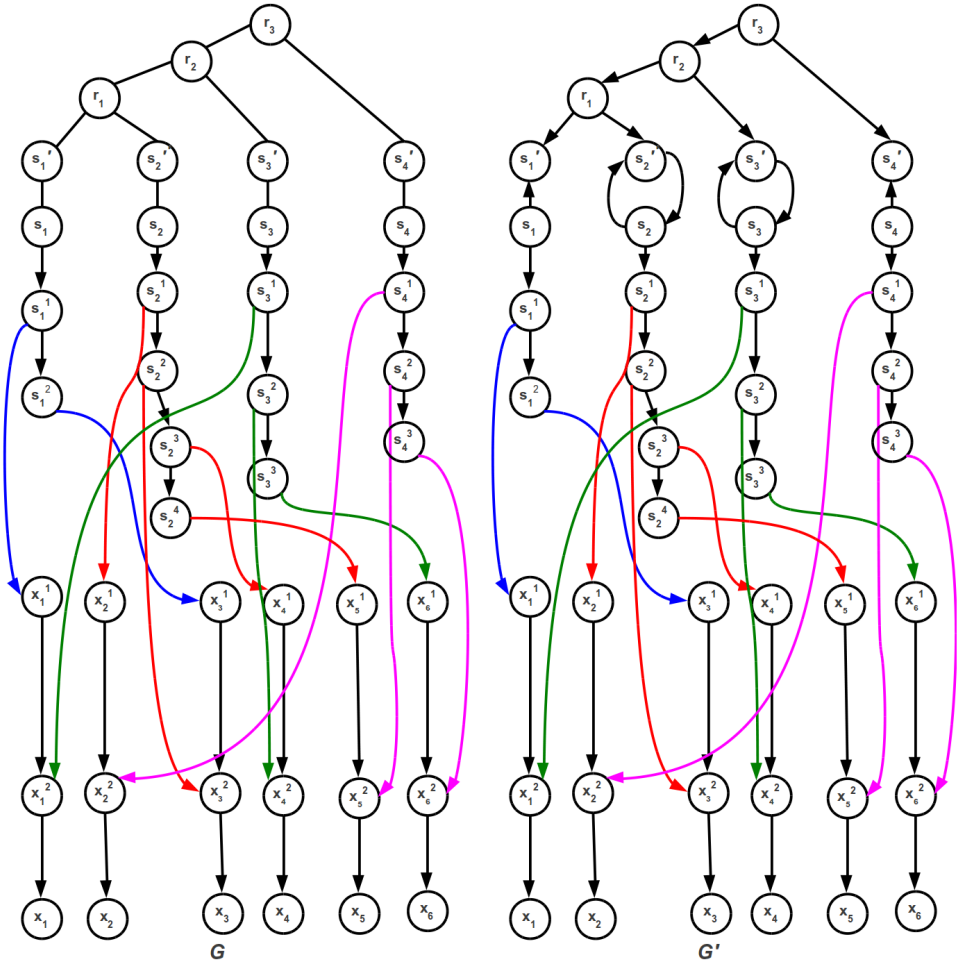


FIGURE 4.9 – *(a) Construction d'une instance* $(G, \mathcal{P})$ *de* MIN-D-GO *à partir d'une instance de* MINIMUM SET COVER-2. *Ici,* $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ *et* $\mathcal{C} = \{S_1, S_2, S_3, S_4\}$ *tel que* $S_1 = \{X_1, X_3\}$, $S_2 = \{X_2, X_3, X_4, X_5\}$, $S_3 = \{X_1, X_4, X_6\}$ *et* $S_4 = \{X_2, X_5, X_6\}$. *L'ensemble de paires est* $\mathcal{P} = \{(s_1, s_1'), (s_2, s_2'), (s_3, s_3'), (s_4, s_4'), (r_3, x_1), (r_3, x_2), (r_3, x_3), (r_3, x_4), (r_3, x_5), (r_3, x_6)\}$. *(b) Le graphe* G' *est une* $(\mathcal{P}, 2)$*-D-orientation pour* G *correspondant à l'ensemble couvrant* $\mathcal{C}' = \{S_2, S_3\}$.

choisi de sorte que le degré entrant de x_l^1 soit égal à 1 et le degré entrant de x_l^2 soit égal à 2. Pour finir la construction de G, nous ajoutons un sommet r_1 lié par deux arêtes à s_1' et s_2'. Ensuite, nous ajoutons un sommet r_2 lié par deux arêtes à r_1 et s_3'. On continue l'ajout des sommets r_i liés par deux arêtes à r_{i-1} et s_{i+1}', pour tout $3 \leq i \leq m-1$. L'ensemble des paires à satisfaire est $\mathcal{P} = \{(r_{m-1}, x_j), 1 \leq j \leq n\} \cup \{(s_i, s_i'), 1 \leq i \leq m\}$ avec $m = |\mathcal{C}|$. Un exemple de construction est donné dans la Figure 4.9.

Le degré de chaque sommet r_i dans G est au plus trois, et chaque sommet s_i^j est lié à un seul sommet $x_{i'}^{j'}$. Donc, nous pouvons facilement vérifier que $\Delta(G^*) = 3$.

Maintenant nous allons prouver la propriété suivante : pour tout entier $k \geq 0$, il existe un ensemble couvrant de $\mathcal{X}$ de cardinalité k si, et seulement si, il existe une $(\mathcal{P}, k)$-D-orientation pour G.

$\Rightarrow$: Étant donné un ensemble couvrant $\{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\}$, nous dou-

blons l'orientation des arêtes (s_{i_j}, s'_{i_j}), pour tout $1 \leq j \leq k$. Ensuite, nous remplaçons toute arête (s_i, s'_i) par l'arc $s_i s'_i$, pour tout $i \in \{1, 2, \ldots, m\} \backslash \{i_1, i_2, \ldots, i_k\}$. Enfin, nous orientons l'arbre induit par l'ensemble $\{r_i, 1 \leq i < m\} \cup \{s'_i, 1 \leq i \leq m\}$, pour créer une arborescence de racine r_{m-1}.

L'ensemble $\{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\}$ est un ensemble couvrant, donc pour tout $1 \leq \alpha \leq m$, il existe un entier $1 \leq j \leq k$ tel que le sommet x_α est joignable par un chemin orienté allant de s_{i_j}. Par ailleurs, grâce à l'arborescence de racine r_{m-1}, le sommet s_{i_j} est joignable par un chemin orienté allant de r_{m-1}. Donc, l'orientation obtenue est une $(\mathcal{P}, k)$-D-orientation de G.

$\Leftarrow$: On note par E_D l'ensemble des arêtes doublement orientées dans une $(\mathcal{P}, k)$-D-orientation de G. On pose $\mathcal{C}' = \{S_i : (s_i, s'_i) \in E_D\}$. Donc, $|\mathcal{C}'| \leq k$. Nous allons montrer que l'ensemble $\mathcal{C}'$ est un ensemble couvrant de $\mathcal{X}$. Supposons qu'il existe un $X_j \in \mathcal{X}$ tel que, pour tout $S_i \in \mathcal{C}'$, $X_j \notin S_i$. On note par $\mathcal{C}_j$ la collection des sous-ensembles qui contiennent X_j, c'est-à-dire, $\mathcal{C}_j = \{S \in \mathcal{C} : X_j \in S\}$. Le graphe G est construit de telle façon que, pour satisfaire une paire (r_{m-1}, x_j), nous devons créer au moins un arc $s'_i s_i$ tel que $S_i \in \mathcal{C}_j$. D'autre part, nous devons orienter chaque arête (s'_i, s_i), de s_i vers s'_i, pour satisfaire la paire $(s_i, s'_i) \in \mathcal{P}$. Donc, l'arête (s'_i, s_i) doit être remplacée par une arête doublement orientée, ce qui implique que $S_i \in \mathcal{C}'$. On obtient donc une contradiction, parce que $\mathcal{C}' \cap \mathcal{C}_j = \emptyset$. Donc, $\mathcal{C}'$ est un ensemble couvrant. Par ailleurs, $|\mathcal{C}'| \leq k$. Si $|\mathcal{C}'| \neq k$, nous pouvons compléter $\mathcal{C}'$ en ajoutant arbitrairement des éléments de $\mathcal{C}$ pour assurer que $|\mathcal{C}'| = k$ ($\mathcal{C}'$ reste également un ensemble couvrant).

La réduction proposée est bien une L-réduction préservant le paramètre k (le cardinal de l'ensemble couvrant et le nombre d'arêtes doublement orientées). Or, le problème Minimum Set Cover-2 est APX-difficile [PY91], donc on en déduit que le problème MIN-D-GO est aussi APX-difficile, même si $B = 2$ et $\Delta(G^*) = 3$.

□

Remarque 4.2 *Pour toute constante B supérieure ou égale à à 2, le problème* Minimum Set Cover-B *reste* APX*-difficile [PY91]. Donc, le résultat du Théorème 4.10 peut être généralisé pour toute constante B supérieure ou égale à à 2.*

En terme de complexité paramétrée, nous allons montrer que le problème MIN-D-GO est W[1]-difficile.

Théorème 4.11 *Le problème* MIN-D-GO *est* W[1]*-difficile (paramétré par le nombre d'arêtes doublement orientées) quand G est un MAG et B est non-borné, même si $\Delta(G^*) = 3$.*

Preuve. Nous utilisons la même réduction que dans la preuve précédente (la preuve du Théorème 4.10), mais nous considérons le problème Minimum Set Cover sans aucune contrainte sur le nombre d'occurrences des éléments de $\mathcal{X}$. Ce problème est W[1]-difficile, paramétré par par la taille de l'ensemble couvrant [RS97, PM81]. Par ailleurs, la réduction proposée est une L-réduction préservant le paramètre k (la taille de l'ensemble couvrant d'une et d'autre part le nombre d'arêtes doublement orientées). Donc, le problème MIN-D-GO est W[1]-difficile. □

Conclusion du chapitre

Dans ce chapitre, nous avons étudié deux problèmes d'orientation des graphes. Les deux problèmes sont motivés par des applications biologiques. Nous avons étudié la complexité de ces problèmes, et en particulier nous avons proposé des algorithmes polynomiaux pour certaines instances restreintes, ainsi que des résultats de difficulté et de non-approximabilité.

Cependant, des questions intéressantes restent encore ouvertes. Premièrement, puisque nous avons montré que MIN-D-GO et S-GO sont faciles quand $|\mathcal{P}| \leq 2$, et difficiles quand $|\mathcal{P}|$ est non-bornée, il est intéressant d'étudier la complexité de ces problèmes quand $|\mathcal{P}| = 3$ (ou plus généralement quand $|\mathcal{P}|$ est une constante supérieure ou égale à 3). Cette étude nous permettra d'identifier la frontière (vis-à-vis de $|\mathcal{P}|$) entre les instances faciles et celles difficiles pour ces problèmes. Il sera également intéressant de chercher s'il y a des classes des graphes spécifiques pour lesquelles ces problèmes peuvent être résolus par un algorithme d'approximation de ratio constant ou par un algorithme FPT.

Enfin, comme la plupart de réseaux biologiques sont scale-free et donc ayant un diamètre relativement faible [CHBA02, CH03, MV07], il serait intéressant d'étudier la complexité de MIN-D-GO et S-GO en fonction du diamètre du graphe d'entrée.

Conclusion générale et perspectives

Nous avons divisé ce manuscrit en quatre chapitres. Dans le premier chapitre, nous avons rappelé des notions de base en biologie, en théorie de graphes et en théorie de complexité. Dans le deuxième chapitre, nous avons proposé une approche innovante pour comparer deux réseaux biologiques hétérogènes représentés respectivement par un graphe orienté D et un graphe non-orienté G. Nous avons formulé le problème Skew SubGraph Mining (abrégé SkewGraM) issu de notre modélisation. Dans un premier temps, nous avons considéré que le graphe orienté D est acyclique (DAG). Pour étudier la complexité du problème SkewGraM, nous avons commencé par sa version la plus simple, appelée One-to-One SkewGraM, dans laquelle $V(D) = V(G)$. Nous avons montré que malgré ces restrictions, le problème One-to-One SkewGraM est (i) APX- difficile même si G est un arbre de diamètre 4 et (ii) NP-complet même si D^* est un graphe planaire extérieur et G est un arbre de diamètre 4. Nous avons montré que le paramètre $tw(D^*)$ définit la frontière entre les instances faciles $(tw(D^*) = 1)$ et les instances difficiles $(tw(D^*) = 2)$ du problème. Pour les cas difficiles, nous avons proposé deux algorithmes : une heuristique appelée AlgoH et un algorithme exact, appelé AlgoBB, basé sur la méthode branch-and-bound.

Nous avons également proposé une méthode permettant de résoudre le problème SkewGraM à partir d'une solution de One-to-One SkewGraM, obtenue par l'algorithme AlgoH ou AlgoBB. Dans le but de montrer l'efficacité de notre heuristique, nous l'avons appliquée sur des données simulées et sur des données biologiques.

Pour traiter One-to-One SkewGraM dans le cas où le graphe D est cyclique, une manière de procéder est de décomposer D en DAGs. C'est pour cette raison que nous avons introduit et étudié, dans le troisième chapitre, une méthode de décomposition motivée par des hypothèses biologiques. La méthode étudiée consiste à décomposer D en DAGs tels que les sommets de chaque DAG induisent dans G un sous-graphe connexe.

Nous nous sommes intéressés, dans le dernier chapitre, au problème biologique de l'inférence des voies de signalisation en combinant les informations sur les causes et sur les effets des événement extra-cellulaires. Nous avons montré que ce problème peut être modélisé par un problème d'orientation des graphes mixtes. Nous avons étudié la complexité de deux problèmes d'orientation, appelés respectivement S-GO et MIN-D-GO. Dans le premier problème, chaque arête de G est remplacée par un seul arc. Tandis que, dans le deuxième problème nous avons autorisé certaines arêtes à être doublement orientées, mais nous avons

cherché à minimiser le nombre de telles arêtes. Pour chacun de ces deux problèmes, nous avons identifié des cas faciles et des cas difficiles en considérant différents paramètres.

Les travaux et résultats présentés dans cette thèse nous ouvrent de nombreuses perspectives.

1. **Questions ouvertes.** Nous allons commencer par étudier les questions ouvertes que nous avons présentées en fin de chaque chapitre. Dans la plupart de ces questions, nous cherchons d'abord s'il existe un algorithme polynomial qui résout le problème. Par exemple, peut-on résoudre en temps polynomial les problèmes S-GO et MIN-D-GO quand G est un MAG, $\Delta(G) = 3$ et $|\mathcal{P}| = 3$?

   Dans le cas où nous sommes en face d'un problème NP-difficile, nous proposons de contourner la NP-difficulté en cherchant un algorithme d'approximation ou un algorithme FPT. Par exemple, en s'inspirant des algorithmes FPT, présentés dans [Mon85, AYZ95], pour résoudre le problème de la recherche d'un plus long chemin dans un graphe orienté, peut-on trouver un algorithme FPT résolvant le problème One-to-One SkewGraM de la recherche d'un plus long chemin consistant ?

   Toutefois, dans certains cas, nous avons montré que le problème en question était non-approximable et aussi W[1]-difficile (en considérant le paramètre standard), comme par exemple pour le problème MIN-D-GO (voir le Tableau 4.1). Pour ces cas, il serait intéressant de trouver des classes de graphes ou des paramètres spécifiques qui rendent le problème moins difficile, ou de trouver un algorithme exact efficace, par exemple un algorithme dont la complexité est de $\mathcal{O}((c)^n)$ où n est la taille des instances et c est une constante aussi petite que possible.

2. **Généralisation.** Dans une perspective à court terme, nous souhaitons généraliser notre approche de comparaison de réseaux hétérogènes pour effectuer une comparaison multiple. Une généralisation possible du problème One-to-One SkewGraM est le problème de maximisation M-One-to-One SkewGraM défini comme suit :

   > M-One-to-One SkewGraM
   > **Instance** : Un ensemble de DAGs $\{D_1, \ldots, D_{m_1}\}$ et un ensemble de graphes non-orientés $\{G_1, \ldots, G_{m_2}\}$ ayant tous le même ensemble de sommets $\{1, 2, \ldots, n\}$.
   > **Solution** : Un chemin commun entre tous les graphes D_i, $1 \leq i \leq m_1$, et dont les sommets induisent un sous-graphe connexe dans tous les graphes G_j, $1 \leq j \leq m_2$.
   > **Mesure** : La longueur du chemin recherché.

3. **WEB/Logiciel.** A moyen et à long terme, nous envisageons de produire une interface web ou un logiciel permettant de faciliter aux biologistes l'utilisation de nos algorithmes, en particulier l'heuristique AlgoH.

4. **Nouvelle variante d'orientation de graphes.** Dans cette thèse, nous avons étudié divers types de données biologiques, à savoir les don-

nés protéomiques, métabolomiques et transcriptomiques. Mais, bien entendu cette liste n'est pas exhaustive, car les données biologiques sont en accroissement continu. Des études récentes se sont intéressées à l'analyse de réseaux dits *réseaux de co-expression de gènes* [SSKK03, LHS+04, OHS+08, RDZ10]. Un réseau de co-expression est modélisé par un graphe non-orienté dont les sommets sont les gènes, et les arêtes représentent une corrélation entre l'expression de deux gènes. Dans tous les problèmes d'orientation que nous avons étudiés, nous cherchons à satisfaire une paire de sommets par un chemin arbitraire. Il est intéressant de prendre en compte, dans les orientations, la proximité génomique des gènes dans le réseau de co-expression. En effet, les gènes qui se trouvent sur une même voie de signalisation ont tendance à avoir une corrélation d'expression [RDZ10]. A long terme, nous envisageons donc d'étudier le problème de la recherche d'une orientation (simple) G' d'un graphe mixte $G = (V, E, A)$ qui satisfait un ensemble de paires $\mathcal{P} \subseteq V \times V$, par des chemins (G', H, f)-consistants (voir Définition 2.5 du Chapitre 2), telle que (i) H est un graphe non-orienté représentant un réseau de co-expression et f est une fonction de correspondance entre $V(G)$ et $V(H)$.

Annexes

A

Nous rappelons que les modules biologiques d'un réseau donné peuvent être modélisés par des structures (sous-graphes) dont la topologie dépend de la nature du réseau en question. Dans cet annexe, nous considérons les trois réseaux biologiques les plus étudiés en littérature, à savoir les réseaux *PPI*, les réseaux métaboliques et les réseaux de régulation de la transcription des gènes. Nous présentons pour chaque réseau (i) des propriétés topologiques, (ii) des modules biologiques et (iii) des structures modélisant les modules biologiques identifiés dans (ii).

A.1 Caractéristiques générales

A.1.1 Réseaux *PPI*

Comme la plupart des réseaux réels, les réseaux *PPI* sont scale-free, c'est-à-dire il existe peu des nœuds hautement connectés (*hubs*), le reste des nœuds ayant un degré faible (voir la Section 2.8 du Chapitre 2). Les réseaux *PPI* possèdent également la propriété "petit-monde" [YOB04] (ou en anglais *small-world*) : le diamètre du réseau est petit par rapport à la taille du réseau.

Topologie du graphe (distribution de nœuds)	**Modules Biologiques**	**Structures recherchées**
• Scale-free • Petit-monde	• Complexes de protéines • Groupes orthologues • Voies métaboliques (*Metabolic pathways*) • Voies de signalisation (*Signaling pathways*)	• Sous-graphes fréquents • Sous-graphes denses • Clusters • Composantes connexes communes (CCCs) • Chemins • *Graphlet* • Sous-arbres centrés • Motifs

Tableau A.1 – *Caractéristiques des réseaux PPI. Les structures sont détaillées dans le Tableau A.4.*

A.1.2 Réseaux métaboliques

La plupart des études montre que les réseaux métaboliques sont scale-free. En revanche, certains auteurs trouvent des chemins très longs dans ces réseaux, ce qui contredit la propriété petit-monde (voir [Ari04]).

Topologie du graphe (distribution de nœuds)	Modules Biologiques	Structures recherchées
• Scale-free • Petit-monde ?	• Complexes d'enzymes • Classification des enzymes	• Sous-graphes fréquents • Sous-graphes denses • Cluster • Composantes connexes communes (CCC) • Chemin • Graphes orientés sans cycles (DAG) • Cycles • Motifs

Tableau A.2 – *Caractéristiques des réseaux métaboliques. Les structures sont détaillées dans le Tableau A.5.*

A.1.3 Réseaux de régulation de la transcription des gènes

En ce qui concerne la topologie de ces réseaux, certains auteurs ont montré que la distribution des degrés entrants suit une loi exponentielle et celle des degrés sortants suit une loi de puissance, c'est-à-dire scale-free. D'autres auteurs ont observé un comportement hybride entre le scale-free et la loi exponentielle.

Topologie du graphe (distribution de nœuds)	Modules Biologiques	Structures recherchées
• **Degrés entrants** : Loi exponentielle • **Degrés sortants** : Scale-free • Comportement hybride	• Opérons • Modules de transcription ou *régulons* (ensembles des gènes régulés par le même facteur de transcription)	• Clusters • 3-Couches (*Three-layers*) • *Dense overlapping regulons* (*DORs*) • Composantes fortement connexes (SCCs) • Motifs

Tableau A.3 – *Caractéristiques des réseaux de régulation de la transcription des gènes. Les structures sont détaillées dans le Tableau A.6.*

A.2 Structures recherchées

A.2.1 Réseaux *PPI*

Structure	Références
Sous-graphes	[GH09, ZZW+07, TGH09, ESU08, SXB07, KGS05, BHMK+09, ZRBV09, SXB08b]
Sous-graphe denses	[AS06, SUS07, KBS09, FNS+06, LCTS08]
Clusters	[TGH09, SUS07, Kla09, LLB+09, SLF06, LCTS08, KSK+03]
CCCs	[BML+05, DBVS09, NK07, TF09, KMM+10]
Chemins	[SSRS06, ZZW+07, LLB+09, KYL+04, FNS+06, HMD06, SLF06, KSK+03]
Graphlets	[Prz09, KMM+10]
Motifs	[Gri05, DSS05, CG08, SHM06]
Arbres	[DSG+08, HMD06]
Sous-arbre centrés	[JHM08]
Graphes de *treewidth* borné	[DSG+08]
2*SPG*	[HMD06]

Tableau A.4 – *Structures recherchées dans les réseaux PPI. Les complexes de protéines peuvent être modélisés par clusters ou sous-graphes denses dans le graphe d'alignement multiple de réseaux PPI (voir la Section 2.1 du Chapitre 2). Les chemins modélisent les voies de signalisation (signaling pathways).*

A.2.2 Réseaux métaboliques

Les voies métaboliques et les complexes d'enzymes peuvent être modélisés par des composantes connexes communes ou sous-graphes communs dans les graphes représentant les réseaux métaboliques. Certains auteurs modélisent les voies métaboliques par des graphes orientés acycliques (DAGs).

Structure	Références
Sous-graphes fréquents (alignement multiple)	[KGS04, Kla09, ZZW+07, SXB08a]
Sous-graphe	[WR07, TGK04, BLC+07]
Cluster	[BLC+07]
CCC	[BML+05, DBVS09]
Chemin	[Alb05, ZZW+07, BLC+07, LCTS08]
DAG	[GHM+02, PRYLZU05]
Cycle	[BLC+07]
Motifs	[ELJ06, LCTS08]

Tableau A.5 – *Structures recherchées dans les réseaux métaboliques.*

A.2.3 Réseaux de régulation de la transcription des gènes.

On remarque la présence de motifs dans les réseaux de régulation de la transcription des gènes, beaucoup plus en comparant avec les autres réseaux biologiques. Des études montrent que ces réseaux sont structurés par couches (entrée, intermédiaire, sortie). La couche intermédiaire est formée par des régions de connexions très denses.

Certains auteurs trouvent des composantes fortement connexes (ou SCC, pour Strongly Connected Components) dans les réseaux de régulation de la transcription des gènes [AA03]. En revanche, d'autres auteurs confirment l'absence des telles structures dans ces réseaux [FWC$^+$06].

Structure	**Références**
Cluster	[AA03]
Three-layers	[FWC$^+$06]
DOR (Dense overlapping regulons)	[AA03, Alo07a, SHM06, FWC$^+$06]
SCC ?	[FWC$^+$06, AA03]
Motifs	[FWC$^+$06, Alo03, AA03, ELJ06, Alo07a, Alo07b, BL04, SHM06]

Tableau A.6 – *Structures recherchées dans les réseaux de régulation de la transcription des gènes. DOR est une région dense de connexions entre les facteurs de transcription et les opérons.*

Bibliographie

[AA03] E. Alm and A. Arkin. Biological networks. *Current opinion in structural biology*, 13 :193–202, 2003. (Cité page 128)

[ADF93] K.R. Abrahamson, R.G. Downey, and M.R. Fellows. Fixed-parameter intractability II (extended abstract). In Proc. *Symposium on Theoretical Aspects of Computer Science (STACS 1993)*, volume 665 of *LNCS*, pages 374–385, 1993. (Cité page 28)

[AGS04] O. Aparicio, J.V. Geisberg, and K. Struhl. Chromatin immunoprecipitation for determining the association of proteins with specific genomic sequences in vivo. *Current Protocols in Molecular Biology*, 21 :21.3.1–21.3.33, 2004. (Cité page 15)

[AH02] E. Arkin and R. Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116 :271–278, 2002. (Cité pages 89, 91, 92, 97, 100, 101, 112 et 113)

[AHK77] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21 :429–567, 1977. (Cité page 20)

[AJ07] Y. Aifeng and Y. Jinjiang. On the vertex arboricity of planar graphs of diameter two. *Discrete Mathematics*, 307 :2438–2447, 2007. (Cité pages 69 et 70)

[AK00] P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237 :123–134, 2000. (Cité page 45)

[Alb05] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118 :4947–4957, 2005. (Cité pages 12, 13, 14, 16 et 127)

[Alo03] U. Alon. Biological networks : the tinkerer as an engineer. *Science*, 301(5641) :1866–1867, 2003. (Cité page 128)

[Alo07a] U. Alon. Network motifs : theory and experimental approaches. *Nature Reviews Genetics*, 8 :450–461, 2007. (Cité pages 15, 16 et 128)

[Alo07b] U. Alon. Simplicity in biology. *Nature*, volume 446, article no. 497, 2007. (Cité page 128)

[APT79] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8 :121–123, 1979. (Cité page 99)

[Ari04] M. Arita. The metabolic world of escherichia coli is not small. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 101 :1543–1547, 2004. (Cité page 125)

[ASo6] T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7 :243–255, 2006. (Cité page 127)

[AYZ95] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42 :844–856, 1995. (Cité page 120)

[Baro9] A.-L. Barabási. Scale-free networks : a decade and beyond. *Science*, 325 :412–413, 2009. (Cité page 57)

[Ber76] C. Berge. *Graphs and hypergraphs*, volume 6. North-Holland Mathematical Library, second revised edition, 1976. (Cité page 17)

[BFMB+11] G. Blin, G. Fertin, H. Mohamed-Babou, I. Rusu, F. Sikora, and S. Vialette. Algorithmic aspects of heterogeneous biological networks comparison. In Proc. *Conference on Combinatorial Optimization and Applications (COCOA 2011)*, volume 6831 of *LNCS*, pages 272–286, 2011. (Cité page 67)

[BHMK+09] S. Bruckner, F. Hüffner, R.M. M. Karp, R. Shamir, and R. Sharan. Topology-free querying of protein interaction networks. In Proc. *Research in Computational Molecular Biology (RECOMB 2009)*, volume 5541 of *LNCS*, pages 74–89, 2009. (Cité pages 10 et 127)

[BJGo1] J. Bang-Jensen and G. Gutin. *Digraphs : theory, algorithms, and applications*. Springer, 2001. (Cité page 17)

[BLo4] J. Berg and M. Lassig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 101 :14689–14694, 2004. (Cité pages 13, 14, 15 et 128)

[BLA+04] M.M. Babu, N.M. Luscombe, L. Aravind, M. Gerstein, and S.A. Teichmann. Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology*, 14 :283–291, 2004. (Cité page 16)

[BLC+07] R. Bourqui, V. Lacroix, L. Cottret, D. Auber, P. Mary, M-.F. Sagot, and F. Jourdan. Metabolic network visualization eliminating node redundance and preserving metabolic pathways. *BMC Systems Biology*, volume 1, article no. 29, 2007. (Cité pages 13, 14 et 127)

[BML+05] F. Boyer, A. Morgat, L. Labarre, J. Pothier, and A. Viari. Syntons, metabolons and interactons : an exact graph-theoretical approach for exploring neighbourhood between genomic and functional data. *Bioinformatics*, 21 :4209–4215, 2005. (Cité pages 5, 10, 12, 13, 32, 36, 37, 63, 68 et 127)

[Bod88] H.L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36 :116–125, 1988. (Cité page 40)

[Broo8] S. Brohée. *Etude bioinformatique du réseau d'interactions entre protéines de transport chez les Fungi*. PhD thesis, Université Libre de Bruxelles, Bruxelles, Belgique, 2008. (Cité pages 15 et 57)

[BT80] F. Boesch and R. Tindell. Robbins's theorem for mixed graphs. *American Mathematical Monthly*, 86 :716–719, 1980. (Cité page 89)

[Bun00] H. Bunke. Graph matching : theoretical foundations, algorithms and applications. In Proc. *International Conference on Vision Interface (VI 2000)*, pages 82–88, 2000. (Cité page 33)

[BW07] F. Bruggeman and H. Westerhoff. The nature of systems biology. *Trends in Microbiology*, 15 :45–50, 2007. (Cité page 5)

[CFSV04] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18 :265–298, 2004. (Cité page 33)

[CG08] G. Ciriello and C. Guerra. A review on models and algorithms for motif discovery in protein-protein interaction networks. *Briefings in Functional Genomics*, 7 :147–56, 2008. (Cité pages 10 et 127)

[CH69] G. Chartrand and Kronk H.V. The point-arboricity of planar graphs. *Journal of the London Mathematical Society*, 44 :612–616, 1969. (Cité page 69)

[CH03] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Physical Review Letters*, 90 :058701–1–058701–4, 2003. (Cité page 118)

[CHBA02] R. Cohen, S. Havlin, and D. Ben-Avraham. *Structural properties of scale free networks*, chapter 4. Handbook of graphs and networks : from the genome to the internet. Wiley-VCH, 2002. (Cité page 118)

[Che81] C.C. Chen. On a characterization of planar graphs. *Bulletin of the Australian Mathematical Society*, 24 :289–294, 1981. (Cité page 20)

[CHSW94] G. Chartrand, F. Harary, M. Schultz, and C. Wall. Forced orientation numbers of a graph. *Congressus Numerantium*, 100 :183–191, 1994. (Cité page 89)

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In Proc. *Symposium on Theory of Computing (STOC 1971)*, ACM Press, pages 151–158, 1971. (Cité page 24)

[CSH+05] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. In Proc. *ACM-SIGKDD Workshop on Link Discovery : Issues, Approaches and Applications (LinkKDD 2005)*, pages 58–65. ACM Press, 2005. (Cité page 38)

[DBVS09] Y-.P. Deniélou, F. Boyer, A. Viari, and M-.F. Sagot. Multiple alignment of biological networks : a flexible approach. In Proc. *Combinatorial Pattern Matching (CPM 2009)*, volume 5577 of *LNCS*, pages 263–273, 2009. (Cité pages 5, 10, 32, 36, 68 et 127)

[Deb08a] R. Deb. Acyclic partitioning problem is NP-complete for k=2. 2008. Private communication,Yale University, United States. (Cité page 69)

[Deb08b] R. Deb. An efficient nonparametric test of the collective household model. 2008. Private communication,Yale University, United States. (Cité page 69)

[DF92] R.G. Downey and M.R. Fellows. Fixed parameter tractability and completeness. In Proc. *Complexity Theory : Current Research, Dagstuhl Workshop*, pages 191–225. Cambridge University Press, 1992. (Cité page 28)

[DFR98] R.G. Downey, M.R. Fellows, and K.W. Regan. Parameterized circuit complexity and the W hierarchy. *Theoretical Computer Science*, 191 :97–115, 1998. (Cité pages 28 et 29)

[DHK+11] B. Dorn, F. Hüffner, D. Kruger, R. Niedermeier, and J. Uhlmann. Exploiting bounded signal flow for graph orientation based on cause-effect pairs. *Algorithms for Molecular Biology*, 6 :21, 2011. (Cité page 88)

[Die05] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 2005. (Cité page 17)

[DSG+08] B. Dost, T. Shlomi, N. Gupta, E. Ruppin, Bafna V., and Sharan R. QNet : a tool for querying protein interaction networks. *Journal of Computational Biology*, 15 :913–925, 2008. (Cité pages 10 et 127)

[DSS05] E. De Silva and M.-P Stumpf. Complex networks and simple models in biology. *Journal of the Royal Society Interface*, 2 :419–430, 2005. (Cité page 127)

[DW08] P. Durek and D. Walther. The integrated analysis of metabolic and protein interaction networks reveals novel molecular organizing principles. *BMC Systems Biology*, volume 2, article no. 100, 2008. (Cité pages 36, 59 et 61)

[ELJ06] Y-.H. Eom, S. Lee, and H. Jeong. Exploring local structural organization of metabolic networks using subgraph patterns. *Journal of Theoretical Biology*, 241 :823–829, 2006. (Cité pages 11, 127 et 128)

[ER59] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6 :290–297, 1959. (Cité page 57)

[ESD+11] M. Elberfeld, D. Segev, C.R. Davidson, D. Silverbush, and R. Sharan. Approximation algorithms for orienting mixed graphs. In Proc. *Combinatorial Pattern Matching (CPM 2011)*, volume 6661 of *LNCS*, pages 416–428, 2011. (Cité pages 86 et 88)

[ESU08] P. Evans, T. Sandler, and L. Ungar. Protein-protein interaction network alignment by quantitative simulation. In Proc. *Bioinformatics and Biomedicine (BIMB 2008)*, pages 325–328. IEEE Computer Society, 2008. (Cité pages 10 et 127)

[FAC+08] J. Flannick, F.N Antal, B.D Chuong, S. Balaji, and S. Batzoglou. Automatic parameter learning for multiple network alignment. In Proc. *Research in Computational Molecular Biology (RECOMB 2008)*, volume 4955 of *LNCS*, pages 214–231, 2008. (Cité pages 5 et 35)

[Fel04] Stefan Felsner. *Geometric graphs and arrangements : some chapters from combinatorial geometry*. Vieweg, 2004. (Cité page 20)

[FMBR12a] G. Fertin, H. Mohamed-Babou, and I. Rusu. Algorithms for subnetwork mining in heterogeneous networks. In *Symposium on Experimental Algorithms (SEA)*, volume 7276 of *LNCS*, pages 184–194, 2012. (Cité page 31)

[FMBR12b] G. Fertin, H. Mohamed-Babou, and I. Rusu. On the complexity of two problems on orientations of mixed graphs. In Proc. *Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM 2012)*, Actes de JOBIM, pages 161–170, 2012. (Cité page 85)

[FNS+06] J. Flannick, A. Novak, B. Srinivasan, H. McAdams, and S. Batzoglou. Graemlin : General and robust alignment of multiple large interaction networks. *Genome Research*, 16 :1169–1181, 2006. (Cité pages 10, 35 et 127)

[FS89] S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340 :245–246, 1989. (Cité page 10)

[FWC+06] I. Farkas, Ch. Wu, C. Chennubhotla, I. Bahar, and Z. Oltvai. Topological basis of signal integration in the transcriptional-regulatory network of the yeast, Saccharomyces cerevisiae. *BMC Bioinformatics*, volume 7, article no. 478, 2006. (Cité pages 15 et 128)

[GH09] X. Guo and A. Hartemink. Domain-oriented edge-based alignment of protein interaction networks. *Bioinformatics*, 25 :i240–i246, 2009. (Cité pages 10 et 127)

[GHM+02] A. Goesmann, M. Haubrock, F. Meyer, J. Kalinowski, and R. Giegerich. PathFinder : reconstruction and dynamic visualization of metabolic pathways. *Bioinformatics*, 18 :124–9, 2002. (Cité pages 38, 59, 68 et 127)

[GHPR03] A.-T. Gai, M. Habib, C. Paul, and M. Raffinot. Identifying common connected components of graphs. Technical Report RR-LIRMM-03016, LIRMM, Université de Montpellier 2, 2003. (Cité pages 41, 48, 49 et 51)

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. (Cité pages 20, 23, 24, 38, 42, 69, 74, 75 et 82)

[GK00] M.Y. Galperin and E.V. Koonin. Who's your neighbour ? New computational approaches for functional genomics. *Nature Biotechnology*, 18 :609–613, 2000. (Cité pages 5 et 35)

[GL09] Z. Gotthilf and M. Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109 :352–355, 2009. (Cité pages 98 et 99)

[Gri05] M.-G Grigorov. Global properties of biological networks. *Drug Discovery Today*, 10 :365–372, 2005. (Cité page 127)

[GSS10] I. Gamzu, D. Segev, and R. Sharan. Improved orientations of physical networks. In Proc. *Workshop on Algorithms in Bioinformatics (WABI 2010)*, volume 6293 of *LNCS*, pages 215–225, 2010. (Cité pages 86, 87 et 88)

[Han08] J-.D. Han. Understanding biological functions through molecular networks. *Cell Research*, 18 :224–237, 2008. (Cité page 5)

[HGH08] C. Huthmacher, C. Gille, and H.G. Holzhütter. A computational analysis of protein interactions in metabolic networks reveals novel enzyme pairs potentially involved in metabolic channeling. *Journal of Theoretical Biology*, 252 :456–464, 2008. (Cité page 36)

[HHLM99] L.H. Hartwell, J.J. Hopfield, S. Leibler, and A.W. Murray. From molecular to modular cell biology. *Nature*, 402 :47–52, 1999. (Cité page 32)

[HM89] R. Hassin and N. Megiddo. On orientations and shortest paths. *Linear Algebra and its Applications*, 114 :589–602, 1989. (Cité pages 89 et 92)

[HMD06] E. Hirsh, S. Mahleb, and B. Davidovich. Analysis of biological networks : protein modules-color coding. Unpublished, 2006. (Cité pages 10, 35 et 127)

[HSY97] S.L. Hakimi, E.F. Schmeichel, and N.E. Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63 :229–235, 1997. (Cité page 89)

[ICO+01] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 98 :4569–4574, 2001. (Cité page 10)

[ILB04] J. Ihmels, R. Levy, and N. Barkai. Principles of transcriptional control in the metabolic network of Saccharomyces cerevisiae. *Nature Biotechnology*, 22 :86–92, 2004. (Cité page 36)

[JHM08] P. Jancura, J. Heringa, and E. Marchiori. Dividing protein interaction networks by growing orthologous articulations. In Proc. *Pattern Recognition in Bioinformatics (PRIB 2008)*, volume 5265 of *LNCS*, pages 187–200, 2008. (Cité pages 10 et 127)

[JMBO01] H. Jeong, S.P. Mason, A.-L. Barabási, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411 :41–42, 2001. (Cité page 10)

[KBS09] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. *Computational Biology*, 16 :989–999, 2009. (Cité pages 10, 35 et 127)

[KGS04] M. Koyutürk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20 :200–207, 2004. (Cité pages 10, 11, 13 et 127)

[KGS05] M. Koyutürk, A. Grama, and W. Szpankowski. Pairwise local alignment of protein interaction networks guided by models of evolution. In Proc. *Research in Computational Molecular Biology (RECOMB 2005)*, volume 3500 of *LNCS*, pages 48–65, 2005. (Cité pages 10 et 127)

[Kla09] G. Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, volume 10, article no. 59, 2009. (Cité pages 10, 12, 13 et 127)

[KMM+10] O. Kuchaiev, T. Milenkovic, V. Memisevic, W. Hayes, and N. Przulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7 :1341–1354, 2010. (Cité pages 32, 68 et 127)

[KS08] G. Karlebach and R. Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9 :770–780, 2008. (Cité page 15)

[KSK+03] B.P. Kelley, R. Sharan, R.M. Karp, T. Sittler, D. Root, B. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 100(20) :11394–11399, 2003. (Cité pages 5, 33, 34, 35 et 127)

[KVC04] P. Kharchenko, D. Vitkup, and G.M. Church. Filling gaps in a metabolic network using expression information. *Bioinformatics*, 20 :1178–1185, 2004. (Cité page 36)

[KYL+04] B.P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. Stockwell, and T. Ideker. PathBLAST : a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32 :83–88, 2004. (Cité pages 10 et 127)

[LCTS08] V. Lacroix, L. Cottret, P. Thébault, and M-.F. Sagot. An introduction to metabolic networks and their structural analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5 :594–617, 2008. (Cité pages 11 et 127)

[LD60] A.H. Land and A.G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28 :497–520, 1960. (Cité page 27)

[LDAM04] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306 :1555–1558, 2004. (Cité page 36)

[LDM+97] D. A. Lashkari, J. L. DeRisi, J. H. McCusker, A. F. Namath, C. Gentile, S.Y. Hwang, P. O. Brown, and R.W. Davis. Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 94 :13057–13062, 1997. (Cité pages 15 et 20)

[LHS+04] H.K. Lee, A.K. Hsu, J. Sajdak, J. Qin, and P. Pavlidis. Coexpression analysis of human genes across many microarray data sets. *Genome Research*, 14 :1085–1094, 2004. (Cité page 121)

[LLB+09] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. IsoRankN : spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25 :253–258, 2009. (Cité pages 10, 35 et 127)

[Luc08] U. Lucia. Probability, ergodicity, irreversibility and dynamical systems. *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 464 :1089–1104, 2008. (Cité page 90)

[MBZS08] A. Medvedovsky, V. Bafna, U. Zwick, and R. Sharan. An algorithm for orienting graphs based on cause-effect pairs and its applications to orienting protein networks. In Proc. *Workshop on Algorithms in Bioinformatics (WABI 2008)*, volume 5251 of *LNCS*, pages 222–232, 2008. (Cité pages 16, 86, 87 et 88)

[MHT+06] Y. Matsuo, M. Hamasaki, H. Takeda, J. Mori, D. Bollegara, Y. Nakamura, T. Nishimura, K. Hasida, and M. Ishizuka. Spinning multiple social networks for semantic web. In Proc. *Association for the Advancement of Artificial Intelligence (AAAI 2006)*, AAAI Press, pages 1381–1387, 2006. (Cité page 38)

[Mon85] B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics*, 25 :239–254, 1985. (Cité page 120)

[MS10] T. Mchedlidze and A. Symvonis. Unilateral orientation of mixed graphs. In Proc. *Software Seminar (SOFSEM 2010)*, volume 5901 of *LNCS*, pages 588–599, 2010. (Cité page 90)

[MV07] O. Mason and M. Verwoerd. Graph theory and networks in biology. *IET Systems Biology*, 1 :89–119, 2007. (Cité page 118)

[NCDR+10] F. Nobibon, L. Cherchye, B. De-Rock, J. Sabbe, and F. Spieksma. Heuristics for deciding collectively rational consumption behavior. *Computational Economics*, 38(2) :173–204, 2010. (Cité page 69)

[NHLS10] F.T. Nobibon, C. Hurkens, R. Leus, and F. Spieksma. Exact algorithms for coloring graphs while avoiding monochromatic cycles. In Proc. *Algorithmic Aspects in Information and Management (AAIM 2010)*, volume 6124 of *LNCS*, pages 229–242, 2010. (Cité page 69)

[NK07] M. Narayanan and R.M. Karp. Comparing protein interaction networks via a graph match-and-split algorithm. *Computational Biology*, 14 :892–907, 2007. (Cité pages 10, 32, 68 et 127)

[OFGK00] H. Ogata, W. Fujibuchi, S. Goto, and M. Kanehisa. A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucleic Acids Research*, 28 :4021–4028, 2000. (Cité pages 5 et 36)

[OHS+08] T. Obayashi, S. Hayashi, M. Shibaoka, M. Saeki, H. Ohta, and K. Kinoshita. COXPRESdb : a database of coexpressed gene networks in mammals. *Nucleic Acids Research*, 36 :77–82, 2008. (Cité page 121)

[PH04] C. Pal and L.D. Hurst. Evidence against the selfish operon theory. *Trends in Genetics*, 20 :232–234, 2004. (Cité page 36)

[PM81] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theorical Computer Science*, 15 :251–277, 1981. (Cité page 117)

[PRYLZU05] R. Pinter, O. Rokhlenko, E. Yeger-Lotem, and M. Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics*, 21(16) :3401–3408, 2005. (Cité pages 13, 32, 38, 59, 68 et 127)

[Prz09] N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23 :177–183, 2009. (Cité page 127)

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43 :425–440, 1991. (Cité pages 25, 26, 78, 115 et 117)

[RDZ10] J. Ruan, A. Dean, and W. Zhang. A general co-expression network-based approach to gene expression analysis : comparison and applications. *BMC Systems Biology*, volume 4, article no. 8, 2010. (Cité page 121)

[Rob39] E. Robbins. A theorem on graphs with an application to a problem on traffic control. *American Mathematical Monthly*, 46 :281–283, 1939. (Cité page 89)

[RS86] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7 :309–322, 1986. (Cité page 40)

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In Proc. *Symposium on Theory of Computing (STOC 1997)*, pages 475–484. ACM Press, 1997. (Cité page 117)

[RSK95] A. Roychoudhury and S. Sur-Kolay. Efficient algorithms for vertex arboricity of planar graphs. In Proc. *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1995)*, volume 1026 of *LNCS*, pages 37–51, 1995. (Cité page 69)

[RTT02] S.C Rison, S.A Teichmann, and J.M. Thornton. Homology, pathway distance and chromosomal localisation of the small molecule metabolism enzymes in Escherichia coli. *Journal of Molecular Biology*, 318 :911–932, 2002. (Cité page 36)

[SBR+06] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID : a general repository for interaction datasets. *Nucleic Acids Research*, 34 :535–539, 2006. (Cité page 61)

[SES11] D. Silverbush, M. Elberfeld, and R. Sharan. Optimally orienting physical networks. In Proc. *Workshop on Algorithms in Bioinformatics (WABI 2011)*, volume 6577 of *LNCS*, pages 424–436, 2011. (Cité pages 16, 86, 88 et 93)

[SHM06] R. Sharan, A. Halpern, and E. Mashiach. Analysis of biological networks : network motifs. Unpublished, 2006. (Cité pages 127 et 128)

[SI06] R. Sharan and T. Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 4 :427–433, 2006. (Cité page 5)

[SIK+04] R. Sharan, T. Ideker, B.P. Kelley, R. Shamir, and R.M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. In Proc. *Research in Computational Molecular Biology (RECOMB 2004)*, pages 282–289. ACM, 2004. (Cité pages 5, 33 et 35)

[SLF06] R. Sharan, O. Lavi, and L. Ferdinskoif. Protein-protein interaction : network alignment. Unpublished, 2006. (Cité pages 10 et 127)

[SLW+06] S. Sinha, X. Ling, C. Whitfield, C. Zhai, and G. Robinson. Genome scan for cis-regulatory DNA motifs associated with social behavior in honey bees. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 103 :16352–16357, 2006. (Cité page 15)

[SSK+05] R. Sharan, S. Suthram, B.P. Kelley, T. Kuhn, S. Mccuine, P. Uetz, T. Sittler, R.M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 102 :1974–1979, 2005. (Cité pages 5, 33 et 35)

[SSKK03] J.M. Stuart, E. Segal, D. Koller, and S.K. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302 :249–255, 2003. (Cité page 121)

[SSRS06] T. Shlomi, D. Segal, E. Ruppin, and R. Sharan. QPath : a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, volume 7, article no 199, 2006. (Cité pages 10 et 127)

[SUS07] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, volume 3, article no. 88, 2007. (Cité pages 10 et 127)

[SXB07] R. Singh, J. Xu, and B. Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In Proc. *Research in Computational Molecular Biology (RECOMB 2007)*, volume 4453 of *LNCS*, pages 16–31, 2007. (Cité page 127)

[SXB08a] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks. In Proc. *Pacific Symposium on Biocomputing (PSB 2008)*, volume 13, pages 303–314, 2008. (Cité page 127)

[SXB08b] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 105(35) :12763–12768, 2008. (Cité pages 10, 35 et 127)

[TF09] W. Tian and N. F.Samatova. Pairwise alignment of interaction networks by fast identification of maximal conserved patterns. In Proc. *Pacific Symposium on Biocomputing (PSB 2009)*, pages 99–110, 2009. (Cité pages 10, 68 et 127)

[TGH09] F. Towfic, M.-H. Greenlee, and V. Honavar. Aligning biomolecular networks using modular graph kernels. In Proc. *Workshop on Algorithms in Bioinformatics (WABI 2009)*, volume 5724 of *LNCS*, pages 345–361, 2009. (Cité pages 5, 10 et 127)

[TGK04] N. Tanaka, S. Goto, and M. Kanehisa. An algorithm for graph isomorphism and its application to kegg compound search. In Proc. *Genome Informatics Workshop (GIW 2004)*, 2004. Poster Abstracts P061. (Cité page 127)

[Tov84] C. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8 :85–89, 1984. (Cité page 100)

[TSB+08] W. Timm, A. Scherbart, S. Böcker, O. Kohlbacher, and T. W. Nattkemper. Peak intensity prediction in MALDI-TOF mass spectrometry : a machine learning study to support quantitative proteomics. *BMC Bioinformatics*, volume 9, article no. 443, 2008. (Cité page 10)

[VM09] R. Vicentini and M. Menossi. *Data mining and knowledge discovery in real life applications*. In-tech, Julio Ponce and Adem Karahoca edition, 2009. (Cité page 38)

[VVCS05] V. Vlachos, V. Vouzi, D. Chatziantoniou, and D. Spinellis. NGCE - network graphs for computer epidemiologists. In Proc. *Panhellenic Conference on Informatic (PCI 2005)*, volume 3746 of *LNCS*, pages 672–683, 2005. (Cité page 58)

[WB04] E.J. Williams and D. J. Bowles. Coexpression of neighboring genes in the genome of Arabidopsis thaliana. *Genome Research*, 14 :1060–1067, 2004. (Cité page 36)

[WPM+06] H. Wei, S. Persson, T. Mehta, V. Srinivasasainagendra, L. Chen, G.P. Page, C. Somerville, and A. Loraine. Transcriptional coordination of the metabolic network in Arabidopsis thaliana. *Plant Physiology*, 18 :762–774, 2006. (Cité page 36)

[WR07] S. Wernicke and F. Rasche. Simple and fast alignment of metabolic pathways by exploiting local diversity. *Bioinformatics*, 23 :1978–1985, 2007. (Cité pages 11, 14 et 127)

[WYZ96] Y.Q. Wu, Y.C. Yuan, and Y.C. Zhao. Partition a graph into two induced forests. *Journal of Mathematical Study*, 29 :1–6, 1996. (Cité page 69)

[YIJ04] C-.H. Yeang, T. Ideker, and T. Jaakkola. Physical network models. *Journal of Computational Biology*, 11 :243–262, 2004. (Cité pages 15, 16, 32 et 85)

[YOB04] S. Yook, Z.N. Oltvai, and A.-L. Barabási. Functional and topological characterization of protein interaction networks. *Journal of Proteomics*, 4 :928–942, 2004. (Cité page 125)

[ZKW+05] L. Zhang, O. King, S. Wong, D. Goldberg, A. Tong, G. Lesage, B. Andrews, H. Bussey, C. Boone, and F. Roth. Motifs,

themes and thematic maps of an integrated Saccharomyces cerevisiae interaction network. *Journal of Biology*, volume 4, article no. 6, 2005. (Cité page 32)

[ZRBV09] M. Zaslavskiy, F. R. Bach, and J-.P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Bioinformatics*, 25 :259–1267, 2009. (Cité pages 10 et 127)

[ZSF+02] Y. Zheng, J.D. Szustakowski, L. Fortnow, R.J. Roberts, and S. Kasif. Computational identification of operons in microbial genomes. *Genome Research*, 12 :1221–1230, 2002. (Cité page 36)

[Zuc07] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1) :103–128, 2007. (Cité pages 78 et 83)

[ZZW+07] L. Zhenping, S. Zhang, Y. Wang, X.-S. Zhang, and L. Chen. Alignment of molecular networks by integer quadratic programming. *Bioinformatics*, 23 :1631–1639, 2007. (Cité pages 10, 13 et 127)

Ce document a été préparé en utilisant l'éditeur Kile et le modèle de thèse élaboré par Julien Chiquet (http ://stat.genopole.cnrs.fr/members/jchiquet/theselatex). La plupart des figures ont été effectuées à l'aide de l'outil TikZ (http ://sourceforge.net/projects/pgf/).

Titre Comparaison de réseaux biologiques

Résumé La comparaison de réseaux biologiques est actuellement l'une des approches les plus prometteuses pour aider à la compréhension du fonctionnement des organismes vivants. Elle apparaît comme la suite attendue de la comparaison de séquences biologiques dont l'étude ne représente en réalité que l'aspect génomique des informations manipulées par les biologistes. Dans cette thèse, nous proposons une approche innovante permettant de comparer deux réseaux biologiques modélisés respectivement par un graphe orienté D et un graphe non-orienté G, et dotés d'une fonction f établissant la correspondance entre les sommets des deux graphes. L'approche consiste à extraire automatiquement une structure dans D, biologiquement significative, dont les sommets induisent dans G, par f, une structure qui soit aussi biologiquement significative. Nous réalisons une étude algorithmique du problème issu de notre approche en commençant par sa version dans laquelle D est acyclique (DAG). Nous proposons des algorithmes polynomiaux pour certains cas, et nous montrons que d'autres cas sont algorithmiquement difficiles (NP-complets). Pour résoudre les instances difficiles, nous proposons une bonne heuristique et un algorithme exact basé sur la méthode branch-and-bound. Pour traiter le cas où D est cyclique, nous introduisons une méthode motivée par des hypothèses biologiques et consistant à décomposer D en DAGs tels que les sommets de chaque DAG induisent dans G un sous-graphe connexe. Nous étudions également dans cette thèse, l'inférence des voies de signalisation en combinant les informations sur les causes et sur les effets des événements extra-cellulaires. Nous modélisons ce problème par un problème d'orientation de graphes mixtes et nous effectuons une étude de complexité permettant d'identifier les instances faciles et celles difficiles.

Mots-clés Réseaux hétérogènes, Biologie computationnelle, NP-difficulté, APX-difficulté, Complexité paramétrée, Heuristiques, Branch-and-Bound.

Title Comparison of biological networks

Abstract The comparison of biological networks is now one of the most promising approaches that help in understanding the functioning of living organisms. It appears as the expected continuation of the comparison of biological sequences, whose study represents in reality only the genomic aspect of the information manipulated by biologists. In this thesis, we propose an innovative approach allowing to compare two biological networks modeled respectively by a directed graph D and an undirected graph G, and provided with a correspondence function f between the vertices of both graphs. The approach consists in extracting automatically a biologically significant structure in D whose vertices induce in G a biologically significant structure as well. We realize an algorithmic study of the problem arising in our approach by starting with its variant in which D is acyclic (DAG). We provide polynomial algorithms for several cases and we show that other cases are algorithmically difficult (NP-completes). In order to solve the difficult instances, we propose a reliable heuristic and an exact algorithm based on the branch-and-bound method. To deal with the case where D is cyclic, we introduce a method motivated by biological hypotheses and consisting in decomposing D into DAGs such that the vertices of each DAG induce in G a connected subgraph. We also study in this thesis, the problem of signaling pathways inference by combining the information on causes and effects of extra-cellular events. We model this problem by a problem of mixed graphs orientation and we perform a complexity study allowing to identify the easy and the difficult instances.

Keywords Heterogeneous networks, Computational biology, NP-hardness, APX-hardness, Parametrized complexity, Heuristics, Branch-and-Bound.