Complex graphs in biology : problems, algorithms and evaluation

Julien Fradin

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Julien FRADIN

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le sceau de l'Université Bretagne Loire

École doctorale : MathSTIC

Discipline : Informatique, section CNU 27
Unité de recherche : Laboratoire des Sciences du Numériques de Nantes (LS2N)

Soutenue le 4 décembre 2018

Graphes complexes en biologie : problèmes, algorithmes et évaluations

JURY

Président :	**M. Michel HABIB**, Professeur des universités, Université Paris Diderot
Rapporteurs :	**M^me^ Annie CHATEAU**, Maîtresse de conférences, Université de Montpellier 2 **M. Ioan TODINCA**, Professeur des universités, Université d'Orléans
Examinateurs :	**M^me^ Géraldine JEAN**, Maîtresse de conférences, Université de Nantes **M. Florian SIKORA**, Maître de conférences, Université Paris Dauphine
Invité :	**M. Damien EVEILLARD**, Maître de conférences, Université de Nantes
Directeur de thèse :	**M. Guillaume FERTIN**, Professeur des universités, Université de Nantes

Table des matières

Liste des tableaux

Liste des figures

Introduction

La biologie est une science des modèles. L'histoire de cette science montre que ces modèles sont associés à la capacité d'observation des systèmes vivants. Par exemple, l'essor de l'observation microscopique a abouti à considérer la cellule comme unité du vivant, tandis que les dernières avancées biotechnologiques proposent l'ADN comme composante fondamentale biologique. L'ADN, ou acide désoxyribonucléique, est une molécule qui contient le génome, c'est-à-dire toute l'information génétique, d'une espèce. En février 2018, la base de données GenBank contenait plus de deux cents millions de génomes [1]. Toutes ces données biologiques peuvent être modélisées sous la forme de séquences. Ces séquences sont ensuite étudiées dans le but de déterminer tous les gènes présents dans un organisme, c'est-à-dire toutes les régions du génome qui sont utilisées par une cellule pour synthétiser des constituants essentiels de l'organisme appelés des protéines. Toutefois, déterminer l'ensemble des gènes d'un organisme n'est pas suffisant pour comprendre le fonctionnement de celui-ci. Pour cela, il est en effet nécessaire d'étudier les interactions entre les différentes entités biologiques, comme les gènes ou les protéines, qui le composent [27, 63].

Ces réseaux d'interactions biologiques se modélisent intuitivement sous la forme de graphes, et non plus de séquences. Par exemple, dans le cadre des réseaux d'interaction protéine-protéine (appelés réseaux PPI, pour "Protein-Protein Interaction"), les sommets représentent des protéines et les arêtes des interactions physiques entre ces protéines. De même, les réseaux métaboliques sont utilisés pour représenter le métabolisme d'un organisme, c'est-à-dire l'ensemble des réactions chimiques effectuées au sein de cet organisme pour assurer son fonctionnement. Ces réseaux métaboliques sont constitués de métabolites, des molécules issues du métabolisme, ainsi que de ces réactions, qui prennent en entrée un ensemble de métabolites afin d'en produire un nouveau. Ainsi, dans le graphe des réactions d'un réseau métabolique, les sommets correspondent aux réactions, et deux sommets sont incidents s'il existe un métabolite commun parmi les métabolites de sortie de la réaction associée au premier sommet et les métabolites d'entrée de la réaction associée au deuxième sommet.

Les sommets des graphes représentant des réseaux biologiques peuvent également être colorés. Dans ce cas, la couleur d'un sommet donne une information supplémentaire sur la nature de l'entité biologique associée à celui-ci. Par exemple, si un graphe représente un réseau PPI, alors on peut colorer les sommets relativement à la fonction de la protéine qui leur est associée.

Alors que les modèles utilisés pour représenter des données biologiques ont évolué, on observe que plusieurs problématiques, et notamment la recherche de motifs, se modélisent de manière similaire dans les séquences et dans les graphes. Dans le cadre de la recherche de motifs, on cherche ainsi à déterminer si une séquence contient une sous-séquence d'intérêt, ou bien si un graphe contient un sous-graphe d'intérêt. En effet, déterminer si une séquence d'ADN contient une sous-séquence particulière permet par exemple d'identifier des zones promotrices de cette séquence et de détecter si ces zones comportent des anomalies [95]. On peut ainsi plus facilement développer des procédés biochimiques pour modifier le comportement des cellules dans l'espoir de corriger ces anomalies. Dans le contexte des réseaux PPI, un complexe protéique est un sous-ensemble de protéines participant à la même tâche au sein de d'une cellule. Déterminer si le réseau PPI d'un organisme méconnu contient un complexe protéique connu permet ainsi d'améliorer la connaissance sur cet organisme [37]. De même, il est intéressant de rechercher des voies

1. https://www.ncbi.nlm.nih.gov/genbank/statistics/

métaboliques, c'est-à-dire des ensembles de métabolites qui participent également à la même tâche au sein d'une cellule, dans les réseaux métaboliques [116].

D'un point de vue algorithmique, on peut lier la recherche de motifs dans ces graphes d'origine biologique à des problématiques d'isomorphisme de graphe. Dans ce cas, le motif – dit topologique – est un sous-graphe, et une occurrence du motif dans le graphe étudié est un sous-graphe isomorphe au motif. Cette définition d'un motif peut néanmoins poser plusieurs problèmes, comme par exemple lorsque la structure exacte du sous-graphe recherché n'est pas connue. De plus, les réseaux biologiques dont on dispose comportent beaucoup de bruit [38], c'est-à-dire qu'ils contiennent des relations entre entités erronnées et/ou manquantes.

Afin de répondre à ces problèmes, Lacroix *et al.* ont proposé une nouvelle définition d'un motif dans les graphes aux sommets colorés [79]. Dans leur modèle, puisque les réactions d'un réseau métabolique sont en grande majorité catalysées par des protéines (appelées des enzymes) qui ont une fonction précise dans l'organisme, la couleur de chaque sommet du graphe des réactions, associé à ce réseau métabolique, correspond à la fonction de l'enzyme (principale) qui catalyse la réaction associée à ce sommet. D'un point de vue biologique, Lacroix *et al.* recherchent ainsi un ensemble de réactions qui ont les caractéristiques d'une voie métabolique de part l'ensemble d'enzymes qui les catalyse, mais dont on ne sait pas comment elles interagissent entre elles. D'un point de vue algorithmique, le motif – dit fonctionnel – que l'on recherche dans ce graphe est alors un multi-ensemble de couleurs, et une occurrence du motif est un sous-graphe connexe dont le multi-ensemble de couleurs des sommets correspond à celui du motif. Contraindre l'occurrence du motif à être connexe garantit que toutes les réactions associées au motif appartiennent à la voie métabolique recherchée. La contrainte sur les couleurs des sommets d'une occurrence garantit quant à elle que la voie métabolique recherchée exécute la tâche désirée dans l'organisme.

Le problème GRAPH MOTIF, introduit par Lacroix *et al.* [79], consiste à déterminer s'il existe une occurrence d'un motif fonctionnel dans un graphe. Il existe maintenant une large littérature algorithmique pour résoudre le problème et ses nombreuses variantes, qui sont algorithmiquement difficiles. Ma thèse s'inscrit dans le cadre de la recherche de motifs dans les graphes aux sommets colorés : l'étude de GRAPH MOTIF et de ses variantes a donc été primordiale au début de celle-ci.

Alors que l'utilisation de modèles de graphes aux sommets colorés a été introduite dans le but d'étudier les interactions présentes dans des systèmes biologiques, on montre dans ce manuscrit que ces modèles peuvent également être appliqués dans d'autres domaines, et notamment en spectrométrie de masse. Le problème MAXIMUM COLORFUL ARBORESCENCE (MCA), auquel on a consacré l'essentiel de cette thèse, a ainsi pour but d'identifier la formule moléculaire d'un métabolite [18].

Pour cela, on fragmente dans un premier temps ce métabolite dans un spectromètre de masse afin d'obtenir un spectre, qui enregistre la masse de chaque fragment obtenu (et qui est ensuite potentiellement re-fragmenté) pendant le processus de fragmentation. Dans un deuxième temps, on construit une liste de formules moléculaires, dites candidates, pouvant correspondre à la formule moléculaire du métabolite d'entrée. Enfin, pour chacune de ces formules candidates, on construit et résout une instance de MCA afin de déterminer le scénario de fragmentation le plus probable du métabolite d'entrée, en supposant que la formule de celui-ci correspond à la formule candidate considérée.

Le graphe d'entrée d'une instance de MCA est orienté et enraciné : les sommets représentent des formules moléculaires et les arcs sont pondérés afin de représenter la plausibilité de la fragmentation d'une molécule dont la formule moléculaire correspond au sommet d'origine en une autre dont la formule moléculaire correspond au sommet d'arrivée. De plus, il existe une couleur pour chaque masse enregistrée dans le spectre ; les sommets sont ainsi colorés en fonction de la masse de la formule moléculaire qu'ils représentent. Dans cette instance, on recherche un sous-arbre (i) contenant la racine du graphe d'entrée, (ii) ne contenant pas deux sommets ou plus de même couleur et (iii) dont la somme du poids des arcs est maximum. D'un point de vue biologique, on rappelle que cette solution correspond au scénario de fragmentation le plus plausible d'une molécule dont la formule correspond à la formule candidate du métabolite d'entrée.

Ainsi, une solution de MCA doit contenir la racine du graphe d'entrée, qui représente la formule candidate considérée. De plus, une solution de MCA ne peut pas contenir deux sommets ou plus de même couleur, puisque chaque pic enregistre la masse d'une seule molécule. Enfin, plus le poids d'une solution de MCA est élevé, plus il est probable que la formule correspondant à la racine de cette solution soit celle du métabolite d'entrée.

Fondamentalement, la première partie de ce manuscrit propose un état de l'art des résultats algorithmiques liés au problème GRAPH MOTIF et à ses variantes. Pour cela, on donne tout d'abord dans le Chapitre 1 des notions de théorie des graphes et de théorie de la complexité qui sont nécessaires à la bonne compréhension de ce manuscrit. Le problème GRAPH MOTIF et ses variantes sont algorithmiquement difficiles [79, 51], ce qui signifie que les algorithmes utilisés pour les résoudre possèdent une complexité exponentielle. Ainsi, on présente également dans ce premier chapitre des techniques pour obtenir des algorithmes de complexité paramétrée (dits FPT), c'est-à-dire des algorithmes dont la partie exponentielle de la complexité porte sur un (ou plusieurs) paramètre(s) qu'on espère petit(s) en pratique [42, 89].

Dans le Chapitre 2, on modélise formellement le problème GRAPH MOTIF, ainsi que trois de ses variantes. Le problème COLORFUL GRAPH MOTIF est une variante de GRAPH MOTIF dans laquelle le motif – et donc la solution recherchée – ne contient pas de répétitions de couleurs. Le problème MAX GRAPH MOTIF cherche à déterminer la plus grande sous-occurrence d'un motif, c'est-à-dire à rechercher un sous-graphe connexe de cardinalité maximum et dont le multi-ensemble de couleurs des sommets est inclus dans le motif. Enfin, la dernière version étudiée dans ce manuscrit, appelée WEIGHTED GRAPH MOTIF, introduit une fonction de pondération sur les arêtes du graphe d'entrée : le but devient alors de chercher une occurrence du motif dont la somme des poids des arêtes est supérieure à une valeur fixée. Bien qu'il n'existe pas de motif explicite dans la modélisation de MCA, on observe que ce problème présente des similarités avec GRAPH MOTIF et ses variantes. En effet, tout comme pour le problème COLORFUL GRAPH MOTIF, une solution de MCA ne doit pas contenir de répétitions de couleurs. De plus, le multi-ensemble de couleurs d'une solution de MCA doit être inclus dans l'ensemble des couleurs du graphe, mais, tout comme pour le problème MAX GRAPH MOTIF, peut ne pas être égal à ce motif implicite. Enfin, tout comme pour le problème WEIGHTED GRAPH MOTIF, la modélisation de MCA inclut une fonction de pondération qui est cette fois-ci appliquée aux arcs, et non pas aux arêtes, du graphe d'entrée. On verra cependant dans ce manuscrit que la modélisation de MCA inclut une contrainte supplémentaire, d'origine biologique, et qui instaure une "hiérarchie" sur les couleurs d'une instance.

Dans la suite du Chapitre 2, on procède à un état de l'art de GRAPH MOTIF et de ses variantes. Puisque ces problèmes sont algorithmiquement difficiles, le but de cet état de l'art consiste à montrer comment contourner leur difficulté afin de pouvoir les résoudre dans des délais raisonnables. Par conséquent, on commence par tracer une frontière entre les instances pour lesquelles GRAPH MOTIF et ses variantes peuvent être résolus en temps polynomial et les instances pour lesquelles ces problèmes sont algorithmiquement difficiles. On montre ainsi que COLORFUL GRAPH MOTIF est NP-dur dans des classes de graphe très contraintes, comme par exemple dans les superstars ou les comb-graphs. On étudie ensuite la complexité de ces problèmes relativement à la taille du motif et au nombre de sommets qui n'appartiennent pas à une occurrence de celui-ci. On présente ainsi des résultats de complexité paramétrée pour chacun de ces deux paramètres. En particulier, on montre qu'un algorithme de complexité paramétrée relativement à la taille du motif est "optimal" sous certaines hypothèses de complexité.

Dans la deuxième partie de ce manuscrit, on procède à une étude algorithmique détaillée du problème MCA. Pour cela, on décrit tout d'abord dans le Chapitre 3 le contexte biologique qui a mené à la modélisation de ce problème, puis on montre que celui-ci est une modélisation plus juste d'un problème de la littérature appelé MAXIMUM COLORFUL SUBTREE (MCS) [18]. En particulier, on introduit la notion de graphe de hiérarchie de couleurs et on montre que ce graphe est en fait nécessairement un graphe orienté acyclique et enraciné. On initie ensuite une étude algorithmique de MCA, dans laquelle on commence par déterminer la frontière entre les instances faciles et difficiles de MCA. Dans les instances difficiles,

on présente également des algorithmes d'approximation, qui permettent de retourner en temps polynomial une solution dont l'écart à l'optimal est borné [6, 121]. Ainsi, on commence par observer que MCA est hautement inapproximable, même si le graphe d'entrée appartient à des classes d'arbres très contraints. En revanche, la notion du graphe de hiérarchie de couleurs introduite dans la définition de MCA nous permet d'obtenir plusieurs algorithmes d'approximation dans ces mêmes classes d'arbres contraints. De plus, on prouve que MCA se résout en temps polynomial si le graphe de hiérarchie de couleurs est un arbre.

On poursuit dans le Chapitre 4 l'étude algorithmique du problème MCA en montrant des résultats de complexité paramétrée pour ce problème. Puisque MCA peut se résoudre en temps polynomial quand le graphe de hiérarchie de couleurs est un arbre, on introduit deux nouveaux paramètres liés à la structure de ce graphe. On introduit également deux autres paramètres liés au nombre de sommets qui n'appartiennent pas à une solution de MCA. On détermine ensuite si MCA admet des algorithmes FPT, voir même des kernels polynomiaux, relativement à (des combinaisons de) ces nouveaux paramètres. En particulier, on obtient un algorithme FPT relativement à un sous-ensemble des couleurs d'une instance de MCA. Dans la dernière partie de ce chapitre, on implémente cet algorithme et on compare ses performances avec l'unique algorithme FPT disponible dans la littérature pour le problème MCS [18] sur des instances construites à partir de données biologiques. On montre ainsi que notre algorithme permet de résoudre plus d'instances et plus rapidement que l'algorithme issu de la littérature.

Pour finir, on récapitule les principaux résultats obtenus et on évoque les perspectives que cette thèse a engendrées.

1

Généralités algorithmiques

On présente dans ce chapitre tous les outils nécessaires à la bonne compréhension de cette thèse. On aborde tout d'abord le concept de graphes, puis on définit la notion de problèmes algorithmiques et d'algorithmes. On présente ensuite les classes de complexité utilisées dans ce manuscrit pour classer les problèmes étudiés dans ce manuscrit. Puisque ces problèmes sont réputés pour être "difficiles", on présente également plusieurs techniques majeures, qui seront utilisées dans ce manuscrit, pour résoudre de manière efficace ces problèmes.

1.1 Théorie des graphes

On décrit le concept de graphes ainsi que les propriétés basiques d'un graphe. On montre également différentes classes de graphes et on explique le concept de décomposition arborescente d'un graphe. Ceci nous servira à mieux appréhender la difficulté des problèmes rencontrés dans les chapitres suivants.

1.1.1 Définitions des propriétés d'un graphe

Les graphes sont des objets mathématiques utilisés pour représenter des interactions entre des entités. Leur première utilisation connue remonte en 1735 quand le mathématicien suisse Leonhard Euler a résolu le problème des sept ponts de Königsberg [49]. Dans ce problème, un fleuve sépare la ville de Königsberg (maintenant Kaliningrad) en quatre parties, qui sont reliées par sept ponts (voir la partie gauche de la Figure 1.1). Le but du problème est de déterminer s'il existe un chemin qui traverse chaque pont une et une seule fois, et dont la position d'arrivée est la même que celle de départ. Dans les paragraphes suivants, on utilise ce problème pour illustrer des notions clefs communément utilisées en théorie des graphes.

Un graphe $G = (V, E)$ est défini par un ensemble V d'entités, appelés des *sommets*, ainsi que par un ensemble E d'*arêtes*, qui relient des paires de ces sommets. On utilise également la notation $V(G)$ (resp. $E(G)$) pour décrire l'ensemble des sommets (resp. arêtes) de G, avec $n = |V(G)|$ et $m = |E(G)|$. Dans le graphe créé par Euler pour résoudre le problème des sept ponts de Königsberg, chaque partie de la ville est associée à un sommet et chaque pont entre deux parties de la ville est représenté par une arête entre les deux sommets correspondants à ces parties. On peut voir une illustration de ce graphe dans la Figure 1.1.

Une arête entre deux sommets u et v du graphe est notée (u, v). Dans la Figure 1.1, l'arête (A, C) représente le pont reliant la partie A de la ville à la partie C. On dit également que A et C sont les *extrémités* de l'arête (A, C) et que l'arête (A, C) est *incidente* aux sommets A et C. Par ailleurs, un graphe est dit

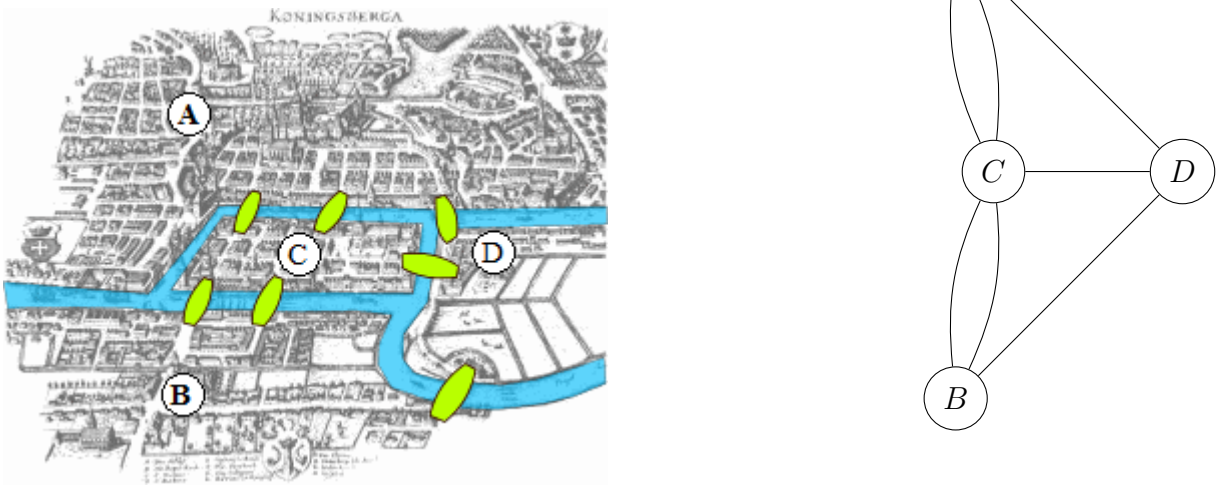


Figure 1.1 – Modélisation d'un graphe à partir du problème des ponts de Königsberg. La ville de la figure de gauche est découpée en quatre parties A, B, C et D qui sont représentées par des sommets dans la figure de droite. Chaque pont est symbolisé par une arête. Source Wikipedia.

simple s'il ne contient pas plusieurs arêtes entre une même paire de sommets et s'il ne contient aucune arête d'un sommet à lui-même.

Une *couverture par les sommets* d'un graphe $G = (V, E)$ est un ensemble de sommets $V' \subseteq V$ tel que chaque arête de E est incidente à au moins un sommet de V'. Par exemple, on observe que le sous-ensemble de sommets $\{C, D\}$ est une couverture par les sommets du graphe G dans la Figure 1.1.

On dit que deux sommets u et v appartenant à V sont *adjacents*, ou *voisins*, s'il existe une arête $(u, v) \in E$. Pour tout sommet $v \in V$, le voisinage de v, noté $N(v) = \{u \in V : (v, u) \in E\}$, est l'ensemble des voisins de v dans G. On note $d(v) = |N(v)|$ le *degré* d'un sommet v. S'il existe une arête reliant chaque paire de sommets de G, alors G est une *clique*.

Un ensemble de sommets $V' \subseteq V$ est *indépendant* s'il n'existe aucune paire de sommets v_1 et v_2 qui appartiennent à V' tels que v_1 et v_2 sont voisins dans G. Dans la Figure 1.1, l'ensemble de sommets $V' = A, B$ est indépendant.

Un chemin τ dans G contient une liste ordonnée de sommets de V tel qu'il existe une arête appartenant à E entre chaque paire de sommets consécutifs de τ. Un cycle est un chemin dont le sommet de départ est le même que celui d'arrivée. Enfin, un chemin (resp. cycle) est dit *simple* s'il ne contient pas deux fois la même arête. Ainsi, dans la Figure 1.1, $\tau_1 =< A, C, B >$ est un chemin, $\tau_2 =< A, D, B, C, D, A >$ est un cycle qui n'est pas simple car il contient deux fois l'arête (A, D), $\tau_3 =< A, C, D, A >$ est un cycle simple, et $\tau_4 =< A, B, C >$ n'est ni un cycle ni même un chemin. La longueur d'un chemin (resp. d'un cycle) est égale à son nombre d'arêtes.

Un graphe est *connexe* s'il existe un chemin entre chaque paire de sommets qu'il contient. Dans notre exemple, le graphe construit est connexe puisqu'on peut accéder à n'importe quelle partie de la ville en traversant un ou plusieurs ponts selon notre position d'origine.

Un graphe $G' = (V', E')$ est un sous-graphe de $G = (V, E)$ si $V' \subseteq V$, si $E' \subseteq E$ et si les extrémités de toutes les arêtes de E' appartiennent à V'. La relation $G' \subseteq G$ signifie que G' est un sous-graphe de G. Dans la Figure 1.1, le graphe $G' = (V', E')$ avec $V' = \{A, C, D\}$ et $E' = \{(A, D)\}$ est un sous-graphe de G qui n'est pas connexe.

Pour tout $V' \subseteq V$, $G[V']$ désigne le *sous-graphe induit* de G pour lequel $V(G[V']) = V'$ et $E(G[V']) = \{(u, v) \in E : (u \in V') \wedge (v \in V')\}$. En d'autres termes, $E(G[V'])$ contient toutes les arêtes de E dont les deux extrémités appartiennent à V'.

Une *composante connexe* de G est un sous-graphe induit et connexe $G' \subseteq G$ tel qu'il n'existe pas

d'autre sous-graphe induit et connexe $G'' \subseteq G$ avec $G' \subset G''$.

Un chemin (resp. cycle) est dit *hamiltonien* dans G s'il passe une et une seule fois par tous les *sommets* de G. De même, un chemin (resp. cycle) qui passe une et une seule fois par toutes les *arêtes* de G est dit *eulérien* en l'honneur du mathématicien éponyme pour sa contribution pour le problème des ponts de Königsberg. Euler a en effet montré qu'un graphe connexe G possède un tel cycle si et seulement si chacun des sommets de G est incident à un nombre pair d'arêtes. Intuitivement, il faut en effet une arête pour arriver à chaque sommet du graphe et une arête pour en repartir.

Soit $\mathrm{col} : V \to \{1, \ldots, k\}$ une *fonction de coloration* d'un graphe $G = (V, E)$. Pour tout ensemble $S \subseteq V$, $\mathrm{col}(S)$ désigne le multi-ensemble de couleurs assignées aux sommets de S et $\mathrm{col}^*(S)$ son ensemble de couleurs induit. On dit qu'un ensemble de sommets S est *colorful* si $\mathrm{col}(S) = \mathrm{col}^*(S)$, c'est-à-dire si S ne contient pas deux sommets de la même couleur. De la même manière, un graphe $G = (V, E)$ est colorful si V est colorful. Enfin, un chemin τ est colorful si l'ensemble des sommets de τ est colorful.

1.1.2 Exemples de classes de graphes

On verra dans les chapitres suivants qu'il est intéressant d'étudier un problème dans un graphe contraint quand ce problème est "dur" (voir Section 1.2.2).

Pour tout graphe $G = (V, E)$, deux sous-ensembles de sommets $V_1 \subseteq V$ et $V_2 \subseteq V$ sont dits *disjoints* si $V_1 \cap V_2 = \emptyset$, *i.e.* s'il n'existe aucun sommet en commun entre les deux sous-ensembles. Un graphe $G = (V, E)$ est *biparti* si on peut partitionner son ensemble de sommets V en deux sous-ensembles disjoints V_1 et V_2 tel qu'il n'existe aucune arête entre deux sommets de V_1, ni entre deux sommets de V_2.

Un *arbre* est un graphe connexe et sans cycles. En particulier, un *arbre couvrant* d'un graphe G est un sous-graphe de G qui est un arbre et dont l'ensemble de sommets est $V(G)$. La Figure 1.2 illustre la notion de graphe biparti et d'arbre. Dans cette figure, on remarque que les arbres sont une sous-classe de graphes bipartis.

On note $d(u, v)$ la *distance* d'un sommet $u \in V$ à un sommet $v \in V$, *i.e.* la longueur d'un plus court chemin qui relie u à v. Un sommet de degré 1 dans un arbre est appelé une *feuille*. On peut enraciner un arbre en choisissant arbitrairement un sommet $r \in V$ comme *racine*. Dans un arbre enraciné en r, pour toute paire de sommets $\{u, v\} \in V$ telle que $(u, v) \in E$ et telle que $d(r, u) < d(r, v)$, on dit que v est un *enfant* de u et que u est un *parent* de v. Enfin, la *hauteur* d'un arbre G enraciné en r est égale à la plus grande distance entre r et une des feuilles de G.

On définit quatre classes d'arbres supplémentaires qui seront utiles dans le reste de ce document. Un *comb-graph* est un arbre dans lequel le degré maximum de tout sommet est inférieur où égal à trois et dans lequel il existe au moins un chemin contenant tous les sommets de degré 3. La *spine* ("colonne vertébrale" en français) d'un comb-graph désigne un chemin arbitrairement choisi parmi ceux contenant tous les sommets de degré 3 dans le comb-graph. Une star (resp. *superstar*) est un arbre enraciné de hauteur au plus égale à 1 (resp. 2). Enfin, un *caterpillar* est un arbre qui devient un chemin si on lui supprime ses feuilles. On montre un exemple de chacun de ces graphes dans la figure 1.3.

1.1.3 Graphes orientés

Dans le problème des ponts de Königsberg, tous les ponts de la ville peuvent s'emprunter dans les deux sens. Le graphe de la Figure 1.1 qui représente cette situation est appelé un graphe non-orienté. Si un de ces ponts était à sens unique, on le modéliserait par un *arc* au lieu d'une arête. Un arc entre entre deux sommets u et v est également noté (u, v) mais on ne peut pas l'emprunter de v à u. Visuellement, on représente un arc (u, v) par une flèche de u vers v pour symboliser sa direction.

On définit un graphe orienté $G = (V, A)$ par un ensemble de sommets V et un ensemble d'arcs A. La *version non-orientée* de G est un graphe non-orienté $U(G) = (V, E)$ dans lequel chaque arc de type $(u, v) \in A$ est remplacé par une arête de type $(u, v) \in E$. On présente maintenant quelques notions similaires entre les graphes non-orientés et les graphes orientés.

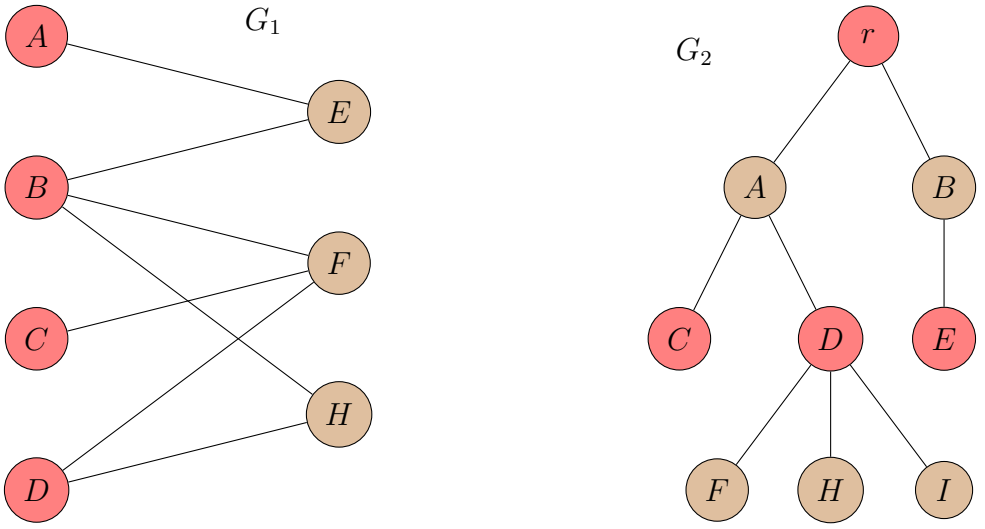


Figure 1.2 – Exemples de graphe biparti (G_1 à gauche) et d'arbre (G_2 à droite). Dans l'exemple de droite, le sommet r a arbitrairement été désigné comme la racine de G_2. Ainsi, le graphe G_2 enraciné en r a une hauteur de 3 puisque le chemin de longueur maximale entre r et une des feuilles du graphes (ici C, E, F, H ou I) est de longueur 3. On note que G_2 est également un graphe biparti puisqu'il n'existe aucune arête entre deux sommets rouges ni entre deux sommets noirs (comme dans G_1).

Un arc (u, v) est appelé un arc *entrant* de v et un arc *sortant* de u. L'ensemble des arcs entrants (resp. sortants) de v s'écrit $N^-(v)$ (resp. $N^+(v)$) et s'appelle le *voisinage entrant* (resp. *sortant*) de v. De plus, le *degré entrant* (resp. *degré sortant*) d'un sommet v est noté $d^-(v) = |N^-(v)|$ (resp. $d^+(v) = |N^+(v)|$).

Un chemin τ dans un graphe orienté $G = (V, A)$ contient une liste ordonnée de sommets de V tel qu'il existe un arc $(u, v) \in A$ pour chaque paire $\{u, v\}$ de sommets consécutifs de τ. La notion de *circuit* dans les graphes orientés est équivalente à la notion de cycle dans les graphes non-orientés. Ainsi, un circuit est un chemin dont le sommet de départ est le même que le sommet d'arrivée.

Un graphe orienté est connexe si la version non-orientée de ce graphe est connexe. Un graphe orienté, connexe et sans circuit est appelé un *Directed Acyclic Graph* (*DAG*). Un *ordre topologique* d'un graphe orienté $G = (V, A)$ est un ordre total sur l'ensemble des sommets de G, dans lequel $u \in V$ précède $v \in V$ s'il n'existe pas de chemin de v vers u dans G. Si G est un DAG, alors G admet nécessairement un ordre topologique de ses sommets puisque G ne contient pas de circuit (voir Figure 1.4).

Dans les graphes orientés, une racine est un sommet de degré entrant 0. On note qu'un DAG peut contenir plusieurs racines. En revanche, une *arborescence* est un DAG qui ne contient qu'une racine et dont le degré entrant maximum de tous les sommets est égal à 1. Ainsi, si un graphe orienté est une arborescence alors sa version non-orientée est un arbre. On illustre les différences entre un DAG et une arborescence dans la Figure 1.5.

Pour tout $V' \subseteq V$ et tout $v \in V'$, on note $G_v[V']$ le sous-graphe induit $G[V']$ enraciné en v et dans lequel on supprime tous les sommets $v' \in V'$ tel qu'il n'existe pas de chemin de v vers v'.

Par défaut, les graphes considérés dans le reste de ce chapitre ainsi que dans le Chapitre 2 sont non-orientés alors que ceux des autres chapitres sont orientés.

1.1.4 Décomposition arborescente d'un graphe

La *treewidth* est un paramètre d'un graphe qui indique à quel point un graphe "ressemble" à un arbre. On décrit formellement cette notion, qui sera très utilisée dans le Chapitre 4.

Définition 1. Une *décomposition arborescente* d'un graphe $G = (V, E)$ est une paire $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$, dans laquelle $\mathcal{T}$ est un arbre dont l'ensemble de sommets est I, et dans laquelle chaque X_i est un sous-ensemble de V appelé un *sac*. Les trois propriétés suivantes doivent être respectées :

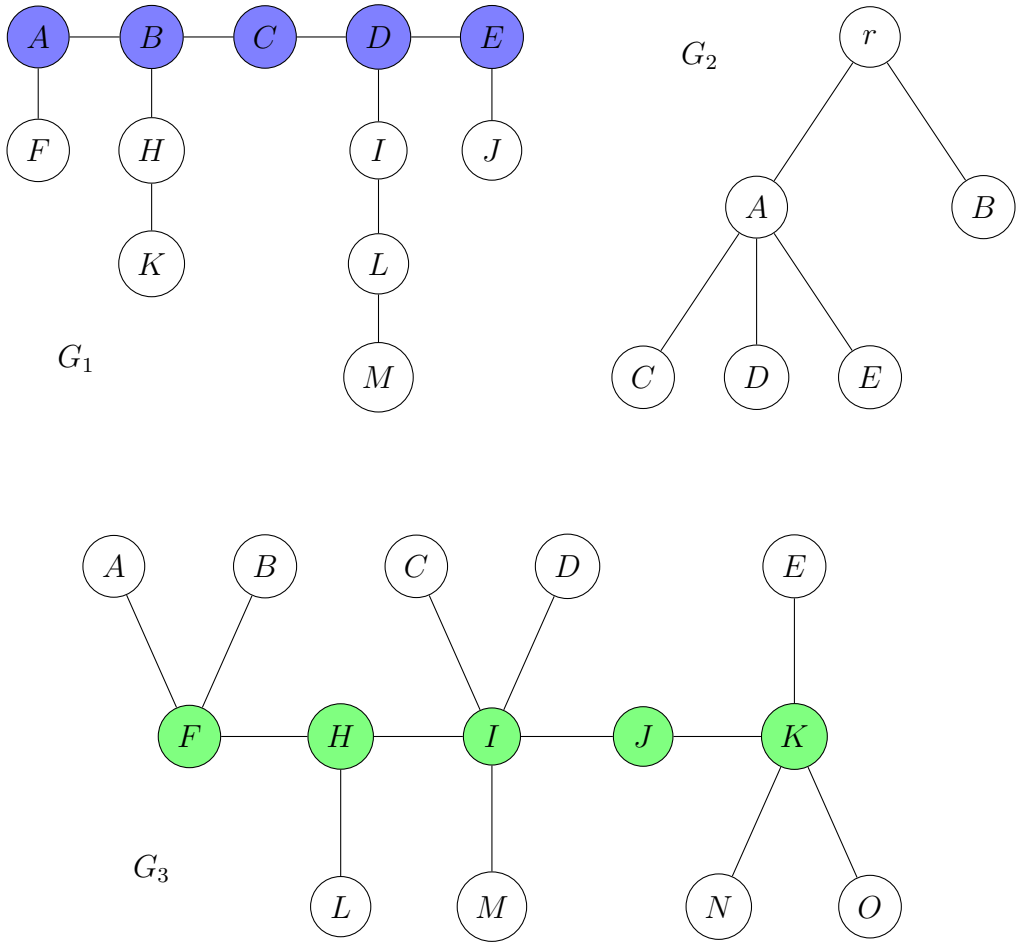


Figure 1.3 – Exemples de comb-graph (G_1 en haut à gauche), de superstar (G_2 en haut à droite) et de caterpillar (G_3 en bas). Les sommets de la *spine* de G_1 sont colorés en bleu. Les sommets colorés en vert dans G_3 correspondent aux sommets restants après suppression de toutes les feuilles de G_3. Dans cette figure, on note que $G_2[r, A, B]$ est une star.

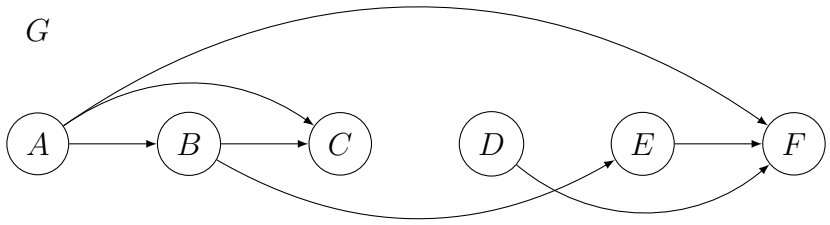


Figure 1.4 – Illustration d'un ordre topologique d'un DAG G. Les sommets de G sont ordonnés de la gauche vers la droite.

1. $\cup_{i \in I} X_i = V$,
2. Pour chaque arête $(u, v) \in E$, il existe un $i \in I$ tel que $\{u, v\} \subseteq X_i$,
3. Pour chaque triplet $i, j, k \in I$, si j appartient au chemin entre i et k dans $\mathcal{T}$, alors $X_i \cap X_k \subseteq X_j$.

La *width* de $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ est égale à $\max\{|X_i| : i \in I\} - 1$, et la *treewidth* de G est le plus petit entier k tel que G admet une décomposition arborescente de width k.

On définit maintenant le concept de *bonne* décomposition arborescente. Nous verrons dans le Chapitre 4 que cette décomposition arborescente particulière permet d'obtenir de nouveaux résultats pour le problème MAXIMUM COLORFUL ARBORESCENCE.

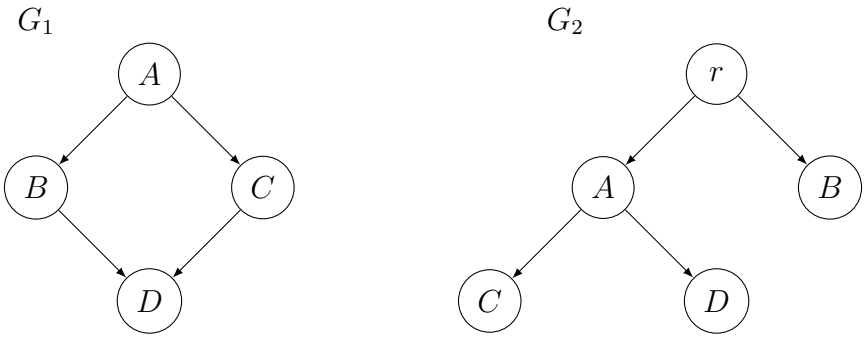


Figure 1.5 – Exemples de DAG (G_1 à gauche) et d'arborescence (G_2 à droite). On observe que le graphe G_2 est également un DAG puisqu'il est connexe et qu'il ne comporte aucun circuit. En revanche, G_1 n'est pas une arborescence puisque le degré entrant de D est égal à 2 dans G_1.

Définition 2. Une décomposition arborescente $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ enracinée en X_0 est dite *bonne* si les conditions suivantes sont satisfaites:

1. Chaque sommet de $\mathcal{T}$ a au plus deux enfants.
2. Si un sommet i a deux enfants j et k, alors $X_i = X_j = X_k$ et on appelle X_i un *nœud joint*.
3. Si un sommet i a un enfant j, on se trouve dans une des situations suivantes:
   a) $|X_i| = |X_j| + 1$ et $X_j \subset X_i$ et dans ce cas on appelle X_i un *nœud introductif*, ou
   b) $|X_i| = |X_j| - 1$ et $X_i \subset X_j$ et dans ce cas on appelle X_i un *nœud de suppression*.
4. Si un sommet i n'a pas d'enfant, alors $|X_i| = 1$ et dans ce cas on appelle X_i un *nœud feuille*.

On donne maintenant un exemple de décomposition arborescente et de bonne décomposition arborescente dans la Figure 1.6.

1.2 Théorie de la complexité

L'essentiel de cette thèse a été consacrée à l'étude des aspects algorithmiques de problèmes possédant des motivations biologiques. En particulier, on étudie le problème GRAPH MOTIF dans le Chapitre 2 afin de trouver des sous-réseaux d'intérêt dans des réseaux biologiques et on étudie dans les Chapitres 3 et 4 le problème MCA, qui a pour but d'identifier des petites molécules appelées métabolites. Dans cette section, on définit la notion de problème, puis on présente les principales classes de complexité qui permettent de caractériser la difficulté d'un problème.

1.2.1 Problèmes et algorithmes

On commence par définir les deux types de problèmes qui seront étudiés dans ce manuscrit : les problèmes de *décision* et les problèmes d'*optimisation*. On montre ensuite les relations entre ces deux types de problèmes, avant d'aborder les notions d'*algorithme* et surtout de *complexité* d'un algorithme.

On définit un problème de *décision* par des données d'entrées, dont l'ensemble constitue une *instance*, ainsi que par une question sur cette instance. Pour ce type de problèmes, la réponse à la question posée est soit "oui" (l'instance est appelée une YES-instance), soit "non" (l'instance est appelée une NO-instance). On donne un exemple de problème de décision avec le problème k-CYCLE ci-dessous.

k-CYCLE

- **Instance**: Un graphe $G = (V, E)$, un entier k
- **Question**: Existe-t-il un cycle simple C de longueur supérieure ou égale à k dans G ?

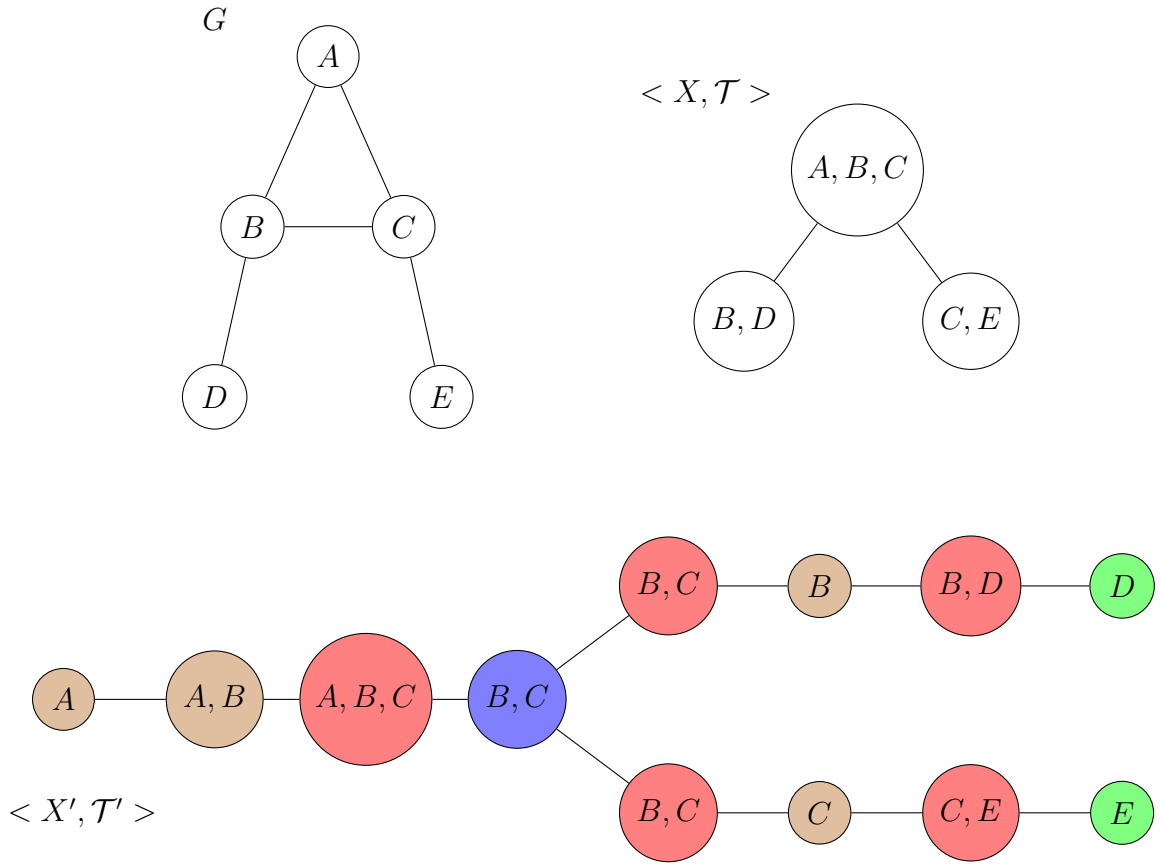


Figure 1.6 – Exemples de décompositions arborescentes. La paire $< X, \mathcal{T} >$ en haut à droite est une décomposition arborescente du graphe G (en haut à gauche) mais n'est pas une bonne décomposition arborescente puisqu'elle ne contient aucun nœud feuille. La paire $< X', \mathcal{T}' >$ est une *bonne* décomposition arborescente, dans laquelle on colorie les nœuds introductifs en rouge, les nœuds de suppression en marron, les nœuds joints en bleu et les nœuds feuilles en vert.

Un problème d'*optimisation* est défini par une instance et une solution à donner en sortie qui doit être la "meilleure" solution contenue dans l'instance par rapport à un critère de mesure. Cette mesure doit soit être maximisée (problème de maximisation) ou minimisée (problème de minimisation) ; les noms des problèmes étudiés dans ce manuscrit sont assez explicites pour ne pas avoir à préciser si un problème d'optimisation est un problème de maximisation ou de minimisation. On donne un exemple de problème d'optimisation avec le problème LONGEST CYCLE ci-dessous.

LONGEST CYCLE

- **Instance**: Un graphe $G = (V, E)$
- **Sortie**: Un cycle simple C de longueur $k \in \mathbb{N}$
- **Mesure**: k

On observe que les problèmes k-CYCLE et LONGEST CYCLE permettent tous les deux de répondre au problème des ponts de Königsberg qui est présenté dans la Section 1.1.1. En effet, un graphe $G = (V, E)$ contient un cycle eulérien si la réponse à k-CYCLE avec $k = |E|$ est "oui", ou bien si k est au moins égal à $|E|$ dans la solution donnée en sortie de LONGEST CYCLE. En règle générale, on peut facilement transformer un problème de décision (resp. d'optimisation) Q en problème d'optimisation Q' (resp. de décision) ; on appelle Q' la version optimisation (resp. décision) de Q. Dans ce manuscrit, tous les problèmes dont le nom commence par "k-" sont des problèmes de décision.

Un algorithme est une suite finie d'actions effectuées sur l'instance d'un problème dans le but de résoudre ce problème. Analyser la complexité d'un problème consiste à déterminer la quantité de ressources dont a besoin un algorithme pour le résoudre. Cette complexité doit être calculée indépendamment des machines ainsi que du langage de programmation utilisés pour exécuter l'algorithme afin de pouvoir comparer directement les algorithmes entre eux. Dans cette thèse, on distingue la complexité en *temps* de la complexité en *espace* d'un algorithme. Pour calculer la complexité en temps d'un algorithme, on compte les opérations élémentaires effectuées par celui-ci, comme le nombre d'affectations de variables et d'accès mémoire. La complexité en espace d'un algorithme est déterminée par la quantité de mémoire dont celui-ci a besoin pour fonctionner. Dans la suite, on dit que la *taille* d'un objet est la taille du meilleur codage informatique pour stocker en mémoire cet objet. Ainsi, la complexité en espace d'un algorithme est déterminée par la taille de l'instance et des variables utilisées par celui-ci pour résoudre un problème.

Par abus de langage, on dira que la complexité en temps (resp. en espace) d'un algorithme qui résout un problème Q est polynomiale si elle est polynomiale en la taille de l'instance de Q. De même, on dira que la complexité d'un algorithme est polynomiale si à la fois ses complexités en temps et en espace sont polynomiales.

En règle générale, on utilise les notations de Landau pour donner un ordre de grandeur sur la complexité d'un algorithme. Ainsi, si f et g sont deux fonctions telles que $f = \mathcal{O}(g)$, cela signifie qu'il existe deux constantes c et n_0 telles que $f(n) \leq c \cdot g(n)$ pour tout $n \geq n_0$. On utilise également les notation $\mathcal{O}^*()$ et $\tilde{\mathcal{O}}()$ pour respectivement masquer les facteurs polynomiaux et les facteurs polylogarithmiques. Ces trois notations ont l'avantage de constituer des bornes supérieures sur la complexité d'un algorithme. De plus, elles permettent de comparer deux algorithmes entre eux. Par exemple, un algorithme avec une complexité linéaire (en $\mathcal{O}(n)$) est à priori plus efficace pour résoudre un problème dans des instances de grande taille qu'un algorithme avec une complexité quadratique (en $\mathcal{O}(n^2)$). Dans la suite, s'il existe un algorithme avec une complexité en $\mathcal{O}(n)$ pour résoudre un problème Q, alors on dit que Q peut être résolu en $\mathcal{O}(n)$.

1.2.2 Les classes P et NP

Le but de cette sous-section est de présenter des classes de complexité qui permettent de distinguer les problèmes "faciles" des problèmes "difficiles". On considère ici uniquement les problèmes de décision puisque la théorie de la complexité ne considère pas les problèmes d'optimisation dans les classes de complexité présentées. On précise toutefois que la difficulté d'un problème de décision et la difficulté d'un problème d'optimisation sont les mêmes : il existe un algorithme de complexité polynomiale pour résoudre un problème de décision si et seulement il existe un algorithme de complexité polynomiale pour résoudre le problème d'optimisation correspondant.

La classe P contient tous les problèmes qui peuvent être résolus par un algorithme de complexité polynomiale. On note $Q \in \mathsf{P}$ pour indiquer qu'un problème Q appartient à P. Tous les problèmes appartenant à P sont considérés comme "faciles".

La classe NP contient tous les problèmes pour lequels on peut prouver en temps polynomial qu'une solution donnée est correcte. On observe que l'ensemble P est inclus dans NP, puisque vérifier qu'une solution est correcte est plus simple que de trouver une solution. Il existe néanmoins de nombreux problèmes appartenant à NP qu'on ne sait pas résoudre en temps polynomial. Par exemple, on peut montrer que le problème k-CYCLE, défini dans la Section 1.2.1, appartient à la classe NP. En effet, la taille d'une solution de k-CYCLE est polynomiale en la taille de l'instance – puisque la solution contient au plus $|V| = n$ sommets – et on peut vérifier en temps polynomial qu'il existe une arête entre chaque paire de sommets consécutifs de la solution. Cependant, il n'existe pas – à notre connaissance – d'algorithme de complexité polynomiale pour résoudre k-CYCLE. Plus généralement, une question fondamentale en théorie de la com-

plexité consiste à déterminer si $\mathsf{P} = \mathsf{NP}$.

On présente maintenant le concept de *réduction polynomiale*, qu'on utilise afin de montrer qu'un problème Q_2 est "au moins aussi dur" qu'un problème Q_1.

Définition 3. (Réduction polynomiale) Il existe une *réduction polynomiale* d'un problème de décision Q_1 vers un problème de décision Q_2 si, pour toute instance I_1 de Q_1, on peut construire en temps polynomial une instance I_2 de Q_2 telle que la réponse à Q_1 sur I_1 est la même que la réponse à Q_2 sur I_2.

Le concept de réduction polynomiale permet d'identifier les problèmes qui appartiennent à P. En effet, si un problème Q_2 appartient à P et qu'il existe une réduction polynomiale d'un problème Q_1 vers Q_2, alors on en conclut que $Q_1 \in \mathsf{P}$. On peut également utiliser les réductions polynomiales afin de déterminer les problèmes les plus durs de NP. Tout d'abord, on dit qu'un problème Q_2 est NP-dur (ou NP-difficile) si on peut effectuer une réduction polynomiale de n'importe quel problème $Q_1 \in \mathsf{NP}$ vers Q_2. De plus, si Q_2 est NP-dur et si Q_2 appartient à NP, alors on dit que Q_2 est NP-complet. Les problèmes NP-complets sont donc les problèmes les plus difficiles qui appartiennent à la classe NP – puisque les problèmes NP-durs n'appartiennent pas nécessairement à NP.

On montre dans la Figure 1.7 les relations entre les différentes classes présentées dans cette section selon que $\mathsf{P} = \mathsf{NP}$ ou $\mathsf{P} \neq \mathsf{NP}$. Les problèmes NP-complets présentent un avantage certain pour déterminer si $\mathsf{P} = \mathsf{NP}$: s'il existait *un* algorithme polynomial pour résoudre *un* problème Q NP-complet, tous les problèmes appartenant à NP pourraient alors être résolus en temps polynomial. A contrario, si $\mathsf{P} \neq \mathsf{NP}$ alors Q ne pourrait pas être résolu par un algorithme de complexité polynomiale et donc pourrait ne pas pouvoir être résolu dans des délais raisonnables si la taille de l'instance est trop élevée. A l'heure actuelle, l'essentiel des résultats en théorie de la complexité est obtenu sous la conjecture $\mathsf{P} \neq \mathsf{NP}$.

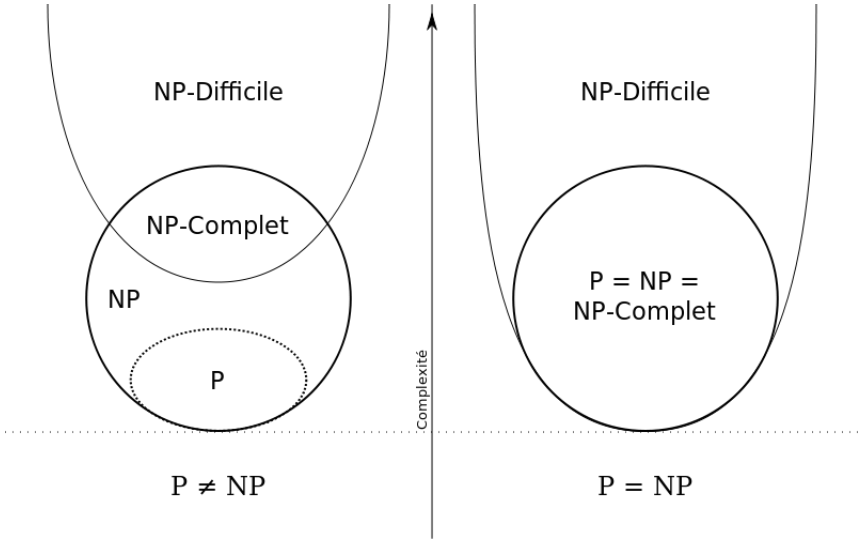


Figure 1.7 – Relations entre les différentes classes présentées dans la Section 1.2.2 selon si $\mathsf{P} \neq \mathsf{NP}$ ou si $\mathsf{P} = \mathsf{NP}$. Source Wikipedia.

Le problème BOOLEAN SATISFIABILITY (SAT) est un problème central en théorie de la complexité ; c'est notamment le premier problème à avoir été montré NP-complet [69]. Puisqu'on utilisera ce problème pour effectuer plusieurs réductions tout au long de ce manuscrit, on introduit ici quelques définitions pour pouvoir définir SAT. Soit $X = \{x_1, \ldots, x_p\}$ un ensemble de variables booléennes, *i.e.* dont la valeur appartient à $\{\texttt{vrai}, \texttt{faux}\}$. Un *littéral* est une variable x_i ou sa négation $\neg x_i$. Une *formule* ϕ est construite à partir des littéraux d'un ensemble X de variables booléennes et des connecteurs booléens ET et OU. Cette formule est satisfaite s'il existe une affectation des variables de X qui satisfait toutes les clauses de ϕ. Une *clause* est une disjonction (OU logique) de littéraux. La *taille* d'une clause est le nombre de littéraux

contenus dans cette clause. Une clause est *satisfaite* s'il existe une affectation de ses variables telle qu'au moins un de ses littéraux est fixé à `vrai`. Enfin, on dit qu'une formule ϕ est *CNF* (pour "Conjunctive Normal Form") si ϕ est une conjonction (ET logique) de clauses. On précise que toute formule ϕ peut se transformer en formule CNF ϕ' telle que la taille de ϕ' est linéaire en la taille de ϕ ; toutes les formules considérées dans cette thèse seront donc CNF par défaut.

BOOLEAN SATISFIABILITY (SAT)

• **Instance**: Un ensemble de variables booléennes $X = \{x_1, \ldots, x_p\}$, une formule ϕ sur un ensemble $C = \{C_1, \ldots, C_q\}$ de clauses contenant des variables de X

• **Question**: Existe-t-il une affectation $\beta : X \to \{\texttt{vrai}, \texttt{faux}\}$ de chaque $x_i \in X$ qui satisfait ϕ ?

Pour finir, il existe des hypothèses de complexité plus fortes que la conjecture $\mathsf{P} \neq \mathsf{NP}$. En particulier, l'hypothèse de complexité SETH ("Strong Exponential-Time Hypothesis") suppose que le problème SAT avec p variables ne peut pas être résolu en temps $\mathcal{O}^*((2-\epsilon)^p)$, pour tout $\epsilon > 0$ [67]. Si cette conjecture est vraie, alors on précise que SAT ne pourrait pas être résolu en temps polynomial, et donc que $\mathsf{P} \neq \mathsf{NP}$ puisque SAT appartient clairement à NP. De même, certains résultats obtenus dans ce manuscrit seront corrects sauf si $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$, où coNP/Poly est une classe de complexité dont je fais le choix de ne pas parler dans le manuscrit.

1.2.3 La classe APX

Tous les problèmes présentés dans cette sous-section sont des problèmes d'optimisation dont la version décision appartient à NP. Lorsqu'un problème de décision ne peut pas être résolu en temps polynomial sous la conjecture $\mathsf{P} \neq \mathsf{NP}$, on utilise dans cette section des algorithmes d'approximation afin de déterminer en temps *polynomial* une solution *approchée* de leur version optimisation. On définit tout d'abord la notion d'algorithme d'approximation de ratio r.

Définition 4. (Algorithme d'approximation de ratio r) Soit Q un problème d'optimisation dont la version décision appartient à NP. Pour toute instance I de Q, soit $OPT(I)$ la mesure d'une solution de Q pour I et soit $A(I)$ la mesure de la solution potentiellement non-optimale de Q qui est retournée en temps polynomial par un algorithme A dans I. Le problème Q admet un algorithme d'approximation de ratio $r \in \mathbb{R}^+$ – l'algorithme A – si, pour toute instance I de Q, on a :

$$\max\left\{\frac{A(I)}{OPT(I)}; \frac{OPT(I)}{A(I)}\right\} \leq OPT(I) \cdot r$$

Pour illustrer ces notions, on montre un exemple d'algorithme d'approximation pour une variante de SAT appelée MAX 2-SAT.

MAX 2-SAT

• **Instance**: Un ensemble de variables booléennes $X = \{x_1, \ldots, x_p\}$, une formule CNF ϕ sur un ensemble $C = \{C_1, \ldots, C_q\}$ de clauses de taille 2 contenant des variables de X

• **Sortie**: Une affectation $\beta : X \to \{\texttt{vrai}, \texttt{faux}\}$ de chaque $x_i \in X$ qui satisfait k clauses de ϕ

• **Mesure**: k

Proposition 5. *Le problème* MAX 2-SAT *peut être approximé en temps polynomial avec un ratio de 2.*

Preuve. Soit β une affectation arbitraire des variables de X, et $\overline{\beta}$ une autre affectation telle que toutes les variables affectées à `vrai` (resp. à `faux`) dans β sont affectées à `faux` (resp. à `vrai`) dans $\overline{\beta}$. Si β satisfait $k \leq q$ clauses de ϕ, alors on remarque que $\overline{\beta}$ en satisfait au moins $q - k$. Ainsi, choisir l'affectation qui satisfait le plus de clauses entre β et $\overline{\beta}$ garantit d'avoir une affectation qui satisfait *au moins* $\frac{q}{2}$ clauses de ϕ. Puisqu'une affectation ne peut pas satisfaire plus de q clauses, la solution donnée en sortie de notre algorithme satisfait *au pire* deux fois moins de clauses qu'une solution optimale. Ainsi, on vient de décrire un algorithme d'approximation dont le ratio d'approximation est *constant* et égal à 2. □

La classe APX contient tous les problèmes d'optimisation dont la version décision appartient à NP et qui admettent un algorithme d'approximation de ratio constant. Le problème MAX 2-SAT appartient donc à APX selon la Proposition 5. La classe PTAS contient tous les problèmes qui admettent un algorithme d'approximation (appelé algorithme PTAS) de ratio $(1+\epsilon)$, pour tout $\epsilon > 0$. Intuitivement, plus ϵ est proche de 0, plus la solution donnée en sortie est proche d'une solution optimale et plus le temps d'exécution de l'algorithme augmente. Clairement, tous les problèmes qui appartiennent à PTAS appartiennent également à APX.

On peut utiliser des réductions *linéaires* pour montrer qu'un problème d'optimisation appartient à APX. Ces réductions sont similaires aux réductions polynomiales, mais elles préservent également le ratio d'approximation [94]. En d'autres termes, s'il existe une réduction linéaire d'un problème Q_1 vers un problème Q_2 pour lequel il existe un algorithme d'approximation de ratio r_2, alors il existe un algorithme d'approximation de ratio $f(r_2)$ pour le problème Q_1, où f est une fonction *linéaire* de r_2. Ainsi, si un problème Q_1 admet une réduction linéaire vers un problème Q_2 et que Q_2 appartient à APX, alors Q_1 appartient également à APX.

On peut également utiliser les réductions linéaires afin de déterminer les problèmes les plus durs de APX. Ainsi, un problème Q est APX-dur s'il existe une réduction linéaire de n'importe quel problème de APX à Q. De plus, Q est APX-complet si Q appartient à APX et si Q est APX-dur. De manière similaire aux problèmes NP-complets qui sont les problèmes les plus durs de NP, les problèmes APX-complets sont les problèmes les plus durs de APX. Par ailleurs, si P $\neq$ NP, alors PTAS $\neq$ APX et aucun problème APX-complet n'admet d'algorithme PTAS.

Pour finir, on peut utiliser les réductions linéaires afin de borner les ratios d'approximation des algorithmes d'approximation permettant de résoudre un problème. En effet, s'il existe une réduction linéaire d'un problème Q_1 vers un problème Q_2 et que Q_1 n'admet pas d'algorithme d'approximation avec un ratio d'approximation inférieur à r, alors Q_2 n'admet pas d'algorithme d'approximation avec un ratio d'approximation inférieur à une fonction linéaire de r qui se calcule à partir de la réduction linéaire.

1.2.4 Les classes FPT, W[1], W[2] et XP

Dans cette section, de manière similaire à la Section 1.2.2, tous les problèmes considérés sont des problèmes de décision – puisque la théorie de la complexité ne considère pas les problèmes d'optimisation dans les classes de complexité présentées. On a vu dans la Section 1.2.2 qu'un problème NP-dur ne peut pas être résolu par un algorithme de complexité polynomiale si P $\neq$ NP. La complexité d'un algorithme utilisé pour résoudre un problème NP-dur est donc exponentielle, et le problème peut ne pas pouvoir être résolu avec des temps d'exécution raisonnables si la taille de l'instance est trop élevée.

Le domaine de la *complexité paramétrée* a pour but de confiner la partie exponentielle de la complexité d'un algorithme à un paramètre k qu'on espère petit en pratique. Soit (I, k) une instance paramétrée d'un problème Q paramétré par k. Le problème Q appartient à la classe XP si Q peut être résolu en $\mathcal{O}^*(|I|^{f(k)})$, où f est une fonction calculable qui dépend uniquement de k ; tous les problèmes étudiés dans cette thèse appartiennent à XP. Maintenant, on dit que le problème Q appartient à la classe FPT (pour

"Fixed-Parameter Tractable") relativement à k si Q peut être résolu en temps $\mathcal{O}(f(k) \cdot \text{poly}(|I|))$, où f est une fonction calculable qui dépend uniquement de k. En d'autres termes, si Q appartient à FPT relativement à k, alors Q peut être résolu par un algorithme dont la complexité est uniquement exponentielle par rapport au paramètre k ; cet algorithme est dit FPT relativement à k dans la suite.

On verra différentes techniques pour développer des algorithmes de complexité paramétrée dans la Section 1.3. Cependant, on peut également montrer qu'un problème n'est pas dans FPT relativement à un paramètre k si P $\neq$ NP. On donne deux exemples avec les problèmes k-COLORATION et k-INDEPENDENT SET définis ci-dessous.

k-COLORATION

• **Instance**: Un graphe $G = (V, E)$, un entier k

• **Question**: Existe-t-il une fonction de coloration $\text{col} : V \to \{1, \ldots, k\}$ tel que $\text{col}(u) \neq \text{col}(v)$ pour toute arête $(u, v) \in E$?

k-INDEPENDENT SET

• **Instance**: Un graphe $G = (V, E)$, un entier k

• **Question**: Existe-t-il un ensemble indépendant $V' \in V$ tel que $|V'| = k$?

Le problème k-COLORATION appartient à P quand $k = 2$: il suffit en effet de déterminer si G est biparti. Cependant, k-COLORATION est NP-dur même quand $k = 3$ [33]. D'après la définition donnée ci-dessus, si un problème Q appartient à FPT relativement à k, alors celui-ci peut être résolu en temps polynomial pour chaque valeur constante de k. Par conséquent, k-COLORATION n'appartient pas à FPT par rapport à k – sauf si P = NP – puisque 3-COLORATION est NP-dur. A contrario, k-INDEPENDENT SET se résout en temps polynomial pour chaque valeur constante de k et pourtant le problème n'appartient pas à FPT relativement à k [60]. Montrer qu'un problème Q paramétré par k appartient à P pour chaque valeur constante de k n'est donc pas suffisant pour prouver que Q appartient à FPT relativement à k. Pour cela, on peut directement exhiber un algorithme FPT relativement à k pour Q, ou bien réaliser une *réduction paramétrée* d'un problème Q_2 qui appartient à FPT relativement à k_2 vers le problème Q paramétré par k. On introduit le concept de réduction paramétrée ci-dessous.

Définition 6. (Réduction paramétrée [89]) Il existe une réduction paramétrée d'un problème Q_1 paramétré par k_1 vers un problème Q_2 paramétré par k_2 si pour chaque instance (I_1, k_1) de Q_1 on peut construire une instance (I_2, k_2) de Q_2 telle que :

1. la réponse à Q_1 sur (I_1, k_1) est la même que la réponse à Q_2 sur (I_2, k_2),
2. $k_2 = g(k_1)$, où g est une fonction calculable qui ne dépend que de k_1,
3. la construction de (I_2, k_2) est réalisée en temps $f(k_1).|I_1|^c$, où c est une constante et f est une fonction calculable qui ne dépend que de k_1.

Concrètement, le paramètre k_2 ainsi que la complexité de la construction de (I_2, k_2) peuvent uniquement être exponentiels en k_1. Ainsi, si Q_2 est FPT relativement à k_2, alors Q_1 est FPT relativement à k_1.

Les réductions paramétrées peuvent également être utilisées afin de déterminer que des problèmes paramétrés n'appartiennent pas à FPT si P $\neq$ NP. Pour cela, on utilise deux nouvelles classes de complexité W[1] et W[2] telles que FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq$ XP. Un problème paramétré Q est W[1]-dur (resp. W[2]-dur) si tous les problèmes paramétrés de W[1] (resp. W[2]) admettent une réduction paramétrée vers Q. Les problèmes paramétrés W[1]-durs (resp. W[2]-durs) sont les problèmes paramétrés les plus durs de W[1] (resp. W[2]) : si P $\neq$ NP, alors FPT $\neq$ W[1] et les problèmes paramétrés W[1]-durs n'admettent pas d'algorithme FPT – le raisonnement est similaire pour les problèmes paramétrés W[2]-durs. Ainsi, pour montrer qu'un problème paramétré Q n'est vraisemblablement pas dans FPT, on montrera des réductions de problèmes paramétrés W[1]-durs (ou W[2]-durs) vers Q.

1.3 Quelques techniques pour obtenir des algorithmes FPT

On présente ici plusieurs techniques utilisées dans les chapitres suivants pour obtenir des algorithmes de complexité paramétrée.

1.3.1 Algorithme de branchement

Un algorithme de branchement effectue une recherche systématique exhaustive sur une instance I d'un problème Q. Cette recherche systématique est organisée à l'aide d'un arbre de recherche T_S : l'instance I constitue la racine de T_S et tous les autres nœuds de T_S sont obtenus en appliquant récursivement des *règles de branchement*, qui permettent de restreindre la taille de l'instance I en découpant l'espace des solutions de Q dans I.

Si Q est un problème paramétré par k, le but est de borner relativement à k le nombre final de nœuds contenus dans l'arbre de recherche. Ainsi, si chaque étape de la recherche systématique exhaustive est réalisée en temps polynomial, alors l'algorithme de branchement créé est FPT relativement à k. Déterminer des règles de branchement efficaces est essentiel afin de minimiser le nombre final de nœuds contenus dans l'arbre. On montre un exemple d'algorithme de branchement avec le problème k-VERTEX COVER défini ci-dessous.

k-VERTEX COVER

- **Instance**: Un graphe $G = (V, E)$, un entier k
- **Question**: Existe-t-il une couverture par les sommets $V' \subseteq V$ de G telle que $|V'| \leq k$?

Proposition 7. *k-VERTEX COVER peut être résolu en $\mathcal{O}^*(2^k)$ en utilisant un arbre de recherche.*

Preuve. On construit un algorithme de branchement basé sur les arêtes de G. Pour cela, on initialise un ensemble de sommets $S = \emptyset$ qui représente une solution partielle potentielle de k-VERTEX COVER. S'il existe une arête (u, v) dans G telle que (i) u et v n'appartiennent pas à S et (ii) $|S| < k$, alors on branche récursivement sur deux cas : on ajoute soit u à S, soit v à S. Dans chacun des deux cas, on note que la taille de l'instance est réduite puisqu'on ne considère plus les arêtes incidentes à u (resp. v) si on ajoute u (resp. v) à S dans le reste de l'algorithme. Pour tout ensemble S correspondant à une feuille de l'arbre de recherche induit T_S, on peut déterminer en temps polynomial si S est une couverture par les sommets de G. Si c'est le cas, alors on fixe $V' = S$, et V' est clairement une solution de k-VERTEX COVER puisque $|S| \leq k$.

Chaque application de la règle de branchement est réalisée en temps polynomial. Ainsi, la complexité de l'algorithme est exponentielle uniquement par rapport au nombre de nœuds de T_S. Comme T_S est un arbre binaire de hauteur inférieure ou égale à k, son nombre de nœuds ne peut pas excéder 2^k et en conséquence la complexité en temps de notre algorithme est en $\mathcal{O}^*(2^k)$. □

En règle générale, on utilise la notion de *vecteur de branchement* pour calculer le nombre de sommets de T_S. Soit Q un problème et I une instance de Q de taille n. Si une règle de branchement crée récursivement t instances de taille $n - a_1, n - a_2, \ldots n - a_t$ à partir de I, alors on note $(a_1, a_2, \ldots a_t)$ le vecteur de branchement de T_S. Si chaque nœud de T_S a au moins deux enfants, alors T_S contient $\mathcal{O}(\alpha^k)$ sommets, où α est égal à la racine de valeur absolue maximum du polynôme

$$z^a = z^{a-a_1} + z^{a-a_2} + \ldots + z^{a-a_t},$$

dans lequel $a = \max\{a_i \in \{a_1, \ldots, a_t\}\}$. Dans notre arbre de recherche utilisé pour résoudre k-VERTEX COVER, on obtient le vecteur de branchement $(1, 1)$. Le polynôme $z^1 = z^0 + z^0$ admet une racine $\alpha = 2$, et on retrouve donc la complexité en $\mathcal{O}^*(2^k)$ calculée précédemment.

1.3.2 Programmation dynamique

Le concept de la programmation dynamique consiste à résoudre un problème principal en résolvant récursivement des problèmes intermédiaires plus petits. Pour cela, on utilise une (ou plusieurs) *table(s)* de programmation dynamique qui stocke(nt) les solutions de ces problèmes intermédiaires.

On montre une application de cette technique provenant de [2] sur le problème COLORFUL k-PATH défini ci-dessous.

COLORFUL k-PATH

• **Instance**: Un graphe $G = (V, E)$, un entier k, une fonction de coloration $\mathrm{col} : V \to \{1, \ldots, k+1\}$

• **Question**: Existe-t-il un chemin simple et colorful de longueur k dans G ?

Théorème 8. *([2])* COLORFUL k-PATH *peut être résolu en* $\mathcal{O}^*(2^k)$.

Preuve. On crée une table de programmation dynamique $A[v, i]$ pour tout $v \in V$ et tout $i \in \{0, \ldots, k+1\}$. Une case $A[v, i]$ contient un ensemble de couleurs $\mathrm{col}(\tau)$ par chemin τ qui est simple, colorful, de longueur i et dont le sommet d'arrivée est v. L'algorithme proposé consiste donc à remplir la table A afin de regarder s'il existe un sommet $v \in V$ tel que $A[v, k+1] \neq \emptyset$.

Pour remplir la table A, on commence par initialiser $A[v, 0] = \{\{\mathrm{col}(v)\}\}$ pour tous les sommets $v \in V$. En effet, pour tout $v \in V$, $\tau = (\{v\}, \emptyset)$ est le seul chemin de longueur 0 dont le sommet d'arrivée est v. Ensuite, pour tout $v \in V$ et $i \in \{1, \ldots, k+1\}$, s'il existe un chemin τ de longueur $i-1$ et un sommet $u \in N(v)$ tels que (i) $\mathrm{col}(\tau) \in A[u, i-1]$ et (ii) $\mathrm{col}(v) \notin \mathrm{col}(\tau)$, alors on ajoute l'ensemble $\mathrm{col}(\tau) \cup \{\mathrm{col}(v)\}$ dans $A[v, i]$.

On calcule maintenant la complexité en temps de l'algorithme proposé ci-dessus. Premièrement, on remarque que chaque case de type $A[v, i]$ contient au plus $\binom{k}{i}$ ensembles de couleurs. Deuxièmement, déterminer si un élément est contenu dans un ensemble de taille i s'effectue en $\mathcal{O}(i)$. Enfin, lors du calcul du contenu de la case $A[v, i]$, on rappelle que l'algorithme analyse le contenu de chaque case $A[u, i-1]$ telle que $u \in N(v)$. Ainsi, l'algorithme possède une complexité temporelle de $\mathcal{O}(\sum_{i=0}^{k} i\binom{k}{i}|E|)$. On rappelle la formule de Newton $(x+y)^k = \sum_{l=0}^{k} \binom{k}{l} x^{k-l} y^l$. Par conséquent, en substituant $x = y = 1$, la complexité en temps de l'algorithme est en $\mathcal{O}(2^k k|E|)$. □

1.3.3 Le color-coding et ses améliorations

Le color-coding est une technique créée par Alon *et al.* [2] pour obtenir des algorithmes de complexité paramétrée. Le principe consiste à "colorier" arbitrairement une instance I d'un problème Q afin d'ajouter une contrainte supplémentaire (les couleurs) sur les solutions de Q dans I. L'espace des solutions de la version plus contrainte de Q dans une coloration de I est nécessairement plus restreint que l'espace des solutions initial de Q dans I, ce qui améliore l'efficacité des algorithmes exhaustifs. En contrepartie, il est important de résoudre la version contrainte de Q dans chaque coloration de I, puisqu'une solution initiale de Q dans I peut ne plus être correcte pour la version contrainte de Q dans une coloration donnée de I. La technique du color-coding est particulièrement efficace pour résoudre des problèmes d'isomorphisme de graphe. En particulier, on montre pourquoi et comment cette technique peut être appliquée pour résoudre le problème k-PATH, que nous définissons ci-dessous.

k-PATH

• **Instance**: Un graphe $G = (V, E)$, un entier k

• **Question**: Existe-t-il un chemin simple de longueur k dans G ?

Le problème k-PATH est très similaire au problème COLORFUL k-PATH introduit dans la Section 1.3.2. On propose donc dans un premier temps de créer un algorithme de programmation dynamique qui est également similaire à celui utilisé pour résoudre COLORFUL k-PATH. Pour cela, on crée une table $B[v, i]$ pour chaque sommet $v \in V$ et $i \in \{0, \ldots, k\}$. Cette fois, une case $B[v, i]$ contient tous les chemins simples de longueur i et dont le sommet d'arrivée est v. On insiste sur le fait qu'une case $B[v, i]$ contient des chemins alors qu'une case $A[v, i]$ dans l'algorithme de la Section 1.3.2 contient les ensembles de couleurs de ces chemins. Par conséquent, chaque case $B[v, i]$ peut contenir jusqu'à $\binom{n}{i} = \mathcal{O}(n^i)$ chemins et un algorithme qui remplit la table B pour résoudre k-PATH n'est pas FPT relativement à k puisqu'il possède une complexité temporelle exponentielle en le nombre de sommets n de G.

Afin de mieux comprendre le principe du color-coding, on rappelle qu'il suffit qu'il existe *une* case $B[v, k]$ qui contienne *un* chemin pour que (G, k) soit une YES-instance de k-PATH. Le principe d'un algorithme utilisant le color-coding consiste donc à ne *pas* calculer les potentielles $\binom{n}{i}$ solutions pour chaque case de type $B[v, i]$ afin que cet algorithme ne possède *pas* une complexité exponentielle en n. Pour cela, on va modifier l'instance de départ de k-PATH afin de pouvoir appliquer l'algorithme FPT présenté dans la Section 1.3.2.

Le principe du color-coding consiste ainsi à appliquer une fonction de coloration aléatoire $\mathrm{col} : V \to \{1, \ldots, k + 1\}$ et à chercher un chemin de longueur k qui est *colorful* dans G. Toutefois, si un chemin simple τ de longueur k existe dans G, alors il existe seulement $k + 1!$ colorations possibles de τ tels que τ est colorful parmi $(k + 1)^{k+1}$ colorations possibles de τ au total. Ainsi, τ a une probabilité $\frac{k+1!}{(k+1)^{k+1}} \sim \frac{1}{e^{k+1}}$ d'être colorful après l'application d'une fonction de coloration arbitraire sur G. Pour trouver τ dans G avec une probabilité constante de 1, on doit donc créer et résoudre e^{k+1} instances de COLORFUL k-PATH correspondant à e^{k+1} colorations aléatoires de I. Puisque chaque instance de COLORFUL k-PATH peut être résolu en temps $\mathcal{O}^*(2^k)$ selon le Théorème 8, on en déduit le théorème suivant :

Théorème 9. *([2])* k-PATH *peut être résolu en* $\mathcal{O}^*((2e)^k)$ *avec une probabilité constante de 1.*

L'algorithme proposé ci-dessus fait intervenir de l'aléatoire ; il est dit *probabiliste* – au contraire des algorithmes dits *déterministes* qui renvoient systématiquement la même réponse. Puisque l'algorithme décrit ci-dessus est probabiliste, on note qu'il existe des cas dans lesquels tous les chemins simples de (G, k) possèdent des répétitions de couleur après chacune des e^k fonctions de coloration appliquées sur (G, k). Si l'algorithme obtenu ne renvoie pas de faux-positifs, il peut donc en revanche renvoyer des faux-négatifs.

On montre maintenant comment *dérandomiser* les algorithmes de color-coding, *i.e.* comment les transformer en algorithmes déterministes. Pour cela, soit $\mathcal{F}$ une liste de fonctions de coloration du graphe G. Pour tout sous-ensemble de sommets $V' \subseteq V$ de taille $|V'| = k$, s'il existe une fonction de coloration col appartenant à $\mathcal{F}$ telle que $\mathrm{col}(V')$ est colorful, alors $\mathcal{F}$ est une *famille parfaite de hachage* [2]. Ainsi, appliquer l'algorithme présenté dans le Théorème 8 pour chaque fonction de coloration de $\mathcal{F}$ sur G nous assure de trouver un chemin simple de longueur k dans G si (G, k) est une YES-instance de k-PATH. Alon *et al.* ont montré qu'on peut obtenir une famille parfaite de hachage $\mathcal{F}$ pour colorer un graphe avec k couleurs telle que $|\mathcal{F}| = \mathcal{O}^*(2^{\mathcal{O}(k)})$ [2].

On montre maintenant différentes améliorations du color-coding en utilisant le problème k-PATH comme fil conducteur. Comme pour le color-coding, on verra que chacune de ces améliorations a également pour but de ne pas calculer l'ensemble des solutions d'un problème.

Divide-and-Color. La technique divide-and-color utilise à la fois le color-coding et la technique divide-and-conquer. Intuitivement, la technique divide-and-conquer divise une instance en plusieurs parties, résout les sous-instances indépendamment et combine les solutions des sous-instances pour trouver une solution

de l'instance initiale.

On illustre la technique divide-and-color sur le problème k-PATH avec l'Algorithme 1 ci-dessous, qui provient d'un article de Kneis *et al.* [72]. Si G contient plus d'un sommet, on partitionne V en deux sous-ensembles disjoints V' et $V \setminus V'$ pour séparer l'instance en deux parties à la ligne (5). Pour faire le lien avec le color-coding, cela revient à appliquer une fonction de coloration aléatoire $\mathrm{col} : V \to \{1, 2\}$ et à définir $V' = \{v \in V : \mathrm{col}(v) = 1\}$. On applique récursivement l'algorithme sur chaque sous-instance jusqu'à obtenir des sous-instances ne contenant qu'un sommet. La ligne (8) permet ensuite de relier deux sous-chemins de deux instances disjointes. Si τ est un chemin de longueur k, la valeur $3 \cdot 2^k$ est calculée pour que la probabilité de trouver τ avec l'algorithme soit au moins égale à 0.75. Kneis *et al.* montrent ensuite que l'algorithme effectue au plus $\mathcal{O}^*(4^k)$ appels récursifs. Puisque toutes les opérations en-dehors des appels s'effectuent en temps polynomial, la complexité en temps de l'algorithme probabiliste est de $\mathcal{O}^*(4^k)$ [72, 28]. On peut également dérandomiser l'algorithme avec une famille parfaite de hachage pour obtenir un algorithme déterministe de complexité $\mathcal{O}^*(16^k)$ [72].

Algorithme 1 DIVIDE-AND-COLOR ([72])

Entrée: Un graphe $G = (V, E)$, un entier k.
Sortie: Une collection de paires de sommets R tel qu'il existe un chemin de longueur k entre u et v pour tout $\{u, v\} \in R$.

1: **si** $(k = 1)$ **alors**
2: **retourne** $\{\{v, v\} : v \in V\}$;
3: **sinon**
4: **pour tout** $(i \in \{1, \ldots, 3 \cdot 2^k\})$ **faire**
5: Choisir aléatoirement et uniformément $V' \subset V$;
6: $G_1 = G[V']$; $G_2 = G[V \setminus V']$; $R = \emptyset$;
7: **pour tout** $(\{u, v, x, y\} \in V)$ **faire**
8: **si** $(\{u, v\} \in$ DIVIDE-AND-COLOR$(G_1, \lceil \frac{k}{2} \rceil)) \wedge ((v, x) \in E) \wedge (\{x, y\} \in$ DIVIDE-AND-COLOR$(G_2, \lfloor \frac{k}{2} \rfloor))$ **alors**
9: $R = R \cup \{u, y\}$;
10: **fin si**
11: **fin pour**
12: **fin pour**
13: **fin si**
14: **retourne** R;

Détection de monômes multilinéaires. On montre dans ce paragraphe une technique qui permet de décrire l'ensemble des solutions d'un problème par le biais d'un *polynôme*. On commence par introduire quelques notions.

Soit $X = \{x_1, x_2, \ldots, x_k\}$ un ensemble de variables avec $k \in \mathbb{N}$. Pour tout sous-ensemble $X' \subseteq X$, le produit des variables de X' est appelé un monôme. Par exemple, le monôme du sous-ensemble $X' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$ de X est $x_{i_1} x_{i_2} \ldots x_{i_p}$. Une somme de monômes est appelée un *polynôme*. Le *degré* d'un monôme est la somme des degrés de chacune de ses variables. Ainsi, le degré de $x_{i_1}^2 x_{i_2}^3 x_{i_3}$ est $2+3+1 = 6$. Enfin, un monôme est dit *multilinéaire* si chacune de ses variables est de degré 1.

On peut représenter un polynôme P sur un ensemble de variables X sous la forme d'un *circuit arithmétique* A, qui est un *DAG* à racine unique dont chaque feuille correspond à une variable de X et dont le reste des sommets correspond à une opération $\oplus$ ou $\otimes$. Le polynôme décrit par une feuille est la variable

contenue dans cette feuille. Le polynôme décrit par un sommet $\oplus$ (resp. $\otimes$) est l'addition (resp. la multiplication) des voisins sortants de ce sommet. Enfin, le polynôme P décrit par un circuit arithmétique A est décrit par la racine de A. On montre un exemple dans la Figure 1.8.

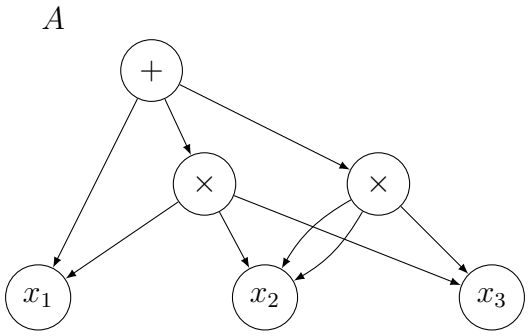


Figure 1.8 – Exemple de circuit décrivant le polynôme $P = x_1 + x_1x_2x_3 + x_2^2x_3$.

On introduit maintenant le problème k-MULTILINEAR DETECTION ci-dessous, puis on explique comment utiliser ce problème pour obtenir des algorithmes de complexité paramétrée.

k-MULTILINEAR DETECTION (k-MLD)

• **Instance**: Un circuit arithmétique A représentant un polynôme P sur un ensemble X de variables, un entier k

• **Question**: Existe-t-il un monôme multilinéaire de degré exactement k dans P ?

Pour résoudre un problème Q paramétré par k en utilisant une technique de détection de monômes multilinéaires, le principe consiste à effectuer une réduction de Q vers k-MULTILINEAR DETECTION. On peut ensuite appliquer le théorème suivant, pour lequel on rappelle que la notation $\tilde{\mathcal{O}}()$ permet de masquer les facteurs polylogarithmiques.

Théorème 10. *([73]) Le problème* k-MLD *peut être résolu par un algorithme probabiliste de complexité en temps* $\tilde{\mathcal{O}}(2^k|A|)$ *et de complexité en espace* $\tilde{\mathcal{O}}(|A|)$.

On montre maintenant une réduction de k-PATH vers k-MLD afin de résoudre k-PATH avec un algorithme de complexité paramétrée [73].

Théorème 11. *([73]) Le problème* k-PATH *peut être résolu par un algorithme probabiliste de complexité en temps* $\mathcal{O}^*(2.83^k)$ *et de complexité polynomiale en espace.*

Preuve. On va construire un polynôme P dans lequel chaque monôme multilinéaire de degré k représente un chemin simple de longueur k dans $G = (V, E)$, où $|V| = n$.

On commence par définir deux ensembles de variables $X = \{x_{v_1}, \ldots, x_{v_n}\}$ et $Y = \cup_{t=1}^{k} \cup_{(v_i,v_j)\in E} y_{v_i,v_j,t}$. On représente un chemin τ par un monôme $enc(\tau)$, qui contient une variable de type x_{v_i} si $v_i \in \tau$ et qui contient une variable de type $y_{v_i,v_j,t}$ si l'arête (v_i, v_j) est la t-ième arête visitée par τ et que t est impair. Plus généralement, si $\tau =< v_1, v_2, \ldots, v_t >$ est un chemin de longueur $t-1$ alors on définit :

$$enc(\tau) = (\prod_{i=1}^{t} x_{v_i})(\prod_{j=1}^{\lfloor \frac{t}{2} \rfloor} y_{v_{2j-1},v_{2j},2j-1})$$

Pour donner un exemple, si $\tau =< v_1, v_2, v_3, v_4 >$ est un chemin, alors $enc(\tau) = x_{v_1}x_{v_2}x_{v_3}x_{v_4}y_{v_1,v_2,1}y_{v_3,v_4,3}$. Pour tout chemin τ dans G, on montre que $enc(\tau)$ est multilinéaire si et seulement si τ est un chemin simple. En effet, on observe qu'on peut exprimer τ uniquement à partir de son sommet d'arrivée v_t ainsi que

les arêtes (u, v) de τ telles qu'il existe un nombre pair de sommets entre u et v_t dans la liste de sommets consécutifs visités par τ. Un monôme exprime donc un unique chemin. De plus, si τ n'est pas un chemin simple alors on remarque que τ passe au moins deux fois par un sommet v_i et le degré de x_{v_i} est au moins égal à 2 dans $enc(\tau)$.

Soit $P_{t,v}$ un polynôme dont chaque monôme représente un chemin simple de longueur t et dont le sommet d'arrivée est $v \in V$. Pour tout $v \in V$, on observe que $P_{0,v} = x_v$, puisqu'un chemin de longueur 0 ne contient qu'un sommet. Plus généralement, pour tout $v \in V$ et pour tout $t \in \{1, \ldots, k\}$, on obtient :

$$P_{t,v} = \sum_{(u,v)\in E} x_v (y_{u,v,t})^{(t \mod 2)} P_{t-1,u}$$

Dans cette formule, le but est d'étendre chaque chemin de longueur $t-1$ dont le sommet d'arrivée est $u \in N(v)$ par une arête (u, v) pour obtenir un chemin de longueur t dont le sommet d'arrivée est v. Ainsi, on multiplie chaque monôme de $P_{t-1,u}$ par x_v et par $y_{u,v,t}$ (une fois sur deux selon la position de l'arête dans le chemin) pour que tous les chemins de $P_{t,v}$ soient correctement encodés.

Si on connaît les valeurs de $P_{t-1,u}$ pour tout $u \in V$, on peut calculer $P_{t,v}$ en $\mathcal{O}(m)$ opérations – où m est le nombre d'arêtes de G – et calculer $P_{k,v}$ en $\mathcal{O}(km)$ opérations. Par conséquent, on peut calculer le polynôme $P = \sum_{v\in V} P_{k,v}$ en $\mathcal{O}(kmn)$ opérations. De plus, on observe que le circuit A représentant P est de taille polynomiale puisque A possède $\mathcal{O}(kn)$ nœuds et que chaque nœud de A est de degré $\mathcal{O}(m)$. Maintenant, si un chemin τ est de longueur k, alors on observe que $enc(\tau)$ est de degré $k - 1 + \lfloor \frac{k-1}{2} \rfloor$. Par conséquent, on peut utiliser le Théorème 10 pour obtenir un algorithme probabiliste qui réduit une instance I de k-PATH vers une instance I' de k-MLD, puis qui résout l'instance I'. L'algorithme proposé possède une complexité en temps de $\mathcal{O}^*(2^{\frac{3k}{2}}) = \mathcal{O}^*(2.83^k)$, et une complexité polynomiale en espace. □

Par la suite, Williams a amélioré le Théorème 11 et a proposé un algorithme probabiliste de complexité en temps $\mathcal{O}^*(2^k)$ [120].

Narrow sieves. On aborde ici la technique des *narrow sieves* ("tamis serrés" en français), une autre technique de détection de monômes qui nécessite d'abord de construire un polynôme à partir de l'instance d'un problème.

Björklund a tout d'abord utilisé la technique des narrow sieves pour résoudre le problème HAMILTONIAN PATH, qui est équivalent à k-PATH avec $k = |V|$, avec un algorithme probabiliste en temps $\mathcal{O}^*(1.657^n)$ [11]. Le problème HAMILTONIAN PATH est considéré comme un problème majeur de la littérature et fait notamment partie des 21 premiers problèmes à avoir été montrés NP-complets par Karp en 1972 [69]. L'algorithme proposé par Björklund peut lui aussi être considérée comme une avancée majeure puisque les meilleurs algorithmes jusque-là possédaient une complexité temporelle de $\mathcal{O}^*(2^n)$ et avaient été proposé en 1962 [7, 65].

On illustre maintenant le concept des narrow sieves avec l'algorithme proposé par Björklund pour résoudre le problème k-PATH [12].

Théorème 12. *([12]) Le problème k-PATH peut être résolu par un algorithme probabiliste de complexité en temps $\mathcal{O}^*(1.66^k)$.*

Preuve. Tout d'abord, on rappelle que si un polynôme P contient un monôme par chemin de longueur k dans G, alors P peut contenir $\mathcal{O}(n^k)$ monômes. De manière similaire au color-coding, la technique des narrow sieves cherche donc un sous-ensemble particuliers de solutions. Pour décrire les monômes correspondant à ces solutions, on commence par partitionner aléatoirement l'ensemble des sommets V de G en deux sous-ensembles V_1 et V_2 de cardinalités égales modulo 1. On note E_1 (resp. E_2) l'ensemble des arêtes du graphe $G[V_1]$ (resp. $G[V_2]$). On définit ensuite deux entiers $k_1 = \lfloor 0.5k \rfloor$ et $k_2 = \lfloor 0.2017707k \rfloor$; la valeur

de k_1 et k_2 a été calculée pour minimiser la complexité finale de l'algorithme. Dans la suite, on s'intéresse aux chemins simples de longueur k dans G qui contiennent k_1 sommets dans V_1 et k_2 arêtes dans $G[V_2]$. La recherche de ces chemins particuliers explique en partie la complexité en $\mathcal{O}^*(2^{k_1+k_2})$ de l'algorithme proposé.

On montre maintenant comment encoder un chemin de G par un monôme. Pour cela, on définit deux ensembles de couleurs $K_1 = \{1, \ldots, k_1\}$ et $K_2 = \{1, \ldots, k_2\}$ ainsi que trois ensembles de variables $X = \{x_e : e \in E\}$, $Y = \{y_{v,i} : v \in V_1, i \in K_1\}$ et $Z = \{z_{e,j} : e \in E_2, j \in K_2\}$. Pour tout chemin τ, un monôme représentant τ est constitué d'une variable x_e pour chaque arête de τ, d'une variable de type $y_{v,i}$, où i est une couleur arbitraire de K_1, pour chaque occurrence d'un sommet de τ qui appartient à V_1, et enfin d'une variable de type $z_{e,j}$, où j est une couleur arbitraire de K_2, pour chaque occurrence d'une arête de τ qui appartient à E_2. Par exemple, soit $\tau =< v_1, v_2, v_3 >$ un chemin avec $v_1 \in E_1$, $(v_2, v_3) \in E_2$, $K_1 = \{1, 2\}$ et $K_2 = \{1\}$. On observe qu'on peut encoder τ avec un monôme $x_{(v_1,v_2)}x_{(v_2,v_3)}y_{v_1,1}z^2_{(v_2,v_3),1}$ ou bien un monôme $x_{(v_1,v_2)}x_{(v_2,v_3)}y_{v_1,2}z_{(v_2,v_3),1}z_{(v_2,v_3),2}$.

En règle générale, on remarque que l'ensemble des chemins non-simples peut être partitionné en paires de façon à ce que deux éléments d'une même paire soient représentés par le même monôme. Ainsi, un monôme $x_{(v_1,v_2)}x_{(v_2,v_3)}x_{(v_3,v_4)}x_{(v_4,v_2)}y_{v_1,1}z_{(v_2,v_3),1}$ peut à la fois représenter le chemin $\tau_1 =< v_1, v_2, v_3, v_4, v_2 >$ et le chemin $\tau_2 =< v_1, v_2, v_4, v_3, v_2 >$. Par conséquent, un monôme représentant un chemin simple possède la particularité d'être *unique* et non plus multilinéaire comme vu précedemment.

L'algorithme consiste à calculer un polynôme P dans lequel chaque monôme présente deux caractéristiques. Premièrement, chaque monôme de P doit représenter un chemin simple de longueur k dans G qui contient k_1 sommets dans V_1 et k_2 arêtes dans $G[V_2]$. Deuxièmement, chaque monôme de P doit être *colorful*, *i.e.* ne doit pas contenir deux variables y_{v_1,i_1} et y_{v_2,i_2} (resp. z_{e,j_1} et z_{e,j_2}) telles que $i_1 = i_2$ (resp. $j_1 = j_2$). Ces deux caractéristiques permettent de borner le nombre maximum de monômes contenus dans P afin d'obtenir un algorithme FPT.

Le calcul de P se décompose en deux parties. Tout d'abord, on remplit une table de programmation dynamique $A[K'_1, K'_2]$ pour tout $K'_1 \subseteq K_1$ et $K'_2 \subseteq K_2$. Une case $A[K'_1, K'_2]$ contient la somme des monômes ne contenant aucune variable de type $y_{v,i}$ avec $i \in K'_1$ et aucune variable de type $z_{e,j}$ avec $j \in K'_2$. On dit que $A[K'_1, K'_2]$ contient la somme des monômes qui *évitent* les ensembles K'_1 et K'_2. Lors de l'évaluation des cases de A, toutes les paires identiques de monômes, qui représentent donc des chemins non-simples, s'annulent par des artifices algébriques. Les cases de A ne contiennent donc que des monômes représentant des chemins simples.

Pour chaque case de type $A[K'_1, K'_2]$, on note que les monômes contenus dans $A[K'_1, K'_2]$ ne sont pas nécessairement colorfuls et ne contiennent pas nécessairement k_1 sommets dans V_1 et k_2 arêtes dans $G[V_2]$. Pour calculer P, on applique donc le *principe d'inclusion/exclusion* : on initialise $P = 0$ puis, pour tout $K'_1 \subseteq K_1$ et $K'_2 \subseteq K_2$, on ajoute $(-1)^{|K'_1|+|K'_2|}A[K'_1, K'_2]$ à P. Pour illustrer ce principe, on fait une analogie avec un problème simplifié. Soit $X = \{x_1, x_2, x_3\}$ et $P(X) = x_1x_2x_2 + x_1x_2x_3 + x_2x_2x_2$ un polynôme sur X. On note $P_{COL}(X)$ la somme des monômes de $P(X)$ qui ne contiennent pas deux fois la même variable et, pour tout ensemble $S \subseteq X$, on note $P_S(X)$ la somme des monômes de P_X qui ne contiennent pas de variables appartenant à S. En appliquant le principe d'inclusion/exclusion, on obtient :

$$\begin{aligned} P_{COL}(X) &= \textstyle\sum_{S\subseteq X}(-1)^{|S|}P_S(X) \\ &= P_{\{\emptyset\}}(X) \\ &\quad -P_{\{1\}}(X) - P_{\{2\}}(X) - P_{\{3\}}(X) \\ &\quad +P_{\{1,2\}}(X) + P_{\{1,3\}}(X) + P_{\{2,3\}}(X) \\ &\quad -P_X(X) \\ &= x_1x_2x_2 + x_1x_2x_3 + x_2x_2x_2 \\ &\quad -(x_2x_2x_2) - (x_1x_2x_2 + x_2x_2x_2) \\ &\quad +(x_2x_2x_2) \\ &= x_1x_2x_3 \end{aligned}$$

Si un graphe G contient un chemin simple de longueur k, alors Björklund a montré que la probabilité que ce chemin appartienne à P dépend uniquement de k, k_1 et k_2. Dès lors, appliquer l'algorithme présenté ci-dessus sur *un nombre suffisant* de partitions arbitraires de V en deux sous-ensembles V_1 et V_2 permet de trouver un tel chemin avec une probabilité constante de 1. Ce nombre suffisant dépend également uniquement de k, k_1 et k_2 et l'algorithme possède une complexité en temps de $\mathcal{O}^*(1.66^k)$. □

Familles représentatives. Pour résoudre le problème k-PATH dans la Section 1.3.3, on a commencé par présenter un algorithme utilisant une table de programmation dynamique A telle que chaque case $A[v,i]$ contient les ensembles de sommets de tous les chemins de longueur $0 \leq i \leq k$ et dont le sommet d'arrivée est $v \in V$. Puisqu'une case de A contient $\mathcal{O}(n^i)$ chemins, on a expliqué comment utiliser le color-coding (et ses améliorations) afin d'obtenir des algorithmes probabilistes de complexité paramétrée qui résolvent – plusieurs fois – une version plus contrainte de k-PATH. On montre maintenant comment construire une table $\hat{A}$, pour tout $v \in V$ et $i \in [k+1]$, telle que le nombre de chemins contenus dans une case $\hat{A}[v,i]$ soit uniquement exponentiel relativement à i, et telle que $\hat{A}[v,k+1] \neq \emptyset$ si et seulement si il existe un chemin de longueur k dans G dont le sommet d'arrivée est v. En d'autres termes, on montre maintenant comment obtenir un algorithme *déterministe* de complexité paramétrée pour le problème k-PATH.

Pour définir $\hat{A}$, on introduit la définition suivante :

Définition 13. Soit U un ensemble d'éléments et S une famille de sous-ensembles de U telle que chaque sous-ensemble de S est de taille p. Une sous-famille $S' \subseteq S$ est q-représentative de S si, pour chaque sous-ensemble $Y \subseteq U$ de taille au plus q, on a : s'il existe un sous-ensemble $X \in S$ qui est disjoint de Y, alors il existe un sous-ensemble $X' \in S'$ qui est disjoint de Y.

On utilise cette définition pour construire $\hat{A}$, dans laquelle chaque case $\hat{A}[v,i]$ est une famille $(k-i-1)$-représentative de $A[v,i]$. Ainsi, pour chaque case $\hat{A}[v,i]$, s'il existe un chemin $\tau \in A[v,i]$, avec $V(\tau) = X$, et un ensemble $Y \subseteq V$ de $k-i-1$ sommets tels qu'il existe un chemin de longueur k dans G dont les sommets sont $X \cup Y$, alors il existe un chemin $\hat{\tau} \in \hat{A}[v,i]$ avec $V(\hat{\tau}) = \hat{X}$ tel qu'il existe un chemin de longueur k dans G dont les sommets sont $\hat{X} \cup Y$. En d'autres termes, chaque case $\hat{A}[v,i]$ contient uniquement un sous-ensemble des chemins de longueur i dont le sommet d'arrivée est v ; ce sous-ensemble de chemins est calculé afin d'assurer que $\hat{A}[v,k+1] \neq \emptyset$ s'il existe un chemin de longueur k dans G dont le sommet d'arrivée est v.

Pour savoir comment calculer une famille représentative, on conseille au lecteur de lire la thèse de Panolan [50]. La complexité en temps d'un algorithme déterministe qui consiste à calculer la table $\hat{A}$ est de $\mathcal{O}^*(2.619^k)$ [56].

Bilan color-coding. Afin de comparer les améliorations du color-coding, on commence par récapituler dans le Tableau 1.1 les meilleurs algorithmes qui ont été conçus pour résoudre k-PATH, le fil conducteur de cette section, à partir de ces améliorations. On rappelle qu'avant l'article d'Alon *et al.* [2], la complexité en

temps des meilleurs algorithmes FPT (qui sont déterministes) pour résoudre k-PATH était de $\mathcal{O}^*(k!)$ [86] et $\mathcal{O}^*(k!2^k)$ [19].

Technique	Complexité en temps (Algorithmes probabilistes)	Complexité en temps (Algorithmes déterministes)
Color-coding	$\mathcal{O}^*((2e)^k)$ [2]	$\mathcal{O}^*(c^k), c > 8000$ [2]
Divide-and-color	$\mathcal{O}^*(4^k)$ [72, 28]	$\mathcal{O}^*(16^k)$ [72]
k-MLD	$\mathcal{O}^*(2^k)$ [120]	-
Narrow Sieves	$\mathcal{O}^*(1.66^k)$ [12]	-
Familles représentatives	-	$\mathcal{O}^*(2.619^k)$ [56]

Tableau 1.1 – Résultats principaux de complexité paramétrée pour k-PATH relativement à k. On désigne la technique de détection de monômes multilinéaires par l'acronyme k-MLD.

Koutis et Williams ont montré qu'il est peu vraisemblable de résoudre le problème k-MLD (et donc le problème k-PATH) plus rapidement que leur algorithme en $\mathcal{O}^*(2^k)$ [75]. Selon Koutis [73], les méthodes du color-coding et du divide-and-conquer ont également des limites et ne permettent pas de résoudre le problème k-PATH en temps inférieur à $\mathcal{O}^*(2^k)$ – même si l'algorithme proposé est déterministe. La technique des narrow sieves est donc celle qui permet d'obtenir la meilleure complexité en temps pour résoudre k-PATH.

On discute maintenant brièvement des possibles applications des méthodes décrites dans cette section pour le problème k-SHORT CHEAP TOUR. Ce problème est une variante de k-PATH dans laquelle on ajoute des poids sur les arêtes. Il est intéressant de donner ici des résultats pour cette variante de k-PATH puisqu'on étudiera dans les chapitres suivants plusieurs problèmes dont les graphes sont pondérés. On commence par définir k-SHORT CHEAP TOUR ci-dessous.

k-SHORT CHEAP TOUR (k-SCT)

• **Instance**: Un graphe $G = (V, E)$, une fonction de pondération $w : E \to \mathbb{N}$, un entier k

• **Question**: Existe-t-il un chemin simple dont la somme des poids des arêtes est au moins égale à k dans G ?

Soit W le poids de l'arc de poids maximum dans G. L'algorithme de Björklund qui utilise la technique des narrow sieves peut être adapté afin de résoudre k-SCT en temps $\mathcal{O}(1.66^k n^{\mathcal{O}(1)} W)$ [12]. Williams observe que l'algorithme de divide-and-color de Chen *et al.* [28] peut être utilisé pour réduire la dépendance en W et ainsi résoudre k-SCT en temps $\mathcal{O}(4^k n^{\mathcal{O}(1)} \log W)$ [120]. Le même auteur indique en revanche que son algorithme de détection de monômes pour résoudre k-PATH ne peut pas être adapté pour résoudre k-SCT.

Pour les lecteurs intéressés par les variantes du color-coding, nous conseillons l'excellent article de Koutis et Williams [76].

1.3.4 Recherche de noyau

On montre ici une nouvelle technique afin d'obtenir des algorithmes de complexité paramétrée. La recherche de noyau (ou *kernelisation*) consiste à prétraiter une instance d'un problème paramétré afin de réduire la taille de cette instance. On définit formellement ci-dessous le processus de kernelisation.

Définition 14. (Kernelisation) Soit Q un problème paramétré par k. Il existe une kernelisation d'une instance (I, k) de Q vers une instance (I', k') de Q si

1. la réponse à Q sur (I, k) est la même que la réponse à Q sur (I', k'),

2. $|I'| = f(k)$, où f est une fonction calculable qui ne dépend que de k,
3. $k' \leq g(k)$, où g est une fonction calculable qui ne dépend que de k.

L'instance (I', k') est appelé un kernel (ou noyau).

Intuitivement, le kernel est la partie difficile d'un problème. Ainsi, rechercher un kernel consiste à réduire l'instance (I, k) en une instance (I', k') dont la taille ne dépend que de k tout en préservant l'ensemble des solutions. On rappelle que tous les problèmes étudiés dans cette thèse appartiennent à la classe XP. On suppose donc que le problème Q de la définition ci-dessus peut être résolu en $\mathcal{O}(|I|^{h(k)})$, où h est une fonction calculable. Par conséquent, si toutes les instances (I, k) de Q paramétré par k peuvent être kernelisées, alors on observe que Q appartient à FPT relativement à k puisque $\mathcal{O}(|I'|^{h(k')}) = \mathcal{O}(|f(k)|^{h(g(k))})$.

Pour rechercher un kernel, on utilise des *règles de réduction* afin d'éliminer les parties faciles à résoudre d'une instance. On montre deux exemples de règles de réduction pour le problème k-VERTEX COVER qui est défini dans la Section 1.3.1.

Règle de réduction 15. *Si une instance (G, k) de k-VERTEX COVER contient un sommet $v \in V$ tel que $d(v) > k$ pour $k > 0$, alors on exécute les instructions suivantes:*
— *on supprime v et toutes les arêtes incidentes à v de G,*
— *on fixe $k = k - 1$.*

Lemme 16. *La Règle de réduction 15 est correcte.*

Preuve. Si v n'appartient pas à une solution V' de k-VERTEX COVER dans (G, k), alors tous les voisins de v doivent appartenir à V'. Puisque $d(v) > k$, la cardinalité de V' est supérieure à k et V' n'est donc pas une solution correcte de k-VERTEX COVER dans (G, k). Ainsi, s'il existe une solution V' de k-VERTEX COVER dans (G, k), alors $v \in V'$. □

Règle de réduction 17. *Si une instance (G, k) de k-VERTEX COVER contient un sommet $v \in V$ tel que $d(v) = 0$, alors on supprime v de G.*

Lemme 18. *La Règle de réduction 17 est correcte.*

Preuve. Un sommet de degré 0 n'est incident à aucune arête. Ainsi, s'il existe une couverture par les sommets $V' \subseteq V$ de G telle que $v \in V'$, alors $V' \setminus \{v\}$ est également une couverture par les sommets de G. □

Les Règles de réduction 15 et 17 nous permettent maintenant de prouver que k-VERTEX COVER admet un kernel – de taille polynomiale – relativement à k.

Proposition 19. *k-VERTEX COVER admet un kernel contenant au plus $2k^2$ sommets.*

Preuve. Soit $I = (G, k)$ l'instance de k-VERTEX COVER obtenue après applications récursives des Règles de réduction 15 et 17. Si G contient plus de k^2 arêtes, alors on prouve qu'il n'existe pas de solution V' de k-VERTEX COVER dans G tel que $|V'| \leq k$. Pour cela, on observe que chaque sommet de G est incident à au plus k arêtes puisqu'on ne peut pas appliquer la Règle de réduction 15 sur I. Par conséquent, tout sous-ensemble de sommets V' de cardinalité k couvre au plus k^2 arêtes de G. Si G contient plus de k^2 arêtes, alors on vient de montrer que G n'admet pas de couverture par les sommets de cardinalité au plus k. Pour finir, on rappelle que G ne contient pas de sommets de degré 0, puisqu'on ne peut pas appliquer la Règle de réduction 17 sur I. Ainsi, s'il existe une solution de k-VERTEX COVER de cardinalité au plus k dans G, alors G contient au plus $2k^2$ sommets – puisqu'une arête a deux extrêmités. □

On vient de montrer un exemple simple de kernelisation pour le problème VERTEX COVER. Le meilleur algorithme de kernelisation a été proposé par Lampis, qui a montré que VERTEX COVER admet un kernel contenant au plus $2k - c \log k$ sommets où c est une constante strictement positive [80]. En testant naïvement toutes les combinaisons de sommets, on obtient un algorithme FPT pour VERTEX COVER en temps $\mathcal{O}^*(2^{2k-c\log k})$.

Un problème Q paramétré par k appartient à FPT si et seulement si Q est décidable et admet un kernel relativement à k. Il est légitime de se demander si Q admet un kernel *polynomial* relativement à k afin d'obtenir de meilleurs algorithmes de complexité paramétrée. Dans la suite, on rappelle que le problème k-PATH appartient à FPT relativement à k (voir Section 1.3.3), ce qui implique donc que k-PATH admet un kernel relativement à k. On introduit le concept d'*or-composition* afin de montrer ensuite que k-PATH n'admet pas de kernel *polynomial* relativement à k.

Définition 20. (Or-composition [20]) Soit Q un problème paramétré par k. Une or-composition est un algorithme qui prend en entrée un ensemble d'instances $(I_1, k), \ldots, (I_t, k)$, avec $t > 0$, et retourne en temps polynomial en $\sum_{i=1}^{t} |I_i| + k$ une instance paramétrée (I, k') de Q telle que :

1. (I, k') est une YES-instance de Q si et seulement si il existe $i \in \{1, \ldots, t\}$ tel que l'instance (I_i, k) est une YES-instance de Q,
2. $k' \leq g(k)$ où g est une fonction polynomiale qui dépend uniquement de k.

Si un problème Q paramétré par k est NP-dur et admet une or-composition, alors Q n'admet pas de kernel polynomial relativement à k (sauf si NP $\subseteq$ coNP/Poly). Par exemple, on rappelle que k-PATH est NP-dur. De plus, pour n'importe quel ensemble d'instances $(I_1, k), \ldots, (I_t, k)$ de k-PATH, on observe que le problème admet clairement une or-composition qui construit une instance (I, k) dont le graphe d'entrée est l'union de tous les graphes des instances de type (I_i, k). Par conséquent, k-PATH n'admet pas de kernel polynomial en k (sauf si NP $\subseteq$ coNP/Poly).

Une autre méthode pour prouver qu'un problème Q paramétré par k n'admet pas de kernel polynomial consiste à effectuer une *transformation polynomiale paramétrée*.

Définition 21. (Transformation polynomiale paramétrée [22, 21, 32]) Soit Q_1 et Q_2 deux problèmes paramétrés respectivement par k_1 et k_2. Il existe une transformation polynomiale paramétrée de Q_1 paramétré par k_1 vers Q_2 paramétré par k_2 si, pour chaque instance (I_1, k_1) de Q_1, on peut construire une instance (I_2, k_2) de Q_2 telle que :

1. (I_1, k_1) est une YES-instance de Q_1 si et seulement si (I_2, k_2) est une YES-instance de Q_2,
2. $(I_2, k_2) = f(I_1, k_1)$ où f est une fonction polynomiale ne dépendant que de I_1 et k_1,
3. $k_2 \leq g(k_1)$ où g est une fonction polynomiale ne dépendant que de k_1.

La fonction f est appelée une transformation polynomiale paramétrée.

Si le problème Q_1 est NP-dur et que $Q_2 \in$ NP, alors une transformation polynomiale paramétrée de Q_1 paramétré par k_1 vers Q_2 paramétré par k_2 a la conséquence suivante : si Q_1 admet un kernel polynomial relativement à k_1 alors Q_2 admet un kernel polynomial relativement à k_2. De même, si Q_1 n'admet pas de kernel polynomial par rapport à k_1 (sauf si NP $\subseteq$ coNP/Poly), alors Q_2 n'admet pas de kernel polynomial par rapport à k_2 (sauf si NP $\subseteq$ coNP/Poly).

1.4 Conclusion

Dans ce manuscrit, on étudie des problèmes NP-durs dont les instances sont des graphes. Notre but consiste à distinguer les classes de graphes pour lesquelles ces problèmes se résolvent en temps polynomial

de celles pour lesquelles ils sont NP-durs, à développer des algorithmes d'approximation et des algorithmes FPT, ou bien encore à montrer des kernels afin de réduire la taille des instances de ces problèmes. Pour ce faire, on a respectivement présenté dans les Sections 1.1 et 1.2 des notions essentielles en théorie des graphes et en théorie de la complexité. On a également montré dans la Section 1.3 plusieurs techniques afin d'obtenir des algorithmes FPT. On va maintenant observer dans le Chapitre 2 comment ces techniques ont été appliquées pour résoudre le problème GRAPH MOTIF et ses variantes.

2

Le problème GRAPH MOTIF et ses variantes

Dans ce manuscrit, on propose une étude algorithmique des problèmes liés à la recherche de sous-graphes d'intérêt dans les graphes. Pour cela, on se focalise dans ce chapitre sur le problème GRAPH MOTIF. Il existe en effet une vaste littérature pour résoudre ce problème à l'aide de nombreuses techniques algorithmiques innovantes. Ainsi, on décrit dans ce chapitre un état de l'art de GRAPH MOTIF et de ses variantes avant de travailler sur un nouveau problème, appelé MAXIMUM COLORFUL ARBORESCENCE, dans les chapitres suivants. On présente tout d'abord dans la Section 2.1 des problématiques biologiques liées à la recherche de motifs, puis on montre que ces problématiques ont engendré plusieurs définitions d'un motif. On s'intéresse ensuite plus particulièrement à la recherche de motifs dits fonctionnels, ce qui nous amène à modéliser le problème GRAPH MOTIF ainsi que plusieurs de ses variantes. Dans le reste de ce chapitre, on présente les résultats algorithmiques principaux issus de la littérature pour GRAPH MOTIF et ses variantes : on caractérise dans un premier temps les instances algorithmiquement difficiles de ces problèmes dans la Section 2.3, puis on présente des résultats de complexité paramétrée dans la Section 2.4. En particulier, on montre dans cette section comment les techniques qui ont été décrites dans le Chapitre 1 pour obtenir des algorithmes FPT ont été utilisées pour résoudre GRAPH MOTIF et ses variantes. Enfin, on présente dans la Section 2.5 une liste de logiciels conçus pour résoudre les problèmes évoqués ci-dessus.

2.1 Motivations et modélisation de GRAPH MOTIF

De nombreux phénomènes biologiques peuvent être modélisés à l'aide de graphes appelés des réseaux biologiques. Dans la suite, on définit deux types de réseaux biologiques ainsi que leurs problématiques associées. On verra que ces problématiques impliquent la recherche d'un sous-réseau, appelé *motif*, qui est inclus dans le réseau biologique étudié. En particulier, on distinguera les *motifs topologiques*, assimilés à des problématiques d'isomorphisme de sous-graphe, des *motifs fonctionnels*, introduits par Lacroix *et al.* [79]. Enfin, on utilisera la définition d'un motif fonctionnel pour introduire le problème GRAPH MOTIF et trois de ses variantes, qui seront étudiés dans les sections suivantes.

Description de réseaux biologiques et des problématiques associées. On dispose à ce jour de nombreuses bases de données biologiques sur Internet. Par exemple, la base de données STRING [1] contient actuellement près de dix millions de protéines provenant de plus de deux mille organismes différents. Les protéines sont des molécules complexes occupant des fonctions variées dans l'organisme. Elles peuvent

1. https://string-db.org/

par exemple servir d'anticorps, servir à coordonner différents processus biologiques entre des tissus, des cellules ou des organes, ou bien encore servir à déplacer des atomes ou même d'autres protéines entre les cellules. On peut modéliser les interactions physiques entre ces protéines par le biais de graphes appelés des réseaux d'interaction protéine-protéine (PPI pour "Protein-Protein Interaction"). Les sommets de ces graphes correspondent aux protéines d'un organisme et les interactions physiques entre ces protéines sont symbolisées par des arêtes (voir Figure 2.1).

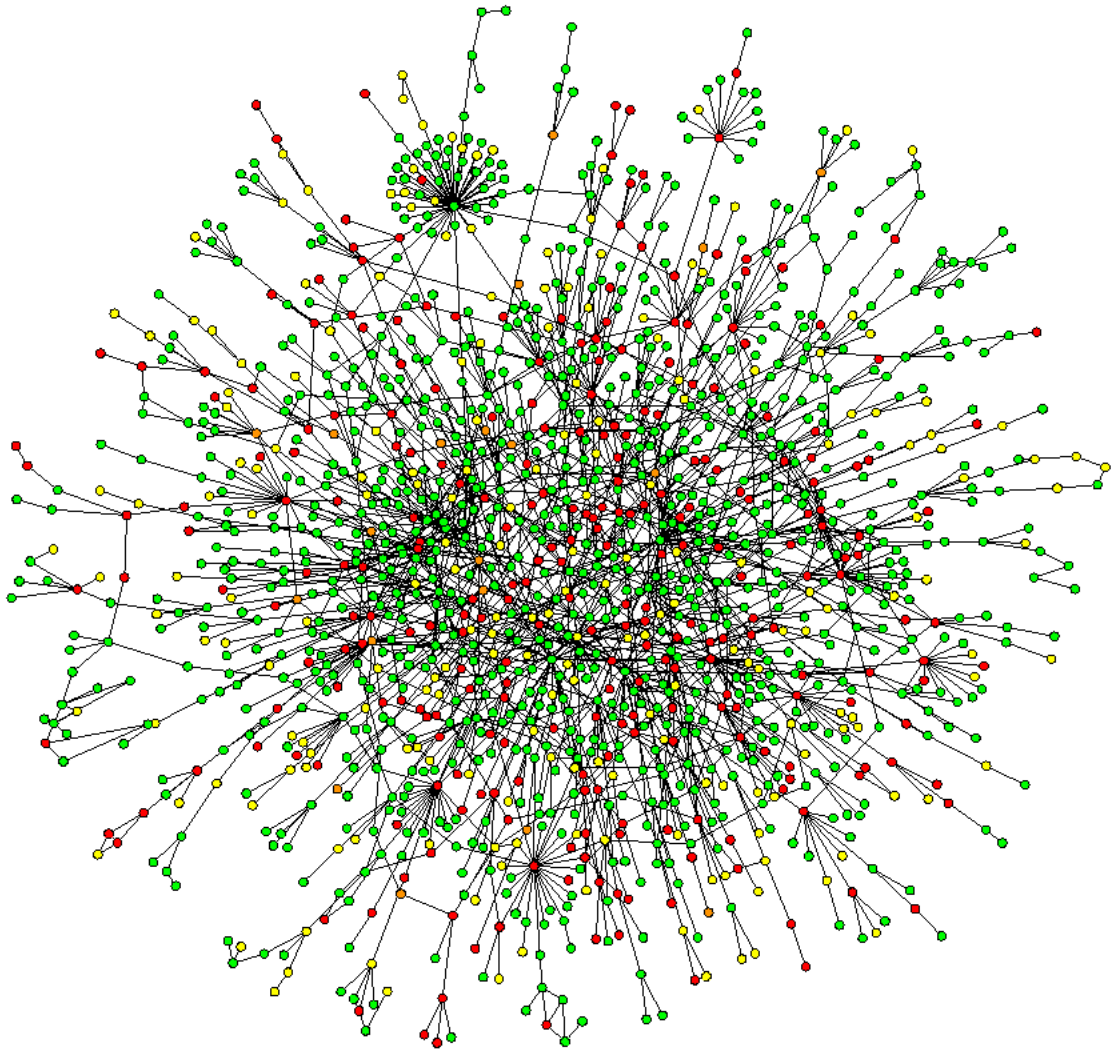


Figure 2.1 – Réseau PPI de la levure provenant de [68]. Dans cet article, les auteurs montrent que supprimer de l'organisme une protéine associée à un sommet de couleur rouge (resp. verte) serait (resp. ne serait pas) léthal pour l'organisme. Supprimer une protéine associée à un sommet de couleur orange entraînerait une croissance modérée de l'organisme. Enfin, les effets provoqués par la suppression d'un sommet de couleur jaune sont inconnus.

Analyser les interactions protéine-protéine est essentiel afin de mieux appréhender le fonctionnement complexe des cellules. Par exemple, la malade d'Huntington est une maladie grave qui provoque d'importants troubles moteurs, cognitifs et psychiatriques. Cette maladie est causée par une anomalie de la huntingtine, une protéine vitale au bon fonctionnement du corps humain. Ainsi, de nombreuses études consistent à déterminer quelles interactions entre protéines sont les plus importantes dans le ou les processus responsable(s) du déclenchement et du développement de la maladie d'Huntington [105, 83]. En règle générale, les études réalisées sur les réseaux PPI consistent soit à chercher un sous-réseau particulier appelé *complexe*

protéique dans un réseau PPI, soit à aligner deux réseaux pour y trouver un sous-réseau de protéines dont les interactions sont similaires [5, 77, 71].

Les réseaux métaboliques servent à décrire le *métabolisme* d'un organisme, *i.e.* le processus à travers lequel l'organisme obtient et utilise de l'énergie pour accomplir différentes tâches [87]. Ces tâches peuvent être séparées en étapes élémentaires appelées des réactions. Chaque réaction transforme des petites molécules appelées métabolites en d'autres métabolites ; les métabolites en entrée sont appelés des *substrats* et les métabolites en sortie sont appelés des *produits*. Généralement, les réactions sont catalysées par des protéines appelés des *enzymes*. Ces enzymes sont classées par une nomenclature EC (pour "Enzyme Commission") en fonction du type de réaction catalysée [118].

Dans la littérature, on trouve plusieurs modélisations en graphe possibles pour représenter un réseau métabolique (voir la thèse de Lacroix pour une description détaillée [78]). Par exemple, dans la Figure 2.2, les sommets correspondent à des métabolites et les arcs correspondent aux réactions impliquant ces métabolites. Dans un graphe des réactions (voir Figure 2.3), les sommets correspondent aux réactions et il existe une arête entre deux métabolites u et v s'il existe une réaction dans laquelle u est un substrat et v un produit.

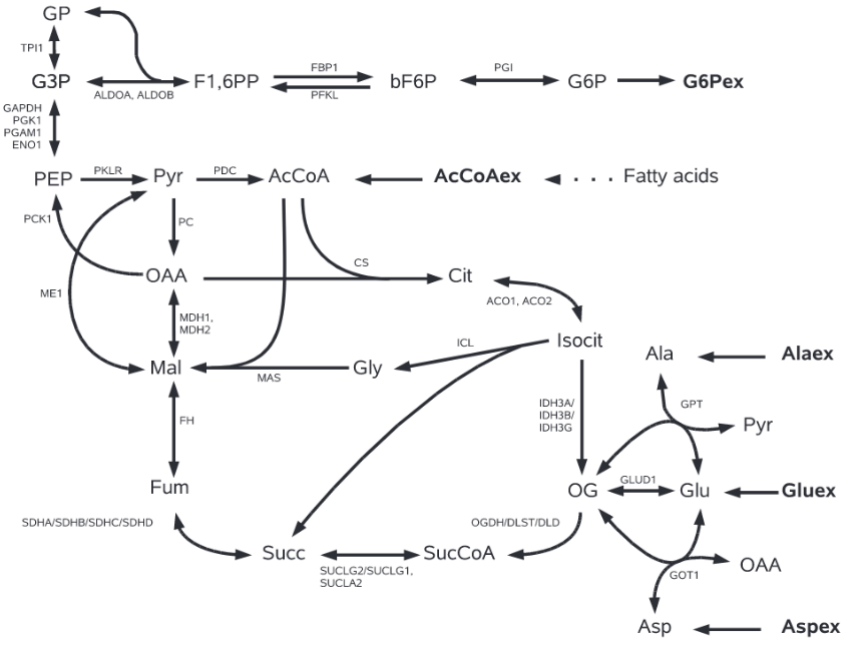


Figure 2.2 – Réseau métabolique de l'humain provenant de [36]. Les sommets représentent des métabolites et les arcs représentent des réactions.

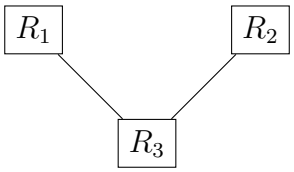


Figure 2.3 – Exemple (fictif) de graphe de réactions produit à partir des réactions $R_1 = A + B \to C$, $R_2 = D + E \to F$ et $R_3 = C + F \to H$. Il existe une arête entre R_1 et R_3 puisque C est un produit de R_1 et un substrat de R_3. De même, il existe une arête entre R_2 et R_3 puisque D est un produit de R_2 et un substrat de R_3.

Une *voie métabolique* est un groupe de réactions qui participent à la même fonction dans le métabolisme. Déterminer les voies métaboliques d'un réseau permet de mieux comprendre comment ce réseau fonctionne.

On peut également comparer deux réseaux métaboliques afin de déterminer des voies métaboliques communes [92, 34, 57, 96].

Recherche de motifs topologiques. Dans le paragraphe ci-dessus, on a vu qu'une des problématiques associées aux réseaux biologiques consiste à se demander si un sous-réseau H est inclus dans un réseau G. Dans la suite, le sous-réseau H est appelé un *motif topologique*. Déterminer si un graphe G contient une *occurrence* d'un motif topologique, c'est-à-dire un sous-graphe isomorphe au motif, est un problème NP-complet [30]. La difficulté du problème est principalement contournée par le biais d'heuristiques [113, 84]. Il existe également quelques algorithmes de complexité paramétrée basés sur le *color-coding* (voir Section 1.3.3, page 26) pour résoudre ce problème. Pour finir, on dénombre plus d'une trentaine de logiciels dont le but est de trouver des motifs topologiques dans [93]. Plus de 80% d'entre eux ont été initialement conçus pour l'analyse de réseaux PPI et près d'un quart [1] pour l'analyse de réseaux métaboliques.

Recherche de motifs fonctionnels : le problème GRAPH MOTIF. En 2006, Lacroix *et al.* proposent une nouvelle méthode pour rechercher des motifs dans des réseaux métaboliques [79]. On rappelle que les réactions sont généralement catalysées par une enzyme. De plus, chaque enzyme possède un numéro dans la nomenclature EC [118], qui correspond au type de réaction catalysée par l'enzyme. Ainsi, Lacroix *et al.* étudient le graphe des réactions d'un réseau métabolique et appliquent une fonction de coloration sur les sommets – les réactions – de ce graphe comme suit : la couleur d'un sommet correspond au numéro EC de l'enzyme – principale s'il en existe plusieurs – qui catalyse la réaction associée à ce sommet. En conséquence, Lacroix *et al.* s'intéressent à la recherche de *motifs fonctionnels*, qu'on définit par un multi-ensemble de couleurs $\mathcal{M}$. On définit ci-dessous le problème GRAPH MOTIF, qui est associé à la recherche de motifs fonctionnels. Un exemple d'instance de GRAPH MOTIF est donné dans la Figure 2.4.

GRAPH MOTIF

• **Instance**: Un graphe $G = (V, E)$, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$, un motif $\mathcal{M}$

• **Question**: Existe-t-il une *occurrence* du motif $\mathcal{M}$ dans G, *i.e.* un ensemble de sommets $V' \subseteq V$ tel que $G[V']$ est connexe et $col(V') = \mathcal{M}$?

L'utilisation de motifs fonctionnels donne plus de souplesse à Lacroix *et al.* pour chercher des voies métaboliques dont ils ne connaissent pas la topologie exacte, puisque $G[S]$ doit seulement être connexe dans la modélisation présentée ci-dessus. D'un point de vue biologique, cela signifie qu'on demande uniquement aux réactions appartenant à une potentielle voie métabolique d'être liées, même si on ne connaît pas la manière dont elles interagissent entre elles. On précise cependant que le multi-ensemble de couleurs associées aux sommets de S doit être égal au motif $\mathcal{M}$ afin d'assurer que les réactions associées aux sommets de S correspondent à la voie métabolique recherchée.

Lacroix *et al.* veulent initialement énumérer toutes les occurrences d'un motif demandé dans [79]. On a ici énoncé la version *décision* du problème GRAPH MOTIF, qui demande s'il existe *une* occurrence de $\mathcal{M}$ dans G. La version proposée par Lacroix *et al.* appartient quant à elle à une catégorie de problèmes qui n'est pas étudiée dans ce manuscrit.

2.2 Présentation de quelques variantes de GRAPH MOTIF

Aujourd'hui, l'article de Lacroix *et al.* [79] a été cité plus de 169 fois dans Google Scholar et le problème GRAPH MOTIF possède une quinzaine de variantes. Ce chapitre n'a pas pour vocation de recenser exhaustivement tous les résultats liés au problème GRAPH MOTIF et à ses variantes. En effet, l'essentiel

1. certains logiciels sont conçus pour plus d'un type de données biologiques

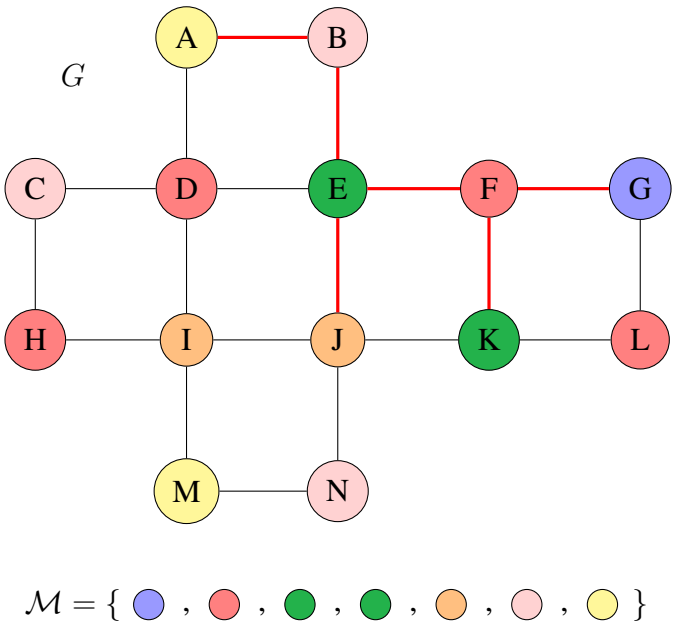


Figure 2.4 – Exemple (fictif) d'une instance positive de GRAPH MOTIF dans laquelle $V' = \{$A, B, E, F, G, J, K$\}$ est une solution possible.

de nos travaux a été effectué sur un problème similaire à GRAPH MOTIF, appelé MAXIMUM COLORFUL ARBORESCENCE, qui n'est pas présenté dans ce chapitre (l'étude de MAXIMUM COLORFUL ARBORESCENCE fera l'objet des Chapitres 3 et 4) ; la lecture de ce chapitre a donc pour but de permettre au lecteur de mieux cerner le paysage algorithmique de MAXIMUM COLORFUL ARBORESCENCE. Pour cela, nous avons sélectionné les variantes de GRAPH MOTIF que nous estimons, en plus du problème originel, être les plus à même d'assurer une bonne compréhension de ce paysage algorithmique. Ainsi, chacune des variantes proposées intègre une spécificité qui sera réutilisée dans la formalisation du problème MAXIMUM COLORFUL ARBORESCENCE.

2.2.1 Suppression des répétitions de couleurs dans le motif

Le problème COLORFUL GRAPH MOTIF est un sous-problème de GRAPH MOTIF dans lequel le motif a la particularité d'être *colorful*, *i.e.* de contenir au plus une fois chaque couleur de $\mathcal{C}$. Cette particularité implique que le motif est en fait égal à l'ensemble des couleurs du graphe, puisque le graphe peut être réduit en supprimant tous les sommets dont la couleur n'appartient pas au motif. On donne la définition de COLORFUL GRAPH MOTIF ci-dessous, puis on montre un exemple d'instance de COLORFUL GRAPH MOTIF dans la Figure 2.5.

COLORFUL GRAPH MOTIF

• **Instance**: Un graphe $G = (V, E)$, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$

• **Question**: Existe-t-il un ensemble de sommets $V' \subseteq V$ tel que $G[V']$ est connexe et $col(V') = \mathcal{C}$?

Puisque toutes les instances de COLORFUL GRAPH MOTIF sont aussi des instances de GRAPH MOTIF telles que $\mathcal{M} = \mathcal{C}$, GRAPH MOTIF peut être vu comme une généralisation de COLORFUL GRAPH MOTIF. Par conséquent, les résultats négatifs obtenus pour COLORFUL GRAPH MOTIF s'appliquent à GRAPH MOTIF, mais les résultats négatifs obtenus pour GRAPH MOTIF ne s'appliquent pas nécessairement à COLORFUL GRAPH MOTIF. De plus, tous les résultats positifs obtenus pour GRAPH MOTIF s'appliquent pour

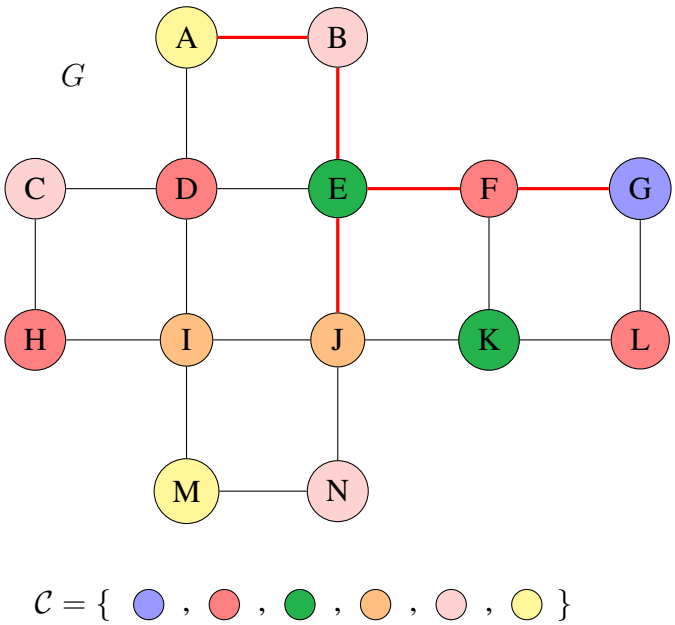


Figure 2.5 – Exemple (fictif) d'une instance positive de COLORFUL GRAPH MOTIF dans laquelle $V' = \{$A, B, E, F, G, J$\}$ est une solution possible.

COLORFUL GRAPH MOTIF. On verra dans la Section 2.3 qu'ajouter cette contrainte supplémentaire sur la nature du motif permet d'améliorer la complexité de certains algorithmes FPT et de trouver des kernels polynomiaux.

2.2.2 Maximisation de la cardinalité d'une sous-occurrence du motif

On rappelle qu'il existe de nombreuses bases de données contenant des réseaux biologiques, telles que KEGG [1] ou STRING [2]. Ces bases de données contiennent des réseaux PPI pour plus de deux mille organismes tels que l'humain [114, 106], la levure [109] ou bien encore la mouche drosophile [61]. En revanche, on estime que la majorité de toutes ces données biologiques comporte beaucoup de bruit [117, 1, 38], c'est-à-dire qu'il existe des arêtes manquantes ou erronnées dans la reconstruction des réseaux biologiques.

Pour contourner ce problème, de nombreux auteurs ont proposé des variantes de GRAPH MOTIF qui permettent d'*éditer* le motif, c'est-à-dire d'ajouter, de supprimer et/ou de substituer des couleurs du motif afin d'obtenir une occurrence [41, 104, 14]. Par exemple, on définit ci-dessous la version optimisation de GRAPH MOTIF, appelée MAX GRAPH MOTIF.

MAX GRAPH MOTIF

- **Instance**: Un graphe $G = (V, E)$, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$, un motif $\mathcal{M}$
- **Sortie**: Un ensemble de sommets $V' \subseteq V$ tel que $G[V']$ est connexe et $col(V') \subseteq \mathcal{M}$
- **Mesure**: $|V'|$

Si le graphe G ne contient pas d'occurrence de $\mathcal{M}$, le fait que MAX GRAPH MOTIF soit un problème d'optimisation permet tout de même d'obtenir une *sous-occurrence* V' de cardinalité maximale du motif – contrairement à GRAPH MOTIF qui est un problème de décision. Par exemple, on montre dans la Figure 2.6 un exemple d'instance de MAX GRAPH MOTIF dans laquelle il n'existe pas d'occurrence du motif.

1. https://www.genome.jp/kegg/pathway.html
2. https://string-db.org/

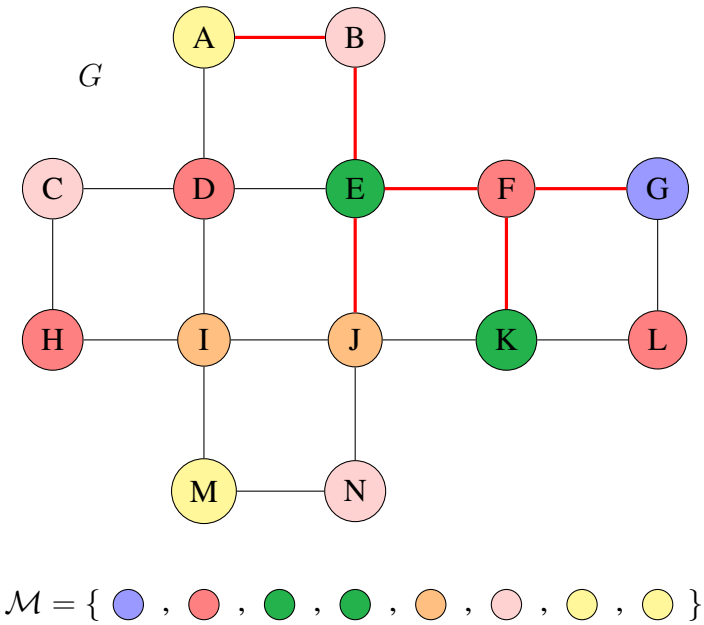


Figure 2.6 – Exemple (fictif) d'une instance de MAX GRAPH MOTIF dans laquelle il n'existe pas d'occurrence du motif $\mathcal{M}$. Ici, $V' = \{$A, B, E, F, G, J, K$\}$ est une sous-occurrence de cardinalité maximale de $\mathcal{M}$ qui ne contient pas deux fois la couleur jaune.

2.2.3 Ajout d'une fonction de pondération sur les arêtes

On rappelle que les réseaux biologiques contiennent beaucoup de bruit [117, 1, 38]. Au lieu d'autoriser le motif à pouvoir être édité comme dans la modélisation de MAX GRAPH MOTIF, une autre solution pour contourner ce problème consiste à pondérer les arêtes des réseaux PPI en fonction de la confiance accordée quant à l'existence de ces arêtes [26]. L'ajout de cette fonction de pondération nous permet de définir ci-dessous une nouvelle variante de GRAPH MOTIF appelée WEIGHTED GRAPH MOTIF. On donne un exemple d'instance de WEIGHTED GRAPH MOTIF dans la Figure 2.7.

WEIGHTED GRAPH MOTIF

- **Instance**: Un graphe $G = (V, E)$, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$, une fonction de poids $w : E \to \mathbb{R}$, un motif $\mathcal{M}$, un réel w^*
- **Question**: Existe-t-il un sous-arbre $T = (V_T, E_T)$ de G tel que $col(V_T) = \mathcal{M}$ de poids $\sum_{e \in E_T} w(e) \geq w^*$?

Le problème WEIGHTED GRAPH MOTIF est une généralisation du problème GRAPH MOTIF. En effet, on peut obtenir une instance de WEIGHTED GRAPH MOTIF en ajoutant un poids de 1 à chaque arête d'une instance de GRAPH MOTIF et en fixant $w^* = 0$. Par conséquent, tous les résultats de complexité négatifs liés à GRAPH MOTIF– et à COLORFUL GRAPH MOTIF– qui sont présentés dans la section suivante s'appliquent également à WEIGHTED GRAPH MOTIF.

2.3 Caractérisation des instances difficiles

Cette section se décline en deux parties. On distingue dans un premier temps les instances pour lesquelles GRAPH MOTIF– ou une variante de GRAPH MOTIF– se résout en temps polynomial de celles pour lesquelles le problème est NP-complet. On montre ensuite des résultats d'approximation qui se révèlent principalement négatifs.

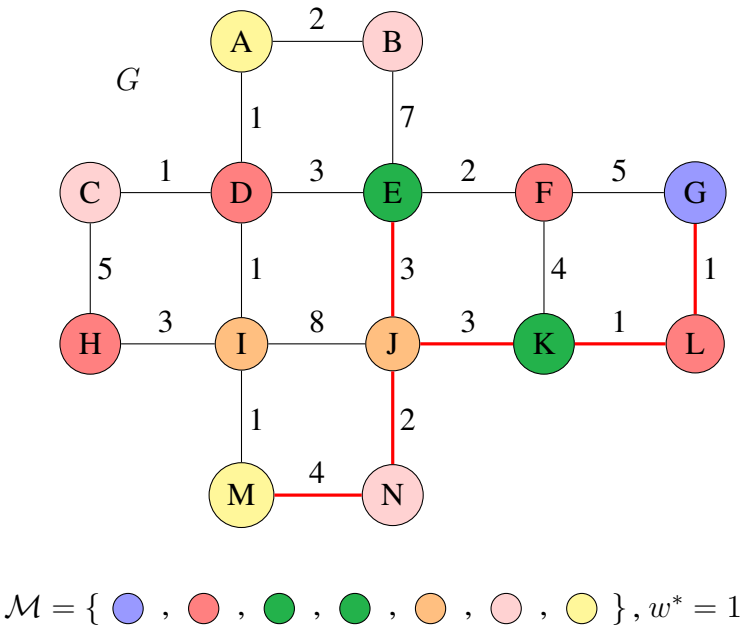


Figure 2.7 – Exemple (fictif) d'une instance positive de WEIGHTED GRAPH MOTIF dans laquelle le sous-arbre $T = (V_T, E_T)$ de G, où $V_T = \{$E, G, J, K, L, M, N$\}$ et $A_T = \{$(M,N), (N,J), (J,E), (J,K), (K,L), (L,G)$\}$, est une solution possible de poids 14.

Dans l'article introduisant GRAPH MOTIF, Lacroix *et al.* ont montré que le problème est NP-complet même si G est un arbre [79]. On montre la preuve de ce résultat ci-dessous.

Théorème 22. *([79])* GRAPH MOTIF *restreint aux arbres est* NP*-complet.*

Preuve. On effectue une *réduction polynomiale* (voir Définition 3, page 21) du problème NP-complet EXACT COVER BY 3-SETS, défini ci-dessous, vers GRAPH MOTIF.

EXACT COVER BY 3-SETS (X3C)

• **Instance**: Un entier q, un ensemble $\mathcal{U} = \{u_1, u_2, \ldots, u_{3q}\}$ d'éléments, un ensemble $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ de sous-ensembles de $\mathcal{U}$ contenant chacun trois éléments

• **Question**: Existe-t-il une *couverture exacte* de $\mathcal{U}$ dans $\mathcal{F}$, *i.e.* un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$ tel que chaque élément de $\mathcal{U}$ est contenu exactement une fois dans $\mathcal{S}$?

On décrit la création du graphe $G = (V, E)$ de l'instance de GRAPH MOTIF. On crée tout d'abord un sommet r ; pour tout sous-ensemble S_i appartenant à $\mathcal{F}$, on crée ensuite un ensemble de sommets $V_i = \{v_i, z_{i,1}, z_{i,2}, z_{i,3}\}$. Pour tout $i \in [p]$, il existe une arête (r, v_i) et une arête entre v_i et chaque autre sommet de V_i. La fonction de coloration est définie comme suit : on attribue une couleur unique c_r au sommet r ; on attribue une couleur $c_{\mathcal{F}}$ à tous les sommets de type v_i ; pour tout $j \in [3q]$, on attribue une couleur c_j à tous les sommets z_{i,j_1} tels que $j_1 = j$. Pour finir, on définit un motif $\mathcal{M}$ qui contient la couleur c_r, q fois la couleur $c_{\mathcal{F}}$ et, pour tout $j \in [3q]$, une fois la couleur c_j. Clairement, G est un arbre contenant $n = 4p + 1$ sommets et le motif $\mathcal{M}$ est de cardinalité $|\mathcal{M}| = 1 + q + 3q = 4q + 1$. L'instance de GRAPH MOTIF qu'on vient de décrire est donc de taille polynomiale en la taille de l'instance initiale de X3C. On montre un exemple de réduction d'une instance de X3C en instance de GRAPH MOTIF dans la Figure 2.8.

On montre maintenant qu'il existe une couverture exacte $\mathcal{S}$ de $\mathcal{U}$ dans $\mathcal{F}$ si et seulement si il existe une occurrence de $\mathcal{M}$ dans G.

($\Rightarrow$) On suppose qu'il existe une couverture exacte $\mathcal{S}$ de $\mathcal{U}$ dans $\mathcal{F}$. Soit $V' = \{r\} \cup \{V_i : i \in [p] \mid S_i \in \mathcal{S}\}$. Par construction de G, on note que $G[V']$ est connexe puisque r appartient à V' et qu'un sommet $z_{i,j}$

appartient à V' uniquement si le sommet v_i correspondant appartient également à V'. De plus, pour tout $j \in [3q]$, on observe que V' contient exactement un sommet de couleur c_j puisque $\mathcal{S}$ contient tous les éléments de $\mathcal{U}$. Enfin, on rappelle que chaque sous-ensemble de $\mathcal{F}$ contient 3 éléments de $\mathcal{U}$ et donc que $\mathcal{S}$ contient exactement q sous-ensembles de $\mathcal{F}$. Par conséquent, V' contient q sommets de couleurs $c_{\mathcal{F}}$ et V' est une occurrence de $\mathcal{M}$ dans G.

($\Leftarrow$) On suppose qu'il existe une occurrence V' de $\mathcal{M}$ dans G. Pour tout sommet de type v_i, on rappelle qu'il existe exactement trois arêtes de type $(v_i, z_{i,j})$ dans G. De plus, on rappelle que V' contient q sommets de type v_i et $3q$ sommets de type $z_{i,j}$. Par conséquent, il n'existe pas d'indice $i^* \in [p]$ tel que $v_{i^*} \in V'$ et $V_{i^*} \not\subset V'$. Puisque $\mathcal{M}$ contient un sommet de couleur c_j pour tout $j \in [3q]$, on en déduit donc que $\mathcal{S} = \{S_i : i \in [p] \mid v_i \in V'\}$ est une couverture exacte de $\mathcal{U}$ dans $\mathcal{F}$.

On vient de réaliser une réduction polynomiale de X3C vers GRAPH MOTIF. Puisque X3C est NP-dur, on en déduit que GRAPH MOTIF est également NP-dur (et même NP-complet). □

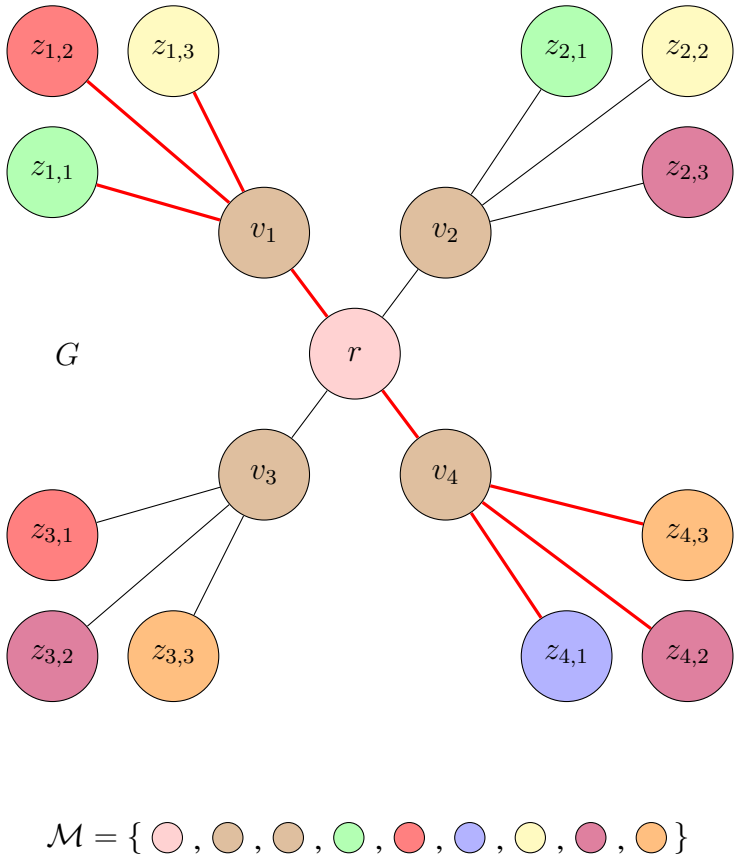


Figure 2.8 – Construction d'une instance de GRAPH MOTIF à partir d'une instance de X3C dans laquelle $\mathcal{U} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ et $\mathcal{F} = \{S_1 = \{u_1, u_2, u_4\}, S_2 = \{u_1, u_4, u_5\}, S_3 = \{u_2, u_5, u_6\}, S_4 = \{u_3, u_5, u_6\}\}$. Les couleurs vert clair, rouge, bleu, jaune, violet et orange des sommets de G correspondent respectivement aux indices $1, 2, 3, 4, 5, 6$ des variables de $\mathcal{U}$. On représente en rouge une solution de GRAPH MOTIF dans G.

Fellows *et al.* ont renforcé le Théorème 22 donné ci-dessus en prouvant – à l'aide d'une autre réduction à partir de X3C – que GRAPH MOTIF est NP-complet quand le motif ne contient que deux couleurs distinctes, et ce même dans les graphes *bipartis* de *degré* maximum 4 [51]. Les mêmes auteurs montrent également que GRAPH MOTIF peut être résolu en temps polynomial si à la fois la *treewidth* de G et le nombre de couleurs différentes dans le motif sont bornés [51]. Par ailleurs, il est également intéressant de

se demander si on peut résoudre GRAPH MOTIF en temps polynomial quand le motif est colorful. Malheureusement, COLORFUL GRAPH MOTIF est NP-complet dans les arbres de degré maximum 3 [51], et même dans les *comb-graphs* [32], une sous-catégorie d'arbres de degré maximum 3 présentée dans la Section 1.1.2 (page 15).

Dans la suite, on note que GRAPH MOTIF se résout en temps polynomial dans les chemins, puisque le nombre de sous-chemins – et donc d'occurrences potentielles d'un motif – d'un chemin est polynomial en la taille de celui-ci [51]. Ainsi, on a montré jusqu'ici que GRAPH MOTIF peut être résolu en temps polynomial dans n'importe quel graphe de degré maximum 2 avec n'importe quel motif, mais que ce problème restreint aux comb-graphs est NP-complet même si le motif est colorful. On propose donc ici d'étudier la complexité de GRAPH MOTIF dans des classes de graphes proches des chemins. Par exemple, on note qu'on peut résoudre GRAPH MOTIF en temps polynomial dans les *caterpillars* [3], des arbres qui deviennent des chemins si on supprime leurs *feuilles* (voir Section 1.1.1, page 13). En revanche, GRAPH MOTIF est NP-complet dans les *lobsters* [3], des arbres qui deviennent des caterpillars si on supprime leurs feuilles. De plus, pour affiner encore plus la frontière entre les instances pour lesquelles GRAPH MOTIF se résout en temps polynomial et les autres, Bonnet et Sikora ont montré que COLORFUL GRAPH MOTIF est NP-complet même dans les graphes où il suffit de retirer un sommet pour les transformer en collection de chemins disjoints [23].

On propose maintenant de tracer la frontière entre les instances pour lesquelles GRAPH MOTIF se résout en temps polynomial et les autres en fonction du *diamètre* d'un graphe, *i.e.* de la plus grande distance entre deux sommets de ce graphe. En effet, GRAPH MOTIF peut trivialement être résolu en temps polynomial dans les *cliques* – des graphes de diamètre 1. Pour cela, il suffit en effet de trouver un sommet $v \in V$ dont la couleur appartient au motif et de regarder si $\mathcal{M} \setminus \{col(v)\}$ appartient au multi-ensemble de couleurs du voisinage de v. Le problème GRAPH MOTIF est cependant NP-complet dans les graphes de *diamètre* 2 et même dans les *superstars*, une sous-classe de lobsters [3]. De plus, Ganian a prouvé que GRAPH MOTIF est NP-complet dans les *split graphs* [58, 59], un ensemble de graphes qui peuvent être partitionnés en une clique et un ensemble indépendant de sommets. Il existe toutefois une catégorie de split graphs, appelée *threshold graphs*, pour laquelle GRAPH MOTIF se résout en temps polynomial. Les threshold graphs sont à la fois des split graphs et des *cographes*, des graphes ne contenant pas de chemin de longueur 4 comme *sous-graphe induit*. Das *et al.* ont montré que GRAPH MOTIF peut être résolu en temps polynomial dans les cographes, et donc dans les threshold graphs [35].

Pour finir, on montre que GRAPH MOTIF peut se résoudre en temps polynomial dans un arbre si chaque couleur apparaît au plus deux fois dans celui-ci [51].

Théorème 23. *([51])* GRAPH MOTIF *restreint aux arbres peut être résolu en temps polynomial si chaque couleur de $\mathcal{C}$ apparaît au plus deux fois dans G.*

Preuve. Dans la suite, on construit une instance de 2-SAT, qui est défini ci-dessous, à partir d'une instance de GRAPH MOTIF.

2-SAT

• **Instance**: Un ensemble de variables booléennes $X = \{x_1, .., x_p\}$, une formule ϕ sur un ensemble $C = \{C_1, .., C_q\}$ de clauses de taille 2 contenant des variables de X

• **Question**: Existe-t-il une affectation $\beta : X \to \{\texttt{vrai}, \texttt{faux}\}$ de chaque $x_i \in X$ qui satisfait ϕ ?

On explique comment construire ϕ en trois temps. Dans un premier temps, on crée une variable x_c pour chaque couleur $c \in \mathcal{C}$. On rappelle que G contient au plus deux sommets de couleur c. Ainsi, pour toute paire de sommets $v_1, v_2 \in V$ de couleur c, on note $l(v_1) = x_c$ et $l(v_2) = \overline{x_c}$ les deux littéraux distincts

associés à v_1 et v_2. Si $v \in V$ est le seul sommet de couleur c dans G, alors on fixe $l(v) = x_c$ et $\beta(x_c)$ sera égale à `vrai`.

Dans un second temps, on désigne arbitrairement un sommet $r \in V$ comme racine de G et on crée une clause contenant uniquement le littéral associé à r. On considère ainsi que r appartient à une occurrence de $\mathcal{M}$ dans G. S'il s'avère par la suite que ϕ ne peut pas être satisfaite, alors on construira une autre formule ϕ' en fixant comme racine l'autre sommet de couleur $\mathrm{col}(r)$ dans G. Si aucune de ces deux formules ne peut être satisfaite, alors il n'existe aucune occurrence de $\mathcal{M}$ dans G.

Dans un troisième temps, pour chaque arête $(u, v) \in E(G)$ telle que $d(r, u) < d(r, v)$, on crée une clause $(l(u) \vee \overline{l(v)})$ qui représente l'implication $l(v) \Rightarrow l(u)$. On rappelle que G est un arbre et donc que u appartient à l'unique chemin entre r et v. Puisque r appartient à toute solution V' de GRAPH MOTIF dans G, on observe que v appartient à V' seulement si u appartient également à V' – et donc $l(v) \Rightarrow l(u)$.

On présente la construction d'une formule ϕ dans la Figure 2.9. Clairement, il existe une occurrence de $\mathcal{M} = \mathcal{C}$ dans G si et seulement si ϕ (ou ϕ') peut être satisfaite par une affectation β des variables de X. Puisque 2-SAT appartient à P, le problème COLORFUL GRAPH MOTIF restreint aux arbres peut donc être résolu en temps polynomial si chaque couleur de $\mathcal{C}$ apparaît au plus deux fois dans G.

□

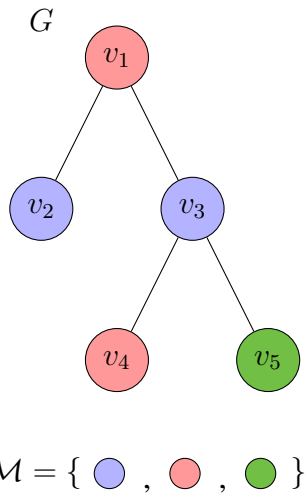


Figure 2.9 – Construction d'une instance de 2-SAT à partir d'une instance de GRAPH MOTIF. On note x_{rouge}, x_{bleu} et x_{vert} les trois variables de X créées à partir de $\mathcal{C}$. On suppose que $l(v_1) = x_{rouge}$, $l(v_2) = x_{bleu}$, $l(v_3) = \overline{x_{bleu}}$, $l(v_4) = \overline{x_{rouge}}$ et $l(v_5) = x_{vert}$. Dans cet exemple, on choisit arbitrairement le sommet v_1 comme racine de G. A partir des arêtes de G, on construit ensuite la formule $\phi = (l(v_1)) \wedge (l(v_1) \vee \overline{l(v_2)}) \wedge (l(v_1) \vee \overline{l(v_3)}) \wedge (l(v_3) \vee \overline{l(v_4)}) \wedge (l(v_3) \vee \overline{l(v_5)})$, ce qui donne $\phi = (x_{rouge}) \wedge (x_{rouge} \vee \overline{x_{bleu}}) \wedge (x_{rouge} \vee x_{bleu}) \wedge (x_{bleu} \vee \overline{x_{rouge}}) \wedge (x_{bleu} \vee \overline{x_{vert}})$. L'affectation $\beta = \{x_{rouge} = \{\mathtt{vrai}\}, x_{bleu} = \{\mathtt{faux}\}, x_{vert} = \{\mathtt{vrai}\}\}$ correspond à une solution $V' = \{v_1, v_3, v_5\}$ de GRAPH MOTIF – on rappelle $\beta(x_{vert})$ est nécessairement égale à `vrai` puisque la couleur verte apparait uniquement une fois dans G.

Dans cette section, on a observé qu'il existe beaucoup de classes de graphes pour lesquelles GRAPH MOTIF est NP-complet. On montre maintenant des résultats d'approximation (voir Section 1.2.3, page 22) obtenus pour MAX GRAPH MOTIF afin de contourner la difficulté du problème. Malheureusement, ces résultats sont principalement négatifs. Pour commencer, Dondi *et al.* montrent que MAX GRAPH MOTIF restreint aux arbres est APX-dur même quand le motif est *colorful* [40]. Les mêmes auteurs prouvent également qu'il n'existe pas d'algorithme d'approximation en temps polynomial pour MAX GRAPH MOTIF dans les arbres avec un ratio $2^{\log^{\delta} n}$, $\forall \delta < 1$ sauf si NP $\subseteq$ DTIME$[2^{\mathrm{poly}\log n}]$. En fixant δ à une valeur proche

de 1, on observe donc qu'on ne peut pas espérer un algorithme d'approximation avec un meilleur ratio d'approximation que n pour résoudre MAX GRAPH MOTIF. Enfin, Rizzi *et al.* ont montré qu'il n'existe pas d'algorithme d'approximation en temps polynomial pour résoudre MAX GRAPH MOTIF avec un ratio $n^{\frac{1}{3}-\epsilon}$, où $\epsilon > 0$, même dans les arbres dans lesquels chaque couleur apparaît au plus deux fois et avec un motif colorful [104].

2.4 Résultats de complexité paramétrée

On rappelle que GRAPH MOTIF est NP-complet même dans des instances très restreintes comme des *comb-graphs* [32] ou des *superstars* [3] et que MAX GRAPH MOTIF est hautement inapproximable même dans les arbres [40, 104]. Par conséquent, la complexité des algorithmes qui résolvent GRAPH MOTIF est exponentielle – sauf si P = NP. Par exemple, Dondi *et al.* ont montré via des *algorithmes de branchement* (voir Section 1.3.1, page 25) que MAX GRAPH MOTIF peut être résolu en temps $\mathcal{O}^*(1.62^n)$, où n est le nombre de sommets du graphe d'entrée G, et même en temps $\mathcal{O}^*(1.33^n)$ quand le motif est colorful [40].

Même si les deux algorithmes évoqués ci-dessus sont nettement plus efficaces qu'un algorithme trivial en $\mathcal{O}^*(2^n)$, le problème GRAPH MOTIF peut cependant ne pas pouvoir être résolu avec des temps d'exécution raisonnables si la taille de l'instance est trop élevée. Ainsi, on présente dans cette section des algorithmes de *complexité paramétrée* (voir Section 1.2.4, page 23), *i.e.* des algorithmes dont la partie exponentielle de la complexité est confinée à un paramètre qu'on espère petit en pratique. Il existe une multitude de paramètres liés à la structure du graphe [23], mais nous nous intéressons dans cette section uniquement aux deux paramètres suivants : la taille du motif $k = |\mathcal{M}|$ et le nombre de sommets n'appartenant pas à la solution $\ell = n - |\mathcal{M}|$. Le premier paramètre est considéré comme le paramètre naturel de GRAPH MOTIF et a concentré l'essentiel des resultats de complexité paramétrée pour GRAPH MOTIF. On considère également important le fait de présenter les résultats de complexité paramétrée liés à ℓ afin de pouvoir les comparer avec ceux obtenus dans le Chapitre 4 sur un problème similaire à GRAPH MOTIF relativement à deux paramètres similaires à ℓ. Dans ce chapitre, pour chacun des deux paramètres k et ℓ, on présente des algorithmes FPT ainsi que des résultats – positifs et négatifs – sur l'existence de kernels. La plupart des techniques utilisées pour obtenir les résultats décrits ci-dessous ont été présentées dans la Section 1.3, à partir de la page 25.

2.4.1 Paramétrer par la taille du motif

On rappelle que le problème GRAPH MOTIF est motivé par la recherche de sous-réseaux biologiques supposés petits à l'intérieur de plus grands réseaux. Dès lors, il peut sembler légitime de chercher s'il existe des algorithmes FPT qui soient paramétrés par la taille de ces sous-réseaux, et donc par $k = |\mathcal{M}|$. Dans cette section, on montre ainsi comment certaines techniques algorithmiques présentées dans le Chapitre 1 ont permis d'obtenir des résultats de complexité paramétrée pour GRAPH MOTIF et ses variantes relativement à k.

Après avoir montré que GRAPH MOTIF est NP-dur même dans les arbres [79], Lacroix *et al.* ont déterminé le premier algorithme FPT pour GRAPH MOTIF relativement à k dans le cas où G est un arbre. La première phase de l'algorithme consiste à générer exhaustivement toutes les *topologies* du motif $\mathcal{M}$, c'est-à-dire tous les sous-arbres colorés contenant k sommets et dont le multi-ensemble de couleurs des sommets est égal à $\mathcal{M}$. La deuxième phase consiste ensuite à appliquer un algorithme d'isomorphisme de graphe à chaque topologie dans le graphe G. On note que l'algorithme d'isomorphisme de graphe possède une complexité polynomiale puisque G est un arbre – ce qui n'est plus le cas si G n'est pas contraint [60]. De plus, la formule de Cayley indique qu'il existe au plus k^{k-2} topologies de $\mathcal{M}$ pour tout $k > 1$. Par conséquent, la complexité en temps de l'algorithme FPT proposé par Lacroix *et al.* est exponentielle uniquement en k.

Fellows *et al.* ont par la suite montré que GRAPH MOTIF admet des algorithmes FPT relativement à k dans toutes les structures de graphes [51]. Les auteurs appliquent des techniques de programmation dynamique et de *color-coding* (voir Section 1.3.3, page 26), ainsi qu'une famille parfaite de fonctions de hachage pour dérandomiser l'algorithme après l'usage du color-coding. La forte complexité de celui-ci – $\mathcal{O}^*(87^k)$ en temps et $\mathcal{O}^*(4^k)$ en espace – est anecdotique puisque le but premier de Fellows *et al.* était de prouver que GRAPH MOTIF admet des algorithmes FPT par rapport à k. On note que l'utilisation du color-coding et de la programmation dynamique a également permis de montrer que MAX GRAPH MOTIF admet des algorithmes FPT relativement à k en temps $\mathcal{O}^*(32^k 4^{\mathcal{O}(k)})$ dans les graphes non-contraints ainsi qu'en temps $\mathcal{O}^*(2^k 2^{\mathcal{O}(k)})$ dans les arbres [40].

L'algorithme de Scott *et al.* [111] pour trouver des sous-arbres colorful de cardinalité k dans un graphe a indépendamment permis à Betzler *et al.* [9] et à Bruckner *et al.* [26] de prouver le Théorème 24 ci-dessous.

Théorème 24. *([9, 26])* COLORFUL GRAPH MOTIF *peut se résoudre avec un algorithme* FPT *probabiliste de complexité temporelle* $\mathcal{O}^*(3^k)$ *et de complexité spatiale* $\mathcal{O}^*(2^k)$.

Preuve. On crée une table de programmation dynamique binaire $\mathcal{B}[v, \mathcal{M}']$, pour tout $v \in V$ et $\mathcal{M}' \subseteq \mathcal{C}$. Une case $\mathcal{B}[v, \mathcal{M}']$ est égale à `vrai` si et seulement si il existe un ensemble de sommets $V' \subseteq V$ contenant v tel que $G[V']$ est connexe et $col(V') = \mathcal{M}'$; on note $T(v, \mathcal{M}')$ l'arbre couvrant de $G[V']$ dans la suite. Pour tout $c \in \mathcal{C}$, on note $\mathcal{B}[v, \{c\}] =$ `vrai` si $\mathrm{col}(v) = c$ et $\mathcal{B}[v, \{c\}] =$ `faux` sinon. Pour tout $\mathcal{M}' \subseteq \mathcal{C}$ tel que $|\mathcal{M}'| > 1$, on calcule ensuite le contenu d'une case $\mathcal{B}[v, \mathcal{M}']$ comme suit :

$$\mathcal{B}[v, \mathcal{M}'] = \bigvee_{\substack{u \in N(v), \\ \mathcal{M}'_1 \uplus \mathcal{M}'_2 = \mathcal{M}', \\ \mathrm{col}(v) \in \mathcal{M}'_1, \mathrm{col}(u) \in \mathcal{M}'_2}} \mathcal{B}[v, \mathcal{M}'_1] \wedge \mathcal{B}[u, \mathcal{M}'_2]$$

Concrètement, il existe un arbre $T(v, \mathcal{M}')$ si et seulement si il existe deux arbres $T(v, \mathcal{M}'_1)$ et $T(u, \mathcal{M}'_2)$ telles que (i) $\mathcal{M}'$ est partitionné en deux sous-ensembles $\mathcal{M}'_1$ et $\mathcal{M}'_2$ et (ii) il existe une arête (u, v) dans G. On calcule exhaustivement les entrées $\mathcal{B}[v, \mathcal{M}']$ pour tout $v \in V$ et tout $\mathcal{M}' \subseteq \mathcal{C}$. Ainsi, le problème COLORFUL GRAPH MOTIF admet une solution dans G si et seulement si il existe une case $\mathcal{B}[v, \mathcal{C}]$ égale à `vrai`. On précise que l'algorithme proposé est correct puisqu'on a calculé exhaustivement toutes les cases de $\mathcal{B}$.

On calcule maintenant la complexité de cet algorithme. Pour chaque entrée $\mathcal{B}[v, \mathcal{M}']$, on note qu'une couleur de $\mathcal{M}$ appartient soit à $\mathcal{M}'_1$, soit à $\mathcal{M}'_2$ ou soit à $\mathcal{M} \setminus \mathcal{M}'$. La complexité temporelle de l'algorithme est donc de $\mathcal{O}^*(3^{|\mathcal{C}|})$. Enfin, la complexité spatiale de l'algorithme est de $\mathcal{O}^*(2^{|\mathcal{C}|})$ puisqu'il existe une entrée $\mathcal{B}[v, \mathcal{C}]$ pour tout $\mathcal{M}' \subseteq \mathcal{C}$ – et tout $v \in V$. Un exemple d'application de l'algorithme de programmation dynamique présenté dans ce théorème est donné dans l'Exemple 25. □

Exemple 25. Illustration de l'algorithme présenté dans le Théorème 24.

Dans cet exemple, on montre comment calculer les cases de la table de programmation dynamique $\mathcal{B}[v, \mathcal{M}']$, où $v \in V$ et $\mathcal{M}' \subseteq \mathcal{C}$, à partir de l'instance de COLORFUL GRAPH MOTIF présentée dans la Figure 2.10. Tout d'abord, toutes les entrées $\mathcal{B}[v, \mathcal{M}']$ telles que (i) $\mathcal{M}' \subseteq \mathcal{C}$ et (ii) $|\mathcal{M}'| = 1$ sont initialisées à `faux` à l'exception des entrées suivantes :

$$\mathcal{B}[v_1, \{\bullet\}] = \texttt{vrai}$$
$$\mathcal{B}[v_2, \{\bullet\}] = \texttt{vrai}$$
$$\mathcal{B}[v_3, \{\bullet\}] = \texttt{vrai}$$
$$\mathcal{B}[v_4, \{\bullet\}] = \texttt{vrai}$$

On calcule ensuite toutes les entrées $\mathcal{B}[v, \mathcal{M}']$ telles que (i) $\mathcal{M}' \subseteq \mathcal{C}$ et (ii) $|\mathcal{M}'| = 2$. Par exemple :

$$\begin{aligned}\mathcal{B}[v_1, \{\bullet, \bullet\}] &= (\mathcal{B}[v_1, \{\bullet\}] \wedge \mathcal{B}[v_2, \{\bullet\}]) \\ &= (\texttt{vrai} \wedge \texttt{vrai}) \\ &= \texttt{vrai}\end{aligned}$$

Enfin, on calcule toutes les entrées $\mathcal{B}[v, \mathcal{M}']$ telles que (i) $\mathcal{M}' \subseteq \mathcal{C}$ et (ii) $|\mathcal{M}'| = 3$. Par exemple :

$$\begin{aligned}\mathcal{B}[v_1, \{\bullet, \bullet, \bullet\}] &= (\mathcal{B}[v_1, \{\bullet\}] \wedge \mathcal{B}[v_2, \{\bullet, \bullet\}]) \\ &\quad \vee (\mathcal{B}[v_1, \{\bullet, \bullet\}] \wedge \mathcal{B}[v_2, \{\bullet\}]) \\ &\quad \vee (\mathcal{B}[v_1, \{\bullet\}] \wedge \mathcal{B}[v_4, \{\bullet, \bullet\}]) \\ &\quad \vee (\mathcal{B}[v_1, \{\bullet, \bullet\}] \wedge \mathcal{B}[v_4, \{\bullet\}]) \\ &= (\texttt{vrai} \wedge \texttt{vrai}) \vee (\texttt{vrai} \wedge \texttt{vrai}) \vee (\texttt{vrai} \wedge \texttt{faux}) \vee (\texttt{vrai} \wedge \texttt{vrai}) \\ &= \texttt{vrai}\end{aligned}$$

On note que $\mathcal{B}[v_1, \mathcal{C}] = \texttt{vrai}$. Par conséquent, il existe une occurrence de $\mathcal{M} = \mathcal{C}$ dans G – qu'on peut retrouver en utilisant du backtracking.

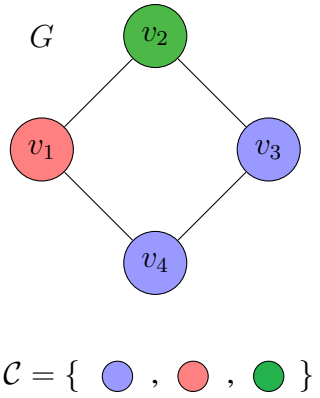


Figure 2.10 – Instance de COLORFUL GRAPH MOTIF sur laquelle on applique l'algorithme du Théorème 24 dans l'Exemple 25.

Betzler *et al.* ont modifié l'algorithme présenté dans le Théorème 24 afin d'obtenir un algorithme probabiliste pour résoudre GRAPH MOTIF avec une complexité temporelle de $\mathcal{O}^*(4.32^k)$ et une complexité spatiale de $\mathcal{O}^*(2^k)$ [9]. Pour cela, Betzler *et al.* utilisent du color-coding sur les répétitions de couleurs contenues dans le motif $\mathcal{M}$ de la manière suivante : si $\mathcal{M}$ contient une couleur c en deux exemplaires, on crée deux couleurs c_a et c_b qui remplacent les deux répétitions de c dans $\mathcal{M}$, puis on attribue via du color-coding une des deux nouvelles couleurs à tous les sommets de couleur c dans G. On applique ensuite l'algorithme présenté dans le Théorème 24 pour résoudre l'instance de COLORFUL GRAPH MOTIF qu'on vient de créer. On note que Bruckner *et al.* ont aussi trouvé un algorithme en temps $\mathcal{O}^*(k!3^k)$ pour résoudre GRAPH MOTIF en utilisant du color-coding [26]. Les auteurs précisent que leur algorithme est plus efficace dans certaines instances que celui de Betzler *et al.* après l'utilisation de règles de réduction. De plus, celui-ci leur permet de résoudre WEIGHTED GRAPH MOTIF avec la même complexité et sans aucune restriction sur les poids. Afin de mieux comprendre la complexité de WEIGHTED GRAPH MOTIF, on précise qu'il n'existe qu'un seul algorithme FPT relativement à k dont la complexité soit meilleure que celle de l'algorithme de Bruckner *et al.* quand les poids sont des réels. Celui-ci a été élaboré par Pinter *et al.* [98] et sa complexité en temps est de $\mathcal{O}^*(20.25^{k+\mathcal{O}(\log^2 k)})$.

Guillemot et Sikora ont montré que GRAPH MOTIF peut être résolu par un algorithme FPT probabiliste en temps $\mathcal{O}^*(4^k)$ et que COLORFUL GRAPH MOTIF peut être résolu par un algorithme FPT probabiliste en

temps $\mathcal{O}^*(2^k)$ avec, pour la première fois, une complexité polynomiale en espace dans les deux cas [62]. Pour cela, ils utilisent une technique de *détection de monômes multilinéaires* (voir Section 1.3.3, page 28) : Guillemot et Sikora créent tout d'abord une instance de k-MLD à partir d'une instance de GRAPH MOTIF (resp. COLORFUL GRAPH MOTIF), puis ils appliquent le Théorème 10 pour déterminer la complexité d'un algorithme qui résout l'instance de k-MLD créée. Dans l'instance de k-MLD créée à partir d'une instance de COLORFUL GRAPH MOTIF, les variables du polynôme créé correspondent aux couleurs des sommets de G. Ainsi, un monôme de degré k correspond au multi-ensemble de couleurs d'un sous-arbre de k sommets. Par construction, ce monôme est multilinéaire si et seulement si le sous-arbre est colorful. Par conséquent, l'instance de COLORFUL GRAPH MOTIF contient une occurrence du motif si et seulement si le polynôme construit à partir de cette instance contient un monôme multilinéaire de degré $k = |\mathcal{M}|$. Par ailleurs, Guillemot *et al.* ont également proposé un algorithme FPT probabiliste qui utilise la technique de détection de monômes multilinéaires afin de résoudre WEIGHTED GRAPH MOTIF avec des poids dans $\mathbb{N}^*$ en temps $\mathcal{O}^*(4^k)$. L'algorithme est cependant *pseudo-polynomial* par rapport aux poids, ce qui signifie que sa complexité est polynomiale relativement à la *valeur* du poids maximum d'une arête, mais pas relativement à la *taille* (voir Section 1.2.1, page 18) de cette valeur en mémoire.

Suite à l'article de Guillemot *et al.*, Koutis a montré un nouvel algorithme probabiliste en temps $\mathcal{O}^*(2.54^k)$ pour résoudre GRAPH MOTIF en réduisant le problème à une variante de k-MLD appelée CONSTRAINED k-MLD [74]. Björklund *et al.* [14] et Pinter *et al.* [100] ont ensuite indépendamment proposé un algorithme probabiliste en temps $\mathcal{O}^*(2^k)$ pour résoudre GRAPH MOTIF en utilisant la technique des *narrow sieves* (voir Section 1.3.3, page 30). Björklund *et al.* ont également démontré que ces algorithmes sont "optimaux" [14] sous l'hypothèse de complexité Set Cover Conjecture [31]. Enfin, Pinter *et al.* ont prouvé que WEIGHTED GRAPH MOTIF pouvait être résolu en temps $\mathcal{O}^*(2^k)$ avec des poids dans $\mathbb{N}^*$ [100], puis avec des poids dans $\mathbb{Z}$ [97].

Les algorithmes FPT optimaux proposés ci-dessus pour résoudre GRAPH MOTIF sont des algorithmes probabilistes. Il existe toujours un écart de complexité entre le meilleur algorithme FPT probabiliste (en temps $\mathcal{O}^*(2^k)$ [14, 100]) et le meilleur algorithme FPT déterministe pour GRAPH MOTIF. Ce dernier algorithme a été obtenu en utilisant la technique des *familles réprésentatives* (voir Section 1.3.3, page 32) et possède une complexité temporelle de $\mathcal{O}^*(5.18^k)$ [99]. On précise en revanche qu'il n'existe pas d'écart de complexité entre le meilleur algorithme FPT probabiliste et le meilleur algorithme FPT déterministe pour COLORFUL GRAPH MOTIF. En effet, Cygan *et al.* ont proposé un algorithme déterministe "optimal" – sous l'hypothèse de complexité Set Cover Conjecture [31] – pour COLORFUL GRAPH MOTIF qui possède une complexité temporelle de $\mathcal{O}^*(2^k)$ et une complexité spatiale polynomiale [32].

On résume maintenant les principaux algorithmes de complexité paramétrée respectivement pour GRAPH MOTIF et COLORFUL GRAPH MOTIF relativement à k dans les Tableaux 2.1 et 2.2.

Complexité en temps	Complexité spatiale polynomiale ?	Nature	Source
$\mathcal{O}^*(87^k)$	NON	Déterministe	[51]
$\mathcal{O}^*(4.32^k)$	NON	Randomisé	[9]
$\mathcal{O}^*(4^k)$	OUI	Randomisé	[62]
$\mathcal{O}^*(2.54^k)$	OUI	Randomisé	[74]
$\mathcal{O}^*(2^k)$	OUI	Randomisé	[14, 100]
$\mathcal{O}^*(5.18^k)$	NON	Déterministe	[99]

Tableau 2.1 – Résultats principaux de complexité paramétrée pour GRAPH MOTIF relativement à $k = |\mathcal{M}|$. Björklund *et al.* ont montré qu'il n'existe pas d'algorithme en temps $\mathcal{O}((2-\epsilon)^k)$, quelque soit $\epsilon > 0$, sous l'hypothèse Set Cover Conjecture [14].

Complexité en temps	Complexité spatiale polynomiale ?	Nature	Source
$\mathcal{O}^*(3^k)$	NON	Déterministe	[9, 26]
$\mathcal{O}^*(2^k)$	OUI	Randomisé	[62]
$\mathcal{O}^*(2^k)$	OUI	Déterministe	[32]

Tableau 2.2 – Résultats principaux de complexité paramétrée pour COLORFUL GRAPH MOTIF relativement à $k = |\mathcal{M}|$.

Ce dernier paragraphe aborde la recherche de kernels polynomiaux relativement à k pour COLORFUL GRAPH MOTIF. Ambalath *et al.* ont d'abord montré qu'il n'existe pas de kernel polynomial pour COLORFUL GRAPH MOTIF, même dans les *comb-graphs*, sauf si $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$ [3]. Ils ont cependant montré que COLORFUL GRAPH MOTIF dans les comb-graphs admet $|V(G)| = n$ kernels, chacun de taille $O(k^2)$. Les auteurs précisent qu'ils connaissent très peu d'autres problèmes admettant non pas un mais plusieurs kernels polynomiaux. Ce dernier résultat leur permet également de montrer qu'une variante de COLORFUL GRAPH MOTIF, dans laquelle il existe un sommet qui doit nécessairement appartenir à la solution, possède un kernel de taille $O(k^2)$ dans les comb-graphs [3]. Finalement, Ambalath *et al.* ont montré que cette variante de COLORFUL GRAPH MOTIF n'admet cependant pas de kernel polynomial dans les arbres binaires.

2.4.2 Paramétrer par le dual

On s'intéresse ici à $\ell = n - |\mathcal{M}|$, le paramètre dual de GRAPH MOTIF qui représente le nombre de sommets qui ne sont *pas* dans la solution. A priori, ce paramètre ne semble pas très intéressant puisque les motifs recherchés sont généralement de petite taille. Pour justifier l'intérêt de leurs travaux sur le paramètre ℓ, Fertin et Komusiewicz montrent qu'il existe d'autres problèmes pour lesquels les algorithmes FPT par rapport au paramètre dual donnent les meilleurs résultats en pratique [64]. En outre, Fertin et Komusiewicz [55] utilisent des *algorithmes de branchement* (voir Section 1.3.1, page 25), dont le temps d'exécution peut-être grandement réduit si on applique des règles de réduction sur l'instance initiale de GRAPH MOTIF. En effet, si par exemple le graphe d'entrée est composé de plusieurs composantes connexes après application des règles de réduction, on peut essayer de trouver une occurrence du motif dans chacune de ces composantes connexes séparément. Si n' désigne le nombre de sommets d'une composante connexe, on note que ℓ devient égal à $n' - |\mathcal{M}|$ et peut par conséquent devenir très petit dans des instances contenant un grand nombre de couleurs [26, 10, 55].

Dans la suite, on présente des algorithmes FPT et des kernels polynomiaux relativement à ℓ. On commence avec le résultat suivant pour COLORFUL GRAPH MOTIF.

Théorème 26. *([10])* COLORFUL GRAPH MOTIF *peut être résolu en temps* $\mathcal{O}^*(2^\ell)$.

Preuve. On propose un algorithme de branchement récursif basé sur les couleurs de G. Pour cela, on définit $S = V$. Si S n'est pas colorful, on considère $u, v \in S$ tel que $\mathrm{col}(u) = \mathrm{col}(v)$ et on branche récursivement sur deux cas : on supprime soit u soit v de S. Maintenant, pour tout S correspondant à une feuille de l'arbre de recherche induit T_S, il suffit de déterminer si $G[S]$ est connexe pour savoir si S est une occurrence de $\mathcal{M}$ dans G.

On observe que l'algorithme décrit ci-dessus est correct puisqu'il existe un ensemble S correspondant à une feuille de T_S pour chaque sous-ensemble colorful de $|\mathcal{C}|$ sommets de G. Puisque chaque étape de l'algorithme est effectuée en temps polynomial, la complexité en temps de celui-ci est exponentielle uniquement par rapport au nombre de nœuds de T_S. Puisque T_S est un arbre binaire de hauteur $\ell = n - |\mathcal{M}|$, son nombre de nœuds ne peut pas excéder 2^ℓ et en conséquence la complexité temporelle de l'algorithme proposé est de $\mathcal{O}^*(2^\ell)$. On montre une application de l'algorithme dans la Figure 2.11. □

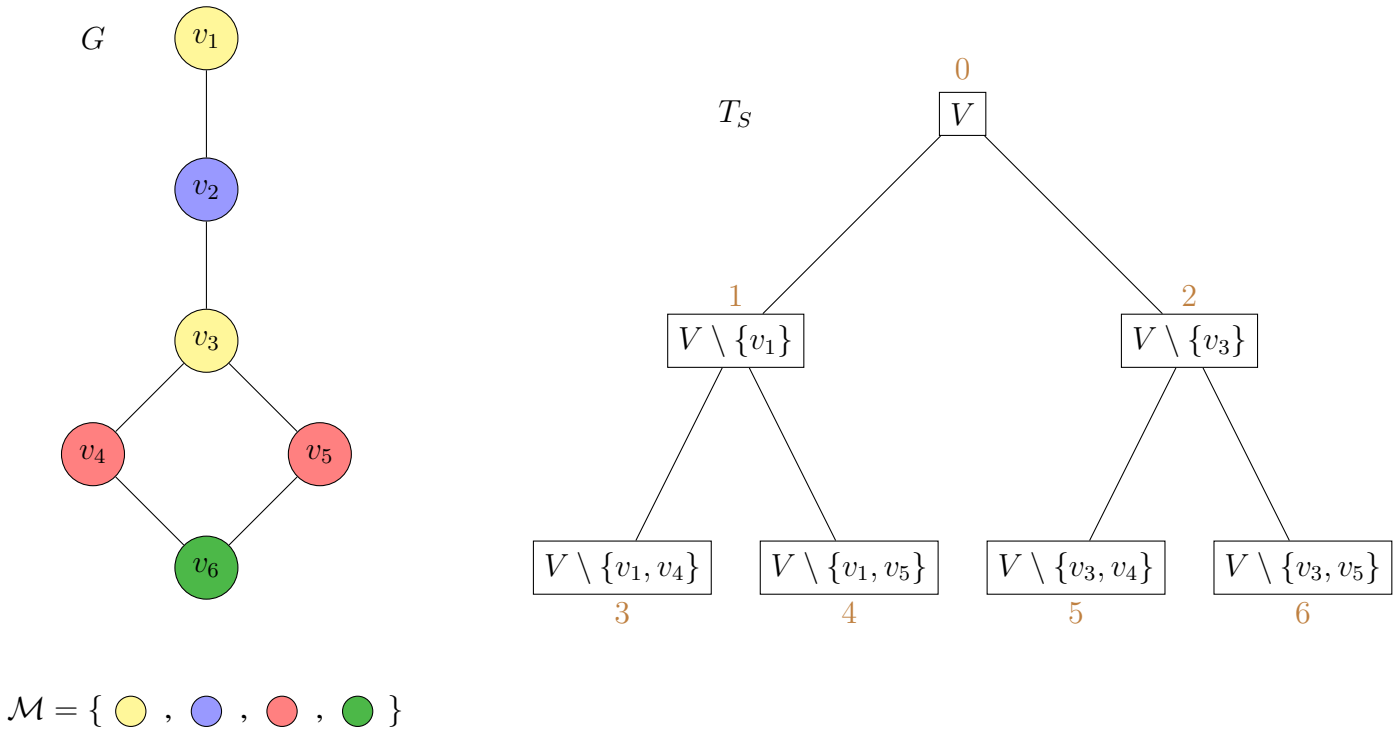


Figure 2.11 – Illustration de l'algorithme de branchement décrit dans le Théorème 26 sur une instance de COLORFUL GRAPH MOTIF (à gauche). Pour faciliter les explications, on donne des noms en marron aux nœuds de l'arbre de recherche induit T_S (à droite). On observe que v_1 et v_3 sont de même couleur et appartiennent tous les deux à l'ensemble S du nœud 0. Ainsi, on crée deux nœuds 1 et 2 à partir du nœud 0, dans lesquels on supprime respectivement v_1 et v_3 de S. Puisque v_4 et v_5 sont de même couleur, on applique également cette règle de branchement pour obtenir les nœuds 3 et 4 à partir du nœud 1, et pour obtenir les nœuds 5 et 6 à partir du nœud 2. Les nœuds 3, 4, 5 et 6 sont des feuilles de T_S dont l'ensemble S correspondant est colorful. On observe que le graphe $G[S]$ est connexe pour les ensembles S des nœuds 3 et 4. Ainsi, les ensembles qui correspondent aux nœuds 3 et 4 sont des occurrences du motif $\mathcal{M}$ dans G.

Fertin et Komusiewicz ont amélioré la complexité de l'algorithme présenté ci-dessus dans les arbres en montrant que COLORFUL GRAPH MOTIF peut être résolu en temps $\mathcal{O}^*(\sqrt{2}^\ell)$ [55] dans ce type d'instances.

Théorème 27. *([55])* COLORFUL GRAPH MOTIF *restreint aux arbres peut être résolu en temps* $\mathcal{O}^*(\sqrt{2}^\ell)$.

Preuve. Tout comme l'algorithme présenté dans le Théorème 26, on décrit un algorithme de branchement récursif basé sur les couleurs de G et on initialise S à V. On applique cependant une règle de branchement différente afin que le graphe $G[S]$ soit connexe pour tout ensemble S correspondant à un nœud de T_S. Ainsi, pour toute paire de sommets $u, v \in S$ partageant la même couleur et telle que u et v ne sont pas des feuilles de $G[S]$, on applique la règle de branchement suivante : on supprime de S (i) tous les sommets u' tels que u appartient à l'unique chemin entre v et u' (u inclus), ou (ii) tous les sommets v' tels que v appartient à l'unique chemin entre u et v' (v inclus). Pour tout ensemble S correspondant à une feuille de T_S, il n'existe donc pas deux sommets de même couleur tels qu'aucun de ces sommets n'est une feuille de $G[S]$. Pour de tels ensembles S, on commence donc par supprimer itérativement toutes les feuilles de $G[S]$ dont la couleur n'est pas unique dans $\mathrm{col}(S)$. Puisqu'un arbre reste connexe après la suppression de n'importe laquelle de ses feuilles, il suffit ensuite de regarder si $\mathrm{col}(S) = \mathcal{C}$ pour déterminer si S est une occurrence de $\mathcal{M}$ dans G.

On observe que l'algorithme décrit ci-dessus est correct puisqu'il existe un ensemble S correspondant à une feuille de T_S pour chaque sous-ensemble colorful de $|\mathcal{C}|$ sommets de G. Par ailleurs, chaque étape de l'algorithme présenté ci-dessus est exécutée en temps polynomial ; la complexité de celui-ci est donc

exponentielle uniquement en son nombre de nœuds. Puisque la règle de branchement supprime au minimum 2 sommets dans chacun des deux cas possibles, on note que le *vecteur de branchement* (voir Section 1.3.1, page 25) de l'algorithme est (2,2). On en déduit que l'algorithme proposé par Fertin et Komusiewicz a une complexité temporelle de $\mathcal{O}^*(\sqrt{2}^{\ell})$. □

On vient de montrer que COLORFUL GRAPH MOTIF peut être résolu en temps $\mathcal{O}^*(\sqrt{2}^{\ell})$ dans les arbres. Afin d'améliorer l'efficacité de cet algorithme, Fertin et Komusiewicz ont montré que COLORFUL GRAPH MOTIF admet également un kernel polynomial de taille au plus $(2\ell + 1)$ dans les arbres, lequel peut être calculé en temps $\mathcal{O}(m)$, où m est le nombre d'arêtes de G [55]. Malheureusement, COLORFUL GRAPH MOTIF n'admet pas de kernel polynomial dans les graphes en général et GRAPH MOTIF n'admet même pas de kernel polynomial dans les arbres, sauf si $\mathsf{NP} \subseteq \mathsf{NP/Poly}$ [55]. Fertin et Komusiewicz ont en revanche proposé un algorithme de programmation dynamique en temps $\mathcal{O}^*(4^{\ell})$ dans les arbres [55]. Enfin, ils ont montré que GRAPH MOTIF ne peut pas être résolu en temps $\mathcal{O}((2-\epsilon)^{\ell})$ sous l'hypothèse SETH (voir page 22) [55]. On rappelle que l'algorithme proposé par Betzler *et al.* pour COLORFUL GRAPH MOTIF dans le Théorème 26 possède une complexité en temps de $\mathcal{O}^*(2^{\ell})$, mais que celui-ci est restreint aux motifs colorfuls. Par conséquent, le fait que GRAPH MOTIF ne peut vraisemblablement pas être résolu en temps $\mathcal{O}((2-\epsilon)^{\ell})$ ne suffit pas à prouver l'optimalité de l'algorithme de Betzler *et al.*.

On résume dans le Tableau 2.3 les résultats obtenus dans cette section.

	Restriction sur G	Statut FPT	Statut kernel
GRAPH MOTIF	–	Pas d'algorithme en $\mathcal{O}((2-\epsilon)^{\ell})$ sous SETH [55]	–
	Arbre	$\mathcal{O}^*(4^{\ell})$	Pas de kernel [55]
COLORFUL GRAPH MOTIF	–	$\mathcal{O}^*(2^{\ell})$ [10]	Pas de kernel [55]
	Arbre	$\mathcal{O}^*(\sqrt{2}^{\ell})$ [55]	Kernel de taille $\mathcal{O}(\ell)$ [55]

Tableau 2.3 – Tableau récapitulatif des résultats de complexité paramétrée liés à ℓ pour GRAPH MOTIF et COLORFUL GRAPH MOTIF dans la Section 2.4.2. Ici, ℓ est le nombre de sommets qui ne font pas partie de la solution et SETH est une hypothèse de complexité définie page 22.

2.5 Liste de logiciels pour résoudre GRAPH MOTIF

De nombreux logiciels ont vu le jour afin de résoudre GRAPH MOTIF dans des réseaux biologiques. On peut ainsi citer le logiciel de Lacroix *et al.* appelé Motus qui a été implémenté pour les réseaux métaboliques et qui utilise un algorithme de branch-and-bound [79]. Betzler *et al.* ont, quant à eux, créé un web-serveur appelé Torque pour déterminer des complexes protéiques dans les réseaux PPI [25, 26]. Si celui-ci ne retrouve pas le motif par le biais d'une heuristique, il utilise de la programmation linéaire ou un algorithme de programmation dynamique selon l'instance donnée en entrée.

Toujours dans les réseaux PPI, on retrouve deux algorithmes stochastiques appelés SIMBio [107] et APPAGATO [24], ainsi qu'un plugin pour Cytoscape, appelé GraMoFoNe [15], qui utilise un solveur pseudo-booléen pour résoudre GRAPH MOTIF. Cytoscape est un logiciel gratuit et open-source qui a été téléchargé plus de 100 000 fois depuis sa création en 2002 (selon le nombre de téléchargements du plugin le plus téléchargé), et dont le but est de visualiser et d'éditer des réseaux biologiques [112]. Si GraMoFoNe n'est plus compatible avec les dernières mises à jour de Cytoscape, l'application aura, quant à elle, été téléchargée plus de 500 fois.

Le dernier exemple de logiciel que nous donnons est celui de Björklund *et al.* [13], qui ont implémenté l'algorithme FPT relativement à la taille du motif possédant une complexité dite "optimale" et évoqué dans la Section 2.4. L'implémentation a été testée sur des graphes aléatoires et peut supporter des graphes contenant des centaines de millions d'arêtes tant que le motif est de petite taille (inférieure à 5) [13]. Pour conclure, la majorité des logiciels présentés ici est également capable de résoudre plusieurs des variantes de GRAPH MOTIF décrites dans la Section 2.2.

2.6 Conclusion

Dans ce chapitre, on a introduit le problème GRAPH MOTIF ainsi que trois de ses variantes. On a ensuite décrit une série de résultats provenant de la littérature pour ces quatre problèmes. Dans un premier temps, on a distingué dans la Section 2.3 les instances pour lesquelles GRAPH MOTIF (et/ou ses variantes) peut se résoudre en temps polynomial de celles pour lesquelles le problème est NP-complet. On a ainsi remarqué que GRAPH MOTIF est NP-complet même dans des graphes très contraints, comme les comb-graphs ou les superstars. Dans un deuxième temps, on a présenté dans la Section 2.4 des résultats de complexité paramétrée pour MCA et ses variantes relativement à deux paramètres k, la taille du motif, et ℓ, le deuxième au nombre de sommets qui n'appartiennent pas à une solution. Pour chacun de ces paramètres, on a montré des algorithmes de complexité paramétrée et des résultats concernant l'existence de kernels pour GRAPH MOTIF (et/ou ses variantes). On précise cependant qu'il existe une quinzaine de variantes de GRAPH MOTIF ainsi que des résultats de complexité paramétrée pour une multitude d'autres paramètres dans la littérature. Ainsi, on écrit actuellement avec Florian Sikora (LAMSADE, Université Paris Dauphine) une revue exhaustive des résultats de complexité liés à GRAPH MOTIF et ses variantes.

Dans le Chapitre 3, on introduit un nouveau problème appelé MAXIMUM COLORFUL ARBORESCENCE dans le but d'identifier des molécules inconnues de taille modérée. On verra que la modélisation de ce problème emprunte des caractéristiques propres à chacune des variantes de GRAPH MOTIF qui a été introduite dans ce chapitre, mais que ce nouveau problème introduit également une nouvelle contrainte sur les couleurs du graphe d'entrée G. Le but des chapitres suivants sera de déterminer comment utiliser au mieux cette contrainte afin de résoudre efficacement le problème MAXIMUM COLORFUL ARBORESCENCE.

3

Le problème MAXIMUM COLORFUL ARBORESCENCE : caractérisation des instances difficiles

Dans ce chapitre, on introduit un problème, appelé MAXIMUM COLORFUL ARBORESCENCE, possédant des similarités avec GRAPH MOTIF (voir Chapitre 2) et auquel l'essentiel de cette thèse a été consacré. Dans la suite, on explique tout d'abord les problématiques biologiques qui ont mené à la formalisation de MAXIMUM COLORFUL ARBORESCENCE, puis on propose une étude algorithmique approfondie du problème. Dans cette étude, on montre dans un premier temps des ratios d'inapproximabilité ainsi que des algorithmes d'approximation pour ce problème dans des classes d'arbres contraintes telles que les *comb-graphs* et les *superstars*. Dans un second temps, on exhibe une propriété importante du problème afin de distinguer une catégorie d'instances qui peut être résolue en temps polynomial.

La plupart de ces travaux a été publiée lors de la 14ème édition de la conférence Theory and Applications of Models of Computation (TAMC) en 2017 [53].

3.1 Le problème MAXIMUM COLORFUL ARBORESCENCE

Contexte biologique. Dans le Chapitre 2, on a introduit la notion de réseaux métaboliques et expliqué en quoi ces réseaux permettent de mieux comprendre le fonctionnement du métabolisme d'un organisme. Pour rappel, ces réseaux décrivent au niveau cellulaire des réactions chimiques entre des petites molécules, appelées des métabolites. Tous les organismes synthétisent de nombreux métabolites différents – entre 4000 et 20000 pour tous les eucaryotes supérieurs –, mais une majorité de ces métabolites est toujours inconnue [52]. Ainsi, l'*identification* de métabolites est un problème majeur en biologie [81, 88], ayant des applications notamment dans la conception de nouveaux médicaments [82, 108].

Dans la suite, on utilise une formule (dite brute) pour décrire le nombre et le type d'atomes d'une molécule. Par exemple, la molécule d'eau H_2O contient deux atomes d'hydrogène et un atome d'oxygène. On considère qu'un métabolite est *identifié* si on connaît sa formule ainsi que sa structure. En effet, deux molécules ayant la même formule peuvent avoir une structure différente. Ces deux molécules, dites *isomères*, peuvent donc avoir des propriétés physiques, chimiques et biologiques différentes. Par exemple, l'éthanol et le méthoxyméthane partagent la même formule C_2H_6O. Cependant, l'éthanol est utilisé dans l'industrie agro-alimentaire (notamment dans la conception de spiritueux), tandis que le méthoxyméthane

est utilisé comme biocarburant par les entreprises pétrolières.

La spectrométrie de masse est une des techniques les plus utilisées pour identifier des métabolites. Concrètement, cette technique consiste à fragmenter une molécule en plusieurs sous-molécules qui peuvent elles-mêmes également être fragmentées. Le spectromètre enregistre le rapport masse/charge m/z, que nous appellerons simplement masse par la suite, de chaque fragment de la molécule d'entrée – y compris la molécule d'entrée elle-même. On obtient en sortie un *spectre* dans lequel la masse de chaque fragment est représentée par un pic (voir Figure 3.1). Pour identifier la molécule d'entrée, on peut ensuite comparer ce spectre à d'autres spectres issus de bases de données [91]. Malheureusement, ces bases de données sont grandement incomplètes [52, 115, 122] et ne permettent donc que l'identification de métabolites déjà connus.

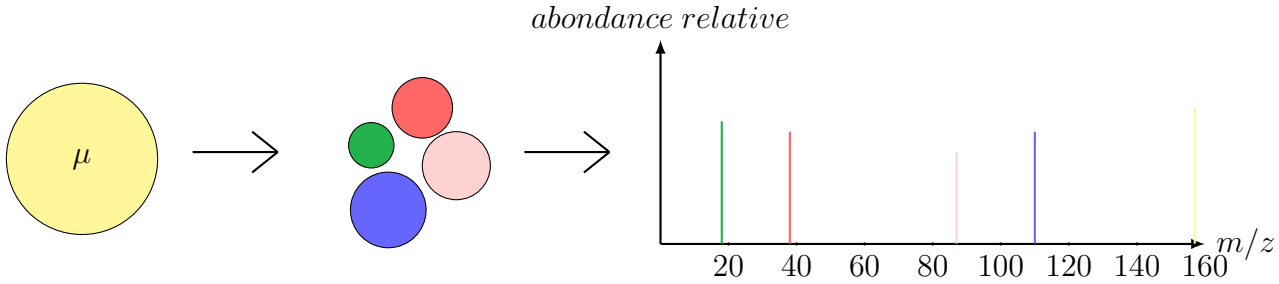


Figure 3.1 – Exemple (fictif) de fragmentation d'une molécule μ dans un spectromètre de masse pour obtenir un spectre s_μ. Chaque pic dans le spectre correspond à la masse d'un fragment. Plus la valeur en ordonnée d'un pic est élevée, plus ce pic est important dans le spectre.

Introduction du problème MAXIMUM COLORFUL SUBTREE. En 2008, Böcker et Rasche ont présenté une méthode pour déterminer la formule d'un métabolite [18]. La méthode proposée est une méthode *de novo*, ce qui signifie qu'elle permet d'inférer la formule du métabolite en entrée uniquement à partir du spectre obtenu – et donc sans recourir aux bases de données. En particulier, cela signifie qu'on peut déterminer la formule d'un métabolite *inconnu*, qui n'apparaît donc dans aucune base de données. De plus, on verra par la suite que cette méthode a non seulement permis de déterminer la formule du métabolite d'entrée, mais également de l'identifier.

Pour illustrer la méthode proposée par Böcker et Rasche, on considère un métabolite inconnu μ dont on veut inférer la formule à partir d'un spectre s_μ. Tout d'abord, pour chaque pic p de s_μ, on génère un ensemble de formules dont la masse, exprimée en Daltons (Da), correspond à celle du pic p [17]. Dans la suite, les formules dont la masse correspond à celle de μ sont appelées des formules *candidates*. De plus, une formule a est une *sous-formule* d'une formule b si a ne contient pas d'atome non-contenu dans b et si la quantité de chaque atome contenu dans a est au plus égale à celle contenue dans b.

Pour chaque formule candidate, on crée un *DAG* $G = (V, A)$ avec une racine unique r qui représente la formule candidate considérée. Tous les autres sommets $v \in V$ représentent des sous-formules de cette formule candidate et il existe un arc (u, v) dans A si la formule représentée par le sommet v est une sous-formule de celle représentée par le sommet u. On crée ensuite un ensemble de couleurs $\mathcal{C}$ dont chaque couleur est associée à un pic distinct de s_μ et on applique une fonction de coloration $\mathrm{col} : V \to \mathcal{C}$ telle que les formules de tous les sommets de même couleur dans G possèdent la même masse. Enfin, on crée une fonction de pondération des arcs $w : A \to \mathbb{R}$ telle que pour tout arc (u, v) de A, le poids $w(u, v)$ représente la plausibilité que la molécule représentée par u ait été fragmentée en la molécule représentée par v dans le processus de spectrométrie de masse [18, 102, 101] ; ce poids est possiblement négatif puisque les auteurs

utilisent des logarithmes sur des probabilités pour le calculer. Pour chaque formule candidate, on vient de décrire une instance $(G, \mathcal{C}, \text{col}, w, r)$ du problème MAXIMUM COLORFUL SUBTREE, que nous définissons ci-après et pour lequel on montre un exemple d'instance dans la Figure 3.2. Le but de ce problème est de trouver un sous-arbre $T = (V_T, A_T)$ de poids maximum dans G qui soit *colorful*, *i.e.* tel qu'il n'existe pas deux sommets de même couleur dans T. Ce sous-arbre correspond à l'*arbre de fragmentation* de la formule candidate considérée : si on suppose que la formule candidate considérée est la formule de μ, alors l'arbre de fragmentation de cette formule décrit le scénario de fragmentation de μ le plus probable qui a conduit au spectre s_μ. On définit formellement le problème MAXIMUM COLORFUL SUBTREE ci-dessous.

MAXIMUM COLORFUL SUBTREE (MCS)

- **Instance**: Un DAG $G = (V, A)$ enraciné en r, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$, une fonction de pondération $w : A \to \mathbb{R}$
- **Sortie**: Un sous-arbre $T = (V_T, A_T)$ colorful et enraciné en r
- **Mesure**: $w(T) = \sum_{a \in A_T} w(a)$

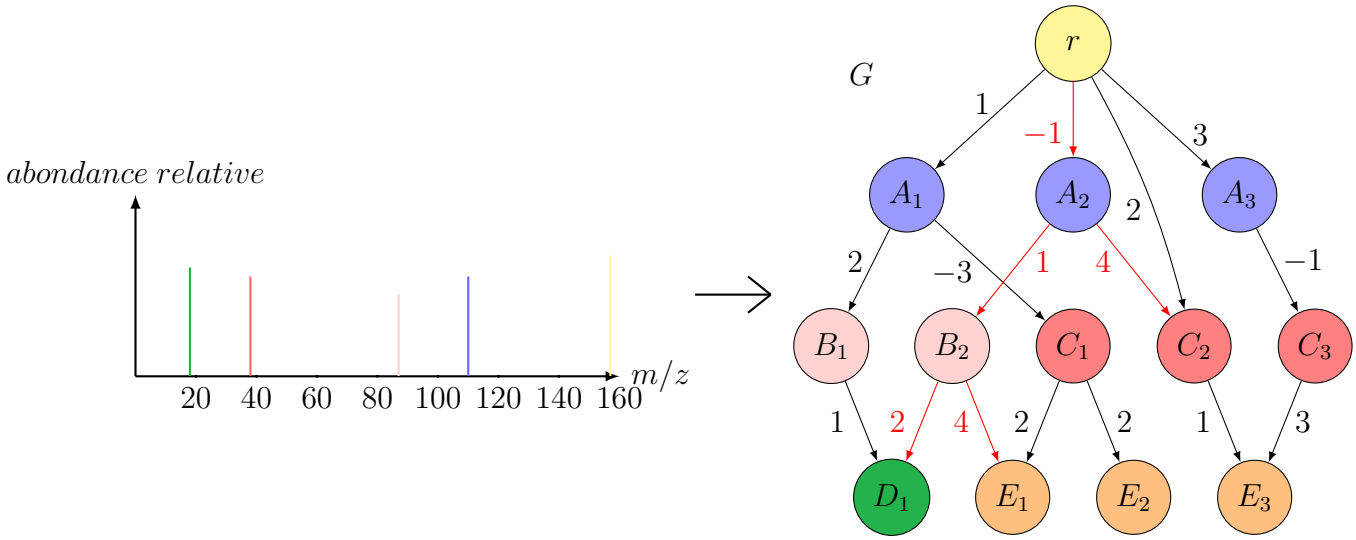


Figure 3.2 – Construction d'une instance de MCS à partir d'un spectre s_μ (exemple fictif). Les arcs d'une solution de MCA dans G sont représentés en rouge.

Intérêt du problème MAXIMUM COLORFUL SUBTREE. Pour déterminer la formule du métabolite inconnu μ, on a généré un ensemble de formules candidates et résolu une instance de MCS pour chacune d'entre elles. Pour finir, on classe maintenant les formules candidates par ordre décroissant du poids de leur arbre de fragmentation et on fournit cette liste à l'utilisateur. Dans le premier article introduisant les arbres de fragmentation [18], Böcker et Rasche montrent que la formule d'un métabolite correspond à la formule candidate en première position du classement opéré – qui contient une centaine de formules candidates testées pour chaque métabolite – pour 26 des 32 métabolites sur lesquels ils ont testé leur méthode. Pour chacun de ces métabolites, ils montrent également que la formule correcte du métabolite est toujours parmi les cinq premières formules candidates de ce classement.

Même s'il existe un algorithme de complexité paramétrée pour le problème MCS [18], celui-ci est principalement résolu dans la littérature en utilisant des heuristiques [44] et de l'optimisation linéaire [119].

Böcker et Dührkop ont ainsi créé un logiciel appelé Sirius 3 [1] (basé sur de l'optimisation linéaire) pour calculer des arbres de fragmentation [16]. Pour tester leur logiciel, ils ont tout d'abord essayé de retrouver la formule de plus de 4000 métabolites dont la formule est enregistrée dans la base de données PubChem [70], qui contient plus de 96 millions de molécules. Ainsi, ils ont montré que que la formule d'un métabolite correspond à la première formule candidate donnée par Sirius 3 pour 76% des instances et appartient au top 5 des formules candidates pour plus de 90% des instances [16]. Böcker et Dührkop ont ensuite comparé leur méthode avec une autre méthode dite "naïve" : pour chaque métabolite en entrée, cette méthode consiste à fournir une liste de formules, contenues dans PubChem, classées telles que la masse de ces formules est la plus proche possible de la masse du métabolite donné en entrée. A titre de comparaison, la formule d'un métabolite correspond à la première formule candidate donnée par la méthode naïve pour seulement 17.1% des instances et appartient au top 5 des formules candidates pour seulement 59.8% des instances. Böcker et Dührkop ont donc montré que Sirius 3 est bien plus performant que la méthode "naïve" pour déterminer la formule d'un métabolite qui appartient à une base de données. De plus, on rappelle que Sirius 3 est basé sur une méthode *de novo* et peut donc également inférer la formule d'un métabolite inconnu dans PubChem (ou toute autre base de données).

La qualité des formules proposées par Sirius 3 est impactée par deux critères majeurs : la masse du métabolite (exprimée en Daltons) et le nombre de formules candidates créées [16]. En effet, Sirius 3 retrouve quasiment la totalité des formules pour les métabolites d'entrée dont la masse est entre 0 Da et 200 Da, contre environ 40% si la masse du métabolite d'entrée est entre 600 Da et 800 Da. De plus, il existe potentiellement 20 000 formules candidates pour un métabolite et plus il en existe, plus la formule exacte du métabolite est loin dans le classement donné par Sirius 3. Ainsi, Sirius 3 retrouve plus de 95% des formules des métabolites d'entrée pour lesquels il existe moins de 7 formules candidates à tester, contre environ 65% s'il existe entre 128 et 265 formules candidates et finalement moins de 25% s'il existe plus de 1000 formules candidates – ce taux est ensuite stable quelque soit le nombre de formules candidates. A titre de comparaison, la méthode naïve est moins efficace pour retrouver des formules que Sirius 3 si le métabolite d'entrée a une masse inférieure à 800 Da et/ou si le métabolite d'entrée possède moins de 1000 formules candidates. Au-delà, Böcker et Dührkop constatent que PubChem contient très peu de candidats pour des métabolites de masse élevée. En effet, on rappelle que la formule des métabolites en entrée appartient nécessairement à PubChem et que Sirius 3 est une méthode *de novo* dont certaines formules candidates n'appartiennent pas à PubChem. Par conséquent, si le métabolite d'entrée a une masse au moins égale à 800 Da et/ou si le métabolite d'entrée possède plus de 1000 formules candidates, alors les performances de la méthode naïve deviennent légèrement meilleures que celles de Sirius 3 (avec toutefois moins de 5 points d'écart entre les pourcentages de formules retrouvées). Pour ces métabolites, il devient donc judicieux de calculer uniquement les arbres de fragmentation des formules candidates appartenant à PubChem dans Sirius 3 ; la combinaison des deux méthodes permet ainsi de retrouver environ 40% des formules des métabolites d'entrée pour lesquels il existe plus de 2047 formules candidates.

Si les arbres de fragmentation permettent de retrouver la formule d'un métabolite, ceux-ci contiennent également des informations sur la structure de ces métabolites. Ainsi, des experts en spectrométrie de masse ont évalué manuellement 79 arbres de fragmentations contenant un total de 808 arcs et ont montré que plus que 78% de ces arcs – et donc toutes les formules moléculaires associées aux sommets incidents de ces arcs – étaient corrects [102]. Rasche *et al.* ont proposé une méthode qui compare directement des arbres de fragmentation – au lieu de comparer le spectre du métabolite d'entrée aux spectres appartenant à une base de données – afin d'identifier des métabolites [101]. Dans le cas où les métabolites recherchés ne figurent pas dans une base de données, leur méthode renvoie un métabolite dont la structure chimique est très proche. Ce concept, dénommé *alignement* d'arbres de fragmentation, est maintenant très répandu pour identifier des métabolites [66, 43, 90]. Le logiciel CSI:FingerID [46], qui utilise entre autres Sirius 3, a d'ailleurs remporté plusieurs compétitions visant à identifier des métabolites [110]. Un web-service utilisant

1. https://bio.informatik.uni-jena.de/software/sirius/

ce logiciel a enregistré plus d'un million de requêtes en trois mois [1].

MAXIMUM COLORFUL ARBORESCENCE, une variante plus juste de MCS. Dans le paragraphe ci-dessus, on a expliqué que les techniques de pointe pour identifier des métabolites en spectrométrie de masse reposent sur le calcul d'arbres de fragmentation, et donc sur la résolution du problème MAXIMUM COLORFUL SUBTREE. Après avoir étudié la littérature autour du problème MCS, on a remarqué que sa définition pouvait être plus précise et qu'ajouter cette précision pouvait permettre de trouver des algorithmes plus performants que ceux qui existent déjà. Ainsi, notre première contribution a donc été de définir le problème MAXIMUM COLORFUL ARBORESCENCE.

Dans la suite, on montre comment obtenir une définition plus précise de MCS en introduisant le *graphe de hiérarchie de couleurs* du graphe d'entrée G d'une instance de MCS. On remarque ensuite que ce graphe est naturellement contraint si l'instance de MCS considérée a été construite à partir d'un spectre, et on utilise cette nouvelle contrainte pour définir le problème MAXIMUM COLORFUL ARBORESCENCE. Le reste du chapitre sera ensuite consacré à une étude algorithmique de MAXIMUM COLORFUL ARBORESCENCE.

Soit $G = (V, A), \mathcal{C}, \text{col}, w, r)$ une instance de MCS. On définit maintenant $\mathcal{H}(G) = (\mathcal{C}, A_{\mathcal{C}})$ le *graphe de hiérarchie de couleurs* de G de la manière suivante : $V(\mathcal{H}(G))$ est l'ensemble de couleurs $\mathcal{C}$ de G et il existe un arc d'une couleur c vers une couleur c' dans $\mathcal{H}(G)$ si et seulement si il existe un arc d'un sommet de couleur c vers un sommet de couleur c' dans G. On montre un exemple dans la Figure 3.3.

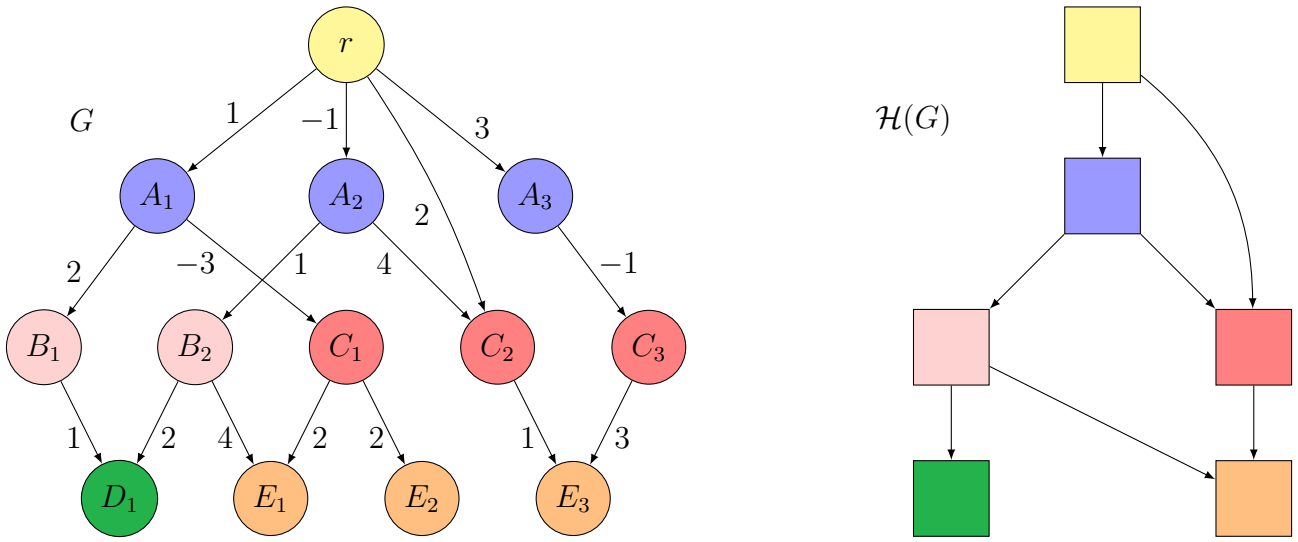


Figure 3.3 – Construction du graphe de hiérarchie de couleurs $\mathcal{H}(G)$ (à droite) du graphe G (à gauche).

On rappelle maintenant que chaque couleur de G est associée à un pic distinct d'un spectre et que ces pics sont eux-mêmes associés à des masses. S'il existe un arc d'un sommet u de couleur c vers un sommet v de couleur c' dans G, alors la formule représentée par v est une sous-formule de la formule représentée par u, et la masse associée à la couleur c' est donc nécessairement inférieure à la masse associée à la couleur c. Ainsi, il n'existe aucun chemin d'un sommet de couleur c' vers un sommet de couleur c dans G puisqu'une molécule ne peut pas être fragmentée en une molécule de plus grande masse. Par conséquent, on en déduit que le graphe $\mathcal{H}(G)$ est toujours un *DAG*. Le problème que l'on veut résoudre n'est donc pas le problème MCS, mais un problème plus précis qu'on définit ci-dessous.

1. https://bio.informatik.uni-jena.de/category/sirius-news

MAXIMUM COLORFUL ARBORESCENCE (MCA)

• **Instance**: Un DAG $G = (V, A)$ enraciné en r, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration $col : V \to \mathcal{C}$ telle que le graphe de hiérarchie de couleurs de G, $\mathcal{H}(G)$, est un DAG, une fonction de pondération $w : A \to \mathbb{R}$

• **Sortie**: Un sous-arbre $T = (V_T, A_T)$ colorful et enraciné en r

• **Mesure**: $w(T) = \sum_{a \in A_T} w(a)$

On observe que MCA présente des similarités avec le problème GRAPH MOTIF et ses variantes qui sont étudiés dans le Chapitre 2. En particulier, tout comme pour le problème COLORFUL GRAPH MOTIF, on observe que le multi-ensemble de couleurs d'une solution de MCA ne doit pas contenir de répétitions de couleurs. Ainsi, le multi-ensemble de couleurs d'une solution de MCA doit être inclus dans un motif implicite correspondant à l'ensemble des couleurs du graphe, mais, tout comme pour le problème MAX GRAPH MOTIF, le multi-ensemble de couleurs d'une solution de MCA peut ne pas être égal à ce motif. Enfin, tout comme pour le problème WEIGHTED GRAPH MOTIF, la modélisation de MCA inclut une fonction de pondération qui est cette fois-ci appliquée aux arcs, et non pas aux arêtes, du graphe d'entrée. En revanche, la modélisation de MCA introduit la notion de graphe de hiérarchie de couleurs, ce qui n'existe dans aucune variante de GRAPH MOTIF, et contraint ce graphe à être un DAG. De plus, on cherche à maximiser le poids d'une solution de MCA, et non pas à trouver une sous-occurrence de cardinalité maximum du motif implicite qu'on vient de décrire.

Dans la suite, pour des raisons de lisibilité, on utilise la notation $\mathcal{H}$ au lieu de $\mathcal{H}(G)$ quand le contexte est clair. Nous aurons également besoin d'étudier des cas contraints du problème MCA que nous définissons ci-après. Le problème MCA$^+$ dénote la restriction de MCA aux DAGs de poids positifs ou nuls et le problème UMCA dénote la restriction de MCA$^+$ aux DAGs de poids unitaires, *i.e.* pour lesquels $w(a) = 1$ pour tout $a \in A$. Si G ne contient que des arcs négatifs, alors on note que l'unique sous-arbre solution est celui limité à $T = (r, \emptyset)$ et nous n'avons donc pas besoin de définir une restriction de MCA aux DAGs de poids négatifs. Le problème MCA-X est une restriction de MCA aux DAGs dans lesquels chaque couleur apparaît au plus x fois dans G, *i.e.* dans lesquels, pour tout $c \in \mathcal{C}$, il existe au plus x sommets de couleur c. Finalement, on peut combiner ces restrictions pour décrire la restriction d'une instance de MCA par rapport aux poids et par rapport au nombre maximum d'occurrences des couleurs de $\mathcal{C}$ dans G. Par exemple, le problème UMCA-2 dénote la restriction de MCA aux DAGs de poids unitaires et dans lesquels chaque couleur apparaît au plus deux fois.

3.2 Résultats de complexité en fonction de la nature du graphe d'entrée et du graphe de hiérarchie de couleurs

Dans cette section, on cherche à déterminer la complexité du problème MCA ou de certaines de ses restrictions en fonction de la nature du graphe d'entrée G et du graphe de hiérarchie de couleurs. Pour cela, on étudie le problème dans différentes familles de graphes afin de distinguer les instances pouvant être résolues en temps polynomial de celles pour lesquelles le problème est NP-dur. Pour ces dernières, on montre des algorithmes d'approximation ainsi que des bornes d'inapproximibilité. Les résultats principaux obtenus sont récapitulés dans le Tableau 3.1.

3.2.1 Résultats de complexité quand le graphe d'entrée est un arbre

Par définition, on note que toute instance de MCA est également une instance de MCS. Par conséquent, les résultats positifs liés à MCS s'appliquent à MCA. En revanche, les résultats négatifs liés à MCS ne

Restriction		Contrainte	Variante	Résultat
	sur G	arbre	UMCA	Pas de ratio d'approximation en $2^{log^{\delta}n}$, $\delta < 1$ (Thm. 31) Approximation en $\mathcal{O}(\frac{n}{\log n})$ (Prop. 32)
		comb-graph	UMCA-2	Pas de ratio d'approximation en $\mathcal{O}(n^{\frac{1}{3}-\epsilon})$ (Thm. 35)
			UMCA	Approximation en $\mathcal{O}(n^{\frac{1}{2}})$ (Prop. 38)
		superstar	UMCA-2	APX-dur (Thm. 40) 2-approximable (Prop. 41)
		caterpillar	MCA	Appartient à P (Prop. 43)
	sur $\mathcal{H}$	arbre	MCA	Appartient à P (Thm. 46)

Tableau 3.1 – Tableau récapitulatif des résultats obtenus dans le Chapitre 3, où n est le nombre de sommets du graphe d'entrée d'une instance de MCA. Les résultats positifs d'approximation sont liés au fait que le graphe de hiérarchie de couleurs $\mathcal{H}$ est un DAG.

s'appliquent pas nécessairement à MCA, puisque toute instance de MCS n'est pas nécessairement une instance de MCA.

On va maintenant décrire une série de résultats négatifs pour MCS qui s'adaptent facilement pour obtenir des résultats négatifs pour MCA. Premièrement, Böcker et Rasche ont réduit le problème SAT (introduit dans la Section 1.2.2, page 20) au problème MCS pour prouver que MCS est NP-dur même si G est un arbre dont les arcs ont un poids unitaire [18]. Les instances de MCS décrites dans cette réduction sont clairement aussi des instances de UMCA, ce qui implique que UMCA est NP-dur même si G est un arbre. Deuxièmement, Rauf *et al.* affirment que MCS est APX-dur même dans les arbres binaires [103] en réutilisant la preuve de la Proposition 1 de Dondi *et al.* [39]. De manière similaire, nous affirmons que cette preuve peut facilement être adaptée afin de prouver que UMCA-2 est APX-dur même dans les arbres binaires. Enfin, Rauf *et al.* ont démontré que le problème MCS n'admet pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$, sauf si P = NP [103]. Nous redonnons la preuve de ce résultat, puis nous l'utilisons afin de déduire deux résultats négatifs pour MCA– dont un restreint au cas où G est un arbre.

Théorème 28. *[103] Le problème* MCS *n'admet pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$, sauf si* P = NP.

Preuve. On effectue une réduction du problème MAXIMUM INDEPENDENT SET, défini ci-dessous, au problème MCS.

MAXIMUM INDEPENDENT SET (MIS)

- **Instance**: Un graphe $G_I = (V_I, E_I)$
- **Sortie**: Un sous-ensemble indépendant $I \subseteq V_I$
- **Mesure**: $|I|$

Dans la suite, on supposera que $V_I = \{v'_1, v'_2, \ldots, v'_{n_I}\}$ et $E_I = \{e_1, e_2, \ldots, e_{m_I}\}$, avec $|V_I| = n_I$ et $|E_I| = m_I$. Pour tout graphe d'entrée G_I de MIS, on construit un DAG $G = (V, A)$ de MCS ayant des sommets qui s'organisent en quatre niveaux. Le premier niveau de G contient uniquement la racine r. Le deuxième niveau de G contient uniquement un sommet u. Le troisième niveau de G contient un ensemble de sommets $\{x_i : i \in [n_I]\}$ et le quatrième niveau contient un ensemble de sommets $\{y_j : j \in [m_I]\}$. Il existe un arc (r, u) de poids $-n_I(m_I - 1)$ entre le premier et le deuxième niveau. Entre le deuxième et le troisième niveau, il existe un arc (u, x_i) de poids -1 pour tout $i \in [n_I]$. Enfin, entre le troisième et le quatrième niveau, il existe un arc (x_i, y_j) de poids n_I pour tout $i \in [n_I]$ et $j \in [m_j]$ tels que v'_i est incident

à e_j dans G_I. Pour finir, on attribue une couleur unique à chaque sommet de G. On montre un exemple de réduction d'une instance de MIS en instance de MCS dans la Figure 3.4.

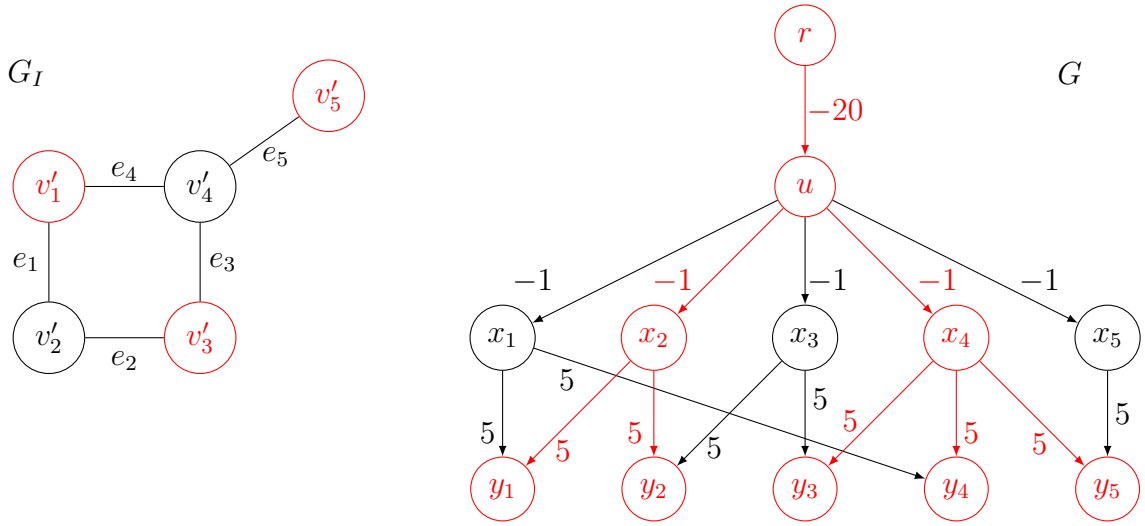


Figure 3.4 – Construction du graphe d'entrée G d'une instance de MCS (à droite) à partir d'une instance G_I de MIS (à gauche). Chaque sommet de G possède une couleur unique. Les sommets de type v'_i (resp. les arêtes de type e_j) dans l'instance de MIS sont représentés par les sommets de type x_i (resp. les sommets de type y_j) dans l'instance de MCS. L'ensemble de sommets $I = \{v'_1, v'_3, v'_5\}$ en rouge est une solution de MIS dans G_I et correspond à la solution de MCS $T = (V_T, A_T)$ dans G qui est également représentée en rouge, avec $V_T = \{r, u, x_2, x_4, y_1, y_2, y_3, y_4, y_5\}$. On observe que $|I| = w(T) = 3$.

On montre maintenant que G_I contient un ensemble indépendant I de cardinalité supérieure ou égale à k si et seulement si G contient un sous-arbre colorful $T = (V_T, A_T)$ de poids supérieur ou égal à k et enraciné en r.

($\Rightarrow$) Si G_I contient un ensemble indépendant I de cardinalité supérieure ou égale à k, alors on fixe $V_T = \{r\} \cup \{u\} \cup \{x_i : i \in [n_I] \mid v'_i \notin I\} \cup \{y_j : j \in [m_I]\}$. Concrètement, les $n_I - k$ sommets de type x_i qui appartiennent à V_T correspondent aux $n_I - k$ sommets de type v'_i qui n'appartiennent pas à I. De plus, on note que les deux voisins entrants de chaque sommet de type y_j dans G correspondent aux deux sommets incidents de la même arête de type e_j dans G_I. Ainsi, puisque I est un ensemble indépendant, pour chaque sommet de type y_j il existe nécessairement au moins un sommet de type $x_i \in V_T$ tel que x_i est incident à y_j dans G. Par conséquent, il existe au moins un *arbre couvrant* T de $G[V_T]$ dans G et $w(T) = -n_I(m_I - 1) + (n_I - k)(-1) + n_I \cdot m_I = k$.

($\Leftarrow$) Soit $T = (V_T, A_T)$ une solution du problème MCA dans G. On commence par montrer que T contient nécessairement u ainsi que tous les sommets de type y_j. Dans un premier temps, on montre que $u \in V_T$. Pour cela, on note que le poids de tout arbre couvrant T' de G est égal à $w(T') = -n_I(m_I - 1) + (-1)n_I + n_I m_I = 0$. De plus, on rappelle que tous les sommets de type y_j ont un degré entrant égal à 2 dans G. Par conséquent, pour tout sommet v de type x_i dans V, il existe au moins un arbre couvrant de $G[V \setminus \{v\}]$ dans G, et le poids de celui-ci est strictement positif. Ainsi, si T est une solution de MCA, alors T doit contenir plus d'un sommet pour que $w(T)$ soit également strictement positif. Comme u est le seul voisin de r dans G, on vient ainsi de montrer que le sommet u appartient nécessairement à V_T. Dans un deuxième temps, on montre maintenant que tous les sommets de type y_j appartiennent nécessairement à V_T. Pour cela, on rappelle que $u \in V_T$. Ainsi, pour tout sommet de type y_j dans V tel que $y_j \notin V_T$, il suffit d'ajouter au plus un sommet de type x_i à V_T en plus de y_j pour que le poids d'un arbre couvrant de $G[V_T]$ augmente au moins de $n_I - 1$. Par conséquent, si $T = (V_T, A_T)$ est une solution de MCA dans G, alors V_T contient nécessairement tous les sommets de type y_j. En particulier, cela implique que l'ensemble $V \setminus V_T$ ne contient aucune paire de sommets de type x_i qui sont adjacents au même sommet de type y_j dans

G ; l'ensemble $I = \{v'_i : i \in [n_I] \mid x_i \notin V_T\}$ est donc indépendant dans G_I. Enfin, puisque $w(T) \geq k$, on en déduit que T contient nécessairement au moins $n_I - k$ sommets de type x_i et donc que $|I| \geq k$.

Pour conclure, on vient de décrire une réduction du problème MIS vers le problème MCS. Dans cette réduction, on note que G contient $n = \mathcal{O}(m_I) = \mathcal{O}(n_I^2)$ sommets. Par ailleurs, MIS n'admet pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n_I^{1-\epsilon})$, pour tout $\epsilon > 0$, sauf si $\mathsf{P} = \mathsf{NP}$ [123]. Par linéarité de la réduction proposée ci-dessus, MCS n'admet donc pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$ [1], pour tout $\epsilon > 0$, sauf si $\mathsf{P} = \mathsf{NP}$. □

On remarque maintenant que l'instance de MCS construite dans la preuve ci-dessus est également une instance de MCA-1. En effet, par construction, le graphe de hiérarchie de couleurs $\mathcal{H}$ du DAG G qui a été créé est lui-même un DAG puisque G est colorful. On obtient donc le corollaire suivant.

Corollaire 29. *Le problème* MCA-1 *n'admet pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$, sauf si* $\mathsf{P} = \mathsf{NP}$.

Maintenant, on remarque que l'instance de MCA-1 décrite ci-dessus peut également être transformée en instance de MCA-2 pour laquelle G est un arbre. Pour cela, on rappelle que tous les sommets de l'instance de MCA-1 ont un degré entrant égal à 1, à l'exception des sommets de type y_j qui ont un degré entrant égal à 2. Dans la suite, pour chaque sommet de type y_j dans l'instance de MCA-1, on nomme arbitrairement $f(y_j)$ l'un des deux voisins entrants de y_j. On construit maintenant une instance de MCA-2 à partir de l'instance de MCA-1. Ainsi, pour tout sommet de type y_j dans V, on ajoute un sommet y'_j de même couleur que y_j dans V et on remplace l'arc $(f(y_j), y_j)$ par un arc $(f(y_j), y'_j)$ dans A. On observe que $G = (V, A)$ est un arbre dans l'instance de MCA-2 puisque tous les sommets du graphe d'entrée ont maintenant un degré entrant égal à 1. Puisque l'instance de MCA-2 contient $n = 2 + n_I + 2 \cdot m_I$ et donc $n = \mathcal{O}(n_I^2)$ sommets, on obtient le corollaire suivant.

Corollaire 30. *Le problème* MCA-2 *restreint au cas où G est un arbre n'admet pas d'algorithme d'approximation en temps polynomial avec un ratio $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$, sauf si* $\mathsf{P} = \mathsf{NP}$.

Le Corollaire 29 et le Corollaire 30 ci-dessus montrent que MCA est fortement inapproximable même si on restreint le nombre maximum d'occurrences de la même couleur dans le graphe d'entrée. On montre maintenant que MCA est également fortement inapproximable même si les poids des arcs sont unitaires dans le graphe d'entrée.

Théorème 31. *Pour tout $\delta < 1$, le problème* UMCA *restreint au cas où G est un arbre ne peut pas être approximé avec un ratio $2^{\log^{\delta} n}$ en temps polynomial, sauf si* $\mathsf{NP} \subseteq \mathsf{DTIME}[2^{\text{poly log } n}]$.

Preuve. On effectue ici une réduction du problème MAXIMUM LEVEL MOTIF, un problème hautement inapproximable introduit par Dondi *et al.* dans [40], vers le problème UMCA. Le problème MAXIMUM LEVEL MOTIF est une version plus contrainte de MAX GRAPH MOTIF dans laquelle le motif est colorful. Ce problème introduit également la notion de fonction de coloration *par niveau* : le graphe d'entrée est un arbre enraciné et deux sommets peuvent partager la même couleur si et seulement si ils sont à la même *distance* de la racine du graphe d'entrée. On donne une définition formelle de MAXIMUM LEVEL MOTIF ci-dessous.

MAXIMUM LEVEL MOTIF (MLM)

• **Instance**: Un arbre $G' = (V, E)$ enraciné en r, un ensemble $\mathcal{C}$ de couleurs, une fonction de coloration par niveau $\text{col}' : V \to \mathcal{C}$

• **Sortie**: Un ensemble de sommets $V' \subseteq V$ tel que $G'[V']$ est connexe et colorful

• **Mesure**: $|V'|$

1. et non pas $\mathcal{O}(n^{1-\epsilon})$ comme énoncé dans [103]

On décrit la construction d'une instance I de UMCA à partir d'une instance I' de MLM. Tout d'abord, on construit un DAG G tel que $V(G) = V(G')$ et dont l'ensemble d'arcs est le suivant : il existe un arc (u, v) pour chaque arête $(u, v) \in E$ telle que $d(r, u) < d(r, v)$ dans G'. Clairement, G est un arbre enraciné en r. On attribue ensuite un poids de 1 à chaque arc de G et on utilise la fonction de coloration col', donnée en entrée de MLM, pour colorer les sommets de G. Comme col' est une fonction de coloration par niveau, il existe un *ordre topologique* des couleurs de $\mathcal{C}$. Par conséquent, $\mathcal{H}$ est un DAG et I est une instance correcte de UMCA. On montre maintenant qu'il existe une solution V' de cardinalité k pour MLM dans G' si et seulement si il existe dans G une arborescence colorful $T = (V_T, A_T)$ telle que $w(T) \geq k - 1$.

($\Rightarrow$) On suppose qu'il existe une solution V' de cardinalité k pour MLM dans G'. Soit $V_T = V'$ et $T = (V_T, A_T)$ une arborescence couvrante de $G[V_T]$. Clairement, T est colorful et de poids $k - 1$. Si $r \notin V_T$, on cherche le sommet $x \in V_T$ tel que $d(r, x) = \min\{d(r, u) : u \in V_T\}$ et on appelle $V_{r,x}$ (resp. $A_{r,x}$) l'ensemble de sommets (resp. arcs) du chemin de r à x dans G. On construit ensuite une nouvelle arborescence $T' = (V_{T'}, A_{T'})$ avec $V_{T'} = V_T \cup V_{r,x}$ et $A_{T'} = A_T \cup A_{r,x}$. Par définition de col', $V_{r,x}$ est colorful et $\mathrm{col}'(V_{r,x}) \cap \mathrm{col}'(V_T) = \mathrm{col}'(x)$. Par conséquent, $V_{T'}$ est colorful et $w(T') \geq k - 1$.

($\Leftarrow$) On suppose qu'il existe une arborescence colorful $T = (V_T, A_T)$ tel que $w(T) \geq k-1$ dans G. On choisit alors $V' = V_T$. Clairement, V' est colorful et de cardinalité k.

Pour tout $\delta < 1$, Dondi *et al.* ont prouvé que MAXIMUM LEVEL MOTIF ne peut pas être approximé avec un ratio $2^{\log^{\delta} n}$ en temps polynomial sauf si $\mathsf{NP} \subseteq \mathsf{DTIME}[2^{\mathrm{poly}\ \log\ n}]$[40]. Par linéarité de la réduction ici présentée, nous obtenons les mêmes conclusions pour UMCA restreint aux arbres. □

Concrètement, si δ est proche de 1, alors le Théorème 31 signifie qu'on ne peut pas approximer UMCA restreint aux arbres avec un ratio proche de n, sauf si $\mathsf{NP} \subseteq \mathsf{DTIME}[2^{\mathrm{poly}\ \log\ n}]$. En particulier, puisque le poids d'une solution est au plus égal à $n - 1$ dans toute instance de UMCA, on en déduit que le ratio d'approximation de tout algorithme d'approximation de UMCA restreint aux arbres est dans ce cas proche de celui d'un algorithme d'approximation qui renvoie systématiquement une arborescence enracinée en r et de poids 1 en sortie.

Dans la suite, on contraint davantage le graphe d'entrée G afin d'obtenir des algorithmes d'approximation en temps polynomial pour UMCA dont le ratio d'approximation est meilleur que le ratio d'inapproximabilité du Théorème 31. L'intuition de ces algorithmes d'approximation est la suivante : un chemin de la racine de G vers n'importe quel autre sommet de G est toujours colorful puisque $\mathcal{H}$ est un DAG. Comme le poids d'une solution est toujours au plus égal à $n - 1$ dans toute instance de UMCA, la difficulté principale de ces algorithmes est donc de déterminer la longueur minimum d'un plus long chemin dans le graphe G. Par exemple, si G est un arbre binaire contenant n sommets, alors sa *hauteur* est toujours au moins égale à $\log n$. Par conséquent, on obtient la proposition suivante.

Proposition 32. *Le problème* UMCA *admet un algorithme d'approximation en temps polynomial de ratio* $\mathcal{O}(\frac{n}{\log n})$ *dans les arbres binaires.*

3.2.2 Résultats de complexité quand le graphe d'entrée est un comb-graph

Dans la Section 3.2.1, on a montré que UMCA restreint aux arbres est fortement inapproximable (voir Théorème 31) mais que UMCA restreint aux arbres *binaires* admet un algorithme d'approximation en temps polynomial de ratio $\mathcal{O}(\frac{n}{\log n})$. Dans cette section, on propose d'étudier le problème MCA dans les *comb-graphs*, qui sont des arbres binaires encore plus contraints définis dans la Section 1.1.2 (page 15). Pour rappel, un comb-graph est un arbre de degré maximum 3 dans lequel il existe au moins un chemin – appelé la *spine* – contenant tous les sommets de degré 3. Malheureusement, on commence par prouver que UMCA-2 reste APX-dur (avec un ratio d'inapproximabilité élevé) dans les comb-graphs. Néanmoins, on

verra également que UMCA admet un algorithme d'approximation en temps polynomial avec un meilleur ratio dans les comb-graphs que dans les arbres binaires.

On commence par prouver que le problème UMCA-2 ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{3}-\epsilon})$, pour tout $\epsilon > 0$, même si G est un comb-graph. Pour cela, on s'inspire de la preuve de la Proposition 1 dans [104] et on effectue une réduction du problème MAXIMUM INDEPENDENT SET (défini à la page 63) au problème UMCA-2. On présente cette réduction dans les paragraphes suivants.

Dans la suite, on considère $V_I = \{v'_1, v'_2, \ldots, v'_{n_I}\}$ et $E_I = \{e_1, e_2, \ldots, e_{m_I}\}$, avec $|V_I| = n_I$ et $|E_I| = m_I$. Pour tout sommet de type v'_i dans V_I et toute arête de type e_j dans E_I, la relation $v'_i \sim e_j$ signifie que v'_i est incident à e_j. Pour tout $i \in [n_I]$, l'ensemble $L_i = \{j \in [m_I] \mid v'_i \sim e_j\}$ est défini comme la liste *ordonnée* des arêtes incidentes au sommet v'_i dans G_I. Ainsi, pour tout $i \in [n_I]$, on observe que $|L_i| = d(v'_i)$, où $d(v'_i)$ est le *degré* de v'_i. Enfin, pour tout $k \in [d(v'_i)]$, on note $L_i(k)$ le k-ième élément de L_i.

Pour toute instance de MIS, on construit une instance de UMCA-2 dont le DAG $G = (V, A)$ est défini de la manière suivante : $V = \{r_i : i \in [n_I]\} \cup \{x_i^{L_i(k)} : i \in [n_I], k \in [d(v'_i)]\} \cup \{z_i^p : i \in [n_I], p \in [n_I^2]\}$ et $A = \{(r_i, r_{i+1}) : i \in [n_I - 1]\} \cup \{(r_i, x_i^{L_i(1)}) : i \in [n_I]\} \cup \{(x_i^{L_i(k)}, x_i^{L_i(k+1)}) : i \in [n_I], k \in [d(v'_i) - 1]\} \cup \{(x_i^{L_i(d(v'_i))}, z_i^1) : i \in [n_I]\} \cup \{(z_i^p, z_i^{p+1}) : i \in [n_I], p \in [n_i^2 - 1]\}$ (voir la Figure 3.5 pour une illustration). Concrètement, les sommets de type r_i dans G représentent les sommets de V_I et constituent la spine du comb-graph G, que l'on enracine en r_1. Pour chaque sommet de type r_i dans G, il existe un chemin "vertical" qui est d'abord composé de sommets de type x_i^h – qui représentent les arêtes incidentes au sommet de type v'_i dans G_I – puis de n_I^2 sommets de type z_i^p. On décrit maintenant la fonction de coloration $\mathrm{col} : V \to \mathcal{C}$. Premièrement, on attribue une couleur unique à tous les sommets de type r_i et z_i^p dans G. Deuxièmement, pour tout $h \in [m_I]$, on attribue la même couleur $c(h)$ aux deux sommets $x_{i_1}^h$ et $x_{i_2}^h$ qui appartiennent à V. Comme $x_{i_1}^h$ et $x_{i_2}^h$ représentent les deux sommets v'_{i_1} et v'_{i_2} qui sont incidents à e_h dans G_I, il existe exactement deux sommets de couleur $c(h)$ dans G pour tout $h \in [m_I]$. De plus, pour tout $i \in [n_I]$, on observe que les sommets de type x_i^h – les seuls à ne pas avoir une couleur unique – sont ordonnés dans G en fonction de la valeur de h, ce qui implique que $\mathcal{H}$ est bien un DAG. Pour finir, on assigne un poids unitaire à chaque arc $a \in A$. Clairement, l'instance $(G, \mathcal{C}, \mathrm{col}, w, r)$ qu'on vient de décrire est une instance correcte de UMCA-2 dans laquelle G est un comb-graph. Dans la suite, pour tout $i \in [n_I]$, on appelle Z_i le chemin induit par les sommets de type z_i^j.

On explique maintenant comment obtenir une solution $T = (V_T, A_T)$ de UMCA-2 à partir d'une solution I de MIS.

Lemme 33. *Si il existe un ensemble indépendant I de cardinalité k dans G_I, alors il existe une arborescence colorful $T = (V_T, A_T)$ de poids $w(T) \geq k \cdot n_I^2$ dans G.*

Preuve. Soit $V_T = \{r_i : i \in [n_I]\} \cup \{x_i^{L_i(j)} : (i \in [n_I] \,|v'_i \in I) \wedge (j \in [d(v'_i)])\} \cup \{z_i^p : (i \in [n_I] \,|v'_i \in I) \wedge (p \in [n_I^2])\}$ et T l'arborescence couvrante de V_T dans G. Dans un premier temps, on observe que T est colorful. En effet, si deux sommets $x_{i_1}^h$ et $x_{i_2}^h$ de même couleur appartenaient à V_T, alors I ne serait pas indépendant puisque les sommets $v'_{i_1}, v'_{i_2} \in I$ sont incidents à la même arête e_h dans G_I. Dans un deuxième temps, on rappelle que T contient k chemins de type Z_i comportant chacun n_I^2 sommets. Par conséquent, $w(T) \geq k \cdot n_I^2$. □

On explique maintenant comment obtenir une solution I de MIS à partir d'une solution $T = (V_T, A_T)$ de UMCA-2.

Lemme 34. *Si il existe une arborescence colorful $T = (V_T, A_T)$ de poids W dans G, alors il existe un ensemble indépendant I de cardinalité $k \geq \left\lceil \frac{W+1-n_I-m_I}{n_I^2} \right\rceil$ dans G_I.*

Preuve. On construit l'ensemble I de la manière suivante : pour tout $i \in [n_I]$, s'il existe $p \in [n_I^2]$ tel que $z_i^p \in V_T$, alors $v'_i \in I$. En d'autres termes, I est composé de chaque sommet de type v'_i dont l'indice

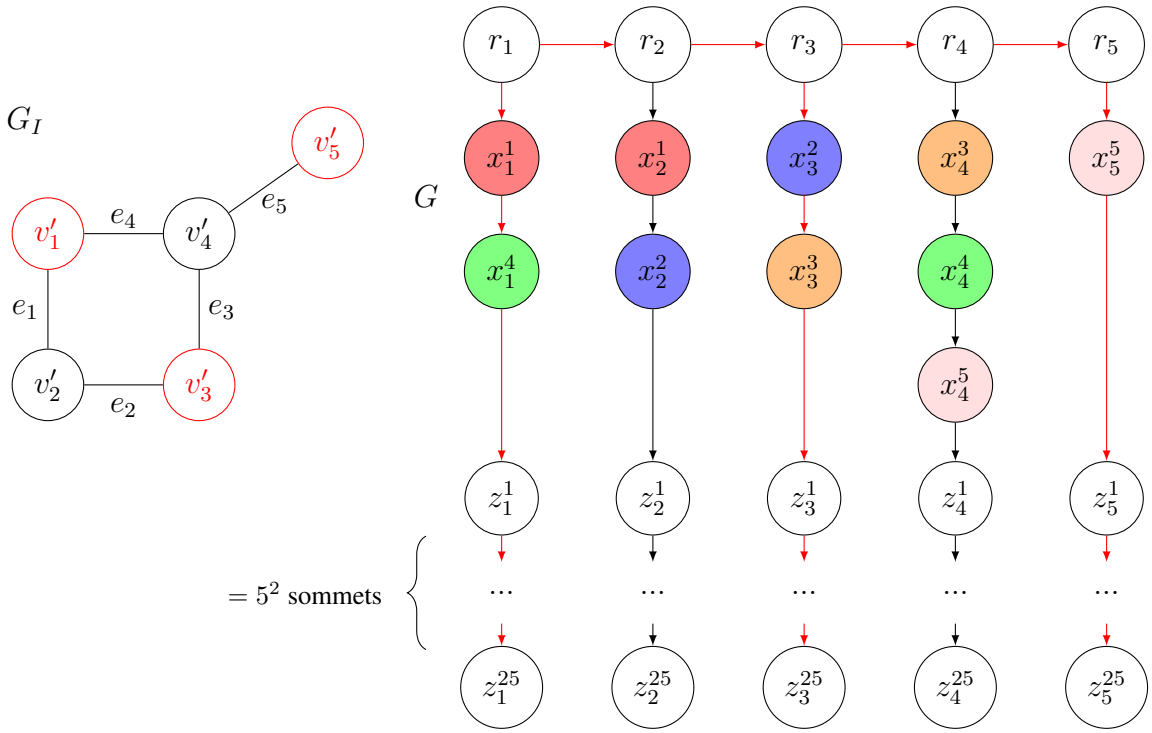


Figure 3.5 – Construction d'une instance de UMCA-2 (à droite) à partir d'une instance de MIS (à gauche). Les arêtes de G_I sont nommées $e_1, e_2, \ldots, e_5$. Chaque sommet de G non-coloré dans la figure ci-dessus possède une couleur unique. Par souci de clarté, les poids unitaires ne sont pas représentés. L'ensemble de sommets $I = \{v'_1, v'_3, v'_5\}$ en rouge est une solution de MIS dans G_I et correspond à la solution T de UMCA-2 dans G dont les arcs sont en rouge.

i correspond à celui d'au moins un sommet de type z^p_i dans V_T. On montre tout d'abord que I est un ensemble indépendant. Pour cela, on suppose qu'il existe deux sommets v'_{i_1} et v'_{i_2} voisins dans G_I et qui appartiennent à I. Par construction de G, il existe dans ce cas un indice $h \in [m_I]$ et deux sommets $x^h_{i_1}$ et $x^h_{i_2}$ qui appartiennent à V_T tel que $col(x^h_{i_1}) = col(x^h_{i_2})$, ce qui contredit le fait que T est colorful. Par conséquent, I est un ensemble indépendant.
On prouve maintenant que $k \geq \left\lceil \frac{W+1-n_I-m_I}{n_I^2} \right\rceil$. Premièrement, on note qu'on peut toujours étendre T à une autre arborescence colorful $T' = (V_{T'}, A_{T'})$ telle que les conditions suivantes sont satisfaites : (i) $V_T \subseteq V_{T'}$, (ii) pour tout $i \in [n_I]$, $r_i \in V_{T'}$ et (iii) si un sommet de type z^p_i appartient à V_T, alors tout le chemin Z_i appartient à T'. L'arborescence $V_{T'}$ contient donc n_I sommets de type r_i, $k \cdot n_I^2$ sommets de type z^p_i et au plus m_I sommets de type x^h_i. Comme tous les arcs ont un poids de 1, on obtient donc $w(T') \leq n_I + m_I + k \cdot n_I^2 - 1$. On conclut la preuve en rappelant que $W \leq w(T')$. □

Les Lemmes 33 et 34 permettent maintenant de prouver le théorème suivant.

Théorème 35. *Le problème* UMCA-2 *restreint aux comb-graphs ne peut pas être approximé en temps polynomial avec un ratio de* $\mathcal{O}(n^{\frac{1}{3}-\epsilon})$*, pour tout* $\epsilon > 0$.

Preuve. Supposons que UMCA-2 soit approximable en temps polynomial avec un ratio ρ. Dans ce cas, s'il existe une solution optimale de UMCA-2 de poids W^* dans G, alors on peut également obtenir une solution approchée de poids $W \geq \frac{W^*}{\rho}$ dans G. Selon le Lemme 33, s'il existe une solution optimale de cardinalié k^* dans G_I, alors $W^* \geq k^* \cdot n_I^2$. Par substitution, on obtient $W \geq \frac{k^* \cdot n_I^2}{\rho}$. Maintenant, selon le Lemme 34, s'il existe une solution de poids W dans G, alors il existe une solution de cardinalité $k \geq \frac{W+1-n_I-m_I}{n_I^2}$ dans G_I. Par substitution encore, on obtient $k \geq \frac{\frac{k^* \cdot n_I^2}{\rho}+1-n_I-m_I}{n_I^2}$. On note que $\frac{m_I+n_I-1}{n_I^2} \leq 1$ puisque $m_I \leq \frac{n_I(n_I-1)}{2}$. On obtient donc $k \geq \frac{k^*}{\rho} - 1$. Par conséquent, si UMCA-2 est approximable en temps polynomial avec un

ratio ρ, alors MIS est approximable en temps polynomial avec un ratio ρ. On observe maintenant que la réduction proposée est linéaire et que $n = |V(G)| = \mathcal{O}(n_I^3)$. De plus, Zuckerman a prouvé que MIS ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n_I^{1-\epsilon})$, pour tout $\epsilon > 0$ [123]. Ainsi, le problème UMCA-2 restreint aux comb-graphs ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{3}-\epsilon})$, pour tout $\epsilon > 0$. □

Dans la preuve ci-dessus, si on remplace chaque chemin vertical composé de n_I^2 sommets de type z_i^p par un unique sommet de type z_i et si on fixe tous les poids à 0 à l'exception des arcs entrants de tous les sommets de type z_i, alors on peut utiliser les Lemmes 33 et 34 pour montrer qu'il existe un ensemble indépendant I de cardinalité supérieure ou égale à k dans l'instance G_I de MIS si et seulement si il existe une solution de poids supérieur ou égal à k dans l'instance de MCA$^+$-2 qu'on vient de décrire. Dans cette nouvelle réduction, on observe que $n = \mathcal{O}(n_I^2)$. Par linéarité de la réduction, on obtient le corollaire suivant.

Corollaire 36. *Le problème* MCA$^+$-2 *restreint aux comb-graphs ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$.*

De manière similaire au Théorème 31 qui concernait le cas où G est un arbre, le Théorème 35 et le Corollaire 36 donnent des ratios d'inapproximabilité très elevés – tous deux en fonction de n – pour MCA quand G est un comb-graph. Nous montrons cependant un résultat positif sur l'approximation de UMCA dans les comb-graphs, qui repose sur le fait que $\mathcal{H}$ est un DAG. Pour cela, nous prouvons tout d'abord le lemme intermédiaire suivant.

Lemme 37. *Soit $G = (V, A)$ un comb-graph enraciné en $r \in V$ et p un entier quelconque. Si $d(r, v) \leq p$ pour tout sommet $v \in V$, alors G contient au plus $(p + 1)^2$ sommets.*

Preuve. Par définition d'un comb-graph, on note qu'on peut décrire un comb-graph G par sa spine, dont on appelle ici S l'ensemble de sommets, ainsi que par un ensemble de chemins indépendants dans G, appelés chemins *pendants*, tels que seule une des deux extrémités de chaque chemin pendant appartient à la spine de G.

Dans un premier temps, on construit un comb-graph G dont on souhaite maximiser le nombre de sommets n et tel que $d(r, v) \leq p$ pour tout $v \in V$. Pour cela, on crée tout d'abord deux sommets $v_a \in S$ et $v_b \notin S$ tel que r appartient au chemin pendant de v_a à v_b. On note $a \leq p$ (resp. $b \leq p$) la distance de r à v_a (resp. de r à v_b) dans G. Ainsi, la distance de v_a à tout autre sommet $v \in S$ est au plus égale à $p - a$. Par conséquent, pour tout $v \in S$ tel que $d(v_a, v) = c$ avec $c \in \{1, \ldots, p - a\}$, on remarque que le chemin pendant de v est de longueur au plus égale à $p - a - c$. On montre un exemple dans la Figure 3.6.

Dans un deuxième temps, on calcule le nombre maximum de sommets que le comb-graph G construit ci-dessus puisse contenir. Tout d'abord, on remarque que $n \leq 1 + a + b + 2 \cdot \sum_{c=1}^{p-a} c$, ce qui donne $n \leq 1+a+b+2 \cdot \frac{(p-a-1)(p-a)}{2}$ et $n \leq (p-a)^2+p+b+1$. Maintenant, on observe que a est nécessairement nul – et donc que $r \in S$ – si on veut maximiser n. Pour finir, on note que $n \leq p^2 + 2p + 1$ puisque $b \leq p$, ce qui mène à $n \leq (p + 1)^2$ et conclut cette preuve. □

Le Lemme 37 nous permet maintenant de démontrer la proposition suivante.

Proposition 38. *Le problème* UMCA *restreint aux comb-graphs peut être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{2}})$.*

Preuve. Dans la suite, on suppose que G contient n sommets. D'après le Lemme 37 ci-dessus, si $d(r, v) \leq \sqrt{n} - 2$ pour tout $v \in V$, alors G peut au plus contenir $(\sqrt{n} - 2 + 1)^2$ sommets, ce qui contredit le fait que G contienne n sommets. Par conséquent, dans tout graphe d'entrée G d'une instance de MCA, il existe au moins un sommet $v \in V$ tel que $d(r, v) \geq \sqrt{n} - 1$. Maintenant, on rappelle que toute solution de UMCA dans G possède un poids au plus égal à $n - 1$. De plus, on rappelle que tout chemin dans G à partir de la racine r est colorful puisque $\mathcal{H}$ est un DAG. Par conséquent, on vient de montrer que UMCA admet un algorithme d'approximation qui renvoie le plus long chemin – nécessairement colorful – à partir de la racine et qui possède un ratio d'approximation de $\mathcal{O}(\frac{n}{n^{\frac{1}{2}}}) = \mathcal{O}(n^{\frac{1}{2}})$. □

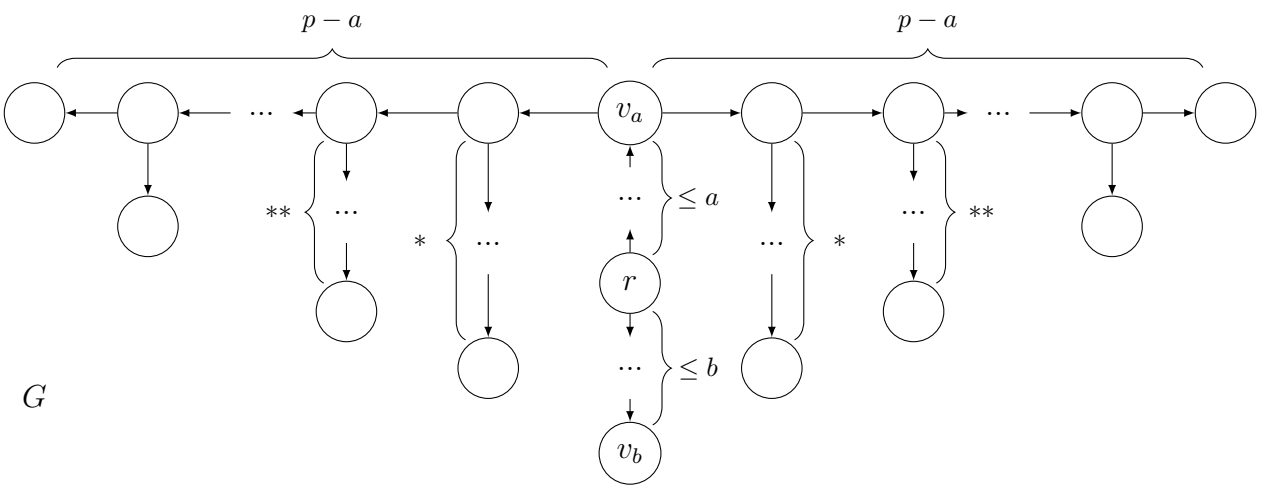


Figure 3.6 – Construction d'un comb-graph $G = (V, A)$ enraciné en r tel que $d(r, v) \leq p$ pour tout $v \in V$ avec p un entier quelconque. Les accolades représentent les distances maximum entre deux sommets, où $*$ signifie "$\leq p - a - 1$" et où $**$ signifie "$\leq p - a - 2$".

On rappelle que le Théorème 35 n'interdit pas l'existence d'un algorithme d'approximation en temps polynomial de ratio $\mathcal{O}(n^{\frac{1}{3}})$ pour UMCA dans les comb-graphs. Par conséquent, déterminer si un tel algorithme existe demeure une question ouverte.

Bien que l'algorithme d'approximation de la Proposition 38 ne s'applique pas aux comb-graphs avec des poids non-uniformes, on peut tout de même montrer un algorithme d'approximation trivial en temps polynomial pour MCA$^+$ dans de tels graphes.

Proposition 39. *Le problème* MCA$^+$ *restreint aux comb-graphs peut être approximé en temps polynomial avec un ratio de* $\mathcal{O}(|\mathcal{C}|)$.

Preuve. Soit T une solution de MCA$^+$ dans G et soit W le poids maximum attribué à une arête de G. On remarque que $w(T) \leq W \cdot (|\mathcal{C}| - 1)$ puisque T ne peut pas contenir plus de $|\mathcal{C}|$ sommets. De plus, on rappelle que tout chemin de r à un sommet $v \in V$ est colorful puisque $\mathcal{H}$ est un DAG. Par conséquent, l'algorithme d'approximation que nous proposons consiste simplement à prendre un chemin arbitraire à partir de r qui inclut l'arc de poids maximum de G. □

3.2.3 Résultats de complexité quand le graphe d'entrée est une superstar

Au lieu de contraindre davantage le degré maximum d'un arbre G comme dans la Section 3.2.2, nous proposons dans cette section de contraindre davantage la *hauteur* de G. En effet, le problème MCA appartient trivialement à P dans les *stars*, dans lesquelles chaque arc a pour sommet d'origine la racine du graphe. On décide donc d'étudier le problème MCA dans les *superstars*, des arbres enracinés de hauteur 2 qu'on peut voir comme des superstars auxquelles on a supprimé leurs *feuilles*. Malheureusement, on montre dans un premier temps que UMCA-2 est APX-hard dans les superstars.

Théorème 40. *Le problème* UMCA-2 *restreint aux superstars est* APX*-dur.*

Preuve. La preuve consiste en une réduction depuis le problème MAX-2-SAT(3), une variante APX-dure de SAT [6]. On peut la voir comme une extension de la preuve du Théorème 1 dans [18].

MAX-2-SAT(3)

• **Instance**: Un ensemble de variables booléennes $X = \{x_1, \ldots, x_p\}$, une formule ϕ sur un ensemble $C = \{C_1, \ldots, C_q\}$ de clauses de taille 2 contenant des variables de X telle que chaque variable apparaît dans au plus 3 clauses

• **Sortie**: Une affectation $\beta : X \to \{\texttt{vrai}, \texttt{faux}\}$ de chaque $x_i \in X$ qui satisfait k clauses dans ϕ

• **Mesure**: k

Dans ce qui suit, étant une superstar de racine r, on appelle $f(v)$ l'unique voisin entrant de tout sommet $v \in V \setminus \{r\}$. Pour tout $j \in [q]$, soit $l_{j,1}$ et $l_{j,2}$ les deux littéraux qui apparaissent dans la clause C_j. Pour toute instance de MAX-2-SAT(3), on construit une superstar $G = (V, A)$ dont les sommets sont organisés en trois niveaux. La racine r est au niveau 1, deux sommets v_i et $\overline{v_i}$ sont créés pour tout $i \in [p]$ au niveau 2, et deux sommets $C_{j,1}$ et $C_{j,2}$ sont créés pour tout $j \in [q]$ au niveau 3. Il existe un arc de r vers chaque sommet du niveau 2. Pour tout $i \in [p]$ et $j \in [q]$, il existe un arc de v_i (resp. $\overline{v_i}$) vers $C_{j,1}$ si $l_{j,1} = x_i$ (resp. $l_{j,1} = \overline{x_i}$) ou de v_i (resp. $\overline{v_i}$) vers $C_{j,2}$ si $l_{j,2} = x_i$ (resp. $l_{j,2} = \overline{x_i}$). La fonction de coloration $\text{col} : V \to \mathcal{C}$ est définie de la manière suivante : la racine r possède une couleur unique ; pour tout $i \in [p]$, on attribue une même couleur $c(v_i)$ aux deux sommets v_i et $\overline{v_i}$; pour tout $j \in [q]$, on attribue une même couleur $c(C_j)$ aux deux sommets $C_{j,1}$ et $C_{j,2}$. Clairement, chaque couleur apparaît au plus deux fois dans G, et $\mathcal{H}$ est un DAG puisque deux sommets sont de même couleur si et seulement si ils appartiennent au même niveau. Enfin, on donne un poids de 1 à chaque arc de G. Un exemple de réduction est donné dans la Figure 3.7.

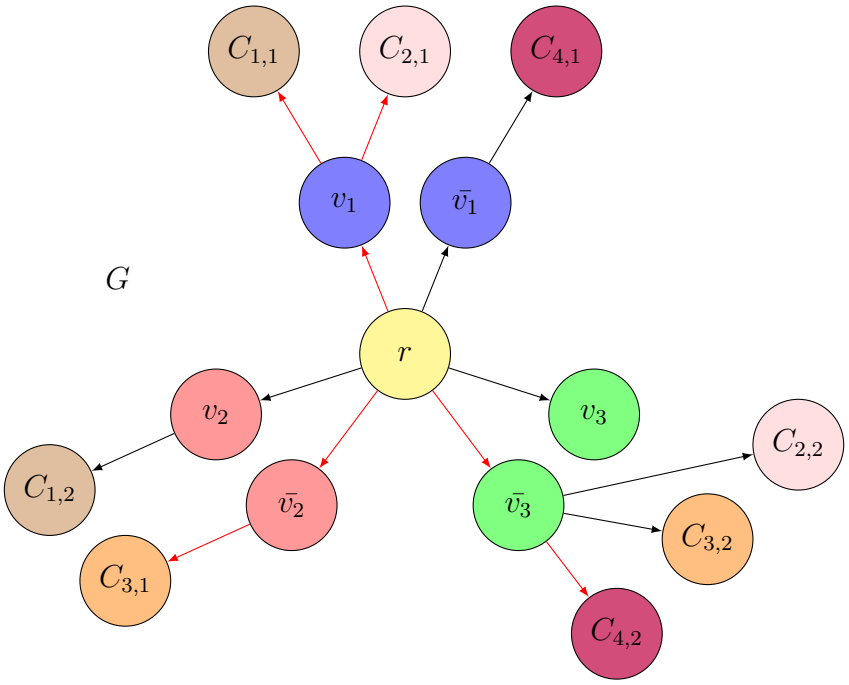


Figure 3.7 – Construction d'une instance de UMCA-2 à partir de l'instance de MAX-2-SAT(3) suivante : $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3})$. Par souci de clarté, les poids unitaires ne sont pas représentés. L'affectation $\beta = \{x_1 = \texttt{vrai}, x_2 = \texttt{faux}, x_3 = \texttt{faux}\}$ satisfait ϕ et correspond à la solution T de UMCA-2 dans G dont les arcs sont en rouge.

On montre maintenant qu'il existe une affectation β de X qui satisfait k clauses dans ϕ si et seulement si il existe une arborescence colorful $T = (V_T, A_T)$ de poids $w(T) \geq p + k$ dans G.

($\Rightarrow$) On suppose qu'il existe une affectation β de X qui satisfait k clauses dans ϕ. Soit $S_t = \{v_i : i \in [p] \mid \beta(x_i) = \texttt{vrai}\}$ et $S_f = \{\overline{v_i} : i \in [p] \mid \beta(x_i) = \texttt{faux}\}$. On définit $V_T = \{r\} \cup S_t \cup S_f \cup \{C_{j,1} : j \in [q] \mid f(C_{j,1}) \in (S_t \cup S_f)\} \cup \{C_{j,2} : j \in [q] \mid (f(C_{j,2}) \in (S_t \cup S_f)) \wedge (f(C_{j,1}) \notin (S_t \cup S_f))\}$ avec T

l'arborescence couvrante de $G[V_T]$. Par construction, il ne peut pas exister $j \in [q]$ et $h \in \{1,2\}$ tels que $C_{j,h} \in V_T$ et $f(C_{j,h}) \notin V_T$. Par conséquent, T est connexe. De plus, puisque β satisfait k clauses, il existe k sommets distincts de type $C_{j,h}$ qui appartiennent à T, en plus des p sommets qui appartiennent à $S_t \cup S_f$ et de la racine r. Par conséquent, T est clairement colorful et $w(T) = p + k$.

($\Leftarrow$) On suppose qu'il existe une arborescence colorful $T' = (V_{T'}, A_{T'})$ de poids $w(T') \geq p + k$ dans G. Si $V_{T'}$ ne contient pas p sommets appartenant au niveau 2, alors il est toujours possible de l'étendre à un ensemble V_T tel que $V_{T'} \subseteq V_T$, tel que V_T est colorful et tel que V_T contient p sommets du niveau 2. Soit T l'arborescence couvrante de $G[V_T]$. Puisque T est colorful, pour tout $i \in [p]$, exactement un des deux sommets v_i et $\overline{v_i}$ appartient à V_T. Par conséquent, pour tout $i \in [p]$, si $v_i \in V_T$ (resp. $\overline{v_i} \in V_T$), alors on pose $\beta(x_i) = \texttt{vrai}$ (resp. $\beta(x_i) = \texttt{faux}$). On montre maintenant que β satisfait au moins k clauses dans ϕ. Tout d'abord, pour tout $j \in [q]$ et pour tout $h \in \{1,2\}$, si le sommet $C_{j,h}$ appartient à V_T alors on observe que $f(C_{j,h})$ appartient nécessairement à V_T et, par construction, que la clause C_j est satisfaite par β. De plus, pour tout $j \in [q]$, les deux sommets $C_{j,1}$ et $C_{j,2}$ ne peuvent pas tous les deux appartenir à V_T puisque V_T est colorful. Puisque $w(V_T) \geq p + k$, V_T contient nécessairement au moins k sommets du niveau 3 et β satisfait donc au moins k clauses dans ϕ.

Pour conclure cette preuve, on rappelle que $k \leq q$ puisqu'on ne peut pas satisfaire plus de clauses qu'il n'en existe. On note aussi que $2q \leq 3p$ puisque chaque variable apparaît au plus 3 fois dans ϕ par définition et que chaque clause est de taille 2. On obtient donc $p \geq \frac{2q}{3} \geq \frac{2k}{3}$ et $p + k \geq \frac{2k}{3} + k \geq \frac{5k}{3}$. Par conséquent, il existe une affectation β de X qui satisfait k clauses dans ϕ si et seulement si il existe une arborescence colorful $T = (V_T, A_T)$ de poids $w(T) \geq \frac{5k}{3}$ dans G. Puisque la réduction proposée est linéaire et puisque MAX-2-SAT(3) est APX-dur [6], on vient de prouver que UMCA-2 est également APX-dur dans le cas où le graphe d'entrée est une superstar. □

Bien que le Théorème 40 implique qu'il n'existe pas d'algorithme PTAS pour UMCA-2 dans les superstars, la structure particulière de ces arbres nous permet d'obtenir un algorithme d'approximation en temps polynomial de ratio constant, que nous décrivons ci-dessous.

Proposition 41. *Le problème* UMCA-2 *restreint aux superstars est 2-approximable.*

Preuve. Dans la suite, on suppose tout d'abord que la racine r d'une superstar G est également le *centre* de G, *i.e.* on suppose qu'il n'existe pas d'autre sommet $z \in V$ tel que $d(z,v) < d(r,v)$ pour tout sommet $v \in V \setminus \{r\}$. Le cas où r n'est pas le centre de G sera étudié ensuite. On rappelle qu'on peut visualiser une superstar comme un graphe à trois niveaux de la manière suivante : la racine r appartient au niveau 1 ; l'ensemble de sommets $V_2 = N^+(r)$ appartient au niveau 2 ; l'ensemble de sommets $V_3 = N^+(V_2)$ appartient au niveau 3. Pour toute couleur $c \in \mathcal{C}$, on définit $V_c = \{v \in V : \text{col}(v) = c\}$. Enfin, on rappelle que le poids de tous les arcs de G est égal à 1 dans une instance de UMCA-2. Ainsi, s'il existe deux sommets de même couleur $v_2 \in V_2$ et $v_3 \in V_3$ tels que v_3 appartient à une solution $T = (V_T, A_T)$ de UMCA-2 dans G, alors on peut substituer v_3 par v_2 dans V_T sans diminuer le poids de T. Dans la suite, on suppose donc sans perte de généralité que $\text{col}(V_2) \cap \text{col}(V_3) = \emptyset$.

Pour tout $1 \leq |col(N^+(r))| \leq |\mathcal{C}| - 1$, on montre par induction qu'il existe une arborescence $T = (V_T, A_T)$ de poids $w(T) \geq \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil$ dans G. Tout d'abord, si $|col(N^+(r))| = 1$, alors V_2 contient soit un sommet, soit deux sommets de même couleur, puisqu'aucune couleur ne peut apparaître plus de deux fois dans une instance de UMCA-2. Dans le premier cas, l'unique sommet de V_2 possède un arc vers tous les sommets de V_3. Dans le deuxième cas, au moins un des deux sommets de V_2 possède un arc vers au moins $\left\lceil \frac{|\mathcal{C}|}{2} \right\rceil - 1$ sommets de couleurs distinctes dans V_3. Dans les deux cas, il existe donc une arborescence T triviale de poids $w(T) \geq \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil$.
Pour tout $k \in [|\mathcal{C}|-2]$, si $|col(N^+(r))| = k$, alors on suppose qu'il existe une arborescence $T = (V_T, A_T)$ de

poids $w(T) \geq \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil$ dans G. On considère maintenant une instance de UMCA-2 telle que $|col(N^+(r))| = k+1$ dans G. Soit c une couleur arbitraire de $\mathcal{C}$ à partir de laquelle on définit deux ensembles de sommets $V^+ = \{r\} \cup V_c \cup N^+(V_c)$ et $V^- = \{r\} \cup (V_2 \setminus V_c) \cup (V_3 \setminus \{v \in V_3 : \text{col}(v) \in \text{col}(N^+(V_c))\})$, avec $\mathcal{C}^+ = \text{col}(V^+)$ et $\mathcal{C}^- = \text{col}(V^-)$. Par hypothèse de récurrence, on remarque que $G[V^+]$ (resp. $G[V^-]$) admet une solution T^+ (resp. T^-) de UMCA-2 de poids $w(T^+) \geq \left\lceil \frac{|\mathcal{C}^+|}{2} \right\rceil$ (resp. $w(T^-) \geq \left\lceil \frac{|\mathcal{C}^-|}{2} \right\rceil$). On montre que $T = T^+ \cup T^-$ est une solution de UMCA-2 dans G. Premièrement, on rappelle que r appartient nécessairement à la fois à T^+ et à T^-. Comme r est le seul sommet de V à appartenir à la fois à V^+ et V^-, cela implique que T est une arborescence dans G. Deuxièment, par construction, on remarque que $\mathcal{C} = \mathcal{C}^+ \oplus (\mathcal{C}^- \setminus \{\text{col}(r)\})$. Par conséquent, T est colorful et $w(T) = w(T^+) + w(T^-)$, ce qui donne $w(T) \geq \left\lceil \frac{|\mathcal{C}^+|}{2} \right\rceil + \left\lceil \frac{|\mathcal{C}^-|}{2} \right\rceil$ et finalement $w(T) \geq \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil$.

Pour finir, si $z \neq r$ est le centre de G, alors on note τ le chemin de r à z et on applique le raisonnement ci-dessus pour obtenir une arborescence colorful T' enracinée en z dans $G[V \setminus (V(\tau) \setminus \{z\})]$ telle que $w(T') \geq \left\lceil \frac{|\mathcal{C} \setminus \text{col}(\tau)|}{2} \right\rceil$. De plus, on rappelle que la couleur de chaque sommet de τ est unique puisque $\mathcal{H}$ est un DAG. Par conséquent, on obtient une solution $T = T' \cup \tau$ de UMCA-2 dans G de poids $w(T) \geq \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil$. □

En effectuant un raisonnement similaire à celui de la preuve de la Proposition 41, on obtient le corollaire suivant, qui généralise le résultat précédent pour tout $p \in \mathbb{N}$.

Corollaire 42. *Le problème* UMCA-P *restreint aux superstars est* P*-approximable.*

3.2.4 Résultats de complexité quand le graphe d'entrée est un caterpillar

On a vu que le problème MCA est APX-dur dans les superstars (voir Théorème 40), alors que MCA peut clairement être résolu par des algorithmes de complexité polynomiale dans les stars. On a également vu que MCA est hautement inapproximable dans les comb-graphs (voir Théorème 35), alors que MCA peut clairement être résolu en temps polynomial dans les chemins. On rappelle maintenant qu'un *caterpillar* est un arbre qui devient un chemin si on lui supprime ses feuilles. En fait, un caterpillar peut également être vu comme un ensemble de stars dont les racines sont reliées par un chemin. Déterminer la complexité de MCA quand G est un caterpillar nous permet donc d'affiner la frontière entre les instances de MCA qui peuvent être résolues en temps polynomial et celles pour lesquelles ce n'est pas le cas.

Proposition 43. *Le problème* MCA *restreint aux caterpillars appartient à* P.

Preuve. Le but de la preuve est uniquement de montrer un algorithme de complexité polynomiale pour MCA dans les caterpillars ; ici la complexité n'a pas été optimisée. On définit $S = \{v \in V \mid d^+(v) \neq 0\} \cup \{r\}$ la *spine* du caterpillar G à laquelle on rajoute r si ce sommet n'y appartient pas déjà. Par construction, $G[S]$ est clairement connexe. On décrit maintenant les étapes de l'algorithme proposé. Premièrement, on génère tous les sous-ensembles colorful $S' \subseteq S$ tels que $r \in S'$ et tels que $G[S']$ est connexe. Deuxièmement, pour chacun de ces sous-ensembles S', on note $N^{++}(S') = \{v \in N^+(S') \mid v \notin S\}$, on crée un ensemble $S'' = S'$ et on procède de la manière suivante : pour chaque couleur $c \in \text{col}(N^{++}(S'))$, on détermine le sommet $x \in N^{++}(S')$ de couleur c dont l'arc entrant a_x est de poids maximum et on ajoute x à S'' si et seulement si $w(a_x) > 0$. On construit ensuite l'arborescence couvrante de chacun de ces ensembles S'', et on renvoie celle de poids maximum en sortie de l'algorithme.
Clairement, l'algorithme décrit ci-dessus est correct puisqu'on a généré tous les sous-ensembles $S' \subseteq S$ colorful et connexes et qu'étendre ces sous-ensembles S' aux feuilles d'un caterpillar peut être effectué en temps polynomial par un algorithme glouton. De plus, on note qu'il existe $\mathcal{O}(n^2)$ – où n est le nombre de sommets de G – sous-ensembles de sommets S' puisque r possède au plus deux voisins dans $G[S']$. Enfin, on observe que la longueur de tout chemin dont r est le sommet d'origine est au plus égale à n. Par conséquent, la complexité de l'algorithme présenté est polynomiale. □

3.2.5 Résultat de complexité quand le graphe de hiérarchie de couleurs est un arbre

On observe que les résultats négatifs obtenus jusqu'ici quand on contraint le graphe d'entrée G font écho à ceux obtenus dans la littérature pour le problème de décision GRAPH MOTIF, qui est NP-dur dans les comb-graphs et les superstars, et qu'on peut résoudre en temps polynomial dans les caterpillars (voir Section 2.3, page 43). Cependant, l'introduction du graphe $\mathcal{H}$ de hiérarchie de couleurs nous permet d'obtenir des algorithmes d'approximation dans les instances où la version décision de MCA est NP-dure. Par conséquent, il nous semble intéressant d'exploiter davantage les spécificités apportées aux instances de MCA par le graphe de hiérarchie de couleurs.

Tout d'abord, on peut aisément remarquer que $\mathcal{H}$ est un arbre si G est un arbre colorful puisque les deux graphes sont dans ce cas *isomorphes*. Néanmoins, il est important de préciser que G (resp. $\mathcal{H}$) n'est pas nécessairement un arbre si $\mathcal{H}$ (resp. G) est un arbre (voir la Figure 3.8 pour une illustration).

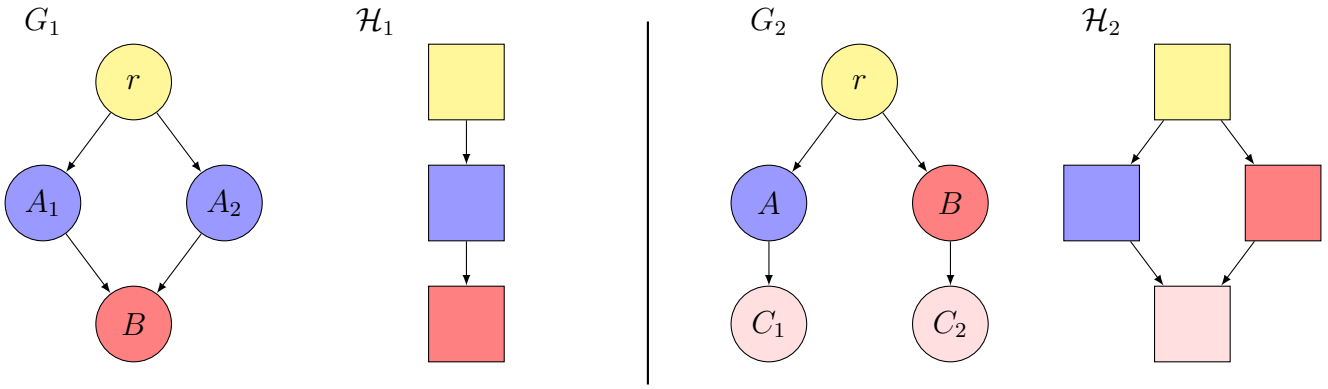


Figure 3.8 – Illustration des relations entre deux graphes d'entrée G_1 et G_2 de MCA et leur graphe de hiérarchie de couleurs respectif $\mathcal{H}_1$ et $\mathcal{H}_2$. On observe que $\mathcal{H}_1$ est un arbre alors que G_1 n'en est pas un, tandis que G_2 est un arbre alors que $\mathcal{H}_2$ n'en est pas un.

Dans la Section 3.2.1, on a vu que MCA est NP-dur, et même hautement inapproximable, quand G est un arbre (voir Théorème 31). En revanche, on montre maintenant que MCA appartient à P quand $\mathcal{H}$ est un arbre. Pour cela, on présente et prouve la validité de la règle de réduction suivante.

Règle de réduction 44. *Soit $I = (G, \mathcal{C}, \mathrm{col}, w, r)$ une instance de* MCA. *Si I contient une couleur $c \in \mathcal{C}$ telle que (i) c ne possède pas de voisin sortant dans $\mathcal{H}$ et (ii) c possède un unique voisin entrant $c^- \neq \mathrm{col}(r)$ dans $\mathcal{H}$, alors on exécute les instructions suivantes:*

- *pour tout sommet $v^- \in V(G)$ de couleur c^-, ajouter $\max\{0, \max_{v \in N^+(v^-) \mid \mathrm{col}(v)=c}\{w(v^-, v)\}\}$ au poids de chaque arc entrant de v^-;*
- *supprimer tous les sommets de couleur c dans $V(G)$ ainsi que tous les arcs incidents à un sommet de couleur c dans $A(G)$.*

Lemme 45. *La Règle de réduction 44 est correcte.*

Preuve. Soit $I' = (G', \mathcal{C}', \mathrm{col}', w', r')$ l'instance de MCA obtenue après l'application de la Règle de réduction 44 sur I. On prouve qu'il existe une arborescence colorful $T = (V_T, A_T)$ de poids au moins égal à W dans I si et seulement si il existe une arborescence colorful $T' = (V_{T'}, A_{T'})$ de poids au moins égal à W dans I'. On montre uniquement le premier sens de l'équivalence ; le deuxième est obtenu en utilisant un raisonnement symétrique. Tout d'abord, si T ne contient aucun sommet de couleur c, alors on peut trivialement définir $T' = T$. Autrement, si T contient un sommet v de couleur c et un arc (u, v), alors on définit $V_{T'} = V_T \setminus \{v\}$ et $A_{T'} = A_T \setminus \{(u, v)\}$. Clairement, T' est une arborescence colorful dans G'. De plus, on observe que $w(T) = w'(T')$ puisque $w'(u^-, u) = w(u^-, u) + w(u, v)$, où u^- est voisin entrant de u dans T. □

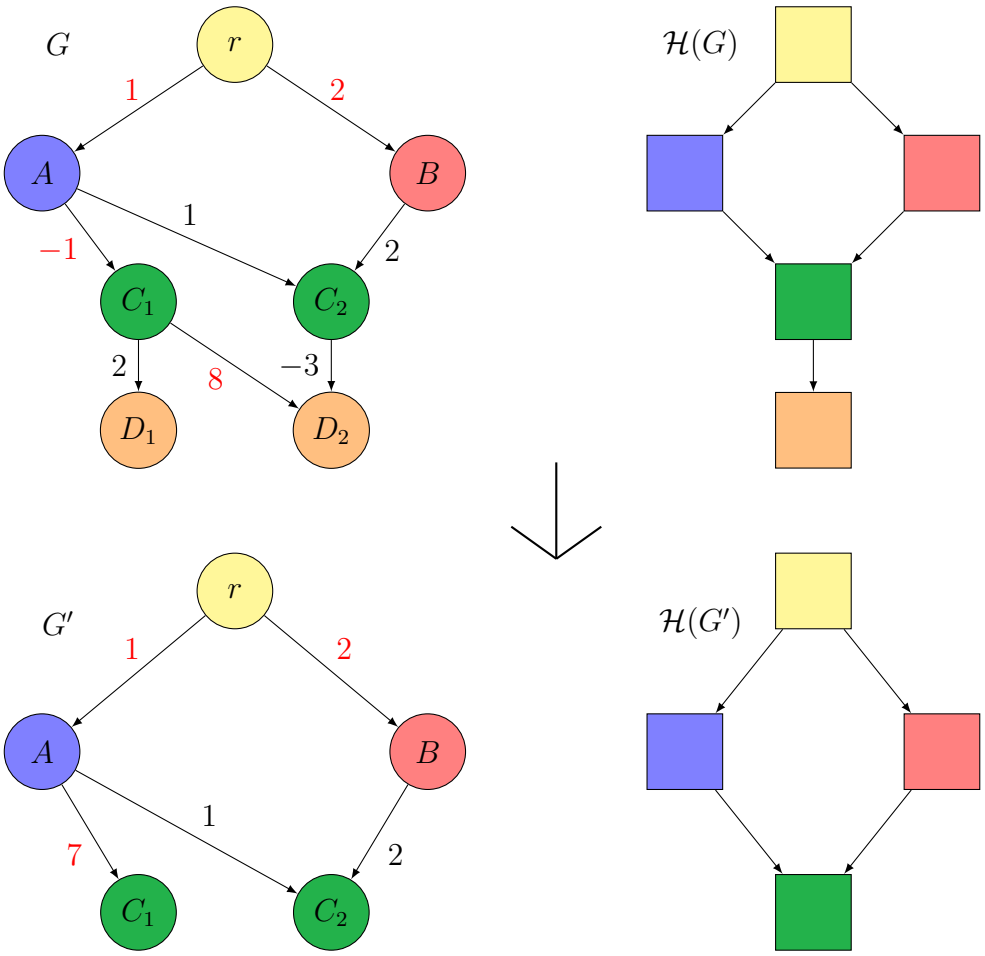


Figure 3.9 – Exemple d'application de la Règle de réduction 44 sur la couleur orange d'une instance initiale de MCA dont le graphe d'entrée G est en haut à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G)$ est en haut à droite. On crée une nouvelle instance de MCA dont le graphe d'entrée G' est en bas à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G')$ est en bas à droite. On initialise $G' = G$, puis on supprime tous les sommets de couleur orange de G' (ainsi que leurs arcs incidents), et enfin on affecte les valeurs $w'(A, C_1) = -1 + \max\{0, \max\{2, 8\}\} = 7$, $w'(A, C_2) = 1 + \max\{0, -3\} = 1$ et $w'(B, C_2) = 2 + \max\{0, -3\} = 2$ dans G'. Dans chacun des deux graphes G et G', les arcs d'une solution de MCA sont représentés en rouge.

On montre un exemple d'application de la Règle de réduction 44 dans la Figure 3.9. On utilise maintenant cette règle afin de prouver le résultat suivant.

Théorème 46. *Le problème* MCA *restreint au cas où $\mathcal{H}$ est un arbre appartient à* P.

Preuve. Tout d'abord, on note que la Règle de réduction 44 s'exécute en temps polynomial puisque son application supprime au plus $n - 1$ sommets de G et modifie en temps polynomial le poids d'au plus $m - 1$ arcs de G. De plus, on peut exécuter la Règle de réduction 44 au plus $|\mathcal{C}| - 2$ fois à partir d'une instance de MCA initiale. Enfin, si on applique la Règle de réduction 44 jusqu'à ce qu'elle ne puisse plus être appliquée sur une instance initiale $(G, \mathcal{C}, \mathrm{col}, w, r)$ de MCA dans laquelle $\mathcal{H}$ est un arbre, alors on observe que G est une star dans l'instance finale obtenue (voir la Figure 3.10). On peut alors déterminer en temps polynomial une solution $T = (V_T, A_T)$ de MCA dans cette instance finale de la manière suivante : pour toute couleur $c \in \mathcal{C}$, on ajoute à T le sommet $v \in V(G)$ de couleur c et l'arc $(r, v) \in A(G)$ tels que (r, v) est l'arc de poids maximum de r vers un sommet de couleur c dans G. Par conséquent, le problème MCA restreint au cas où $\mathcal{H}$ est un arbre appartient à P. □

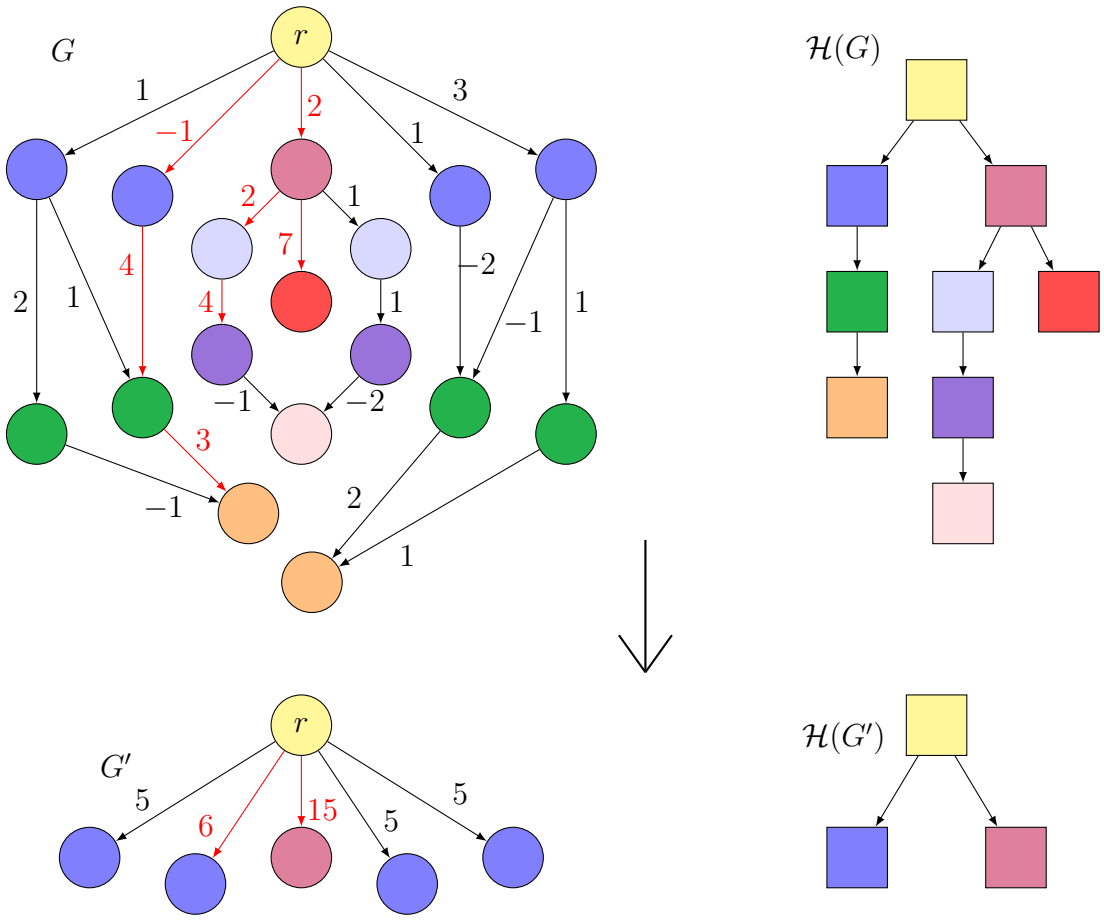


Figure 3.10 – Exemple d'applications successives de la Règle de réduction 44 sur une instance initiale de MCA dont le graphe d'entrée G est en haut à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G)$ est en haut à droite. Comme $\mathcal{H}(G)$ est un arbre, on peut successivement appliquer la Règle de réduction 44 sur la couleur orange, puis vert, rose, violet foncé, bleu clair et enfin rouge (par exemple) jusqu'à obtenir une nouvelle instance de MCA dont le graphe d'entrée G' est en bas à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G')$ est en bas à droite. Dans chacun des deux graphes G et G', les arcs d'une solution de MCA sont représentés en rouge.

3.3 Conclusion

Dans ce chapitre, on a introduit le problème MAXIMUM COLORFUL SUBTREE, dont la résolution a pour but l'identification *de novo* de métabolites inconnus à partir de spectres. On a ensuite observé que le graphe de hiérarchie de couleurs $\mathcal{H}$ du graphe d'entrée de toute instance de MCS est nécessairement un DAG. On a donc modélisé un nouveau problème, MAXIMUM COLORFUL ARBORESCENCE, dont la définition est plus juste que celle de MCS.

On a étudié MCA dans des instances contraintes avec pour premier objectif de distinguer les instances pouvant être résolues en temps polynomial de celles pour lesquelles le problème est NP-dur. Dans les instances où la version décision de MCA est NP-dure, on a obtenu des algorithmes d'approximation en temps polynomial. En particulier, alors que MCA, tout comme MCS, est hautement inapproximable quand le graphe d'entrée G est un arbre, un comb-graph ou encore une superstar, deux des trois algorithmes d'approximation qu'on a obtenu pour MCA dans ces graphes contraints sont uniquement dûs au fait que $\mathcal{H}$ est un DAG. Enfin, on a prouvé que MCA se résout en temps polynomial quand $\mathcal{H}$ est un arbre.

Bien que MCA soit algorithmiquement difficile à résoudre même sur des instances très contraintes, le fait que le problème appartienne à P quand $\mathcal{H}$ est un arbre demande d'étudier plus en profondeur les

avantages potentiels liés à la structure de $\mathcal{H}$. C'est cette étude qu'on va développer dans le Chapitre 4, dans lequel on propose des résultats de complexité paramétrée pour MCA.

4

Résultats de complexité paramétrée pour MAXIMUM COLORFUL ARBORESCENCE

Dans le Chapitre 3, on a introduit le problème MAXIMUM COLORFUL ARBORESCENCE (MCA) et établi le constat suivant : alors que MCA est NP-dur – et même hautement inapproximable – même quand le graphe d'entrée G est très contraint, le même problème peut être résolu en temps polynomial lorsque le graphe de hiérarchie de couleurs $\mathcal{H}(G)$ est un arbre (voir Théorème 46). Dès lors, il est légitime de se demander si MCA admet des algorithmes FPT, et même des kernels polynomiaux, relativement à des paramètres liés à la structure du graphe $\mathcal{H}$. Dans ce chapitre, on commence par définir tous les paramètres d'intérêt qui sont étudiés dans ce chapitre dans la Section 4.1, puis on montre des résultats de complexité paramétrée pour MCA relativement à (des combinaisons de) ces paramètres. En particulier, on s'intéresse dans la Section 4.2.2 au nombre de sommets de degré entrant au moins égal à 2 dans $\mathcal{H}$, puis dans la Section 4.2.3 à la *treewidth* (voir Section 1.1.4, page 16) de la version non-orientée de $\mathcal{H}$. Afin d'obtenir des algorithmes FPT et des kernels polynomiaux, ces paramètres seront parfois associés à des paramètres liés aux répétitions de couleurs dans G, et pour lesquels des algorithmes FPT ont été obtenus dans la Section 4.2.1. On réalise ensuite des expérimentations sur des données biologiques réelles à partir des résultats théoriques obtenus dans ce chapitre et de ceux de la littérature.

Une partie des résultats théoriques obtenus et présentés dans ce chapitre a été publiée lors de la 14ème édition de la conférence Theory and Applications of Models of Computation (TAMC) en 2017 [53] et lors de la 29ème édition de la conférence Combinatorial Pattern Matching (CPM) en 2018 [54].

4.1 Motivations et introduction des paramètres étudiés

Dans cette section, on présente tout d'abord trois paramètres intéressants à étudier pour MCA. Malheureusement, on montre que MCA est W[1]-dur relativement à chacun d'entre eux. On présente ensuite l'unique algorithme de complexité paramétrée disponible dans la littérature – à notre connaissance – pour résoudre MCA. Enfin, on présente trois nouveaux paramètres liés à la structure de $\mathcal{H}$.

Dans le Chapitre 2, on a observé qu'il existe beaucoup de résultats de complexité paramétrée pour GRAPH MOTIF (et ses variantes) relativement à la cardinalité d'une solution et aux répétitions de couleurs dans le graphe d'entrée d'une instance de GRAPH MOTIF. Dans le cadre du problème MCA, on note $\ell_{\mathcal{C}} = n - |\mathcal{C}|$ le nombre de répétitions de couleurs dans G, et on note ℓ le nombre de sommets qui n'appartiennent pas à une solution dans une instance de MCA. Autrement dit, en utilisant le paramètre ℓ on étudie une

version plus contrainte de MCA dans laquelle une solution possède nécessairement au moins $n-\ell$ sommets. Clairement, on observe que $\ell \geq \ell_{\mathcal{C}}$ puisque toute solution de MCA est nécessairement colorful. Par ailleurs, comme on évalue la qualité d'une solution $T = (V_T, A_T)$ en fonction de son poids et non en fonction de la cardinalité de V_T, on définit également le paramètre W^* qui représente le poids d'une solution optimale dans une instance de MCA.

Malheureusement, si on observe plus attentivement la preuve du Théorème 28 (page 63) dans laquelle on effectue une réduction de MAXIMUM INDEPENDENT SET vers MCA, on se rend compte que MCA est W[1]-dur relativement à chacun des trois paramètres présentés ci-dessus, à savoir $\ell_{\mathcal{C}}$, ℓ et W^*. Dans cette preuve, on rappelle dans un premier temps que le graphe G_I d'une instance de MAXIMUM INDEPENDENT SET contient un ensemble indépendant I de cardinalité maximum k si et seulement si G contient un sous-arbre colorful $T = (V_T, A_T)$ de poids $W^* = k$. Par linéarité de la réduction et puisque MAXIMUM INDEPENDENT SET est W[1]-dur relativement à k, on obtient le corollaire suivant.

Corollaire 47. *Le problème* MCA *est* W[1]*-dur relativement au poids W^* d'une solution optimale.*

Dans un second temps, on observe que l'instance de MCA construite dans la réduction est colorful : on en conclut donc que MCA est NP-dur même si $\ell_{\mathcal{C}} = 0$. On obtient donc le corollaire suivant.

Corollaire 48. *Le problème* MCA *est* W[1]*-dur relativement à $\ell_{\mathcal{C}}$.*

Enfin, dans un troisième temps, on observe que $\ell = k$ puisqu'on a montré dans la preuve du Théorème 28 que toute solution de MCA dans G contient nécessairement tous les sommets de G à l'exception des sommets de type x_i qui correspondent aux sommets de I dans G_I. Une nouvelle fois, par linéarité de la réduction et puisque MAXIMUM INDEPENDENT SET est W[1]-dur relativement à k, on obtient le corollaire suivant :

Corollaire 49. *Le problème* MCA *est* W[1]*-dur relativement à ℓ.*

On montre maintenant l'unique algorithme de complexité paramétrée disponible dans la littérature – à notre connaissance – pour résoudre MCA. Dans [18], Böcker et Rasche ont présenté un algorithme FPT relativement à $|\mathcal{C}|$ pour MCS en s'inspirant de celui de Scott *et al.* [111]. On rappelle que les instances de MCA sont plus contraintes que celles de MCS. Par conséquent les résultats positifs liés à MCS s'appliquent également à MCA. On présente ici l'algorithme de Böcker et Rasche – qui est donc également valide pour résoudre MCA.

Théorème 50. *([18])* MCS *peut être résolu en temps $\mathcal{O}^*(3^{|\mathcal{C}|})$.*

Preuve. On crée une table de programmation dynamique $W[v, S]$, pour tout $v \in V$ et $S \subseteq \mathcal{C}$, qui contient le poids d'une arborescence colorful $T(v, S)$ de poids maximum, enracinée en v et telle que $\mathrm{col}(V(T(v, S))) = S$. Pour tout $v \in V$ et $c \in \mathcal{C}$, on initialise les entrées $W[v, \{c\}]$ à 0 si $\mathrm{col}(v) = c$ et à $-\infty$ sinon. On remplit ensuite la table W récursivement de la manière suivante :

$$W[v, S] = \max \begin{cases} \max\limits_{u \in V \mid (\mathrm{col}(u) \in S) \wedge ((v,u) \in A)} W[u, S \setminus \{\mathrm{col}(v)\}] + w(v, u) & (i) \\ \max\limits_{S_1, S_2 \subseteq S \mid (S_1 \cap S_2 = \{\mathrm{col}(v)\}) \wedge (S_1 \cup S_2 = S)} W[v, S_1] + W[v, S_2] & (ii) \end{cases}$$

où $w(v, u) = -\infty$ si $(v, u) \notin A$. Concrètement, la ligne (i) calcule $T(v, S)$ en ajoutant un arc (v, u) à une arborescence enracinée en $u \in N^+(v)$ et telle que $\mathrm{col}(u) \in S$. La ligne (ii) calcule $T(v, S)$ parmi les paires d'arborescences $T(v, S_1)$ et $T(v, S_2)$ dont v est le seul sommet commun, dont $\mathrm{col}(v)$ est la seule couleur commune et dont l'union de S_1 et S_2 est égale à S.

Pour déterminer une solution de MCA dans G, on calcule exhaustivement les entrées de type $W[v, S]$ pour tout $v \in V$ et tout $S \subseteq \mathcal{C}$. Le poids d'une solution de MCA est ensuite donné par la case $W[r, S]$ de valeur maximum parmi tous les sous-ensembles $S \subseteq \mathcal{C}$. On note que cette solution de MCA est correcte, puisqu'on a calculé exhaustivement le poids maximum de $T(v, S)$ pour tout $S \subseteq \mathcal{C}$ dans G. On montre

maintenant que la complexité temporelle d'un algorithme de remplissage de la table W est de $\mathcal{O}^*(3^{|\mathcal{C}|})$. Pour chaque entrée de type $W[v, S]$, on note en effet que cet algorithme effectue $\mathcal{O}(n)$ calculs à la ligne (i), puisque v possède au plus n voisins sortants. De plus, lors du calcul de la valeur retournée à la ligne (ii), on observe que chaque couleur de $\mathcal{C}$ appartient soit à S_1, à S_2 ou bien à $\mathcal{C} \setminus S$. Puisque l'algorithme est exhaustif et que $S \subseteq \mathcal{C}$, on calcule donc au plus $3^{|\mathcal{C}|}$ combinaisons d'arborescences pour déterminer la valeur d'une entrée de type $W[v, S]$ à la ligne (ii). Par ailleurs, on note que la complexité spatiale de l'algorithme est en $\mathcal{O}^*(2^{|\mathcal{C}|})$, puisqu'on doit retenir la valeur de $W[v, S]$ pour tout $S \subseteq \mathcal{C}$ – et pour tout $v \in V$. □

On rappelle maintenant que, d'après le Théorème 46, MCA appartient à P quand $\mathcal{H}$ est un arbre. Ainsi, les deux paramètres que nous introduisons maintenant sont tous deux liés à la structure de $\mathcal{H}$. Tout d'abord, si $\mathcal{H}$ n'est pas un arbre, alors $\mathcal{C}$ contient au moins une *couleur difficile*, *i.e.* une couleur dont le degré entrant est au moins égal à deux dans $\mathcal{H}$. Dans la suite, on note $\mathcal{X}$ l'ensemble des couleurs difficiles de $\mathcal{H}$ et on considère le paramètre $x_{\mathcal{H}} = |\mathcal{X}|$. On étudie également dans ce chapitre la *treewidth* $x_{\mathcal{H}}$ de la *version non-orientée* de $\mathcal{H}$ (voir Section 1.1.4 page 16 pour une définition de la treewidth et la Section 1.1.3 page 15 pour une définition de la version non-orientée d'un graphe). Quand $\mathcal{H}$ est un arbre, ces deux paramètres sont constants puisque $x_{\mathcal{H}} = 0$ et $t_{\mathcal{H}} = 1$ dans ce cas. Ils permettent ainsi de caractériser la distance de $\mathcal{H}$ à un arbre sous deux angles différents (voir la Figure 4.1).

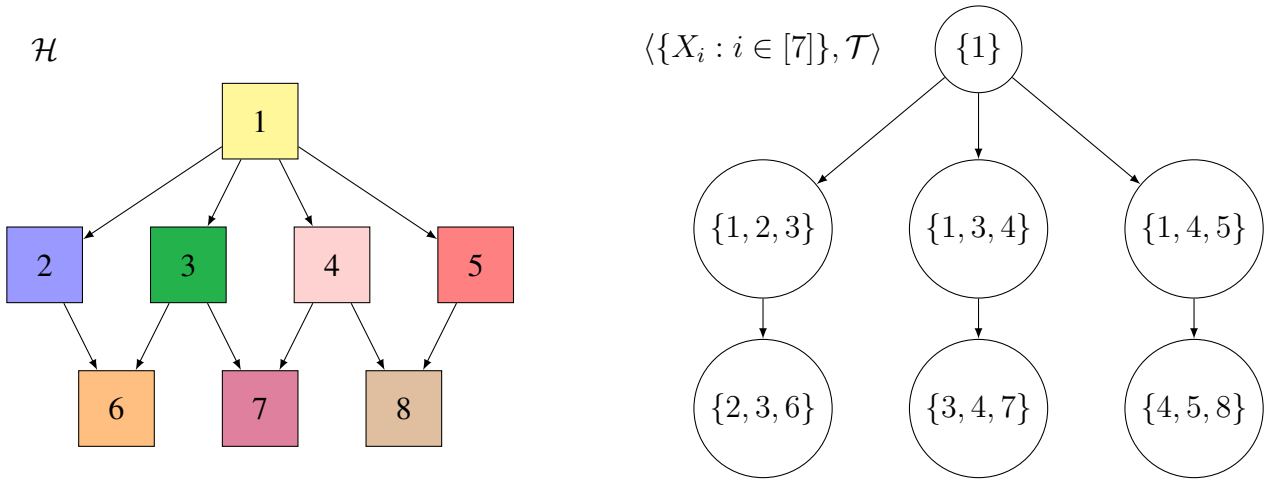


Figure 4.1 – Illustration des relations entre $x_{\mathcal{H}}$ et $t_{\mathcal{H}}$ à partir du graphe de hiérarchie de couleurs $\mathcal{H}$ (à gauche) d'une instance de MCA et d'une décomposition arborescente $\langle \{X_i : i \in [7]\}, \mathcal{T} \rangle$ de $\mathcal{H}$ (à droite) pour laquelle $t_{\mathcal{H}} = 2$. On observe que les couleurs 6,7 et 8 sont des couleurs difficiles de $\mathcal{H}$. Par conséquent, $x_{\mathcal{H}} = 3$ et $x_{\mathcal{H}} > t_{\mathcal{H}}$ dans notre exemple. En revanche, si on définit $\mathcal{C}' = \{1, 2, 3, 6\}$ alors on observe que $\mathcal{H}[\mathcal{C}']$ ne contient plus qu'une seule couleur difficile – la couleur 6 – tandis que toute décomposition arborescente de $\mathcal{H}[\mathcal{C}']$ est nécessairement de treewidth au moins égale à 2, puisque $\mathcal{H}[\mathcal{C}']$ n'est pas un arbre. Puisque $x_{\mathcal{H}} < t_{\mathcal{H}}$ dans ce cas, on en déduit que $x_{\mathcal{H}}$ et $t_{\mathcal{H}}$ sont incomparables en règle générale.

Pour finir, on introduit le paramètre s, qu'on définit comme le nombre minimum d'arcs à supprimer de $\mathcal{H}$ pour que $\mathcal{H}$ soit un arbre. Pour toute instance I de MCA, on rappelle que $\mathcal{H}$ contient un sommet par couleur de I. Par conséquent, $\mathcal{H}$ contient $\mathcal{O}(|\mathcal{C}|^2)$ arcs et $s = \mathcal{O}(|\mathcal{C}|^2)$ puisque $\mathcal{H}$ contient $|\mathcal{C}| - 1$ arcs si $\mathcal{H}$ est un arbre. Il est donc peu vraisemblable que s soit utilisable en pratique, ce qui sera confirmé dans la partie expérimentale de ce manuscrit qui est présentée dans la Section 4.3. On présente toutefois le résultat suivant, dans lequel on note p le degré entrant minimum d'une couleur difficile dans $\mathcal{H}$.

Proposition 51. MCA *peut être résolu en temps* $\mathcal{O}^*(p^{\frac{s}{p-1}})$.

Preuve. On propose un algorithme de branchement récursif basé sur les arcs de $A(\mathcal{H})$ afin d'obtenir toutes les arborescences couvrantes de $\mathcal{H}$. Pour cela, on initialise un ensemble $Z = A(\mathcal{H})$ et on construit un arbre

de recherche T_Z dont chaque sommet correspond à un sous-graphe $\mathcal{H}' = (\mathcal{C}, Z)$ de $\mathcal{H}$. Pour toute couleur difficile $c \in \mathcal{X}$, on branche récursivement sur les $d^-(c)$ cas distincts correspondant au fait qu'un seul arc entrant de c n'est pas supprimé de Z.

Pour tout ensemble Z qui est une feuille de l'arbre de recherche induit T_Z, on observe que toutes les couleurs de Z ont un degré entrant égal à 1 dans $\mathcal{H}'$ et donc que la composante connexe de $\mathcal{H}'$ contenant la couleur $\mathrm{col}(r)$ – où r est la racine de G – est un arbre. Pour chaque feuille Z de T_Z, on crée maintenant un graphe $G' = (V, A_Z)$ avec $A_Z = A \setminus \{(u,v) \in A \mid (\mathrm{col}(u), \mathrm{col}(v)) \notin Z\}$ et on considère la composante connexe G'' de G' qui contient la racine de G. Concrètement, G'' est un sous-graphe de G, construit à partir de Z, et tel que $\mathcal{H}(G'')$ est un arbre. Selon le Théorème 46, on peut donc calculer une solution de MCA T dans G'' en temps polynomial. Maintenant, pour toute solution T de MCA dans G, on observe que le graphe de hiérarchie de couleurs qui est construit uniquement à partir de T est nécessairement un arbre. Par conséquent, calculer une solution de MCA dans sous-graphe $G'' \subseteq G$ correspondant à chaque arbre $\mathcal{H}' \subseteq \mathcal{H}$ permet d'assurer que l'algorithme qu'on vient de décrire est correct.

On calcule maintenant la complexité de l'algorithme présenté. Tout d'abord, on observe que chaque étape de la construction de T_Z est réalisée en temps polynomial. La complexité de l'algorithme présenté est donc uniquement exponentielle relativement au nombre de sommets $|V(T_Z)| = \prod_{c \in \mathcal{X}} d^-(c)$ de T_Z. On montre maintenant que $|V(T_Z)| \leq p^{\frac{s}{p-1}}$ pour prouver que MCA peut être résolu en temps $\mathcal{O}^*(p^{\frac{s}{p-1}})$. Tout d'abord, en supposant que $\mathcal{X}$ n'est pas vide, on cherchele plus petit entier α tel que l'inégalité (1) $d^-(c) \leq \alpha^{d^-(c)-1}$ soit vraie pour tout $c \in \mathcal{X}$. A partir de (1), on obtient $\log(d^-(c)) \leq (d^-(c)-1)\cdot\log(\alpha)$, ce qui donne $\alpha \geq e^{\frac{\log(d^-(c))}{d^-(c)-1}}$ et $\alpha \geq d^-(c)^{\frac{1}{d^-(c)-1}}$. La fonction correspondante $f(u) = u^{\frac{1}{u-1}}$ est monotone et décroissante pour tout $u \in [2; +\infty]$. Par conséquent, par définition de p, on peut fixer α à $p^{\frac{1}{p-1}}$ pour assurer que $\alpha \geq d^-(c)^{\frac{1}{d^-(c)-1}}$ pour tout $c \in \mathcal{C}$. Puisque $|V(T_Z)| \leq \prod_{c \in \mathcal{X}} \alpha^{d^-(c)-1}$ et puisque $s = \sum_{c \in \mathcal{X}} d^-(c) - 1$, on obtient donc $|V(T_Z)| \leq p^{\frac{s}{p-1}}$ et l'algorithme présenté possède une complexité temporelle de $\mathcal{O}^*(p^{\frac{s}{p-1}})$. □

Par exemple, si $p = 3$, alors selon la Proposition 51 on peut résoudre MCA en temps $\mathcal{O}^*(1.733^s)$. En général, on observe que $p \geq 2$ pour toute instance de MCA telle que $\mathcal{H}$ n'est pas un arbre. Par conséquent, en fixant p à 2, on obtent le corollaire "universel" suivant.

Corollaire 52. MCA *peut-être résolu en temps* $\mathcal{O}^*(2^s)$.

4.2 Résultats théoriques

On présente dans cette section des algorithmes FPT et des résultats sur l'existence (ou non) de kernels polynomiaux pour un certain nombre de paramètres présentés dans la Section 4.1. On présente tout d'abord les résultats de complexité paramétrée liés à $\ell_{\mathcal{C}}$ (voir Tableau 4.1). Ceux-ci ont été obtenus en amont de ceux liés à $x_{\mathcal{H}}$ et $t_{\mathcal{H}}$ (voir Tableau 4.2) et nous ont par la suite orientés vers l'obtention de nouveaux résultats combinant $\ell_{\mathcal{C}}$ à $x_{\mathcal{H}}$ ou $t_{\mathcal{H}}$.

Variante	Restriction sur G	Résultat
MCA	-	$\mathcal{O}^*(2^{\ell_{\mathcal{C}}+m^-})$ (Thm. 54)
MCA$^+$	-	$\mathcal{O}^*(2^{\ell_{\mathcal{C}}})$ (Cor. 55)
MCA	G est un arbre	$\mathcal{O}^*(2^{\ell_{\mathcal{C}}})$ (Prop 56)
MCA$^+$	G est un arbre	$\mathcal{O}^*(1.62^{\ell_{\mathcal{C}}})$ (Prop. 57)
UMCA	G est un arbre	$\mathcal{O}^*(1.33^{\ell_{\mathcal{C}}})$ (Prop. 58)
UMCA-2	-	Pas d'algorithme en $\mathcal{O}^*((2-\epsilon)^{\ell_{\mathcal{C}}})$ (Thm. 59)

Tableau 4.1 – Tableau récapitulatif des algorithmes FPT obtenus pour MCA relativement à $\ell_{\mathcal{C}}$ et présentés dans la Section 4.2.1. Ici, $\ell_{\mathcal{C}} = n - |\mathcal{C}|$ et m^- est le nombre d'arcs de poids négatifs dans G.

Paramètre	Statut FPT	Statut kernel
ℓ	W[1]-dur (Cor. 49)	
$x_{\mathcal{H}}$	$\mathcal{O}^*(3^{x_{\mathcal{H}}})$ (Thm. 61)	Pas de kernel polynomial (Thm. 63)
$x_{\mathcal{H}} + \ell_{\mathcal{C}}$	FPT (conséquence du Thm. 61)	Pas de kernel polynomial (Thm. 65)
$x_{\mathcal{H}} + \ell$	Kernel polynomial (Thm. 70)	
$t_{\mathcal{H}}$	W[2]-dur (Thm. 71)	
$t_{\mathcal{H}} + \ell_{\mathcal{C}}$	$\mathcal{O}^*(2^{\ell_{\mathcal{C}}} \cdot 4^{t_{\mathcal{H}}})$ (Thm. 75)	Pas de kernel polynomial (Cor. 77)

Tableau 4.2 – Tableau récapitulatif des résultats de complexité paramétrée liés à $x_{\mathcal{H}}$ et $t_{\mathcal{H}}$ et présentés dans les Sections 4.2.2 et 4.2.3. Ici, $x_{\mathcal{H}}$ est le nombre de sommets de degré entrant au moins égal à 2 dans $\mathcal{H}$, $t_{\mathcal{H}}$ est la treewidth de (la version non-orientée de) $\mathcal{H}$, $\ell_{\mathcal{C}} = n - |\mathcal{C}|$ et $\ell \geq \ell_{\mathcal{C}}$ est le nombre de sommets qui ne font pas partie de la solution.

4.2.1 Etude de MCA en fonction du nombre $\ell_{\mathcal{C}}$ de répétitions de couleurs dans G

Le paramètre $\ell_{\mathcal{C}}$ peut être vu comme un indicateur de la distance de G à $\mathcal{H}$. En effet, on observe que G est isomorphe à $\mathcal{H}$ si et seulement si $\ell_{\mathcal{C}} = 0$. Dans cette section, même si MCA est W[1]-dur relativement à $\ell_{\mathcal{C}}$ selon le Corollaire 48, on montre l'existence d'algorithmes FPT relativement à $\ell_{\mathcal{C}}$ en contraignant la structure de G et/ou la fonction de poids w. En particulier, on prouve que l'un de ces algorithmes est vraisemblablement optimal.

Dans la suite, on appelle sous-graphe *fully-colorful* de G tout sous-graphe de G qui contient *exactement* une occurrence de chaque couleur de G. On calcule dans un premier temps le nombre maximum de sous-graphes fully-colorful de G, puis on montre un algorithme FPT pour MCA relativement à $\ell_{\mathcal{C}}$ et m^-, où m^- est le nombre d'arêtes de poids négatif contenues dans $A(G)$.

Lemme 53. *Tout graphe G avec $|\mathcal{C}|$ couleurs contient au plus $2^{\ell_{\mathcal{C}}}$ sous-graphes fully-colorful.*

Preuve. Pour tout $c \in \mathcal{C}$, on note n_c le nombre de sommets de couleur c dans G. On peut voir que G contient $\prod_{c\in\mathcal{C}} n_c$ sous-graphes fully-colorful. Puisque $x \leq 2^{x-1}$ pour tout $x \in \mathbb{N}$, on obtient $\prod_{c\in\mathcal{C}} n_c \leq 2^{\sum_{c\in\mathcal{C}}(n_c-1)}$ et donc $\prod_{c\in\mathcal{C}} n_c \leq 2^{\ell_{\mathcal{C}}}$ – puisque $\ell_{\mathcal{C}} = \sum_{c\in\mathcal{C}}(n_c - 1)$. □

On prouve maintenant le théorème suivant.

Théorème 54. MCA *peut être résolu en temps $\mathcal{O}^*(2^{\ell_{\mathcal{C}}+m^-})$.*

Preuve. Dans la suite, $G' = (V', A')$ désignera un sous-graphe fully-colorful de G. On dit qu'un sous-ensemble d'arcs de poids négatif $X \subseteq A'$ est *correct* si aucun sommet de V' ne possède strictement plus d'un arc entrant appartenant à X. Pour chaque sous-ensemble correct $X \subseteq A'$, on va dans un premier temps décrire la construction d'un sous-graphe $G'_X = (V'_X, A'_X)$ de G' tel que toute arborescence couvrante T'_X de G'_X contient nécessairement tous les arcs $(v, u) \in X$ pour lesquels il existe un chemin de r vers u dans G'_X. Dans un second temps, on va prouver l'affirmation suivante : si T est une solution de MCA dans G, alors il existe un sous-graphe fully-colorful $G' = (V', A')$ de G, un sous-ensemble correct $X \subseteq A'$ et une solution T'_X de MCA dans G'_X tels que $w(T) = w(T'_X)$.

On explique comment construire $G'_X = (V'_X, A'_X)$ à partir d'un sous-graphe fully-colorful $G' = (V', A')$ et d'un sous-ensemble correct d'arcs de poids négatifs $X \subseteq A'$. Premièrement, on définit $G'_X = G'$. Deuxièmement, pour tout sommet $v \in V'$, si v possède un arc entrant qui appartient à X, alors on supprime de A'_X tous les autres arcs entrants de v. Troisièmement, on supprime tous les sommets $v \in V'_X$ (ainsi que les arcs entrants et sortants qui sont incidents à v dans A'_X) tels qu'il n'existe pas de chemin de r à v dans G'_X ; le graphe G'_X est donc connexe. Maintenant, on observe que toute arborescence couvrante de G'_X contient nécessairement chaque arc de X qui appartient à G'_X puisque tout sommet incident à un arc de X dans G'_X

est nécessairement de degré entrant 1.

On peut maintenant prouver notre affirmation. Soit $T = (V_T, A_T)$ une solution de MCA dans G et soit $G' = (V', A')$ un sous-graphe fully-colorful arbitraire de G tel que $T \subseteq G'$. Comme G' est fully-colorful, on observe qu'il n'existe aucune paire de sommets $u \in V_T$ et $v \in (V' \setminus V_T)$ telle que $w(u, v) > 0$, puisque T ne serait alors pas une solution de MCA dans G' (et donc dans G). Par conséquent, on définit $X = \{a \in A_T \mid w(a) < 0\}$ et on construit G'_X comme décrit ci-dessus. Clairement, T est aussi une arborescence de poids maximum dans G'_X. Par conséquent, calculer de manière exhaustive une solution de MCA pour tout sous-graphe fully-colorful $G' \subseteq G$ et tout sous-ensemble correct X dans un tel graphe G' nous assure de trouver une solution de MCA dans G. Maintenant, on observe qu'il existe au plus 2^{m^-} sous-ensembles d'arcs de poids négatif X dans G et on rappelle que le Lemme 53 nous indique qu'il existe au plus 2^{ℓ_c} sous-graphes fully-colorful de G. De plus, déterminer une arborescence couvrante de poids maximum dans un graphe s'effectue en temps polynomial [29, 48]. Par conséquent, notre théorème est correct puisque toutes les étapes effectuées par notre algorithme s'effectuent en temps polynomial, et ce pour au plus 2^{m^-} ensembles corrects d'arcs de poids négatif dans chacun des 2^{ℓ_c} graphes fully-colorful. □

En fixant $m^- = 0$, le Théorème 54 ci-dessus implique le corollaire suivant.

Corollaire 55. MCA$^+$ *peut être résolu en temps* $\mathcal{O}^*(2^{\ell_c})$.

Dans la suite, pour tout $V' \subseteq V$ et tout $v \in V'$, on note $G_v[V']$ le sous-graphe induit $G[V']$ enraciné en v et dans lequel on supprime tous les sommets $v' \in V'$ tel qu'il n'existe pas de chemin de v vers v'. On montre maintenant que MCA se résout aussi en $\mathcal{O}^*(2^{\ell_c})$ si on impose que le graphe d'entrée G d'une instance de MCA soit un arbre au lieu d'imposer que tous les arcs de G soient positifs.

Proposition 56. MCA *restreint aux arbres peut être résolu en temps* $\mathcal{O}^*(2^{\ell_c})$.

Preuve. On propose un algorithme de branchement récursif basé sur les couleurs de G. Pour cela, on définit $S = V$. Si S n'est pas colorful, on considère $u, v \in S$ tels que $\mathrm{col}(u) = \mathrm{col}(v)$ et on branche récursivement sur deux cas : on supprime soit $V(G_u[S])$, soit $V(G_v[S])$ de S. Maintenant, pour tout S correspondant à une feuille de l'arbre de recherche induit T_S, on observe que $G[S]$ est un arbre colorful et donc que $\mathcal{H}(G[S])$ est un arbre. Par conséquent, pour tout S correspondant à une feuille de T_S, on peut appliquer le Théorème 46 afin d'obtenir une solution de MCA dans $G[S]$ en temps polynomial, et on renvoie celle de poids maximum. Clairement, l'algorithme décrit ci-dessus est correct puisque toute solution de MCA est nécessairement contenue dans l'ensemble S d'au moins une feuille de T_S. Par ailleurs, la complexité en temps de l'algorithme est exponentielle uniquement par rapport au nombre de nœuds de T_S. Comme T_S est un arbre binaire de hauteur $\ell_{\mathcal{C}} = n - |\mathcal{C}|$, son nombre de nœuds ne peut pas excéder 2^{ℓ_c} et en conséquence la complexité en temps de notre algorithme est en $\mathcal{O}^*(2^{\ell_c})$. □

On vient de montrer que MCA se résout en temps $\mathcal{O}^*(2^{\ell_c})$ si tous les arcs de G sont de poids positif (voir Corollaire 55) ou si G est un arbre (voir la Proposition 56). On montre maintenant que cette complexité peut être améliorée si G est un arbre *et* que tous les arcs de G sont de poids positif.

Proposition 57. MCA$^+$ *restreint aux arbres peut être résolu en temps* $\mathcal{O}^*(1.62^{\ell_c})$.

Preuve. Dans la suite, si G est un arbre enraciné en un sommet r, alors on rappelle que $f(v)$ est l'unique voisin entrant de v dans G pour tout $v \in V \setminus \{r\}$. On améliore l'algorithme de branchement proposé dans la preuve de la Proposition 56 en utilisant une règle de branchement différente. A partir d'un ensemble $S = V$, on applique la règle de branchement suivante : s'il existe $u, v \in S$ tels que (i) $\mathrm{col}(u) = \mathrm{col}(v)$ et (ii) $|N^+(u)| > 0$ ou $|N^+(v)| > 0$, alors on branche récursivement sur deux cas : on supprime soit $V(G_u[S])$, soit $V(G_v[S])$ de S.

Pour tout S correspondant à une feuille de l'arbre de recherche induit T_S, on décrit maintenant comment obtenir une solution de MCA$^+$ dans $G[S]$. Soit U_S l'ensemble des sommets dont la couleur est unique dans

S. A partir de la condition (ii) de la règle de branchement, on note que deux sommets $u, v \in S$ sont de même couleur si et seulement si u et v sont tous deux des feuilles de $G[S]$. Par conséquent, $G[U_S]$ est connexe. De plus, comme G est un arbre et que tous les poids des arcs de G sont positifs, les sommets et arcs de $G[U_S]$ appartiennent nécessairement à toute solution $T = (V_T, A_T)$ de MCA$^+$ dans $G[S]$. Maintenant, pour toute couleur $c \in \text{col}(S \setminus U_S)$, on ajoute à V_T le sommet $v \in S$ de couleur c tel que $w(f(v), v)$ est maximum – on note que $f(v)$, le voisin entrant de v dans G, appartient nécessairement à V_T. Finalement, on définit T comme l'arborescence couvrante de $G[V_T]$ (voir Figure 4.2). Clairement, T est connexe, colorful et de poids maximum dans $G[S]$. Ainsi, l'algorithme proposé est correct puisque toute solution de MCA est nécessairement contenue dans le graphe $G[S]$ obtenu à partir d'au moins une feuille de T_S. Pour déterminer la complexité de l'algorithme proposé, on suppose sans perte de généralité que $|N^+(u)| \geq |N^+(v)|$ – et donc que $|N^+(u)| > 0$ – dans la règle de branchement qui est décrite ci-dessus. Ainsi, on note qu'on supprime soit au moins deux sommets – si on supprime $V(G_u[S])$ –, soit au moins un sommet – si on supprime $V(G_v[S])$ – à chaque application de la règle de branchement. Par conséquent, le vecteur de branchement (voir Section 1.3.1, page 25) de l'algorithme ci-dessus est $(2, 1)$ et la complexité en temps de l'algorithme est en $\mathcal{O}^*(1.62^{\ell c})$. □

On montre que l'algorithme donné dans la Proposition 57 peut lui aussi être amélioré quand tous les arcs de G ont un poids de 1.

Proposition 58. UMCA *restreint aux arbres peut être résolu en temps* $\mathcal{O}^*(1.33^{\ell c})$.

Preuve. Dans la suite, pour tout $v \in V$, si v possède un unique voisin sortant dans G et que ce voisin sortant est une feuille dans G, alors v est appelé une *avant-feuille* dans G. De plus, tous les sommets de G qui ne sont ni des feuilles ni des avant-feuilles dans G sont appelés des *sommets-troncs* dans G.

A partir d'un ensemble $S = V$, on applique la règle de branchement suivante : s'il existe $u, v \in S$ tels que (i) $\text{col}(u) = \text{col}(v)$, (ii) u est un sommet-tronc dans $G[S]$ et (iii) v est un sommet-tronc ou une avant-feuille dans $G[S]$, alors on branche récursivement sur deux cas : on supprime soit $V(G_u[S])$, soit $V(G_v[S])$ de S. On note que le vecteur de branchement est $(3, 2)$: en effet on supprime soit au moins trois sommets en supprimant $V(G_u[S])$ – puisque u est un sommet-tronc –, soit au moins deux sommets en supprimant $V(G_v[S])$.

Pour tout S correspondant à une feuille de l'arbre de recherche induit T_S, on note S_1 l'ensemble des sommets-troncs dans $G[S]$, S_2 l'ensemble des avant-feuilles dans $G[S]$ et S_3 l'ensembles des feuilles dans $G[S]$. On commence par montrer que les ensembles de couleurs de ces trois ensembles de sommets sont disjoints deux à deux. Premièrement, si S est une feuille de l'arbre de recherche induit T_S, alors on observe que $\text{col}(S_1) \cap \text{col}(S_2) = \emptyset$, autrement on aurait appliqué la règle de branchement énoncée ci-dessus. Deuxièmement, on note que $G[S_1]$ est colorful puisque S est une feuille de T_S. Enfin, $G[S_1]$ est nécessairement connexe par définition de la règle de branchement. Comme $\text{col}(S_1) \cap \text{col}(S_2) = \emptyset$, il existe donc au moins une solution T de UMCA dans $G[S]$ qui contient tous les sommets de S_1. En effet, si T contient un sommet $v_3 \in S_3$ tel que $\text{col}(v_3) \in \text{col}(S_1)$, alors on peut substituer v_3 par un sommet de même couleur appartenant à S_1 dans T sans diminuer le poids de T – puisque tous les poids sont unitaires et que v_3 est une feuille. Ainsi, on peut supprimer tous les sommets $v_3 \in S_3$ tels que $\text{col}(v_3) \in \text{col}(S_1)$ et supposer sans perte de généralité que $\text{col}(S_1) \cap \text{col}(S_3) = \emptyset$. Enfin, en appliquant un raisonnement similaire, on suppose également sans perte de généralité que $\text{col}(S_2) \cap \text{col}(S_3) = \emptyset$.

Pour tout S correspondant à une feuille de l'arbre de recherche induit T_S, on décrit maintenant comment obtenir une solution $T = (V_T, A_T)$ de UMCA dans $G[S]$. Tout d'abord, on rappelle que tous les sommets de S_1 appartiennent à V_T. On montre maintenant comment déterminer les sommets de S_2 et de S_3 qui appartiennent également à V_T. Pour cela, on définit le problème MAXIMUM MATCHING ci-dessous.

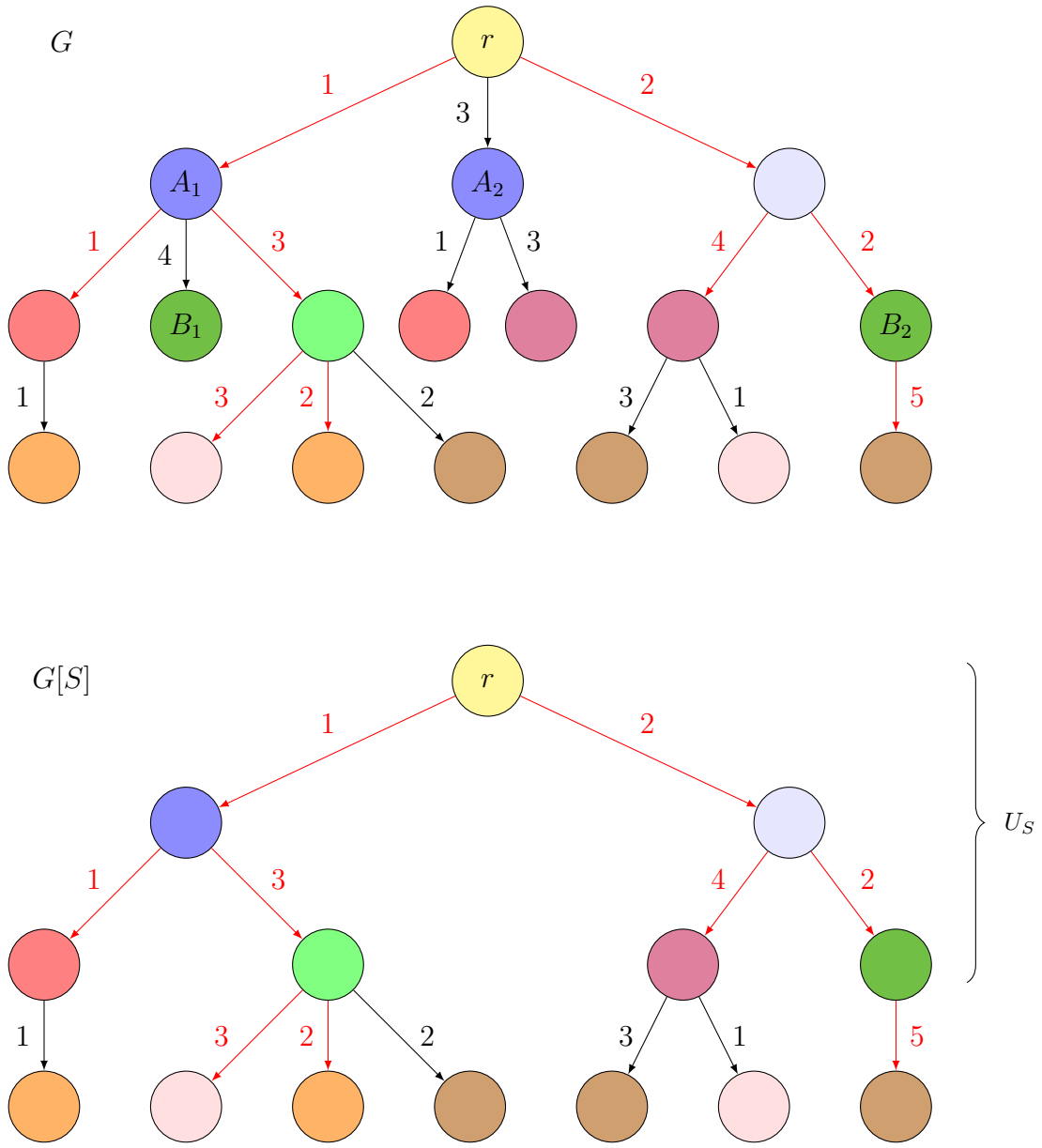


Figure 4.2 – Exemple d'instance de MCA$^+$ obtenue en appliquant l'algorithme de branchement de la Proposition 57. Ici, S est une feuille de l'arbre de recherche induit dans laquelle on a supprimé $V(G_{A_2}[S])$ puis $V(G_{B_1}[S])$. L'ensemble U_S contient tous les sommets de couleur unique dans S. Comme $G[U_S]$ est connexe, on observe que $G[U_S]$ appartient nécessairement à toute solution de MCA$^+$ dans $G[S]$. Pour chaque couleur $c \in \text{col}(S \setminus U_S)$, il ne reste plus qu'à ajouter le sommet de couleur de c dont le poids de l'arc entrant est maximum pour obtenir une solution de MCA. Ainsi, l'arborescence représentée en rouge est une solution de MCA$^+$ dans $G[S]$.

MAXIMUM MATCHING

- **Instance**: Un graphe $G = (V, E)$
- **Sortie**: Un ensemble $M \subseteq E$ d'arêtes deux à deux disjointes
- **Mesure**: $|M|$

Soit M une solution de MAXIMUM MATCHING dans $\mathcal{H}(G[S_2 \cup S_3])$ (voir Figure 4.3). Pour tout arc $(c, c') \in M$, on ajoute à V_T une paire de sommets $u, v \in S$ telle que $\text{col}(u) = c$, $\text{col}(v) = c'$ et $(u, v) \in A(G[S])$. Enfin, pour chaque couleur $c \in \text{col}(S_2)$ telle que $c \notin \text{col}(V_T)$, on ajoute arbitrairement un sommet de S_2 à V_T. Clairement, l'arborescence couvrante T de $G[V_T]$ est connexe et colorful. De plus, on rappelle

que $S_1 \in V_T$ et que $\mathrm{col}(S_2) \subset \mathrm{col}(V_T)$. Ainsi, s'il existe une solution T' de UMCA telle que $w(T') > w(T)$ dans $G[S]$, alors T' contient plus de sommets appartenant à S_3 que T, ce qui contredit le fait que M est une solution de MAXIMUM MATCHING. Par conséquent, T est une solution de UMCA dans $G[S]$. Par ailleurs, l'algorithme proposé est correct puisque toute solution de UMCA est nécessairement contenue dans le graphe $G[S]$ obtenu à partir d'au moins une feuille de T_S. Comme le vecteur de branchement de T_S est $(3, 2)$ et que MAXIMUM MATCHING peut se résoudre en temps polynomial [47], notre algorithme possède une complexité en temps en $\mathcal{O}^*(1.33^{\ell_c})$. □

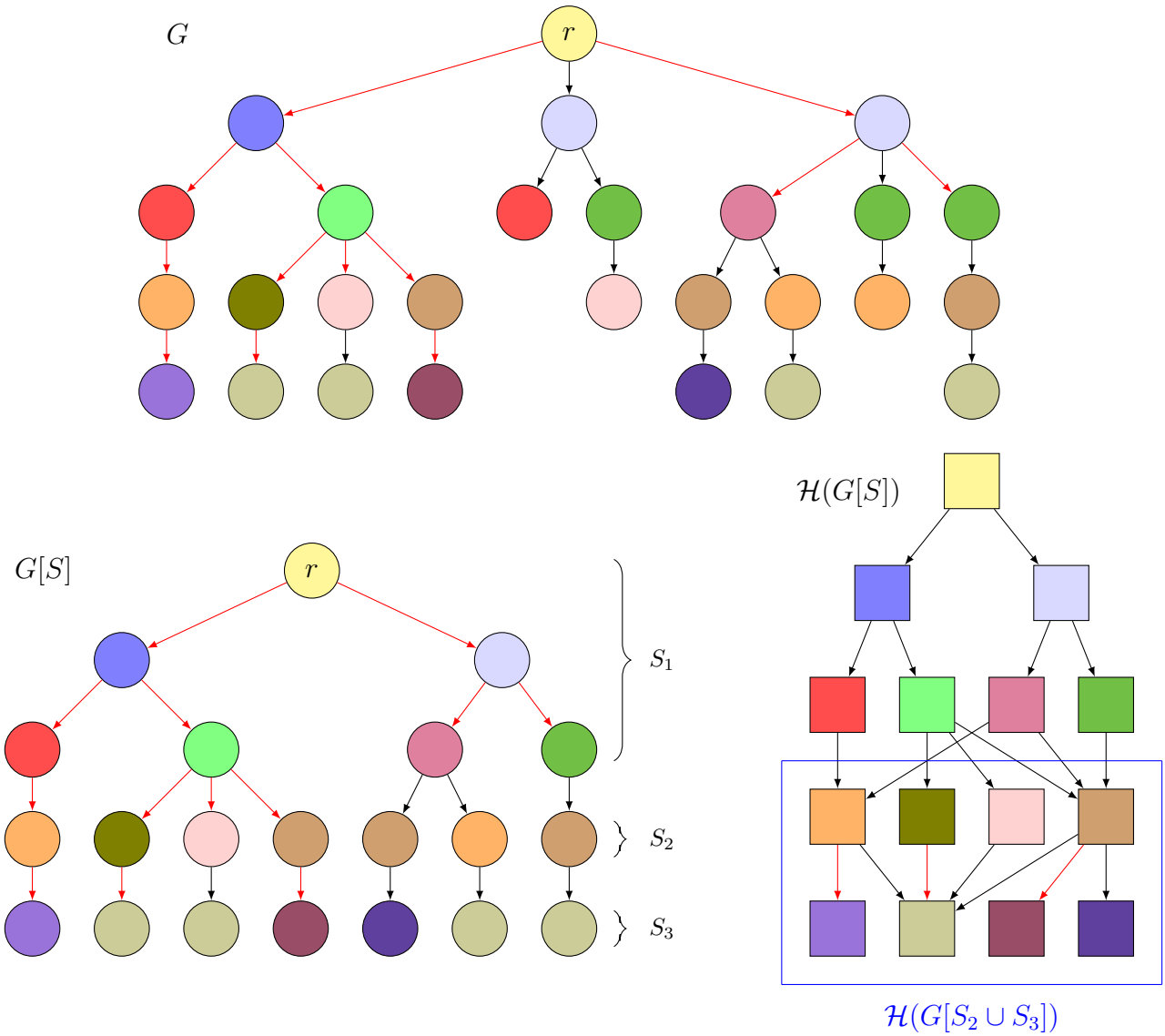


Figure 4.3 – Exemple d'application de l'algorithme de banchement de la Proposition 58. Ici, S est une feuille de l'arbre de recherche induit T_S. Par souci de clarté, les poids unitaires dans G et dans $G[S]$ ne sont pas représentés. On observe que l'arborescence couvrante de $G[S_1]$ appartient nécessairement à toute solution de UMCA dans $G[S]$ puisque tous les sommets de $G[S_1]$ possèdent une couleur unique dans S et que $G[S_1]$ est connexe. Par conséquent, toute solution de UMCA contient également un sommet de couleur c pour tout $c \in \mathrm{col}(S_2)$. Il ne reste plus qu'à résoudre l'instance de MAXIMUM MATCHING $\mathcal{H}(G[S_2 \cup S_3])$ (encadrée en bleu) pour déterminer un ensemble de sommets de S_3 de cardinalité maximum qui appartient à une solution de UMCA.

On utilise maintenant l'hypothèse de complexité SETH (définie page 22) afin d'obtenir une borne inférieure de complexité pour MCA paramétré par ℓ_c. En particulier, le résultat que nous présentons prouve que l'algorithme FPT proposé dans le Corollaire 55 est optimal.

Théorème 59. *Sous l'hypothèse* SETH*,* UMCA-2 *ne peut pas être résolu en temps* $\mathcal{O}^*((2-\epsilon)^{\ell c})$*, pour tout* $\epsilon > 0$*.*

Preuve. La réduction que nous proposons est inspirée de la preuve du Théorème 1 dans [55]. On redonne tout d'abord la modélisation du problème SAT, qui a été présenté page 21.

SAT

• **Instance**: Un ensemble de variables booléennes $X = \{x_1, \ldots, x_p\}$, une formule ϕ sur un ensemble $C = \{C_1, .., C_q\}$ de clauses contenant des variables de X

• **Question**: Existe-t-il une affectation $\beta : X \to \{\texttt{vrai}, \texttt{faux}\}$ de chaque $x_i \in X$ qui satisfait ϕ ?

Pour toute instance ϕ de SAT, on construit un graphe $G = (V, A)$, avec trois niveaux de sommets, de la manière suivante (voir Figure 4.4 pour une illustration). La racine r est au niveau 1, on crée deux sommets v_i et $\overline{v_i}$ pour tout $i \in [p]$ au niveau 2 et on crée un sommet z_j pour tout $j \in [q]$ au niveau 3. On crée ensuite un arc de r vers v_i et vers $\overline{v_i}$ pour tout $i \in [p]$. De même, pour tout $i \in [p]$ et tout $j \in [q]$, on crée un arc de v_i (resp. $\overline{v_i}$) à z_j si et seulement si le littéral x_i (resp. $\overline{x_i}$) apparaît dans la clause C_j. On attribue une couleur unique à chaque sommet appartenant au niveau 1 (resp. 3). Au niveau 2, pour tout $i \in [p]$, les sommets v_i et $\overline{v_i}$ partagent la même couleur $\mathrm{col}(i)$. Par conséquent, chaque couleur $c \in \mathcal{C}$ apparaît au plus deux fois et on note que $\mathcal{H}$ est un DAG puisqu'aucune couleur n'apparaît à deux niveaux différents de G. Pour finir, on attribue un poids unitaire à chaque arc de G. Clairement, $(G, \mathcal{C}, \mathrm{col}, w)$ est une instance de UMCA-2.

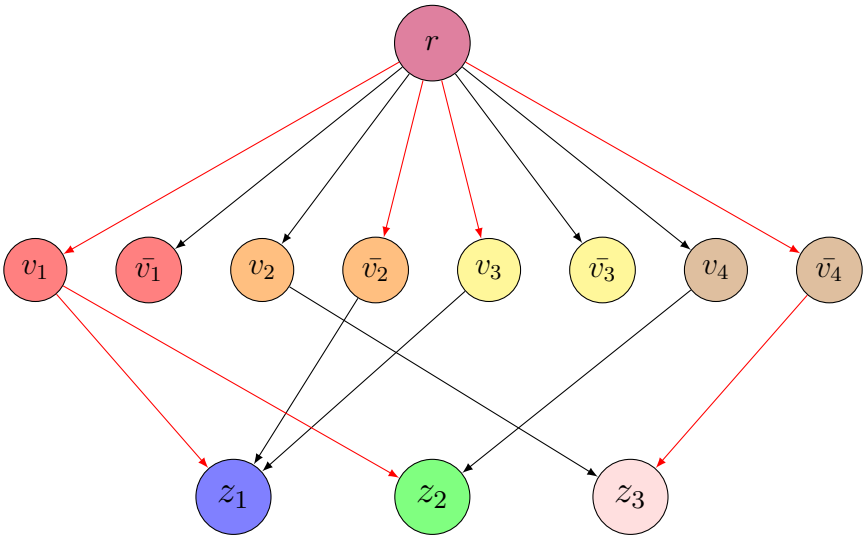


Figure 4.4 – Construction d'une instance de UMCA-2 à partir de la formule $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_4})$ de SAT. Par souci de clarté, les poids unitaires ne sont pas représentés. Les arcs d'une solution optimale T de MCA sont colorés en rouge dans l'instance présentée. A partir des sommets de type v_i^t et v_i^f appartenant à T, on observe que l'affectation $\beta = \{x_1 = \texttt{vrai}, x_2 = \texttt{faux}, x_3 = \texttt{vrai}, x_4 = \texttt{faux}\}$ correspondante satisfait la formule ϕ.

On montre maintenant qu'il existe une affectation β qui satisfait ϕ si et seulement si il existe une arborescence colorful de poids $p + q$ (et donc d'ordre $p + q + 1$) dans G.

($\Rightarrow$) On suppose qu'il existe une affectation β qui satisfait ϕ. Soit $S_t = \{v_i : i \in [p] \mid x_i = \texttt{vrai}$ dans $\beta\}$ et $S_f = \{\overline{v_i} : i \in [p] \mid x_i = \texttt{faux}$ dans $\beta\}$. On définit $V_T = \{r\} \cup S_t \cup S_f$ et T une arborescence couvrante de V_T dans G. On rappelle que r possède un arc sortant vers tous les sommets appartenant au niveau 2. De plus, pour tout $i \in [p]$ et $j \in [q]$, si la clause C_j est satisfaite par la variable x_i (resp. $\overline{x_i}$), alors on rappelle que $v_i \in V_T$ (resp. $\overline{v_i} \in V_T$) par construction et qu'il existe un arc $(v_i, z_j) \in A$.

Par conséquent, T est connexe. Enfin, T est clairement colorful et de poids $p+q$.

($\Leftarrow$) On suppose qu'il existe une arborescence colorful de poids $p+q$ dans G. On note que T contient au plus p sommets appartenant au niveau 2, et donc au moins q sommets appartenant au niveau 3. Puisqu'il existe exactement q sommets au niveau 3, V_T est nécessairement composé de la racine r, d'exactement p sommets appartenant au niveau 2 et de q sommets appartenant au niveau 3. Puisque le niveau 2 contient p paires distinctes de sommets de même couleur et puisque T est colorful, pour tout $i \in [p]$, on observe que V_T contient soit v_i, soit $\overline{v_i}$. On affecte donc les variables de β de la manière suivante : si $v_i \in V_T$ (resp. $\overline{v_i} \in V_T$), alors $x_i = \texttt{vrai}$ (resp. $x_i = \texttt{faux}$) dans β. Pour tout $j \in [q]$, on rappelle qu'au moins un voisin entrant de z_j dans G appartient à T, puisque T est connexe. Par conséquent, l'affectation β implique que chaque clause de ϕ est satisfaite par au moins un littéral.

Puisque $n = 2p+q+1$ et $|\mathcal{C}| = p+q+1$, on observe maintenant que $\ell_{\mathcal{C}} = n - |\mathcal{C}| = p$. Par conséquent, s'il existe un algorithme de complexité en temps $\mathcal{O}^*((2-\epsilon)^{\ell_{\mathcal{C}}})$ pour UMCA-2, alors il existe un algorithme de complexité en temps $\mathcal{O}^*((2-\epsilon)^p)$ pour SAT, ce qui contredit SETH. □

4.2.2 Etude de MCA en fonction des couleurs difficiles de $\mathcal{H}$

On rappelle que $x_{\mathcal{H}}$ est l'ensemble des couleurs difficiles d'une instance de MCA et que MCA appartient à FPT relativement à $|\mathcal{C}|$. Puisque $x_{\mathcal{H}} \leq |\mathcal{C}|$ et puisque MCA se résout en temps polynomial quand $\mathcal{H}$ est un arbre (c'est-à-dire quand $x_{\mathcal{H}} = 0$) selon le Théorème 46 (voir page 75), il est légitime de se demander si MCA appartient à FPT relativement à $x_{\mathcal{H}}$. Dans la suite, pour toute arborescence T dans G, on note $\mathcal{X}(T) = \mathcal{X} \cap \text{col}(V(T))$ l'ensemble de couleurs difficiles des sommets de T. On commence par prouver le lemme suivant.

Lemme 60. *Soit T_1 et T_2 deux arborescences dans $\mathcal{H}$ telles que (i) T_1 est enracinée en c_1, (ii) T_2 est enracinée en $c_2 \neq c_1$ et (iii) $c_1, c_2 \in N^+(c)$ avec $c \in \mathcal{C}$. Si $\mathcal{X}(T_1)$ et $\mathcal{X}(T_2)$ sont disjoints, alors $V(T_1)$ et $V(T_2)$ sont disjoints.*

Preuve. On suppose sans perte de généralité qu'il n'existe pas de chemin de c_2 vers c_1 dans $\mathcal{H}$. Si $V(T_1)$ et $V(T_2)$ ne sont pas disjoints, alors il existe au moins une couleur $c^* \in \mathcal{C}$ qui appartient à la fois à T_1 et à T_2. Afin de prouver que cette couleur c^* ne peut pas exister, on note τ_1 (resp. τ_2) l'ensemble de couleurs du chemin de c_1 (resp. c_2) vers c^* dans T_1 qui inclut c_1 (resp. dans T_2 qui inclut c_2). On remarque que soit $\tau_2 \subset \tau_1$, soit $c_2 \notin \tau_1$ (voir Figure 4.5). Dans le premier cas, si $\tau_2 \subset \tau_1$, alors il existe un sommet $c' \in \tau_1$ tel que $c' \neq c$ avec $(c', c_2) \in A(\mathcal{H})$. Puisque $\mathcal{H}$ contient l'arc (c, c_2), la couleur c_2 est difficile et les deux ensembles $\mathcal{X}(T_1)$ et $\mathcal{X}(T_2)$ ne sont donc pas disjoints. Dans le second cas, si $c_2 \notin \tau_1$, alors $|\tau_1 \cap \tau_2| \geq 1$ puisque $c^* \in \tau_1 \cap \tau_2$. Par conséquent, on définit $\overline{c} \in \tau_1 \cap \tau_2$ tel qu'il existe un chemin de $\overline{c}$ vers chaque autre couleur de $\tau_1 \cap \tau_2$. Selon la définition de $\overline{c}$, l'unique voisin entrant de $\overline{c}$ dans τ_1 est différent de l'unique voisin entrant de $\overline{c}$ dans τ_2. Ainsi, $\overline{c}$ est une couleur difficile, ce qui contredit l'hypothèse que $\mathcal{X}(T_1)$ et $\mathcal{X}(T_2)$ sont disjoints. □

On peut maintenant prouver le théorème suivant.

Théorème 61. MCA *peut être résolu en temps $\mathcal{O}^*(3^{x_{\mathcal{H}}})$ et en espace $\mathcal{O}^*(2^{x_{\mathcal{H}}})$.*

Preuve. Dans la suite, pour tout sommet $v \in V$ qui possède au moins un voisin sortant dans G, on suppose que $\text{col}(N^+(v))$ admet un ordre fixe arbitraire. Ainsi, pour tout $i \in [|\text{col}(N^+(v))|]$, on note $\text{col}^+(v, i)$ la i-ème couleur dans $\text{col}(N^+(v))$. On propose un algorithme de programmation dynamique qui utilise deux tables. La première, $\text{A}[v, \mathcal{X}', i]$, est calculée pour tout $v \in V$, $\mathcal{X}' \subseteq \mathcal{X}$ et $i \in \{0\} \cup [|\text{col}(N^+(v))|]$ et enregistre le poids d'une arborescence $T_{\text{A}}(v, \mathcal{X}', i)$ colorful et de poids maximum dans G telle que

— $T_{\text{A}}(v, \mathcal{X}', i)$ est enracinée en v,
— $(\mathcal{X}(T_{\text{A}}(v, \mathcal{X}', i)) \setminus \{\text{col}(v)\}) \subseteq \mathcal{X}'$, et

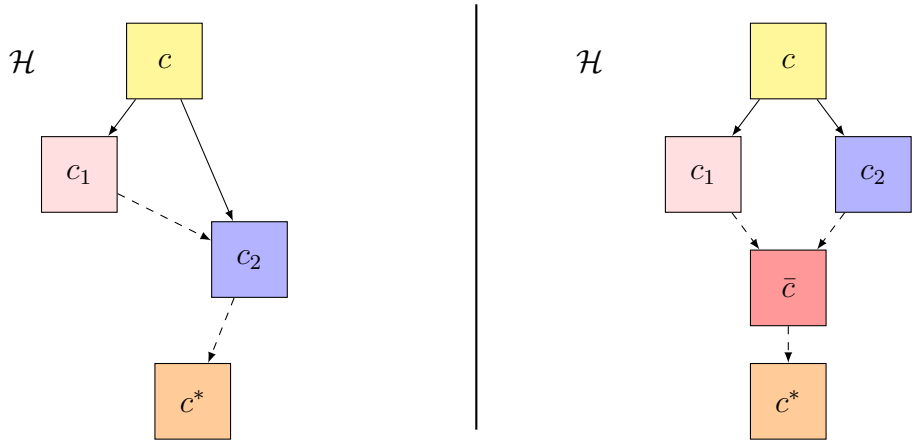


Figure 4.5 – Illustration du Lemme 60. Les arcs en pointillé représentent des chemins entre deux sommets de $\mathcal{H}$. S'il existe quatre couleurs c, c_1, c_2 et c^* appartenant à $V(\mathcal{H})$ tels que (i) c_1 et c_2 sont des voisins sortants de c dans $\mathcal{H}$, (ii) il existe un chemin de c_1 (resp. c_2) à c^* dans $\mathcal{H}$ dont l'ensemble de couleurs est τ_1 (resp. τ_2) et (iii) il n'existe pas de chemin de c_2 à c_1 dans $\mathcal{H}$, alors soit $\tau_2 \subset \tau_1$ (à gauche), soit $c_2 \notin \tau_1$ (à droite). Dans ce dernier cas, il existe alors un sommet $\bar{c} \in \tau_1 \cap \tau_2$ tel qu'il existe un chemin de $\bar{c}$ vers chaque autre couleur de $\tau_1 \cap \tau_2$ – avec la possibilité que $\bar{c} = c^*$.

— si $T_{\mathrm{A}}(v, \mathcal{X}', i)$ contient un arc $(v, u) \in A$, alors $\mathrm{col}(u) = \mathrm{col}^+(v, j)$ avec $j \leq i$.

La deuxième table, $\mathrm{B}[v, \mathcal{X}', i]$, est calculée pour tout $v \in V$, $\mathcal{X}' \subseteq \mathcal{X}$ et $i \in [|\,\mathrm{col}(N^+(v))|]$ et enregistre le poids d'une arborescence $T_{\mathrm{B}}(v, \mathcal{X}', i)$ colorful et de poids maximum dans G telle que
— $T_{\mathrm{B}}(v, \mathcal{X}', i)$ est enracinée en v,
— $(\mathcal{X}(T_{\mathrm{B}}(v, \mathcal{X}', i)) \setminus \{\mathrm{col}(v)\}) \subseteq \mathcal{X}'$, et
— si $T_{\mathrm{B}}(v, \mathcal{X}', i)$ contient un arc $(v, u) \in A$, alors $\mathrm{col}(u) = \mathrm{col}^+(v, i)$.

En résumé, $T_{\mathrm{A}}(v, \mathcal{X}', i)$ et $T_{\mathrm{B}}(v, \mathcal{X}', i)$ partagent la même racine v et le même ensemble de couleurs difficiles $\mathcal{X}'$ (plus éventuellement $\mathrm{col}(v)$), mais v peut posséder plusieurs arcs sortants vers des sommets de $N^+(v)$ dont la couleur est avant $\mathrm{col}^+(i)$ dans $T_{\mathrm{A}}(v, \mathcal{X}', i)$ alors que v possède au plus un arc sortant, dont le sommet d'arrivée doit en outre être de couleur $\mathrm{col}^+(i)$, dans $T_{\mathrm{B}}(v, \mathcal{X}', i)$. Par conséquent, il n'existe aucun sommet $u \in N^+(v)$ tel que $(v, u) \in T_{\mathrm{A}}(v, \mathcal{X}', i-1)$ et $(v, u) \in T_{\mathrm{B}}(v, \mathcal{X}', i)$. On montre maintenant comment remplir les deux tables de programmation dynamique que nous venons de décrire.

$$\mathrm{A}[v, \mathcal{X}', i] = \begin{cases} 0 & \text{si } i = 0, \\ \max\limits_{\mathcal{X}'' \subseteq \mathcal{X}'} \{\mathrm{A}[v, \mathcal{X}'', i-1] + \mathrm{B}[v, \mathcal{X}' \setminus \mathcal{X}'', i]\} & \text{sinon.} \end{cases}$$

Pour toute entrée de type $\mathrm{A}[v, \mathcal{X}', i]$ telle que $i = 0$, on note que $T_{\mathrm{A}}(v, \mathcal{X}', i)$ peut uniquement contenir le sommet v. Par définition, si $i > 0$, alors il ne peut pas exister de sommet $u \in N^+(v)$ tel que u appartienne à la fois à $T_{\mathrm{A}}(v, \mathcal{X}'', i-1)$ et à $T_{\mathrm{B}}(v, \mathcal{X}' \setminus \mathcal{X}'', i)$. Par conséquent, le Lemme 60 indique que $\mathrm{col}(v)$ est l'unique couleur à apparaître à la fois dans $\mathrm{col}(T_{\mathrm{A}}(v, \mathcal{X}'', i-1))$ et dans $\mathrm{col}(T_{\mathrm{B}}(v, \mathcal{X}' \setminus \mathcal{X}'', i))$. L'union de $T_{\mathrm{A}}(v, \mathcal{X}'', i-1)$ et de $T_{\mathrm{B}}(v, \mathcal{X}' \setminus \mathcal{X}'', i)$ est donc une arborescence colorful. Pour finir, on teste exhaustivement tous les sous-ensembles $\mathcal{X}'' \subseteq \mathcal{X}'$ afin d'assurer la validité de la formule.

$$\mathrm{B}[v, \mathcal{X}', i] = \begin{cases} 0 & \text{si } \mathrm{col}^+(v, i) \in \mathcal{X} \setminus \mathcal{X}', \\ \max\limits_{\substack{u \in N^+(v): \\ \mathrm{col}(u) = \mathrm{col}^+(v,i)}} \{0, w(v, u) + \mathrm{A}[u, \mathcal{X}', |\,\mathrm{col}(N^+(u))|]\} & \text{sinon.} \end{cases}$$

Pour toute entrée de type $\mathrm{B}[v, \mathcal{X}', i]$, on rappelle que $\mathrm{B}[v, \mathcal{X}', i]$ enregistre le poids d'une arborescence colorful de poids maximum enracinée en v et qui contient au plus un autre sommet $u \in N^+(v)$

Algorithme 2 CALCUL DES ENTRÉES DES TABLES A ET B

Entrée: Un graphe $G = (V, A)$, deux tables de programmation dynamique A et B vides.
Sortie: Deux tables de programmation dynamique A et B correctement remplies.

pour tout $v \in V$ du dernier au premier sommet d'un ordre topologique arbitraire des sommets de G **faire**
  pour tout $\mathcal{X}' \subseteq \mathcal{X}$ **faire**
    pour tout $i \in \{1, \ldots, |\operatorname{col}(N^+(v))|\}$ **faire**
      Calcul de $\mathrm{B}[v, \mathcal{X}', i]$
    fin pour
  fin pour
  pour tout $\mathcal{X}' \subseteq \mathcal{X}$ **faire**
    pour tout $i \in \{0, \ldots, |\operatorname{col}(N^+(v))|\}$ **faire**
      Calcul de $\mathrm{A}[v, \mathcal{X}', i]$
    fin pour
  fin pour
fin pour

de couleur $\operatorname{col}^+(v, i)$. Si $\operatorname{col}^+(v, i)$ est une couleur difficile qui n'appartient pas à $\mathcal{X}'$, alors on observe que $V(T_{\mathrm{B}}(v, \mathcal{X}', i)) = \{v\}$ et donc que $\mathrm{B}[v, \mathcal{X}', i] = 0$. Sinon, pour chaque sommet $u \in N^+(v)$ de couleur $\operatorname{col}^+(v, i)$, on calcule une arborescence colorful de poids maximum contenant u – et on garde celle de poids maximum – afin de garantir la validité de la formule.

Dans l'Algorithme 2, on montre dans quel ordre remplir toutes les entrées des deux tables de programmation dynamique à partir d'un *ordre topologique* des sommets du DAG G. Une fois que toutes les entrées sont remplies, on note que le poids d'une solution de MCA dans G est contenu dans l'entrée $\mathrm{A}[r, \mathcal{X}, |\operatorname{col}(N^+(r))|]$ – et on peut alors retrouver la solution correspondante en effectuant du backtracking. La complexité totale en temps de l'algorithme provient du fait que celui-ci nécessite au plus $3^{x_{\mathcal{H}}}$ étapes pour calculer une entrée de type $\mathrm{A}[v, \mathcal{X}', i]$ puisqu'une couleur difficile peut appartenir à $\mathcal{X}''$, $\mathcal{X}' \setminus \mathcal{X}''$ ou à $\mathcal{X} \setminus \mathcal{X}'$. Par ailleurs, on rappelle que l'algorithme calcule et stocke toutes les entrées des tables A et B. Ainsi, l'algorithme possède une complexité spatiale en $\mathcal{O}^*(2^{x_{\mathcal{H}}})$. □

Afin de mieux comprendre l'Algorithme 2 présenté ci-avant, on montre dans l'Exemple 62 comment remplir toutes les entrées de A et B de l'Algorithme 2 pour le sommet v appartenant à l'instance de MCA présentée dans la Figure 4.6. Comme indiqué dans la figure, on suppose dans cet exemple que toutes les entrées des tables A et B ont été calculées – et que leur valeur est indiquée dans la figure – pour les voisins sortants du sommet v.

Exemple 62. Exemple d'application de l'Algorithme 2 sur l'instance de MCA présentée dans la Figure 4.6.

$$\begin{aligned}
\mathrm{B}[v, \emptyset, 1] &= \max\{0, \{\max\{w(v, u_1) + \mathrm{A}[u_1, \emptyset, |\operatorname{col}(N^+(u_1))|], w(v, u_2) + \mathrm{A}[u_2, \emptyset, |\operatorname{col}(N^+(u_2))|]\}\} \\
&= \max\{0, \{\max\{4 + 2, 1 + 1\}\} \\
&= 6
\end{aligned}$$

$\mathrm{B}[v, \emptyset, 2] = 0$ puisque $\operatorname{col}^+(v, 2) = \{\bullet\}$ appartient à $\mathcal{X}$.

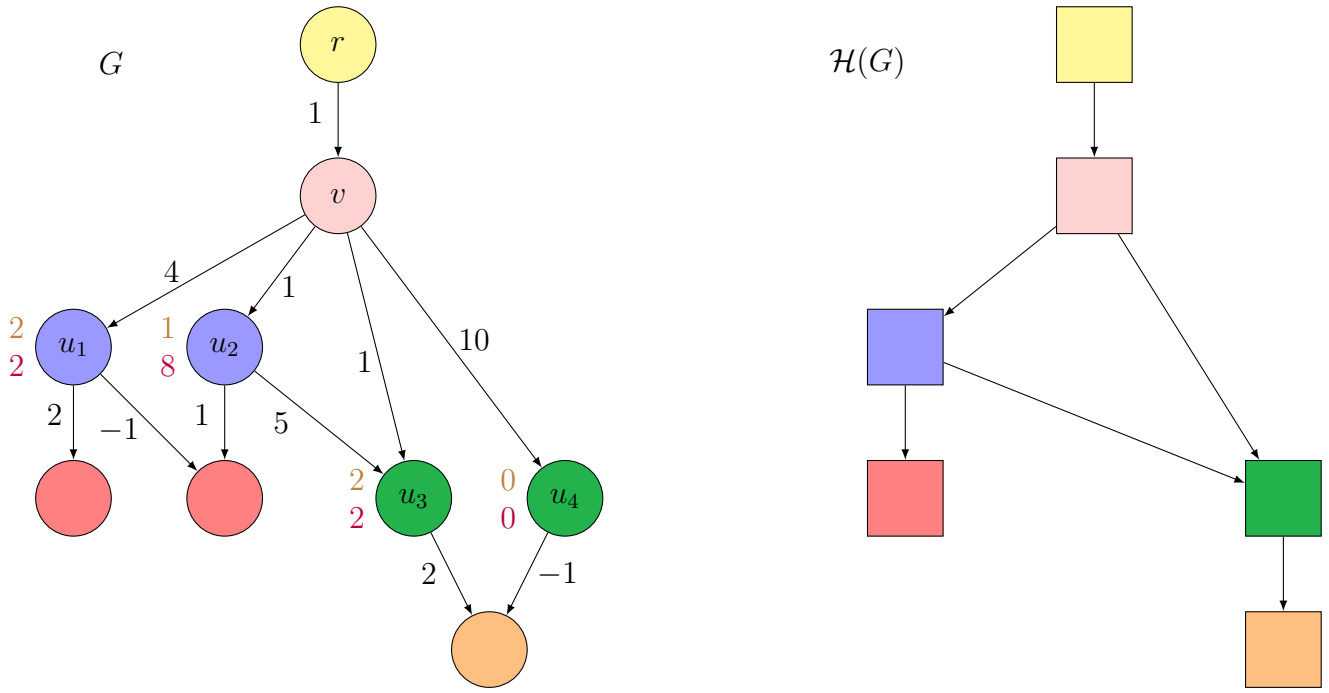


Figure 4.6 – Exemple d'instance de MCA avec le graphe d'entrée G (à gauche) et son graphe de hiérarchie de couleurs $\mathcal{H}$ (à droite) ; l'instance est utilisée pour illustrer l'Algorithme 2. On observe que la couleur verte est la seule couleur difficile de l'instance présentée. Pour tout $u_j \in N^+(v)$, on suppose qu'on a déjà calculé $\mathrm{A}[u_j, \emptyset, |\operatorname{col}(N^+(u_j))|]$ (en marron en haut à gauche de u_j) ainsi que $\mathrm{A}[u_j, \{\bullet\}, |\operatorname{col}(N^+(u_j))|]$ (en violet en bas à gauche de u_j).

$$\begin{aligned}\mathrm{B}[v, \{\bullet\}, 1] &= \max\{0, \{\max\{w(v,u_1) + \mathrm{A}[u_1, \{\bullet\}, |\operatorname{col}(N^+(u_1))|], w(v,u_2) + \mathrm{A}[u_2, \{\bullet\}, |\operatorname{col}(N^+(u_2))|]\}\} \\ &= \max\{0, \{\max\{4+2, 1+8\}\} \\ &= 9\end{aligned}$$

$$\begin{aligned}\mathrm{B}[v, \{\bullet\}, 2] &= \max\{0, \{\max\{w(v,u_3) + \mathrm{A}[u_3, \{\bullet\}, |\operatorname{col}(N^+(u_3))|], w(v,u_4) + \mathrm{A}[u_4, \{\bullet\}, |\operatorname{col}(N^+(u_4))|]\}\} \\ &= \max\{0, \{\max\{1+2, 10+0\}\} \\ &= 10\end{aligned}$$

$$\mathrm{A}[v, \emptyset, 0] = 0$$

$$\begin{aligned}\mathrm{A}[v, \emptyset, 1] &= \mathrm{A}[v, \emptyset, 0] + \mathrm{B}[v, \emptyset, 1] \\ &= 0 + 6 \\ &= 6\end{aligned}$$

$$\begin{aligned}\mathrm{A}[v, \emptyset, 2] &= \mathrm{A}[v, \emptyset, 1] + \mathrm{B}[v, \emptyset, 2] \\ &= 6 + 0 \\ &= 6\end{aligned}$$

$$\mathrm{A}[v, \{\bullet\}, 0] = 0$$

$$\begin{aligned} \mathrm{A}[v, \{\bullet\}, 1] &= \max\{\mathrm{A}[, \emptyset, 0] + \mathrm{B}[v, \{\bullet\}, 1], \mathrm{A}[v, \{\bullet\}, 0] + \mathrm{B}[v, \emptyset, 1]\} \\ &= \max\{0 + 9, 0 + 6\} \\ &= 9 \end{aligned}$$

$$\begin{aligned} \mathrm{A}[v, \{\bullet\}, 2] &= \max\{\mathrm{A}[, \emptyset, 1] + \mathrm{B}[v, \{\bullet\}, 2], \mathrm{A}[v, \{\bullet\}, 1] + \mathrm{B}[v, \emptyset, 2]\} \\ &= \max\{6 + 10, 9 + 0\} \\ &= 16 \end{aligned}$$

On vient de proposer un algorithme FPT relativement à $x_{\mathcal{H}}$ pour MCA dans le Théorème 61. On rappelle que si l'existence d'un tel algorithme implique l'existence d'un kernel pour MCA relativement à $x_{\mathcal{H}}$, la taille de ce kernel n'est pas nécessairement polynomiale par rapport à $x_{\mathcal{H}}$. Prouver l'existence de ce kernel polynomial permettrait de prétraiter efficacement une instance de MCA et, dans le même temps, pourrait mener à la découverte d'un meilleur algorithme FPT pour MCA relativement à $x_{\mathcal{H}}$. Malheureusement, on montre dans le Théorème 63 qu'il n'existe pas de kernel polynomial pour MCA relativement à $x_{\mathcal{H}}$.

Théorème 63. *Sauf si* NP $\subseteq$ coNP/Poly, MCA *restreint aux arbres n'admet pas de kernel polynomial relativement à* $|\mathcal{C}|$*, et par conséquent relativement à* $x_{\mathcal{H}}$*.*

Preuve. Dans la suite, soit t un entier positif. Pour tout $i \in [t]$, soit $G_i = (V_i, A_i)$ le graphe d'entrée, enraciné en r_i, d'une instance de MCA. On suppose que ces t instances de MCA sont construites avec le même ensemble de couleurs $\mathcal{C}' = \{c_1, \ldots, c_{|\mathcal{C}'|}\}$ et telles qu'il existe un ordre topologique de $\mathcal{H}(G_i)$ identique pour tout $i \in [t]$ – si ce n'est pas le cas, on définit un ordre topologique arbitraire des couleurs de $\mathcal{C}'$ et on modifie les couleurs de chacune des t instances pour que leur graphe de hiérarchie de couleurs respecte cet ordre topologique. En particulier, pour tout $i \in [t]$, cela implique que toutes les racines r_i partagent la même couleur de $\mathcal{C}'$.

On effectue maintenant une or-composition (voir Définition 20, page 35) de ces t instances de MCA vers une nouvelle instance de MCA. Soit $G = (V, A)$ le graphe de cette nouvelle instance, avec $V = \{r\} \cup \{v \in V_i \mid i \in [t]\}$ et $A = \{(r, r_i) \mid i \in [t]\} \cup \{(u, v) \in A_i \mid i \in [t]\}$ (voir la Figure 4.7). Ici, la racine r est un sommet qui n'appartient à aucune des t autres instances de MCA et qui possède un arc vers la racine r_i de tout graphe de type G_i ; par conséquent G est un DAG. On note $\mathcal{C}$ l'ensemble des couleurs de G, et on définit $\mathrm{col} : V(G) \to \mathcal{C}$ et $w : A(G) \to \mathbb{R}$ de la manière suivante : on attribue une couleur unique $c_r \notin \mathcal{C}'$ à r ainsi qu'un poids 0 à l'arc $(r, r_i) \in A$ pour tout $i \in [t]$, puis on attribue la même couleur (resp. le même poids) à tous les autres sommets (resp. arcs) de la nouvelle instance que dans leur instance initiale. Clairement, $(G, \mathcal{C}, \mathrm{col}, w, r)$ est une instance correcte de MCA et $|\mathcal{C}| = |\mathcal{C}'| + 1$. De plus, si G_i est une arborescence pour tout $i \in [t]$, alors G est également une arborescence. On montre maintenant qu'il existe un indice $i \in [t]$ tel que G_i contient une arborescence colorful $T = (V_T, A_T)$ enracinée en r_i et de poids $W > 0$ si et seulement si G contient une arborescence colorful $T' = (V_{T'}, A_{T'})$ enracinée en r et de poids $W > 0$.

($\Rightarrow$) S'il existe un indice $i \in [t]$ tel que G_i contient une arborescence colorful $T = (V_T, A_T)$ enracinée en r_i et de poids $W > 0$, alors on définit $T' = (V_{T'}, A_{T'})$ tel que $V_{T'} = V_T \cup \{r\}$ et $A_{T'} = A_T \cup \{(r, r_i)\}$. Clairement, T' est connexe, colorful et de poids W.

($\Leftarrow$) On suppose que G contient une arborescence colorful $T' = (V_{T'}, A_{T'})$ enracinée en r et de poids $W > 0$. Puisque T' est colorful et que tous les sommets de type r_i partagent la même couleur, il ne peut pas exister deux sommets u et v appartenant à deux instances initiales de MCA différentes tels que $u, v \in V_{T'}$.

Par conséquent, on note i^* l'unique indice appartenant à $[t]$ tel que $V_{i^*} \cup V_{T'} \neq \emptyset$ et on définit $T = (V_T, A_T)$ avec $V_T = V_{T'} \setminus \{r\}$ et $A_T = A_{T'} \setminus \{(r, r_{i^*})\}$. Clairement, T est connexe, colorful et de poids W.

Puisque $|\mathcal{C}| = |\mathcal{C}'| + 1$, on vient de décrire une or-composition correcte de MCA relativement à $|\mathcal{C}|$. Maintenant, on rappelle que MCA est NP-dur [18] et que $x_{\mathcal{H}} \leq |\mathcal{C}|$. Par conséquent, MCA n'admet pas de kernel polynomial relativement à $|\mathcal{C}|$, et donc relativement à $x_{\mathcal{H}}$, même si G est une arborescence, sauf si NP $\subseteq$ coNP/Poly. □

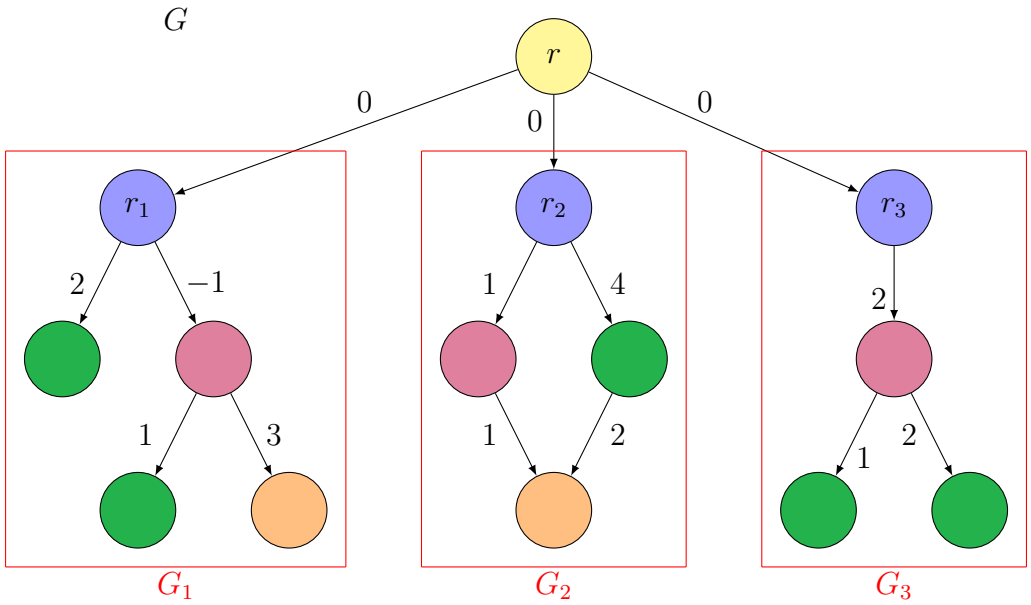


Figure 4.7 – Or-composition de trois instances de MCA (dont les graphes d'entrée G_1, G_2 et G_3 sont encadrés en rouge) vers une nouvelle instance de MCA (dont le graphe d'entrée est G).

On rappelle que s, le nombre minimum d'arcs à supprimer de $\mathcal{H}$ pour que $\mathcal{H}$ soit un arbre, satisfait la relation $s = \mathcal{O}(|\mathcal{C}|^2)$. A partir du Théorème 63 ci-dessus, on obtient donc le corollaire suivant.

Corollaire 64. *Sauf si* NP $\in$ coNP/Poly, MCA *restreint aux arbres n'admet pas de kernel polynomial relativement à* s.

Si MCA n'admet vraisemblablement pas de kernel polynomial relativement à $x_{\mathcal{H}}$, on peut supposer que ce résultat reste valide même si G est colorful, puisque le nombre de répétitions de couleurs dans G n'influe pas sur la structure de $\mathcal{H}$. On prouve ainsi le résultat suivant.

Théorème 65. *Sauf si* NP $\subseteq$ coNP/Poly, MCA *n'admet pas de kernel polynomial relativement à* $x_{\mathcal{H}}$, *même si* $\ell_{\mathcal{C}} = 0$.

Preuve. On effectue une réduction à partir de k-SET COVER, qu'on définit ci-dessous.

k-SET COVER

• **Instance**: Un ensemble $\mathcal{U} = \{u_1, u_2, \ldots, u_q\}$ d'éléments, un ensemble $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ de sous-ensembles de $\mathcal{U}$, un entier k

• **Question**: Existe-t-il un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$, de cardinalité k, et dont l'union des ensembles qu'il contient est $\mathcal{U}$?

On effectue la réduction de la manière suivante : pour toute instance de k-SET COVER, on crée un DAG à trois niveaux de sommets $G = (V = V_1 \cup V_2 \cup V_3, A)$ avec $V_1 = \{r\}$, $V_2 = \{v_i \mid i \in [p]\}$ et

$V_3 = \{z_j \mid j \in [q]\}$. On appelle V_2 le niveau 2 de G et V_3 le niveau 3 de G. Concrètement, on crée un sommet au niveau 2 pour chaque ensemble S_i de $\mathcal{F}$, et un sommet au niveau 3 pour chaque élément u_j de $\mathcal{U}$. Il existe un arc de poids -1 à partir de r vers tous les sommets au niveau 2 et, pour tout $i \in [p]$ et $j \in [q]$, il existe un arc de poids p de v_i à z_j si l'élément u_j appartient à l'ensemble S_i. Enfin, on attribue une couleur unique à chaque sommet de G. On observe donc que $\mathcal{H}$ est également un DAG à trois niveaux de sommets avec $\mathrm{col}(V_1)$, $\mathrm{col}(V_2)$ et $\mathrm{col}(V_3)$ respectivement aux niveaux 1, 2 et 3. Par conséquent, la construction décrite ci-dessus est une instance correcte de MCA. On donne un exemple de construction d'instance dans la Figure 4.8. Maintenant, on montre qu'il existe un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$ de cardinalité k dont l'union des ensembles qu'il contient est $\mathcal{U}$ si et seulement si il existe une arborescence T colorful, enracinée en r et de poids $w(T) = pq - k$ dans G.

($\Rightarrow$) On suppose qu'il existe un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$ de cardinalité k dont l'union des ensembles qu'il contient est $\mathcal{U}$ et on définit $I = \{i \in [p] \mid S_i \in \mathcal{S}\}$. On définit ensuite $V_T = \{r\} \cup \{v_i \mid i \in I\} \cup \{z_j \mid j \in [q]\}$. Clairement, $G[V_T]$ est connexe puisque r possède un arc vers tous les sommets du niveau 2 et puisqu'il existe un arc (v_i, z_j) si un élément u_j est contenu dans un ensemble $S_i \subseteq \mathcal{S}$. On note T une arborescence couvrante de $G[V_T]$. Clairement, T est colorful puisque G est colorful. De plus, T est de poids $pq - k$ puisque T contient k arcs de poids -1 du niveau 1 vers le niveau 2 et q arcs de poids p du niveau 2 au niveau 3.

($\Leftarrow$) On suppose qu'il existe une arborescence T colorful, enracinée en r et de poids $w(T) = pq - k$ dans G. On observe que toute arborescence T' dans G qui contient r et au moins un sommet appartenant à V_3 doit contenir au moins un sommet appartenant à V_2 pour être connexe. De plus, s'il existe un sommet de type z_j qui n'appartient pas à T, alors $w(T) \leq pq - p - 1$ puisque T doit contenir au moins un sommet du niveau 2 (dont le poids d'un arc entrant est égal à 1) pour être connexe. Par conséquent, si $w(T) = pq - k$, alors T contient tous les sommets appartenant au troisième niveau de G ainsi qu'exactement k sommets appartenant au deuxième niveau. Maintenant, soit $\mathcal{S} = \{S_i : i \in [p] \mid v_i \in V_T\}$. Clairement, $\mathcal{S}$ est un sous-ensemble de $\mathcal{F}$ dont l'union des ensembles qu'il contient est $\mathcal{U}$ puisque tous les sommets appartenant au troisième niveau de G appartiennent à T. De plus, $\mathcal{S}$ est de cardinalité k puisque T contient k sommets appartenant au niveau 2. La réduction présentée est donc correcte.

Maintenant, on rappelle que $\mathcal{H}$ est un DAG à trois niveaux de sommets avec $\mathrm{col}(V_1)$, $\mathrm{col}(V_2)$ et $\mathrm{col}(V_3)$ respectivement aux niveaux 1, 2 et 3. Par construction de G, s'il existe $c \in V(\mathcal{H})$ tel que $d^-(c) \geq 1$, alors $c \in \mathrm{col}(V_3)$. De plus, on rappelle que $|\mathrm{col}(V_3)| = q$, ce qui implique que $x_{\mathcal{H}} \leq q$. On vient donc de décrire une transformation polynomiale paramétrée (voir Définition 21, page 35) de k-SET COVER paramétré par q vers MCA paramétré par $x_{\mathcal{H}}$. Maintenant, on rappelle que k-SET COVER n'admet pas de kernel polynomial relativement à q sauf si NP $\subseteq$ coNP/Poly [21] et que k-SET COVER est NP-dur [69]. De plus, la version décision de MCA appartient clairement à NP. Enfin, on observe que le graphe d'entrée de MCA qui est défini dans la réduction ci-dessus est colorful, ce qui implique que $\ell_{\mathcal{C}} = 0$. Par conséquent, MCA n'admet pas de kernel polynomial relativement à $x_{\mathcal{H}}$ même si $\ell_{\mathcal{C}} = 0$, sauf si NP $\subseteq$ coNP/Poly. □

On vient de montrer que MCA n'admet vraisemblablement pas de kernel relativement à $x_{\mathcal{H}} + \ell_{\mathcal{C}}$. On considère maintenant le paramètre ℓ afin de davantage contraindre l'espace des solutions d'une instance de MCA. En effet, après avoir fixé la valeur de ℓ, on rappelle que toute solution $T = (V_T, A_T)$ de MCA contient nécessairement $V_T \geq n - \ell$ sommets, et donc que $\ell \geq \ell_{\mathcal{C}}$. Alors que MCA n'admet vraisemblablement pas de kernel polynomial relativement à $x_{\mathcal{H}}$ même si $\ell_{\mathcal{C}} = 0$ selon le Théorème 65, on peut facilement montrer que MCA se résout en temps polynomial quand $\ell = 0$, et donc que MCA admet un kernel trivial relativement à $x_{\mathcal{H}}$ quand $\ell = 0$. En effet, si $\ell = 0$ alors $\ell_{\mathcal{C}} = 0$ (puisque $\ell \geq \ell_{\mathcal{C}}$) et G est colorful. De plus, si $\ell = 0$ alors on rappelle que toute solution de MCA doit nécessairement contenir tous les sommets de G. Résoudre l'instance de MCA considérée revient donc à trouver une arborescence couvrante de G de poids maximum, ce qui s'effectue en temps polynomial [29, 48]. On peut ainsi légitimement se demander si MCA admet un kernel polynomial relativement à $x_{\mathcal{H}} + \ell$. Dans la suite, pour toute paire de sommets

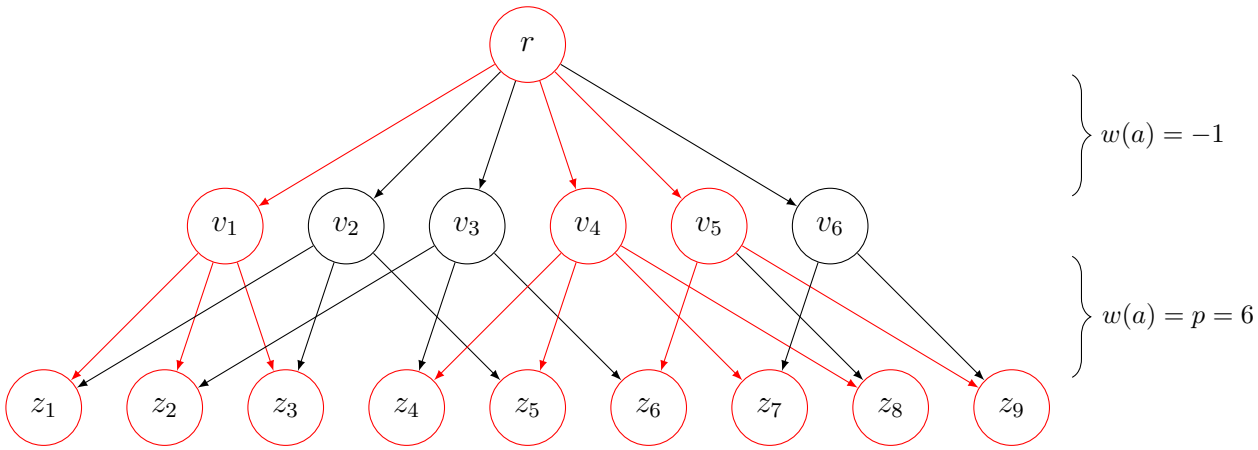


Figure 4.8 – Construction d'une instance de MCA à partir d'une instance de k-SET COVER avec $\mathcal{U} = \{u_1, u_2, \ldots, u_9\}$ et $\mathcal{F} = \{S_1 = \{u_1, u_2, u_3\}, S_2 = \{u_1, u_3, u_5\}, S_3 = \{u_2, u_4, u_6\}, S_4 = \{u_4, u_5, u_7, u_8\}, S_5 = \{u_6, u_8, u_9\}, S_6 = \{u_7, u_9\}\}$. Chaque sommet de G possède une couleur unique. On observe que $\mathcal{S} = \{S_1, S_4, S_5\}$ est un sous-ensemble de $\mathcal{F}$ de cardinalité $k = 3$ et que la solution T de MCA représentée en rouge dans l'instance créée est de poids $w(T) = 3 \cdot (-1) + 9 \cdot 6 = 51$.

$v, v' \in V(G)$ telle qu'il existe un chemin de v vers v' dans G, on note $\pi(v, v')$ le poids d'un chemin de poids maximum allant de v à v' dans G. De plus, pour tout $c \in \mathcal{C}$, on rappelle que $V_c = \{v \in V \mid \text{col}(v) = c\}$. On présente maintenant deux règles de réduction. Dans la première règle de réduction qui est présentée ci-dessous, le but est de supprimer des couleurs "superflues" dans $\mathcal{C}$.

Règle de réduction 66. *Si une instance $(G, \mathcal{C}, \text{col}, w, r)$ de* MCA *contient un triplet $\{c_1, c_2, c_3\} \in V(\mathcal{H})$ tel que (i) c_1 est l'unique voisin entrant de c_2, (ii) c_2 est l'unique voisin entrant de c_3 et (iii) c_3 est l'unique voisin sortant de c_2, alors on exécute les instructions suivantes:*

- *pour tout $v_1 \in V_{c_1}$ et tout $v_3 \in V_{c_3}$ tels qu'il existe un chemin de v_1 vers v_3 dans G, créer un arc (v_1, v_3) de poids $w(v_1, v_3) = \pi(v_1, v_3)$.*
- *créer un sommet v^* de couleur c_3. Pour tout sommet $v_1 \in V_{c_1}$ tel que v_1 possède au moins un voisin sortant de couleur c_2 dans G, ajouter un arc (v_1, v^*) dont le poids est égal au poids maximum d'un arc sortant de v_1 vers un sommet de couleur c_2 dans G.*
- *supprimer tous les sommets appartenant à V_{c_2} dans G.*

On prouve que la Règle de réduction 66 est correcte, puis on montre un exemple dans la Figure 4.9.

Lemme 67. *La Règle de réduction 66 est correcte.*

Preuve. On montre que s'il existe une arborescence colorful $T = (V_T, A_T)$ de poids au moins égal à W dans l'instance initiale $I = (G, \mathcal{C}, \text{col}, w, r)$, alors il existe une arborescence colorful $T' = (V_{T'}, A_{T'})$ de poids au moins égal à W dans la nouvelle instance $I' = (G', \mathcal{C}', \text{col}', w', r')$ obtenue après application de la Règle de réduction 66 sur I ; l'implication inverse est obtenue en utilisant un raisonnement symétrique.

Soit $T = (V_T, A_T)$ une arborescence colorful de poids W dans I. Premièrement, si T ne contient pas de sommet de couleur c_2, alors T est également une arborescence colorful de poids maximum dans la nouvelle instance. Deuxièmement, si T contient un sommet v_2 de couleur c_2 dont le voisin entrant dans T est noté v_1 et si T ne contient aucun sommet de couleur c_3, alors on définit l'arborescence $T' = (V_{T'}, A_{T'})$ avec $V_{T'} = V_T \setminus \{v_2\} \cup \{v^*\}$ et $A_{T'} = A_T \setminus \{(v_1, v_2)\} \cup \{(v_1, v^*)\}$ dans la nouvelle instance. Dans ce cas, on note que $w(T) = w'(T')$ puisque $w(v_1, v_2) = w'(v_1, v^*)$. Troisièmement, si T contient un sommet v_2 de couleur c_2 dont le voisin entrant dans T est noté v_1 et si T contient un sommet v_3 de couleur c_3 dont le voisin entrant est nécessairement v_2, alors on définit l'arborescence $T' = (V_{T'}, A_{T'})$ avec $V_{T'} = V_T \setminus \{v_2\}$ et $A_{T'} = A_T \setminus \{(v_1, v_2), (v_2, v_3)\} \cup \{(v_1, v_3)\}$ dans la nouvelle instance. Dans ce cas, on note que $w(T) =$

$w'(T')$ puisque $w(v_1, v_2) + w(v_2, v_3) = w'(v_1, v_3)$.

□

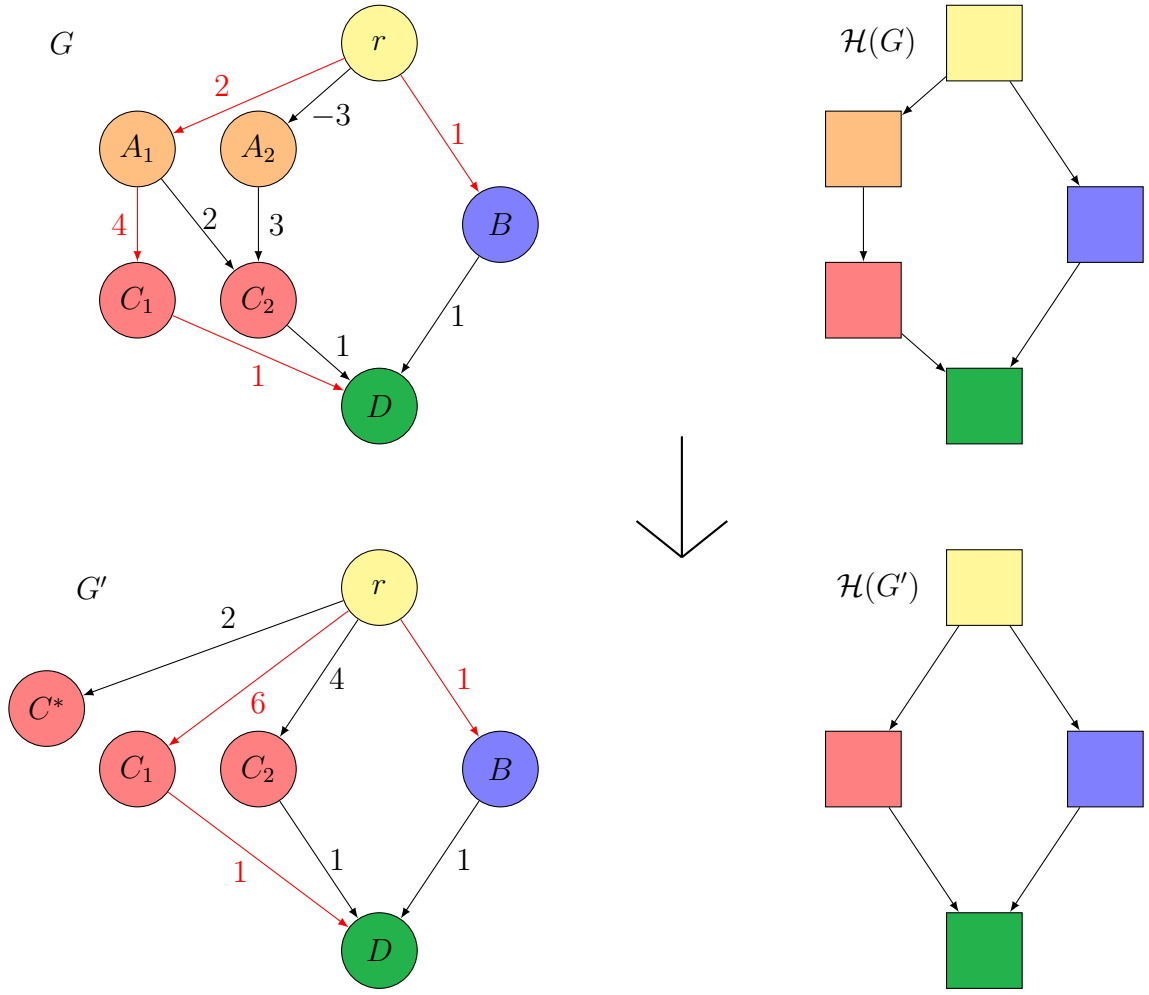


Figure 4.9 – Exemple d'application de la Règle de réduction 66 sur le triplet de couleurs {jaune, orange, rouge} d'une instance initiale de MCA dont le graphe d'entrée G est en haut à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G)$ est en haut à droite. Dans cet exemple, les ensembles V_{c_1}, V_{c_2} et V_{c_3} correspondent respectivement aux sommets de couleur jaune, orange et rouge dans G, et le sommet v^* introduit dans la règle de réduction correspond au sommet C^* sur la figure. On crée une nouvelle instance de MCA dont le graphe d'entrée G' est en bas à gauche et dont le graphe de hiérarchie de couleurs $\mathcal{H}(G')$ est en bas à droite. On initialise $G' = G$, puis on supprime tous les sommets de couleur orange de G' (ainsi que leurs arcs incidents) et on crée un sommet C^* avec $w'(r, C^*) = \max\{w(r, A_1), w(r, A_2)\} = 2$, $w'(r, C_1) = \pi(r, C_1) = w(r, A_1) + w(A_1, C_1) = 6$ et $w'(r, C_2) = \pi(r, C_2) = \max\{w(r, A_1) + w(A_1, C_2), w(r, A_2) + w(A_2, C_2)\} = 4$. Dans chacun des deux graphes G et G', les arcs d'une solution de MCA sont représentés en rouge.

On note que la Règle de réduction 66 possède une complexité en temps polynomiale puisqu'on ajoute au plus $n = |V(G)|$ arcs à une instance, qu'on en supprime au plus $m = |A(G)|$ et enfin puisque $\pi(v_1, v_3)$ peut être calculé en temps polynomial pour tout $v_1 \in V_{c_1}$ et tout $v_3 \in V_{c_3}$.

Avant d'exposer la prochaine règle de réduction, on introduit la notation suivante : $N_U^-(v)$ est l'ensemble des couleurs *uniques* du voisinage entrant d'un sommet v dans G – une couleur c est unique si elle n'apparaît qu'une fois dans G. On rappelle également que ℓ est le nombre maximum de sommets qui n'appartiennent pas à T dans G. Pour tout sommet $v \in V$, on montre maintenant une règle de réduction qui supprime des arcs de G dans le but de borner $|N_U^-(v)|$.

Règle de réduction 68. *Si une instance* $(G, \mathcal{C}, \mathrm{col}, w, r)$ *de* MCA *contient un sommet* $v \in V$ *tel que* $|N_U^-(v)| > \ell + 1$*, alors on peut supprimer de* G *les* $|N_U^-(v)| - \ell - 1$ *arcs entrants de plus petit poids de* v*.*

Lemme 69. *La Règle de réduction 68 est correcte.*

Preuve. Comme $|N_U^-(v)| > \ell + 1$, on observe qu'une solution $T = (V_T, A_T)$ de MCA contient nécessairement au moins deux sommets de $N^-(v)$. Maintenant, soit v_1 un sommet appartenant à $N_U^-(v)$ tel que (v_1, v) est l'arc entrant de poids minimum d'un sommet de couleur unique vers v dans G. Même si v_1 appartient à T, il existera toujours au moins un autre sommet $v_2 \in V_T$ tel que $w(v_1, v) \leq w(v_2, v)$. Par conséquent, on en conclut que T ne contiendra jamais l'arc (v_1, v) et on peut sans perte de généralité le supprimer de G. On prouve que la règle de réduction est correcte en répétant $|N_U^-(v)| - \ell - 1$ fois ce raisonnement. □

On note que la Règle de réduction 68 possède une complexité en temps polynomiale puisqu'on supprime au plus $m - \ell$ arcs de G.

On peut maintenant prouver le théorème suivant.

Théorème 70. MCA *admet un kernel contenant* $\mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$ *sommets.*

Preuve. On décrit un processus de kernelisation pour obtenir une nouvelle instance de MCA à partir d'une instance initiale en appliquant des règles de réduction présentées dans ce manuscrit, puis on montre que le graphe d'entrée de cette nouvelle instance contient $\mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$ sommets.

Tout d'abord, on réduit itérativement l'instance initiale selon les Règles de réduction 66, 68 et 44 (qui est présentée dans la Section 3.2.5, page 74). On note que l'application de ces règles de réduction s'effectue en temps polynomial, puisqu'on peut exécuter les règles de réduction 66, 68 et 44 au plus $|\mathcal{C}|^3$, n et $|\mathcal{C}|$ fois (respectivement) en temps polynomial. On note ensuite $\mathcal{C}^*$ l'ensemble des feuilles de $\mathcal{H}$ qui appartiennent au voisinage sortant de $\mathrm{col}(r)$ et V^* l'ensemble des sommets de G dont la couleur appartient à $\mathcal{C}^*$. On remarque qu'on peut trouver une solution de MCA T^* dans $G[\{r\} \cup V^*]$ en temps polynomial puisque $G[\{r\} \cup V^*]$ est une star. Par conséquent, on supprime V^* de G (ainsi que tous les arcs incidents à V^*), puis on ajoute un sommet v^* de couleur unique c^* dans V et un arc (r, v^*) de poids $w(T^*)$. Par souci de clarté, on renomme dans la suite $(G, \mathcal{C}, \mathrm{col}, w, r)$ la nouvelle instance équivalente obtenue – en temps polynomial – et on définit $T = (V_T, A_T)$ comme une solution de MCA dans cette instance. Dans un premier temps, on montre que le degré entrant de chaque couleur de $\mathcal{H}$ est au plus égal à $(\ell + 1)^2 + \ell$. Cela nous servira à montrer, dans un second temps, que le nombre n de sommets de G est polynomialement borné par une fonction de $x_{\mathcal{H}}$ et de ℓ.

On montre que le degré entrant des sommets de $\mathcal{H}$ est borné. Puisque T est colorful et puisque $|V_T| \geq n - \ell$, il existe au plus ℓ couleurs non-uniques dans $\mathcal{C}$. Par conséquent, le voisinage entrant de chaque couleur $c \in V(\mathcal{H})$ contient au plus ℓ couleurs non-uniques de G dans $\mathcal{H}$. De plus, après application de la Règle de réduction 68, le voisinage entrant de chaque sommet $v \in V(G)$ contient au plus $\ell + 1$ sommets de couleur unique dans G. En outre, pour toute couleur $c \in V(\mathcal{H})$, on suppose qu'il existe au plus $|V_c| \leq \ell + 1$ occurrences de c dans $\mathcal{H}$ puisque T ne peut pas être colorful s'il existe plus de $\ell + 1$ occurrences de c dans G. Par conséquent, pour toute couleur $c \in V(\mathcal{H})$, le voisinage entrant de c contient au plus $|V_c| \cdot (\ell+1) = (\ell+1)^2$ couleurs uniques de G dans $\mathcal{H}$, et c possède au plus $(\ell+1)^2+\ell$ voisins entrants dans $\mathcal{H}$.

Maintenant, on rappelle que $\mathcal{X}$ est l'ensemble des couleurs $c \in \mathcal{C}$ de degré entrant au moins égal à 2 dans $\mathcal{H}$. On considère la forêt $\mathcal{F} \subseteq \mathcal{H}$ dont l'ensemble de sommets est $\mathcal{C}_{\mathcal{F}} = \mathcal{C} \setminus (\mathcal{X} \cup \{c^*\})$ et qui contient chaque arc (c, c') de $\mathcal{H}$ tel que $c, c' \in \mathcal{C}_{\mathcal{F}}$. Dans la suite, on va successivement borner le nombre maximum de feuilles de $\mathcal{F}$, le nombre maximum de sommets de $\mathcal{F}$, de $V(\mathcal{H})$ et finalement de $V(G)$ en fonction de $x_{\mathcal{H}}$ et de ℓ. Tout d'abord, on rappelle que toutes les feuilles de $\mathcal{H}$ qui n'appartiennent pas au voisinage sortant de $\mathrm{col}(r)$ sont des couleurs difficiles, sinon on pourrait encore appliquer la Règle de réduction 44. Par conséquent, toutes les feuilles de $\mathcal{F}$ appartiennent au voisinage entrant d'une couleur difficile dans

$\mathcal{H}$. Puisque le degré entrant maximum de toute couleur de $\mathcal{H}$ est au plus égal à $(\ell+1)^2+\ell$, le nombre maximum de feuilles de $\mathcal{F}$ est au plus égal à $x_{\mathcal{H}} \cdot ((\ell+1)^2+\ell)$. Maintenant, selon la Règle de réduction 66, aucun sommet de $\mathcal{H}$ ne possède un unique voisin entrant ainsi qu'un unique voisin sortant dans $\mathcal{H}$. Par conséquent, tous les sommets de $\mathcal{F}$ qui possèdent un unique voisin entrant ainsi qu'un unique voisin sortant dans $\mathcal{F}$ possèdent également au moins une couleur difficile dans leur voisinage sortant dans $\mathcal{H}$. La forêt $\mathcal{F}$ contient donc au plus $\mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$ sommets qui ne sont pas des feuilles dans $\mathcal{F}$ et qui n'appartiennent pas au voisinage entrant d'au moins une couleur difficile dans $\mathcal{H}$. Par conséquent, on observe que $|\mathcal{C}_{\mathcal{F}}| = \mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$ et que $|\mathcal{C}| = \mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$, puisque $\mathcal{C}_{\mathcal{F}} = \mathcal{C} \setminus (\mathcal{X} \cup \{c^*\})$ et que $|\mathcal{X}| = x_{\mathcal{H}}$. Pour terminer, on rappelle que G contient au plus ℓ sommets de plus que $\mathcal{H}$ puisque $T \subseteq G$ est colorful et contient au moins $n-\ell$ sommets. Ainsi, on obtient bien $n = \mathcal{O}(x_{\mathcal{H}} \cdot \ell^2)$. □

4.2.3 Etude de MCA en fonction de la treewidth de $\mathcal{H}$

On rappelle que la treewidth de (la *version non-orientée* de) $\mathcal{H}$ est notée $t_{\mathcal{H}}$. De plus, selon le Théorème 46, MCA se résout en temps polynomial quand $\mathcal{H}$ est un arbre – et donc quand $t_{\mathcal{H}} = 1$. Par conséquent, il est légitime de se demander si MCA admet des algorithmes FPT relativement à $t_{\mathcal{H}}$. On donne le résultat suivant.

Théorème 71. MCA *est* W[2]*-dur relativement à* $t_{\mathcal{H}}$.

Preuve. On effectue une réduction à partir du problème k-MULTICOLORED SET COVER, qui est très semblable à celle décrite dans la preuve du Théorème 65. On définit k-MULTICOLORED SET COVER ci-dessous.

k-MULTICOLORED SET COVER

• **Instance**: Un ensemble $\mathcal{U} = \{u_1, u_2, \ldots, u_q\}$ d'éléments, un ensemble $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ de sous-ensembles de $\mathcal{U}$, un ensemble de couleurs Δ avec une fonction de coloration $\mathrm{col}' : \mathcal{F} \to \Delta$, un entier k

• **Question**: Existe-t-il un sous-ensemble $\mathcal{S} \subseteq \mathcal{F}$ dont l'union des ensembles qu'il contient est $\mathcal{U}$ et tel que (i) $|\mathcal{S}| = k$ et (ii) $\mathcal{S}$ est colorful, *i.e.* $\mathrm{col}'(S_i) \neq \mathrm{col}'(S_j)$ pour tous $S_i, S_j \in \mathcal{S}$?

On effectue la réduction de la manière suivante : pour toute instance de k-MULTICOLORED SET COVER, on crée un DAG à trois niveaux de sommets $G = (V = V_1 \cup V_2 \cup V_3, A)$ avec $V_1 = \{r\}$, $V_2 = \{v_i \mid i \in [p]\}$ et $V_3 = \{z_j \mid j \in [q]\}$ (voir Figure 4.10). On appelle V_2 le niveau 2 de G et V_3 le niveau 3 de G. Concrètement, on crée un sommet au niveau 2 pour chaque ensemble S_i de $\mathcal{F}$, et un sommet au niveau 3 pour chaque élément de $\mathcal{U}$. Il existe un arc de poids -1 à partir de r vers tous les sommets au niveau 2 et, pour tout $i \in [p]$ et $j \in [q]$, il existe un arc de poids p de v_i à z_j si l'élément u_j appartient à l'ensemble S_i. On définit col de la manière suivante : on attribue tout d'abord une couleur unique à chaque sommet de $V_1 \cup V_3$; on attribue ensuite une même couleur à deux sommets v_{i_1}, v_{i_2} de V_2 si et seulement si la couleur des deux ensembles correspondants S_{i_1} et S_{i_2} est la même. Par conséquent, on observe que $\mathcal{H}$ est également un DAG à trois niveaux de sommets avec $\mathrm{col}(V_1)$, $\mathrm{col}(V_2)$ et $\mathrm{col}(V_3)$ respectivement aux niveaux 1, 2 et 3. La construction décrite ci-dessus est donc une instance correcte de MCA. On montre maintenant qu'il existe un sous-ensemble colorful $\mathcal{S} \subseteq \mathcal{F}$ de cardinalité k dont l'union des ensembles qu'il contient est $\mathcal{U}$ si et seulement si il existe une arborescence $T \subseteq G$ colorful, enracinée en r et de poids $w(T) = pq-k$.

($\Rightarrow$) On suppose qu'il existe un sous-ensemble colorful $\mathcal{S} \subseteq \mathcal{F}$ de cardinalité k dont l'union des ensembles qu'il contient est $\mathcal{U}$ et on définit $I = \{i \in [p] \mid S_i \in \mathcal{S}\}$. On définit ensuite $V_T = \{r\} \cup \{v_i \mid i \in I\} \cup \{z_j \mid j \in [q]\}$. Clairement, $G[V_T]$ est connexe puisque r possède un arc vers tous les sommets au niveau 2 et puisqu'il existe un arc $(v_i, z_j) \in A$ si un élément u_j est contenu dans un ensemble $S_i \subseteq \mathcal{S}$. On note T une arborescence couvrante de $G[V_T]$. Clairement, T est colorful puisque $\mathcal{S}$ est colorful. De plus, T est de poids $pq-k$ puisque T contient k arcs de poids -1 du niveau 1 vers le niveau 2 et q arcs de poids p

du niveau 2 au niveau 3.

($\Leftarrow$) On suppose qu'il existe une arborescence $T \subseteq G$ colorful, enracinée en r et de poids $w(T) = pq-k$ On observe que toute arborescence T' dans G qui contient r et au moins un sommet appartenant à V_3 doit contenir au moins un sommet appartenant à V_2 pour être connexe. De plus, s'il existe un sommet de type z_j qui n'appartient pas à T, alors $w(T) \leq pq - p - 1$ puisque tous les arcs du niveau 1 vers le niveau 2 ont un poids de -1. Par conséquent, si $w(T) = pq - k$, alors T contient tous les sommets appartenant au troisième niveau de G ainsi qu'exactement k sommets appartenant au deuxième niveau. Maintenant, soit $\mathcal{S} = \{S_i : i \in [p] \mid v_i \in V_T\}$. Clairement, $\mathcal{S}$ est un sous-ensemble colorful de $\mathcal{F}$ – puisque T est colorful – dont l'union des ensembles qu'il contient est $\mathcal{U}$ – puisque tous les sommets appartenant au troisième niveau de G appartiennent à T. De plus, $\mathcal{S}$ est de cardinalité k puisque T contient k sommets appartenant au niveau 2. La réduction présentée est donc correcte.

Maintenant, on rappelle que $\mathcal{H}$ est un DAG à trois niveaux de sommets avec $\mathrm{col}(V_1)$, $\mathrm{col}(V_2)$ et $\mathrm{col}(V_3)$ respectivement aux niveaux 1, 2 et 3. Par conséquent, il existe une décomposition arborescente triviale $\langle\{X_i : i \in \{0\} \cup [|\,\mathrm{col}(V_3)+1|]\}, \mathcal{T}\rangle$ de (la *version non-orientée* de) $\mathcal{H}$, présentée dans la Figure 4.11, qu'on décrit de la manière suivante : le sac $X_0 = \{\mathrm{col}(r)\}$ possède un arc vers le sac $X_1 = \{\{\mathrm{col}(r)\} \cup \mathrm{col}(V_2)\}$; tous les autres sacs de type X_i contiennent $\mathrm{col}(V_2)$ ainsi qu'une couleur distincte de $\mathrm{col}(V_3)$ et possèdent un arc entrant à partir de X_1. Ainsi, la width de $\langle\{X_i : i \in \{0\} \cup [|\,\mathrm{col}(V_3)+1|]\}, \mathcal{T}\rangle$ est égale à k, ce qui implique que $t_{\mathcal{H}} \leq k$. Par conséquent, MCA est W[2]-dur relativement à $t_{\mathcal{H}}$ puisque k-MULTICOLORED SET COVER est connu pour être W[2]-dur relativement à k. □

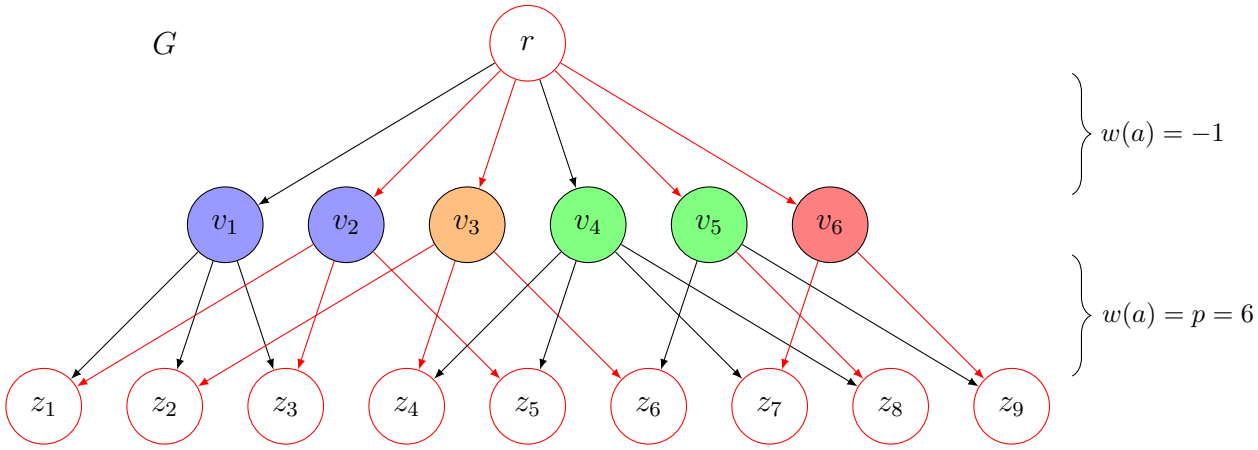


Figure 4.10 – Construction d'une instance de MCA à partir d'une instance de k-MULTICOLORED SET COVER avec $\mathcal{U} = \{u_1, u_2, \ldots, u_9\}$ et $\mathcal{F} = \{S_1 = \{u_1, u_2, u_3\}, S_2 = \{u_1, u_3, u_5\}, S_3 = \{u_2, u_4, u_6\}, S_4 = \{u_4, u_5, u_7, u_8\}, S_5 = \{u_6, u_8, u_9\}, S_6 = \{u_7, u_9\}\}$. Les ensembles S_1 et S_2 sont de même couleur ; les ensembles S_4 et S_5 sont de même couleur (qui est distincte de celle des ensembles S_1 et S_2) ; les ensembles S_3 et S_6 possèdent chacun une couleur unique. Dans l'instance de MCA, tous les sommets de G qui ne sont pas colorés possèdent une couleur unique. On observe que $\mathcal{S} = \{S_2, S_3, S_5, S_6\}$ est un sous-ensemble de $\mathcal{F}$ de cardinalité $k = 4$ et que la solution T de MCA représentée en rouge dans l'instance créée est de poids $w(T) = 4 \cdot (-1) + 9 \cdot 6 = 50$.

Soit $U(\mathcal{H})$ la version non-orientée de $\mathcal{H}$. On utilise maintenant la preuve ci-dessus dans le but de montrer que MCA n'admet vraisemblablement pas d'algorithme FPT relativement à divers paramètres liés à $U(\mathcal{H})$. Premièrement, le *vertex cover number* de $U(\mathcal{H})$ est la cardinalité du plus petit sous-ensemble couvrant de $U(\mathcal{H})$, *i.e.* du plus petit sous-ensemble $S \subseteq V(\mathcal{H})$ tel qu'au moins un des deux sommets incidents à chaque arête de $U(\mathcal{H})$ appartient à S. Dans la réduction qui précède, on remarque que $\mathrm{col}(V_2)$ est un ensemble couvrant de $U(\mathcal{H})$ et donc que $|S| \leq k$, avec k le paramètre de k-MULTICOLORED SET COVER. On obtient le corollaire suivant.

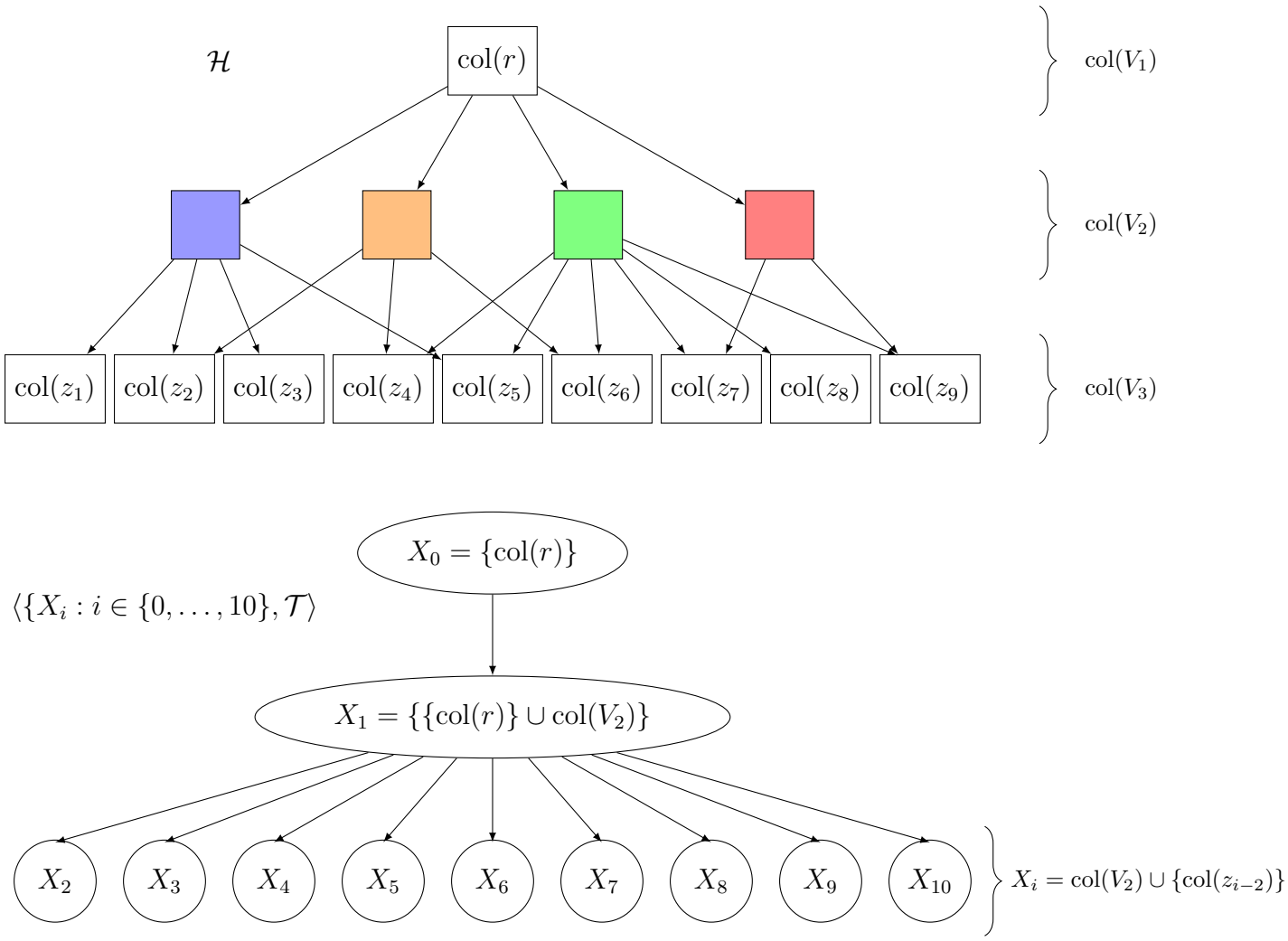


Figure 4.11 – Illustration de la preuve du Théorème 71. La décomposition arborescente $\langle\{X_i : i \in \{0,\ldots,10\}, \mathcal{T}\rangle$ (en bas) a été créée à partir du graphe de hiérarchie de couleurs $\mathcal{H}$ (en haut) de l'instance de MCA présentée dans la Figure 4.10.

Corollaire 72. MCA *est* W[2]*-dur relativement au vertex cover number de* $U(\mathcal{H})$.

Deuxièmement, le *feedback vertex set number* est la cardinalité du plus petit sous-ensemble $S \subseteq V(\mathcal{H})$ tel que $U(\mathcal{H}[V(\mathcal{H}) \setminus S])$ est acyclique. La cardinalité d'un tel ensemble est un paramètre intéressant puisque $x_{\mathcal{H}} = 0$ dans $\mathcal{H}[V(\mathcal{H}) \setminus S]$. Dans la réduction qui précède, on observe que $\mathrm{col}(V_2)$ est un feedback vertex set des sommets de $U(\mathcal{H})$ puisque tout ensemble couvrant de $U(\mathcal{H})$ est également un feedback vertex set de $U(\mathcal{H})$. On en déduit que $|S| \leq k$ et on obtient le corollaire suivant.

Corollaire 73. MCA *est* W[2]*-dur relativement au feedback vertex set number de* $U(\mathcal{H})$.

Troisièmement, on rappelle que dans la preuve du Théorème 71 chaque couleur appartenant au troisième niveau de $\mathcal{H}$ est une feuille. En fait, on remarque que le nombre de couleurs de degré sortant au moins égal à 2 dans $\mathcal{H}$ est égal à $|\mathrm{col}(V_1)| + |\mathrm{col}(V_2)| = k+1$. Bien que le Théorème 61 ait montré que MCA appartient à FPT relativement à $x_{\mathcal{H}}$, le nombre de couleurs de degré entrant au moins égal à 2 dans $\mathcal{H}$, on obtient le résultat suivant.

Corollaire 74. MCA *est* W[2]*-dur relativement au nombre de couleurs de degré sortant au moins égal à 2 dans* $\mathcal{H}$.

Selon le Théorème 71, MCA est W[2]-dur paramétré par $t_{\mathcal{H}}$. Par conséquent, il est intéressant de trouver un paramètre dont la combinaison avec $t_{\mathcal{H}}$ permette de produire un algorithme FPT pour MCA. On considère ici le paramètre $\ell_{\mathcal{C}} = n - |\mathcal{C}|$. On rappelle que MCA paramétré par $\ell_{\mathcal{C}}$ est W[1]-dur d'après le Corollaire 48, mais on a vu que le problème peut être résolu en temps $\mathcal{O}^*(2^{\ell_{\mathcal{C}}})$ quand G est une arborescence (voir Proposition 56). De plus, on rappelle que MCA est dans P quand $\mathcal{H}$ est une arborescence (voir Théorème 46), et donc quand $t_{\mathcal{H}} = 1$. Enfin, on rappelle que toute instance de MCA contenant $|\mathcal{C}|$ couleurs possède au plus $2^{\ell_{\mathcal{C}}}$ sous-graphes *fully-colorful* de G (voir Lemme 53, page 83). On montre maintenant que MCA est dans FPT relativement à $t_{\mathcal{H}} + \ell_{\mathcal{C}}$.

Théorème 75. MCA *peut être résolu en temps* $\mathcal{O}^*(2^{\ell_{\mathcal{C}}} \cdot 4^{t_{\mathcal{H}}})$ *et en espace* $\mathcal{O}^*(3^{t_{\mathcal{H}}})$.

Preuve. Dans la suite, soit $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ une *bonne* décomposition arborescente (voir Définition 2, page 18) de (la *version non-orientée* de) $\mathcal{H}$. On propose dans cette preuve un algorithme de programmation dynamique qui utilise cette décomposition arborescente afin de calculer une solution de MCA dans tout sous-graphe fully-colorful G' inclus dans G. Pour cela, on observe tout d'abord que $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ est également une décomposition arborescente de la version non-orientée du graphe de hiérarchie de couleurs de tout sous-graphe de G. De plus, comme tout graphe colorful est isomorphe à son graphe de hiérarchie de couleurs, on remarque que $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ est en fait une décomposition arborescente de tout sous-graphe fully-colorful G' inclus dans G. Par conséquent, on suppose sans perte de généralité que chaque sac X_i contient les sommets d'un tel graphe G' au lieu des couleurs de $\mathcal{H}(G')$ et que $X_0 = \{r\}$ est la racine de $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$.

Dans la suite, pour toute couleur $c \in \mathcal{C}$ et tout nœud de suppression X_i avec X_j l'enfant de X_i dans $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ tels que $X_j = X_i \cup \{c\}$, on dit que c a été *oublié* au nœud X_i. De plus, si c a été oublié au nœud X_k dans $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ et qu'il existe un chemin de X_i vers X_k dans $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$, alors on dit que c a été oublié *en-dessous* de X_i. On crée maintenant deux tables de programmation dynamique $T_i[L_1, L_2, L_3]$ et $D_i[L_1, L_2, L_3]$ pour tout $i \in I$ et tous sous-ensembles L_1, L_2, L_3 appartenant à X_i tels que $L_1 \oplus L_2 \oplus L_3 = X_i$. Chaque entrée de type $T_i[L_1, L_2, L_3]$ contient le poids d'une solution *partielle* de MCA dans G' qui est constituée d'une collection de $|L_1|$ arborescences telles que :

- tout sommet $v \in L_1$ est la racine d'exactement une de ces arborescences,
- tout sommet $v \in L_2$ est contenu dans exactement une de ces arborescences (mais n'en est pas la racine),
- aucun sommet $v \in L_3$ n'appartient à une de ces arborescences,
- tout sommet $v \in V$ dont la couleur été oubliée en-dessous de X_i dans $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ peut appartenir à l'une de ces arborescences,
- il n'existe aucune autre collection d'arborescences dont la somme des poids est strictement supérieure à celle-ci sous les mêmes contraintes.

Chaque entrée de type $D_i[L_1, L_2, L_3]$ contient le poids de la même solution partielle que celle de l'entrée correspondante $T_i[L_1, L_2, L_3]$, à l'exception des sommets $v \in V$ dont les couleurs ont été oubliées en-dessous de X_i et qui ne peuvent ici appartenir à aucune arborescence de la solution partielle. On décrit maintenant comment remplir chaque entrée de type $T_i[L_1, L_2, L_3]$. On précise que chaque entrée de type $D_i[L_1, L_2, L_3]$ est calculée de la même manière qu'une entrée de type $T_i[L_1, L_2, L_3]$, sauf si X_i est un nœud de suppression – on montrera comment remplir $D_i[L_1, L_2, L_3]$ dans ce cas.

• Si X_i est un nœud feuille: $T_i[L_1, L_2, L_3] = 0$

On observe que les nœuds feuilles sont les cas de base de notre algorithme de programmation dynamique puisque $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ est une bonne décomposition arborescente de G. De plus, on rappelle que les nœuds feuilles de $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ contiennent un unique sommet de G' ; la seule solution partielle pour de tels nœuds est donc de poids nul.

• Si X_i est un nœud introductif possédant un enfant X_j et si v^* est le sommet *introduit*, *i.e.* si $X_i = X_j \cup \{v^*\}$:

$$T_i[L_1, L_2, L_3] = \begin{cases} (a)\ \max\limits_{\forall S \subseteq L_2} \{\sum\limits_{v \in S} w(v^*, v) + T_j[L_1 \cup S \setminus \{v^*\}, L_2 \setminus S, L_3]\} & \text{si } v^* \in L_1 \\ (b)\ \max\limits_{\forall u \in (L_1 \cup L_2)} \{w(u, v^*) + \\ \qquad \max\limits_{\forall S \subseteq (L_2 \setminus \{u\})} \{\sum\limits_{v \in S} w(v^*, v) + T_j[L_1 \cup S \setminus \{v^*\}, L_2 \setminus S, L_3]\}\} & \text{si } v^* \in L_2 \\ (c)\ T_j[L_1, L_2, L_3 \setminus \{v^*\}]\} & \text{si } v^* \in L_3 \end{cases}$$

où on définit $w(u, v) = -\infty$ s'il n'y a pas d'arc de u vers v dans G'. On distingue trois cas : soit v^* est la racine d'une arborescence dans la solution partielle (cas (a)), soit v^* appartient à une arborescence dans la solution partielle mais n'est pas la racine de cette arborescence (cas (b)), ou soit v^* n'appartient pas à la solution partielle (cas (c)). Dans le cas (a), S correspond à l'ensemble des voisins sortants de v^* dans la solution partielle calculée ; les sommets de S n'ont donc aucun autre voisin entrant que v^* dans cette solution partielle. Par conséquent, ces sommets sont les racines des arborescences de la solution partielle dans les entrées correspondantes de type T_j (pour rappel, X_j est l'unique enfant de X_i). Maintenant, on remarque que le cas (b) est très similaire au cas (a). En plus d'un ensemble S de voisins sortants, le fait que $v_2 \in L_2$ implique que v^* possède un voisin entrant $u \in (L_1 \cup L_2)$ dans la solution partielle. Comme ce voisin entrant u ne peut pas également être un voisin sortant de v^*, u n'appartient pas à S. Calculer de manière exhaustive toutes les possibilités à la fois pour S et u nous permet d'assurer que $T_i[L_1, L_2, L_3]$ est correctement remplie. Enfin, par définition de L_3, on observe que v^* n'appartient pas à la solution partielle de $T_i[L_1, L_2, L_3]$ si $v^* \in L_3$. On montre dans l'Exemple 76 comment remplir une entrée de type $T_i[L_1, L_2, L_3]$ si X_i est un nœud introductif.

• Si X_i est un nœud de suppression possédant un enfant X_j et si v^* est le sommet oublié :

$$T_i[L_1, L_2, L_3] = \max\{T_j[L_1, L_2 \cup \{v^*\}, L_3], T_j[L_1, L_2, L_3 \cup \{v^*\}]\}$$

Concrètement, la formule ci-dessus détermine si le poids de la collection d'arborescences contenue dans $T_i[L_1, L_2, L_3]$ est supérieur avec ou sans v^* appartenant à L_2. On observe qu'on ne considère pas le cas où v_2 est la racine d'une arborescence puisqu'une telle arborescence ne pourrait par la suite pas être connexe avec le reste de la solution partielle via un nœud introductif. Par ailleurs, on remarque que $D_i[L_1, L_2, L_3] = D_j[L_1, L_2, L_3 \cup \{v^*\}]$ puisque la solution partielle dans $D_i[L_1, L_2, L_3]$ ne contient par définition aucun sommet oublié.

• Si X_i est un nœud joint possédant deux enfants X_j et X_k :

$$T_i[L_1, L_2, L_3] = T_j[L_1, L_2, L_3] \ + \ T_k[L_1, L_2, L_3] - D_i[L_1, L_2, L_3]$$

Concrètement, une solution partielle de $T_i[L_1, L_2, L_3]$ peut contenir les sommets oubliés d'une solution partielle de $T_j[L_1, L_2, L_3]$ ainsi que ceux d'une solution partielle de $T_k[L_1, L_2, L_3]$. Comme une solution partielle de $D_i[L_1, L_2, L_3]$ ne contient aucun sommet oublié, le poids de chaque arc d'une solution partielle de $T_i[L_1, L_2, L_3]$ est compté exactement une fois.

On remplit les tables à partir des feuilles de $\langle\{X_i : i \in I\}, \mathcal{T}\rangle$ jusqu'à la racine pour tout $i \in I$ et de manière à ce que toute entrée de type $T_i[L_1, L_2, L_3]$ soit calculée juste après l'entrée correspondante de type $D_i[L_1, L_2, L_3]$. Maintenant, toute solution $T' = (V_{T'}, A_{T'})$ de MCA dans un sous-graphe fully-colorful G' inclus dans G est de poids $w(T') = T_0[\{r\}, \emptyset, \emptyset]$ – et on peut alors retrouver la solution correspondante en effectuant du backtracking. Par conséquent, on observe qu'on peut calculer T' en remplissant les deux tables de type T_i et D_i. Chaque table possède exactement $3^{t_{\mathcal{H}}}$ entrées, ce qui donne une borne supérieure

pour la complexité spatiale de l'algorithme. Les entrées les plus coûteuses à calculer en terme de temps d'exécution sont celles des cas (a) et (b) pour les nœuds introductifs, dans lesquels on considère en tout $\mathcal{O}^*(4^{t_\mathcal{H}})$ cas distincts puisque X_i peut être partitionné en quatre sous-ensembles L_1, $L_2 \setminus S$, $L_2 \cap S$ et L_3. Enfin, une solution de MCA dans G est également une solution de MCA dans au moins un sous-graphe fully-colorful G'. Par conséquent, calculer une solution de MCA dans chacun de ces sous-graphes G' nous permet de certifier que l'algorithme est correct et, selon le Lemme 53, ajoute un facteur $\mathcal{O}(2^{\ell_\mathcal{C}})$ à la complexité totale en temps de l'algorithme décrit ci-dessus – ce qui prouve notre théorème. □

Exemple 76. Illustration de l'algorithme présenté dans le Théorème 61.

Dans cet exemple, on montre comment calculer une case de type $T_i[L_1, L_2, L_3]$ quand X_i est un nœud introductif. L'instance de MCA considérée ainsi qu'une partie de la décomposition arborescente dont on dispose sont présentées dans la Figure 4.12. Dans la suite, on suppose que :

- $T_j[\{A, B\}, \{D, E\}, \emptyset] = w(A, D) + w(B, E) + w(D, F) = 1 + -3 + 3 = 1$,
- $T_j[\{A, B, D\}, \{E\}, \emptyset] = w(B, E) + w(D, F) = -3 + 3 = 0$,
- $T_j[\{A, B, E\}, \{D\}, \emptyset] = w(A, D) + w(D, F) = 1 + 3 = 4$,
- $T_j[\{A, B, D, E\}, \emptyset, \emptyset] = w(D, F) = 3$.

On montre maintenant comment calculer formellement l'entrée $T_i[\{A, B\}, \{D, E\}, \emptyset]$.

$$
\begin{aligned}
T_i[\{A,B\},\{D,E\},\emptyset] &= \max\{\ w(A,C) + \max\{T_j[\{A,B\},\{D,E\},\emptyset], \\
&\qquad\qquad w(C,D) + T_j[\{A,B,D\},\{E\},\emptyset], \\
&\qquad\qquad w(C,E) + T_j[\{A,B,E\},\{D\},\emptyset], \\
&\qquad\qquad w(C,D) + w(C,E) + T_j[\{A,B,D,E\},\emptyset,\emptyset]\ \}, \\
&\qquad w(B,C) + \max\{\ \dots\ \}, \\
&\qquad w(D,C) + \max\{\ \dots\ \}, \\
&\qquad w(E,C) + \max\{\ \dots\ \}\ \} \\
&= \max\{\ -1 + \max\{1, 4+0, 2+1, 4+2+3\},\ \dots\ \} \\
&= 5
\end{aligned}
$$

On réutilise maintenant la preuve du Théorème 65 (page 94) afin de montrer que MCA n'admet pas de kernel polynomial relativement à $t_\mathcal{H} + \ell_\mathcal{C}$ sauf si NP $\in$ coNP/Poly. Dans cette preuve, on rappelle que k-Set Cover n'admet pas de kernel polynomial relativement à q sauf si NP $\in$ coNP/Poly [21] et que k-Set Cover est NP-dur [69]. De plus, on rappelle que la version décision de MCA appartient clairement à NP et que le graphe d'entrée de MCA qui est construit dans la preuve du Théorème 65 est colorful, ce qui implique que $\ell_\mathcal{C} = 0$. On montre maintenant que $t_\mathcal{H} \leq x_\mathcal{H} + 1$ dans $U(\mathcal{H})$, la version non-orientée de $\mathcal{H}$. Pour cela, on remarque dans un premier temps que l'ensemble couvrant de $U(\mathcal{H})$ de cardinalité minimum est soit $(\text{col}(V_1) \cup \text{col}(V_3))$ soit $\text{col}(V_2)$. Puisque $|\text{col}(V_1) \cup \text{col}(V_3)| = x_\mathcal{H} + 1$, la cardinalité minimum d'un ensemble couvrant de $U(\mathcal{H})$ est au plus égale à $x_\mathcal{H} + 1$. Dans un deuxième temps, on rappelle que la treewidth d'un graphe est une borne inférieure sur la cardinalité du plus petit ensemble couvrant de ce graphe. Par conséquent, on en déduit que $t_\mathcal{H} \leq x_\mathcal{H} + 1$ dans l'instance de MCA qui est construite dans la preuve du Théorème 65, et on obtient donc le corollaire suivant.

Corollaire 77. MCA *n'admet pas de kernel polynomial relativement à* $t_\mathcal{H}$ *même si* $\ell_\mathcal{C} = 0$*, sauf si* NP $\in$ coNP/Poly.

Dans cette section, on a montré des résultats de complexité paramétrée pour le problème MCA relativement à la structure du graphe de hiérarchie de couleurs et au nombre de répétitions de couleurs dans le graphe d'entrée. Ainsi, on a présenté plusieurs algorithmes FPT pour résoudre ce problème, ainsi qu'un ensemble de règles de réductions qui a ensuite permis de prouver que MCA admet un kernel polynomial relativement à une combinaison de deux paramètres étudiés. Dans la section suivante, on va maintenant déterminer dans quelle mesure ces résultats théoriques peuvent être appliqués sur des instances de MCA issues de données biologiques.

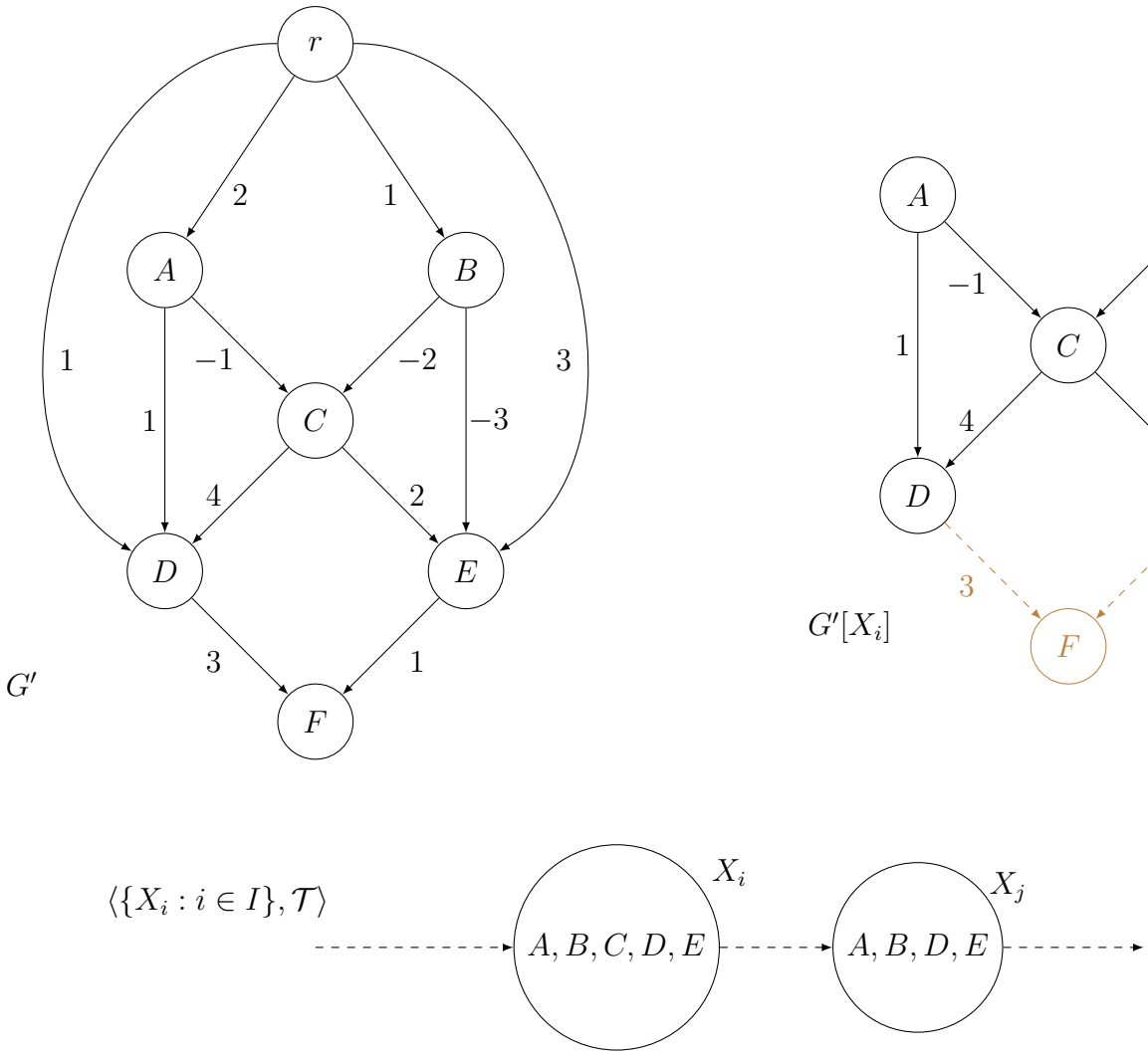


Figure 4.12 – Illustration de l'Exemple 76 présenté pour le Théorème 75. On suppose ici que le graphe d'entrée G d'une instance de MCA est un graphe colorful et donc que le graphe G' (en haut à gauche) est un sous-graphe fully-colorful de G. Le nœud $X_i = \{A, B, C, D, E\}$ est un nœud introductif de la décomposition arborescente $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$ (présentée dans la partie basse) et on représente le graphe $G'[X_i]$ en haut à droite. Dans cet exemple, on suppose que le sommet F a été oublié en-dessous de X_i dans $\langle \{X_i : i \in I\}, \mathcal{T} \rangle$; on représente F en marron dans $G[X_i]$ (ainsi que les arcs incidents à F qui sont représentés en marron et en pointillé) afin de mieux illustrer les calculs effectués dans l'Exemple 76.

4.3 Expérimentations

L'objectif de cette section est de comparer sur des instances réelles les performances de l'unique algorithme FPT disponible dans la littérature pour MCA (à l'exception de notre travail) avec un des algorithmes présentés dans la Section 4.2. Dans la suite, l'algorithme disponible dans la littérature est nommé ALGO-C ; il possède une complexité en temps en $\mathcal{O}^*(3^{|\mathcal{C}|})$ et a été présenté dans le Théorème 50 (page 80). On a choisi de comparer les performances d'ALGO-C avec celles de ALGO-XH, qui possède une complexité en temps en $\mathcal{O}^*(3^{x_{\mathcal{H}}})$ et qui a été présenté dans le Théorème 61 (page 89). Puisque $x_{\mathcal{H}} \leq |\mathcal{C}|$, nous avons de bonnes raisons d'espérer que les performances réalisées par ALGO-XH soient meilleures que celles de ALGO-C.

Avant de continuer, on précise que l'équipe de Sebastian Böcker (Friedrich-Schiller-Universität Jena, Allemagne) est spécialisée dans l'identification de métabolites via l'utilisation des arbres de fragmentation – ce qui a mené à la création des problèmes MCS et MCA. Cette équipe a tout d'abord créé l'algorithme ALGO-C [18], puis a observé que l'utilisation de techniques de programmation linéaire [8] permet de résoudre plus rapidement des instances de MCS qu'en utilisant ALGO-C [41]. L'équipe de Sebastian Böcker a donc abandonné d'un point de vue applicatif le domaine de la complexité paramétrée au profit de ceux de la programmation linéaire [119] et des heuristiques [45] pour résoudre MCS. Ils ont (entre autres) mis à disposition de la communauté le logiciel Sirius 3 [16], déjà évoqué dans le Chapitre 3 (page 59), pour résoudre MCS. Dans le manuel d'utilisation de Sirius 3 [1], Böcker *et al.* expliquent que l'élaboration du logiciel, ainsi que des nombreuses recherches théoriques sous-jacentes, se sont étalées sur dix ans. Dans ce manuscrit, nous ne prétendons pas que l'algorithme ALGO-XH permet d'obtenir de meilleurs résultats que le logiciel Sirius 3 pour obtenir des arbres de fragmentation puisque ce dernier n'implémente pas l'algorithme ALGO-C. En revanche, on montre que l'algorithme ALGO-XH permet d'obtenir de meilleurs résultats que l'algorithme ALGO-C. Par conséquent, on montre qu'il n'est pas vain d'étudier le problème MCA sous le prisme de la complexité paramétrée et qu'on pourrait espérer, à terme, obtenir des résultats similaires à ceux de Sirius 3.

Les implémentations des algorithmes ALGO-C et ALGO-XH sont disponibles à l'adresse `https://github.com/Neojko/MCA`. Celles-ci ont été effectuées en Java dans l'optique d'être intégrées en plugins pour Gephi [2], un outil utilisé pour la visualisation et l'analyse de graphes.

4.3.1 Présentation des instances

On dispose de 626 949 instances de MCA fournies par l'équipe de Sebastian Böcker, que j'ai eu l'occasion de rencontrer pendant un séjour de deux semaines en 2017 [3]. Ces instances sont classifiées en fonction de la masse du métabolite inconnu étudié, qui varie entre 100 Da et 600 Da. De plus, il existe des instances *classiques*, qui ont été modélisées à partir de spectres – comme décrit dans la Section 3.1 –, et des instances *réduites*, qui ont été obtenues à partir des premières en appliquant les règles de réduction décrites dans l'article de White *et al.* [119]. On précise qu'on dispose ici de 330 369 instances classiques et seulement de 296 580 instances réduites – et donc qu'il n'existe pas d'instance réduite pour 10.23% des instances dont on dispose. Il existe au total dix dossiers d'instances, qui sont présentés dans le Tableau 4.3.

Pour chaque dossier d'instances, le Tableau 4.3 indique la valeur moyenne de tous les paramètres étudiés pour le problème MCA dans ce chapitre à l'exception du paramètre $t_{\mathcal{H}}$ – la treewidth de $\mathcal{H}$. En effet, déterminer la treewidth d'un graphe est NP-dur [4] et on a donc indiqué dans ce tableau la *dégénérescence* de

1. disponible à l'adresse `https://bio.informatik.uni-jena.de/sirius-training/`
2. `https://gephi.org/`
3. J'ai effectué deux séjours en Allemagne en 2017. Ces séjours ont été financés par le PHC PROCOPE numéro 37748TL entre la France et l'Allemagne.

$\mathcal{H}$, une borne inférieure de $t_{\mathcal{H}}$ qui se calcule en temps linéaire [85].

Dossier	#instances	n	m	m^-	$\lvert\mathcal{C}\rvert$	$x_{\mathcal{H}}$	s	deg	$\ell_{\mathcal{C}}$
100-200	237	19.41	109.33	93.61	17.66	13.37	85.20	7.38	1.75
200-300	14946	35.92	259.96	240.81	24.23	18.81	155.61	8.72	11.69
300-400	74857	74.34	809.10	775.38	34.31	28.70	319.85	12.96	40.03
400-500	165943	152.28	2753.52	2699.93	40.95	35.64	504.69	17.71	111.33
500-600	74386	320.61	10227.97	10132.59	48.06	43.58	785.70	25.51	272.55
r-100-200	9	26.89	132.67	61.11	26.89	20.33	106.78	7.22	0
r-200-300	7893	17.92	67.62	43.47	14.81	8.76	45.49	3.55	3.11
r-300-400	61886	24.46	125.28	95.54	16.27	10.27	69.41	4.16	8.19
r-400-500	155221	37.82	245.17	206.04	18.22	12.34	97.88	4.97	19.60
r-500-600	71571	79.03	648.08	590.01	23.93	17.67	164.62	7.29	55.10

Tableau 4.3 – Valeurs moyennes des différents paramètres étudiés dans ce chapitre pour le problème MCA sur l'ensemble d'instances dont on dispose. Ici, le dossier 100-200 contient toutes les instances dont le poids du métabolite d'entrée est compris entre 100 Da et 200 Da. Si la lettre 'r' est placée devant un nom de dossier, alors les instances de ce dossier ont été obtenues après application des règles de réduction présentées dans [119]. Pour toute instance de MCA, on note n le nombre de sommets du graphe G, m son nombre d'arcs, m^- le nombre d'arcs de poids négatif, $|\mathcal{C}|$ le nombre de couleurs, $x_{\mathcal{H}}$ le nombre de couleurs difficiles, s le nombre d'arcs à retirer de $\mathcal{H}$ pour qu'il se transforme en arborescence, deg la dégénérescence de $\mathcal{H}$ et $\ell_{\mathcal{C}} = n - |\mathcal{C}|$.

On établit maintenant plusieurs constats à partir du Tableau 4.3. Premièrement, on observe que la valeur des paramètres étudiés est généralement plus élevée dans une instance dont la masse du métabolite étudié est elle-même élevée. En effet, plus la masse d'un métabolite est élevée plus celui-ci est susceptible de se fragmenter pendant le processus de spectrométrie de masse. Le spectre produit contiendra donc plus de pics, ce qui augmentera le nombre de couleurs et donc potentiellement la valeur des autres paramètres du graphe.

Deuxièmement, on constate que la valeur de $x_{\mathcal{H}}$ est toujours inférieure de quelques unités à la valeur de $|\mathcal{C}|$ dans les instances fournies. En particulier, $x_{\mathcal{H}}$ est en moyenne au moins 25% plus petit que $|\mathcal{C}|$ dans les instances réduites que dans les instances classiques. Pourtant, dans le graphe d'entrée $G = (V, A)$ de chaque instance classique qui nous a été fournie, la racine r possède un voisin unique $r' \in V$ et tout le reste du graphe est *transitif*. En d'autres termes, pour tout ensemble de sommets $\{x, y, z\}$ de $V \setminus \{r\}$, s'il existe $(x, y) \in A$ et $(y, z) \in A$ alors il existe $(x, z) \in A$. En effet, si une première molécule est une sous-molécule d'une deuxième molécule qui est elle-même une sous-molécule d'une troisième molécule, alors la première molécule est nécessairement une sous-molécule de la troisième molécule. Cette transitivité des instances explique en partie pourquoi la différence entre le nombre moyen de couleurs d'une instance et le nombre moyen de couleurs difficiles d'une instance est sensiblement la même dans les instances classiques et dans les instances réduites.

Troisièmement, on remarque que les autres algorithmes FPT présentés dans ce chapitre ont peu d'intérêt à être implémentés et testés sur l'ensemble des instances dont nous disposons. Par exemple, le nombre moyen d'arêtes de poids négatif contenues dans une instance de MCA (colonne m^- du Tableau 4.3) exclut tout intérêt à implémenter l'algorithme présenté dans le Théorème 54. De même, on rappelle que la dégénérescence d'un graphe n'est qu'une borne inférieure pour la treewidth et que les valeurs de $\ell_{\mathcal{C}}$ deviennent supérieures à celles de $x_{\mathcal{H}}$ si le métabolite étudié est de masse supérieure ou égale à 300 Da (resp. à 400 Da) pour les instances classiques (resp. réduites). Par conséquent, l'algorithme présenté dans le

Théorème 75 présente également un intérêt pratique limité – même si le métabolite étudié est de masse inférieure à 300 Da.

4.3.2 Analyse des résultats

On a exécuté les algorithmes ALGO-C et ALGO-XH sur l'ensemble des 626 949 instances dont nous disposons avec un temps limite de 60 secondes pour résoudre chaque instance avec chaque algorithme. Les tests ont été effectués sur un serveur contenant 4 processeurs de 20 cœurs hyperthreadés (vus par le système comme 160 cœurs) Intel Xeon E7-8870 à 2.10GHz, 1.5 To de RAM et 10 To de stockage local extensible par SAN et/ou cluster filesystem.

La Figure 4.13 représente le nombre d'instances résolues par les algorithmes ALGO-C et ALGO-XH en fonction du nombre de couleurs contenues dans ces instances. On précise que toutes les instances qui ont été résolues par ALGO-C ont aussi été résolues par ALGO-XH. Dans un premier temps, on observe que les deux algorithmes résolvent une vaste majorité des instances contenant 30 couleurs ou moins dans le délai imparti, puis que les deux algorithmes deviennent beaucoup moins efficaces quand les instances contiennent plus de 30 couleurs. D'après cette première figure, les performances des deux algorithmes semblent similaires puisque l'algorithme ALGO-XH ne résout que 4.6% d'instances de plus que l'algorithme ALGO-C en 60 secondes.

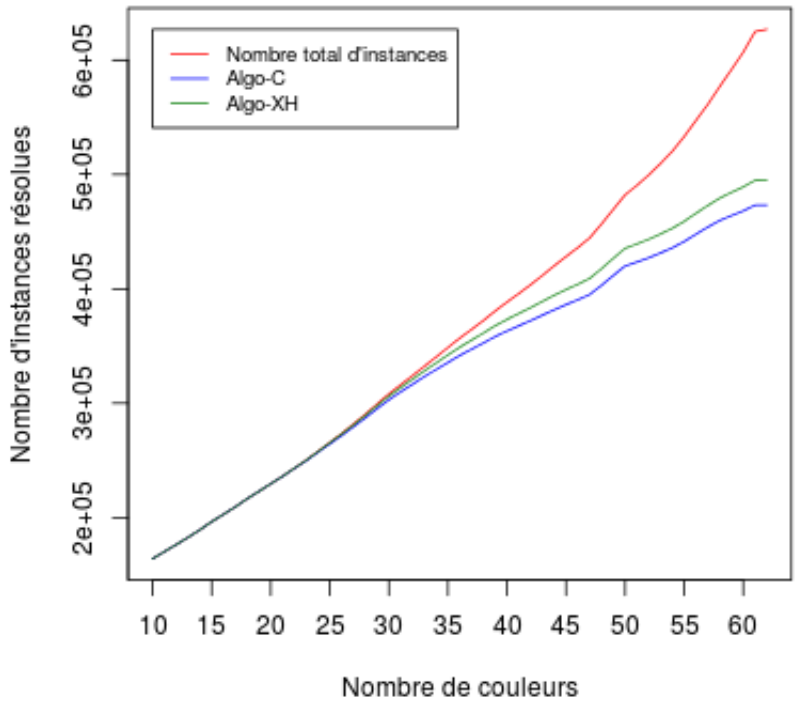


Figure 4.13 – Nombre d'instances de MCA résolues en 60 secondes en fonction du nombre de couleurs contenues dans les instances.

Avant de comparer plus précisément les performances des deux algorithmes, on souligne que, d'après le Tableau 4.3, la majorité des arcs est de poids négatif dans les instances de MCA dont on dispose. En réalité, on a observé que le poids d'une solution de MCA est nul dans 442 297 des 495 056 instances – soit 89.3% – qui ont été résolues par au moins ALGO-C ou ALGO-XH. De plus, l'algorithme ALGO-C a retourné une solution nulle de MCA en moins d'une seconde pour plus de 91% (et plus de 95% pour

ALGO-XH) de ces 442 297 instances. Par conséquent, on compare dans la suite les performances des deux algorithmes uniquement dans les instances que nous appellerons *dures*, c'est-à-dire les instances qui (i) n'ont été résolues ni par ALGO-C ni par ALGO-XH dans le délai imparti, ou (ii) ont été résolues par au moins un de ces deux algorithmes dans le délai imparti et dont le poids de la solution optimale donnée en sortie est strictement positif. Pour chaque dossier d'instances, on reporte dans le Tableau 4.4 la valeur moyenne de tous les paramètres étudiés pour le problème MCA dans ce chapitre – à l'exception du paramètre $t_{\mathcal{H}}$ qui est remplacé par la dégénérescence de $\mathcal{H}$ – en prenant uniquement en compte les instances dures dont on dispose. Globalement, toutes les valeurs moyennes des paramètres sont plus élevées dans ce tableau que dans le Tableau 4.3.

Dossier	#instances	n	m	m^-	$\lvert\mathcal{C}\rvert$	$x_{\mathcal{H}}$	s	deg	$\ell_{\mathcal{C}}$
100-200	96	24.01	161.06	128.71	22.22	18.06	131.60	9.86	1.79
200-300	4124	51.93	516.76	465.41	34.58	29.57	317.76	13.99	17.35
300-400	20492	107.01	1596.64	1507.30	46.64	41.66	609.74	20.68	60.36
400-500	46055	202.12	4512.48	4380.03	53.49	48.80	860.12	26.31	148.64
500-600	23157	343.95	11688.88	11515.78	57.39	53.32	1112.47	32.34	286.56
r-100-200	8	30.0	149.13	68.63	30.0	22.88	120.13	8.13	0
r-200-300	3248	33.91	150.31	97.40	27.13	18.94	104.88	6.87	6.78
r-300-400	19949	57.49	354.43	274.01	35.09	26.42	199.30	9.96	22.40
r-400-500	45188	98.74	764.82	649.86	41.76	33.28	301.39	13.10	56.97
r-500-600	22335	182.88	1820.43	1672.18	47.99	39.97	441.74	17.20	134.89

Tableau 4.4 – Valeurs moyennes des différents paramètres étudiés dans ce chapitre pour le problème MCA sur l'ensemble d'instances dures dont on dispose. Ici, le dossier 100-200 contient toutes les instances dures dont le poids du métabolite d'entrée est compris entre 100 Da et 200 Da. Si la lettre 'r' est placée devant un nom de dossier, alors les instances de ce dossier ont été obtenues après application des règles de réduction présentées dans [119]. Pour toute instance de MCA, on note n le nombre de sommets de G, m son nombre d'arcs, m^- le nombre d'arcs de poids négatif, $|\mathcal{C}|$ le nombre de couleurs, $x_{\mathcal{H}}$ le nombre de couleurs difficiles, s le nombre d'arcs à retirer de $\mathcal{H}$ pour qu'il se transforme en arborescence, deg la dégénérescence de $\mathcal{H}$ et $\ell_{\mathcal{C}} = n - |\mathcal{C}|$.

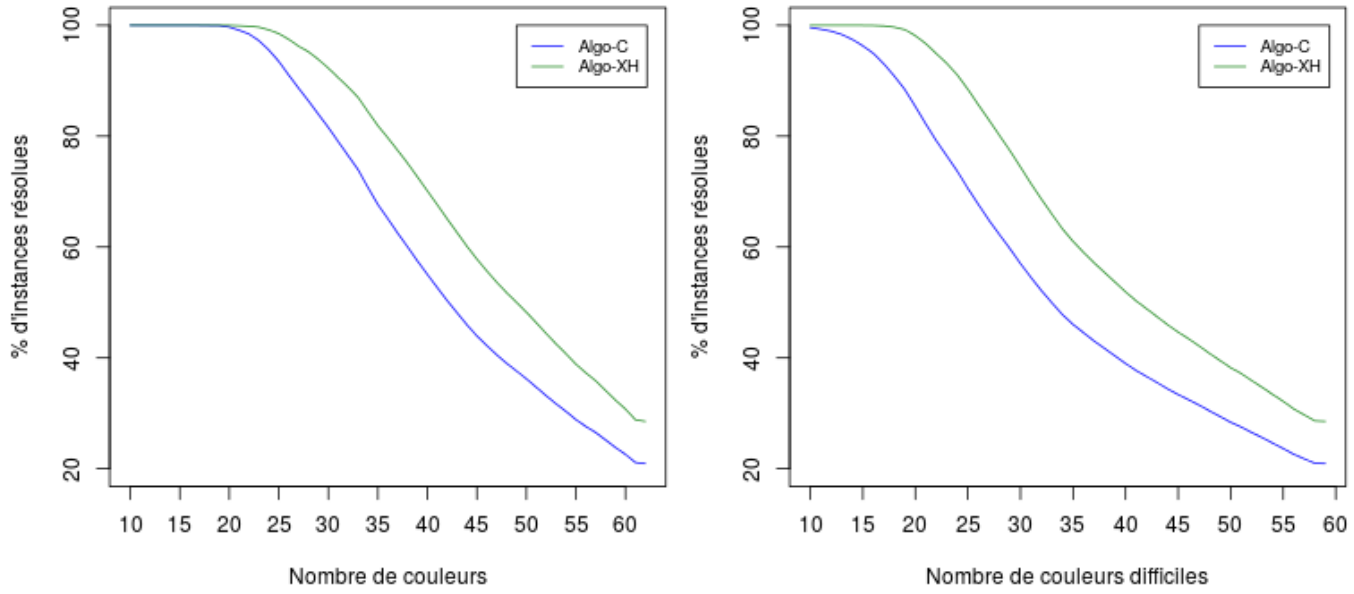
Le Tableau 4.5 compare le nombre d'instances dures de MCA dont nous disposons et qui ont été résolues, avec le nombre d'instances total de MCA dont nous disposons et qui ont été résolues. On rappelle – à partir de ce tableau et de la Figure 4.13 – que ALGO-XH résout 4.6% d'instances de plus que ALGO-C au total. Cependant, on observe que plus de 70% des instances qui ont été résolues par au moins un des deux algorithmes ne sont pas des instances dures. La majorité de ces instances présente donc un intérêt limité – même si 7591 instances dont le poids d'une solution est égal à 0 ont été résolues en moins de 60 secondes uniquement par ALGO-XH. Finalement, on observe que ALGO-XH a en fait résolu plus de 36% d'instances dures de plus que ALGO-C.

	#instances	ALGO-C et ALGO-XH	ALGO-C uniquement	ALGO-XH uniquement
Toutes instances	626 949	473 370 (75.5%)	0 (0%)	21 686 (3.4%)
Instances dures	184 652	38 664 (20.9%)	0 (0%)	14 095 (7.6%)

Tableau 4.5 – Distinction entre le nombre d'instances (classiques et réduites confondues) total et le nombre d'instances dures (classiques et réduites confondues) résolues par ALGO-C et ALGO-XH. Pour chaque catégorie d'instances, on note respectivement en colonnes le nombre d'instances total, le nombre d'instances résolues à la fois par ALGO-C et ALGO-XH, puis le nombre d'instances résolues uniquement par ALGO-C, puis par ALGO-XH.

Dans la Figure 4.14, on compare le pourcentage d'instances dures résolues par les algorithmes ALGO-C

et ALGO-XH en fonction du nombre de couleurs $|\mathcal{C}|$ et du nombre de couleurs difficiles $x_{\mathcal{H}}$ des instances. Ainsi, on remarque que l'algorithme ALGO-XH résout plus d'instances que l'algorithme ALGO-C, quelle que soit la valeur de $|\mathcal{C}|$ ou $x_{\mathcal{H}}$. Maintenant, on rappelle que ALGO-C a une complexité en temps en $\mathcal{O}^*(3^{|\mathcal{C}|})$ et que ALGO-XH a une complexité en temps en $\mathcal{O}^*(3^{x_{\mathcal{H}}})$, où $x_{\mathcal{H}} \leq |\mathcal{C}|$. On compare ensuite dans la Figure 4.15 le pourcentage d'instances dures contenant au plus k couleurs qui ont été résolues par ALGO-C avec le pourcentage d'instances dures contenant au plus k couleurs difficiles qui ont été résolues par ALGO-XH. Ainsi, on observe dans cette figure que ALGO-C résout autant d'instances dures de MCA contenant au plus k couleurs que ALGO-XH dans les instances dures de MCA contenant au plus k couleurs difficiles.



(a) (b)

Figure 4.14 – Performances des algorithmes ALGO-C et ALGO-XH dans les instances dures en fonction (a) du nombre de couleurs $|\mathcal{C}|$ et (b) du nombre de couleurs difficiles $x_{\mathcal{H}}$.

Puisque, pour tout $k \in \mathbb{N}$, le pourcentage d'instances dures contenant au plus k couleurs qui ont été résolues par ALGO-C est similaire au pourcentage d'instances dures contenant au plus k couleurs difficiles qui ont été résolues par ALGO-XH, on pourrait supposer que les instances uniquement résolues par ALGO-XH dans une limite de 60 secondes seraient également résolues par ALGO-C en augmentant la limite de temps par instance. Cette théorie est réfutée par les résultats présentés dans la Figure 4.16, qui décrit le pourcentage d'instances dures résolues par ALGO-C et ALGO-XH en fonction de la limite de temps écoulée. En effet, on observe dans cette figure que ALGO-XH résout plus d'instances dures avec une limite de 5 secondes par instance que ALGO-C avec une limite de 60 secondes par instance. On remarque également que l'écart entre le pourcentage d'instances dures résolues par ALGO-XH est entre 6 et 7 points plus élevé que le pourcentage d'instances dures résolues par ALGO-C, quelque soit le délai de temps accordé aux deux algorithmes pour résoudre une instance de MCA.

Avant d'achever cette étude, on mesure l'impact de l'application des règles de réduction de White *et al.* [119] sur le pourcentage d'instances dures résolues par ALGO-C et ALGO-XH. Les résultats sont présentés dans la Figure 4.17. Dans cette figure, il est à noter que le dossier r-100-200 contient uniquement 9 instances dures et que les statistiques ont donc peu d'intérêt pour ce dossier. En règle générale, on observe que la différence entre le pourcentage d'instances dures résolues par ALGO-XH et le pourcentage

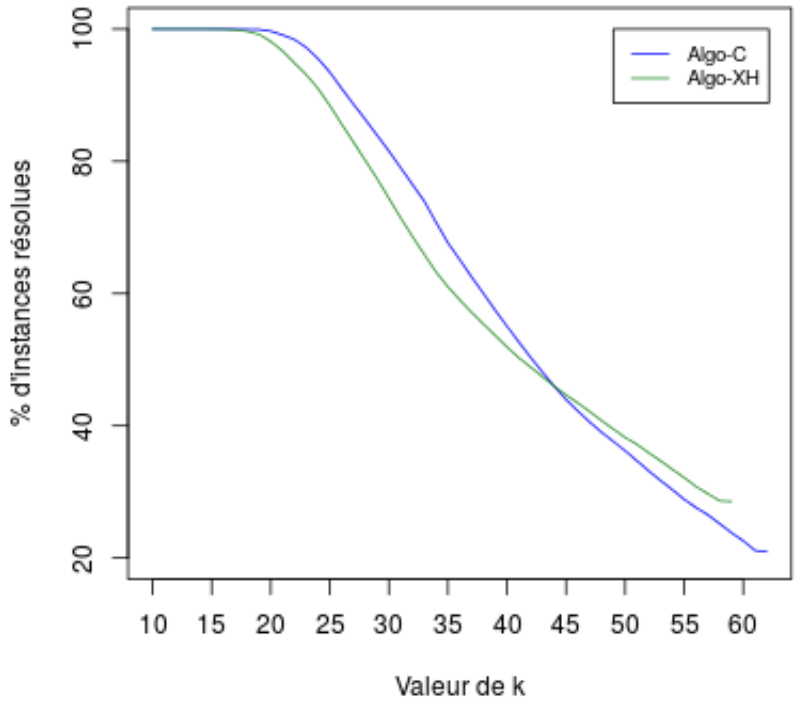
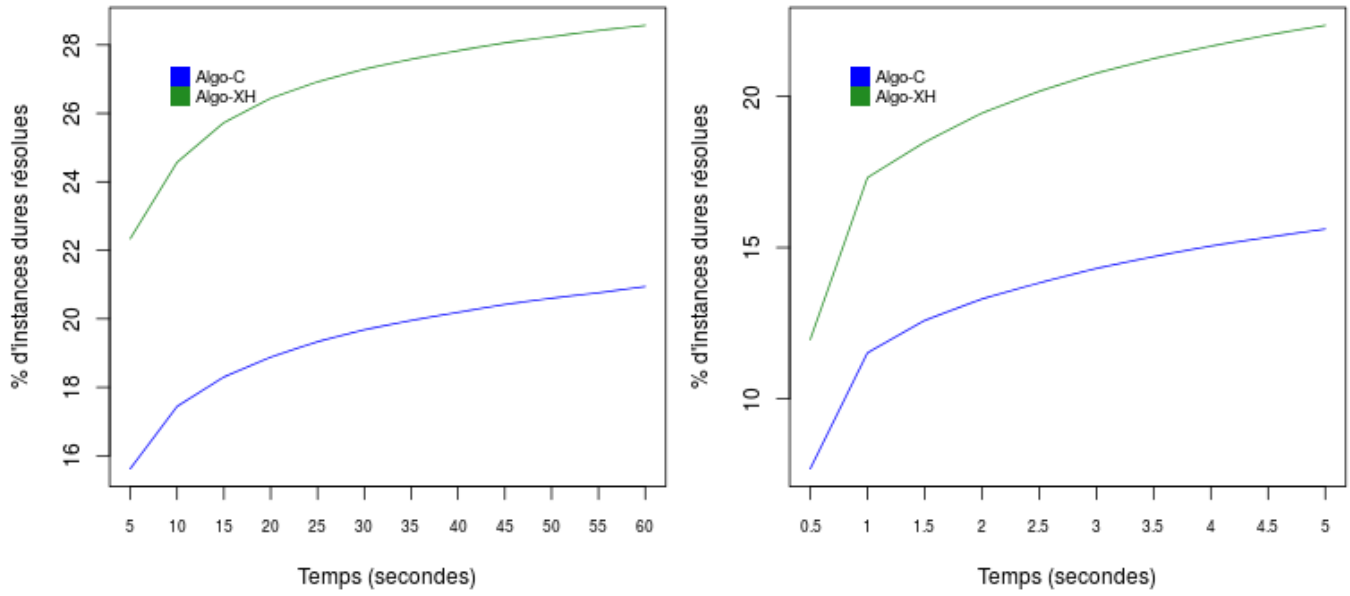


Figure 4.15 – Comparaison des performances de ALGO-C dans les instances dures de MCA contenant au plus k couleurs et de ALGO-XH dans les instances dures de MCA contenant au plus k couleurs difficiles.



(a) Dans une limite de 5 secondes à 60 secondes.

(b) Dans une limite de 0.5 seconde à 5 secondes.

Figure 4.16 – Performances des algorithmes ALGO-C et ALGO-XH en fonction du temps d'exécution imparti (en pourcentage des instances dures résolues).

d'instances dures résolues par ALGO-C est supérieure à la différence entre le pourcentage d'instances classiques dures résolues par ALGO-C (resp. ALGO-XH) et le pourcentage d'instances réduites dures résolues par ALGO-C (resp. ALGO-XH). Cependant, on constate que l'impact des règles de réduction est amplifié au fur et à mesure que la masse du métabolite étudié augmente. Par exemple, l'algorithme ALGO-XH résout 11.12% des instances classiques du dossier 500-600 (à partir desquelles il existe une instance réduite) et 14.41% des instances réduites de ce dossier, ce qui représente une augmentation de près de 30% d'instances résolues supplémentaires.

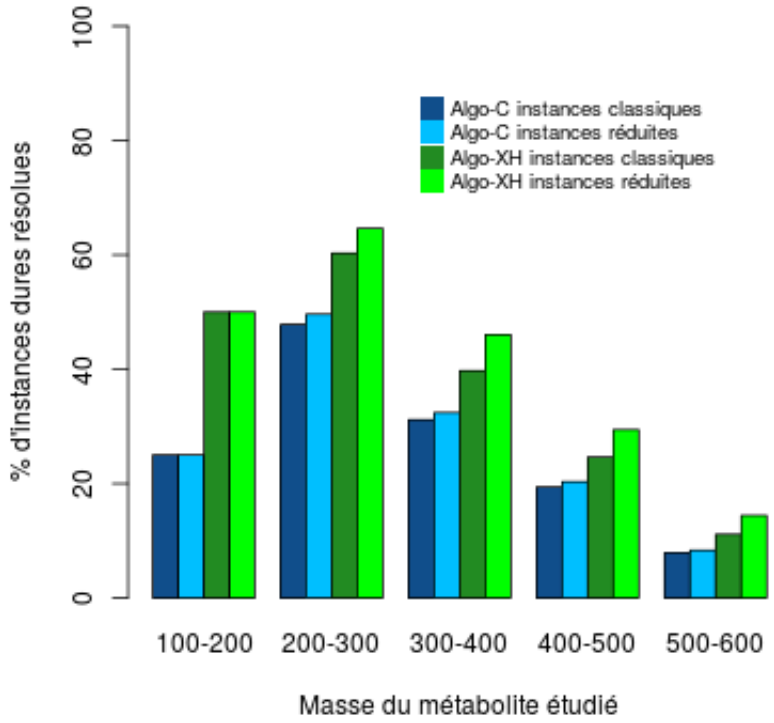


Figure 4.17 – Performances des algorithmes ALGO-C et ALGO-XH avant et après application des règles de réduction de White *et al.* [119] et en fonction de la masse du métabolite étudié. On prend ici uniquement en compte les instances classiques pour lesquelles il existe aussi une instance réduite. De plus, on rappelle que le dossier r-100-200 ne contient que 9 instances.

4.4 Conclusion

On avait conclu le Chapitre 3 en prouvant dans le Théorème 46 (voir page 75) que MCA se résout en temps polynomial quand le graphe de hiérarchie de couleurs $\mathcal{H}$ du graphe d'entrée est un arbre. Dans la première partie de ce chapitre, on a montré des résultats de complexité paramétrée pour MCA relativement à deux paramètres qui sont constants quand $\mathcal{H}$ est un arbre : $x_{\mathcal{H}}$, le nombre de sommets de degré entrant au moins égal à 2 dans $\mathcal{H}$, et $t_{\mathcal{H}}$, la treewidth de (la version non-orientée de) $\mathcal{H}$. On a également montré dans la Section 4.2.1 des algorithmes FPT relativement à $\ell_{\mathcal{C}} = n - |\mathcal{C}|$ dans des instances contraintes de MCA et prouvé que l'un de ces algorithmes était optimal sous l'hypothèse SETH.

Dans la seconde partie de ce chapitre, on a confronté nos résultats à ceux de la littérature sur des données réelles. Pour cela, on a comparé les performances de ALGO-C, l'algorithme FPT relativement à $|\mathcal{C}|$ créé par Böcker et Rasche [18] (voir page 80), à celles de ALGO-XH, l'algorithme FPT relativement à $x_{\mathcal{H}}$ qui

est présenté dans le Théorème 61 (voir page 89). Le bilan de cette partie est globalement positif. En effet, toutes les instances de MCA résolues par ALGO-C ont également été résolues par ALGO-XH. De plus, ALGO-XH a résolu plus d'instances de MCA dans une limite de 5 secondes par instance que ALGO-C dans une limite de 60 secondes par instance. On note toutefois que l'allure générale des instances de MCA a limité l'impact pratique de certains résultats théoriques qu'on a obtenus. En particulier, les règles de Réduction 44 (page 74) et 66 (page 96) ne s'appliquent pas sur des instances réelles de MCA puisque les graphes d'entrée de ces instances sont transitifs – dans le sens où s'il existe deux arcs (x, y) et (y, z) dans ces graphes alors il existe également un arc (x, z) –, ce qui implique que la majorité des couleurs de $\mathcal{C}$ sont également des couleurs difficiles. A la lueur des résultats présentés, il serait donc intéressant de trouver de nouvelles règles de réduction afin d'accroître l'impact des résultats de complexité paramétrée qu'on a obtenus.

Conclusion

Dans ce manuscrit, on s'intéresse à des problématiques biologiques qui se modélisent sous la forme de graphes aux sommets colorés. Dans un premier temps, on observe que ces graphes représentent intuitivement les interactions entre les différentes entités d'un système biologique. On modélise ainsi le problème GRAPH MOTIF, ainsi que quelques unes de ses variantes, dans le but de mieux comprendre ces interactions. Dans un second temps, on étudie un nouveau problème, appelé MAXIMUM COLORFUL ARBORESCENCE (MCA), dont le but est d'identifier des métabolites inconnus en spectrométrie de masse. Bien que possédant des similarités avec GRAPH MOTIF et ses variantes, le problème MCA présente également des contraintes inédites qui sont liées à la nature des données biologiques dont on dispose. Dans la suite, on présente les contributions apportées par ce manuscrit pour les problèmes GRAPH MOTIF et MCA, puis on évoque les perspectives que cette thèse a engendrées.

Contributions. Après avoir rappelé des notions de base en théorie des graphes et en théorie de la complexité dans le Chapitre 1, on modélise dans le Chapitre 2 le problème GRAPH MOTIF, ainsi que trois de ses variantes : le problème COLORFUL GRAPH MOTIF impose au motif d'être colorful, le problème WEIGHTED GRAPH MOTIF introduit une fonction de pondération sur les arêtes, et enfin le problème MAX GRAPH MOTIF est une version optimisation de GRAPH MOTIF dont le but est de déterminer une sous-occurrence de cardinalité maximale du motif. On procède ensuite à la réalisation d'un état de l'art de ces quatre problèmes. Dans un premier temps, on distingue des instances pour lesquelles ces problèmes peuvent être résolus en temps polynomial de celles où ces problèmes sont NP-complets. Dans un second temps, on présente des résultats d'approximation (principalement négatifs). Enfin, on montre des résultats de complexité paramétrée (algorithmes FPT et recherche de noyau) concernant deux paramètres : la cardinalité du motif $k = |\mathcal{M}|$ et le nombre de sommets qui n'appartiennent pas à une occurrence du motif $\ell = n - |\mathcal{M}|$, où n est le nombre de sommets du graphe d'entrée. On rappelle ici que l'état de l'art réalisé dans le Chapitre 2 n'est pas exhaustif, puisqu'il existe en réalité une quinzaine de variantes de GRAPH MOTIF dans la littérature ainsi que des résultats de complexité paramétrée relativement à une multitude de paramètres. Nous collaborons actuellement avec Florian Sikora [1] (LAMSADE, Paris Dauphine) dans le but de proposer un état de l'art, d'un point de vue algorithmique, du problème GRAPH MOTIF et de ses variantes ; l'article fondateur de Lacroix *et al.* [79] a en effet été cité 169 fois selon Google Scholar, ce qui illustre un besoin de réaliser cette étude.

Dans le Chapitre 3, on modélise le problème MAXIMUM COLORFUL ARBORESCENCE (MCA) afin de répondre à une problématique d'identification de métabolites. Dans la modélisation de ce problème, on introduit la notion de *graphe de hiérarchie de couleurs*, appelé $\mathcal{H}$, et on montre que ce graphe est nécessairement un DAG dans une instance de MCA. On initie dans la suite de ce chapitre une étude algorithmique de MCA. En particulier, on montre que MCA est hautement inapproximable même quand le graphe d'entrée est un arbre très contraint. Cependant, le fait que $\mathcal{H}$ est nécessairement un DAG nous permet d'obtenir des algorithmes d'approximation dans ces mêmes arbres contraints. Enfin, on conclut le Chapitre 3 en montrant que MCA appartient à P si $\mathcal{H}$ est un arbre.

1. http://www.lamsade.dauphine.fr/∼sikora/

Puisque MCA peut se résoudre en temps polynomial quand $\mathcal{H}$ est un arbre, on s'intéresse dans le Chapitre 4 à deux paramètres, $x_{\mathcal{H}}$ et $t_{\mathcal{H}}$, qui sont liés à la structure de $\mathcal{H}$. On définit également deux paramètres $\ell_{\mathcal{C}}$ et $\ell \geq \ell_{\mathcal{C}}$ qui sont liés au nombre de répétitions de couleurs dans G. On poursuit ensuite dans ce chapitre l'étude algorithmique de MCA entreprise dans le Chapitre 3 en proposant des résultats de complexité paramétrée. Dans un premier temps, on montre notamment un nouvel algorithme de complexité paramétrée en $\mathcal{O}^*(3^{x_{\mathcal{H}}})$. Ce résultat est une amélioration théorique de l'unique algorithme FPT de la littérature en $\mathcal{O}^*(3^{|\mathcal{C}|})$, où $\mathcal{C}$ est l'ensemble de couleurs d'une instance de MCA, puisque $x_{\mathcal{H}} \leq |\mathcal{C}|$. Dans un second temps, on prouve que MCA admet un kernel polynomial relativement à $x_{\mathcal{H}} + \ell$, mais pas – sous des hypothèses de complexité – relativement à $x_{\mathcal{H}}$ ou même à $x_{\mathcal{H}} + \ell_{\mathcal{C}}$. Dans un troisième temps, on montre que MCA est W[2]-dur relativement à $t_{\mathcal{H}}$, mais que MCA appartient à FPT relativement à $t_{\mathcal{H}} + \ell_{\mathcal{C}}$. Enfin, on présente une série d'algorithmes FPT relativement à $\ell_{\mathcal{C}}$ pour des restrictions de MCA, alors que le problème est W[1]-dur relativement à $\ell \geq \ell_{\mathcal{C}}$. En particulier, on montre que l'un de ces algorithmes est "optimal" sous l'hypothèse SETH.

Questions techniques. On commence par lister ci-dessous, dans l'ordre chronologique, quelques questions techniques que les résultats obtenus dans ce manuscrit ont engendrés.

- On rappelle que des experts en spectrométrie de masse ont évalué manuellement 79 arbres de fragmentations contenant un total de 808 arcs afin de montrer que les arbres de fragmentation permettent non seulement de déterminer la formule moléculaire d'un métabolite, mais également d'identifier ce métabolite [102] (voir page 60). Ainsi, on observe que les 79 solutions de MCA correspondants à ces 79 arbres de fragmentation contiennent en moyenne environ 10 arcs. Alors qu'on s'est principalement intéressé à des paramètres liés à la structure de $\mathcal{H}$ et aux répétitions de couleurs d'une instance dans ce manuscrit, il est ainsi également légitime de se demander si MCA appartient à FPT relativement au nombre d'arcs d'une solution.
- Dans la Section 3.2.2 (page 66), on montre que UMCA-2 restreint aux comb-graphs ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{3}-\epsilon})$, pour tout $\epsilon > 0$. On montre également que MCA$^+$-2 restreint aux comb-graphs ne peut pas être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{2}-\epsilon})$, pour tout $\epsilon > 0$. Enfin, on prouve que UMCA restreint aux comb-graphs peut être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{2}})$. A partir de ces trois résultats, on se demande si UMCA-2 restreint aux comb-graphs peut être approximé en temps polynomial avec un ratio de $\mathcal{O}(n^{\frac{1}{3}})$.
- Dans la Section 3.2.3 (page 70), on montre que UMCA-2 restreint aux superstars est APX-dur et que UMCA-2 restreint aux superstars est 2-approximable. Existe-t-il un meilleur algorithme d'approximation pour UMCA-2 restreint aux superstars ?
- Dans la Section 4.2.1 (page 83), on montre que MCA$^+$ restreint aux arbres peut être résolu en temps $\mathcal{O}^*(1.62^{\ell_{\mathcal{C}}})$ et que UMCA restreint aux arbres peut être résolu en temps $\mathcal{O}^*(1.33^{\ell_{\mathcal{C}}})$. Ces algorithmes sont-ils optimaux comme celui qui résout MCA$^+$ en temps $\mathcal{O}^*(2^{\ell_{\mathcal{C}}})$?
- Dans la Section 4.2.3 (page 99), on montre que MCA est FPT relativement à $t_{\mathcal{H}} + \ell_{\mathcal{C}}$ – et donc relativement à $t_{\mathcal{H}} + \ell_{\mathcal{C}}$ puisque $\ell \geq \ell_{\mathcal{C}}$ – mais que MCA n'admet pas de kernel polynomial relativement à $t_{\mathcal{H}} + \ell_{\mathcal{C}}$. Ainsi, on se demande si MCA admet un kernel polynomial relativement à $t_{\mathcal{H}} + \ell$.
- Pour finir, on remarque que les Règles de réduction 66 et 68 (page 98) ne seraient pas très efficaces dans les instances fournies par l'équipe de Sebastian Böcker puisque leurs graphes d'entrée de MCA sont transitifs. En d'autres termes, s'il existe deux arcs (x, y) et (y, z) dans un de ces graphes, alors il existe également un arc (x, z). De plus, la majorité des arcs sont de poids négatif dans les instances réelles sur lesquelles on a effectué nos expérimentations dans la Section 4.3 (page 106). A partir de ces deux observations, on peut ainsi raisonnablement penser qu'il existe de nouvelles règles de réduction visant à supprimer ces arcs et qui permettent ensuite d'appliquer efficacement les règles décrites dans ce manuscrit.

Perspectives. Dans ce manuscrit, on présente des algorithmes FPT pour MCA relativement à de nombreux paramètres différents. A moyen terme, l'idée est d'implémenter un algorithme "méta-FPT" [1]. Cet algorithme commencerait par calculer les valeurs des différents paramètres pour lesquels MCA admet un algorithme FPT, puis choisirait l'algorithme FPT le plus efficace.

Pour finir, on explique dans le Chapitre 2 que les sommets d'un graphe de GRAPH MOTIF qui représente un réseau métabolique sont colorés relativement à la fonction de l'enzyme qui catalyse la réaction associée à ces sommets. Cependant, il arrive en réalité que réactions soient catalysées par plusieurs enzymes. Pour exprimer cela, il existe une variante de GRAPH MOTIF, appelée LIST-COLORED GRAPH MOTIF, dans laquelle la fonction de coloration col associe les sommets de V non pas à une couleur de $\mathcal{C}$ mais à un sous-ensemble de couleurs de $\mathcal{C}$. On souhaite transposer l'idée du graphe de hiérarchie de couleurs $\mathcal{H}$ du problème MCA au problème LIST-COLORED GRAPH MOTIF. Puisque les instances de LIST-COLORED GRAPH MOTIF sont des graphes non-orientés, on utilise dans la suite le terme de graphe d'*adjacence* de couleurs.
On a imaginé deux modèles de graphes d'adjacence de couleurs du graphe d'entrée (non-orienté) d'une instance de LIST-COLORED GRAPH MOTIF. Actuellement, on précise que ces deux modèles ont été imaginés avec des considérations purement algorithmiques. Pour construire le graphe Adj_{red} (pour "réduit") du premier modèle, on crée une arête entre deux couleurs c et c' dans Adj_{red} si et seulement si il existe une arête d'un sommet de couleur c vers un sommet de couleur c' dans G. Ce premier modèle est ainsi très similaire au graphe de hiérarchie de couleurs d'une instance de MCA. Toutefois, celui-ci ne prend pas en compte les relations entre les couleurs qui sont associées à un même sommet. Ainsi, pour construire le graphe d'adjacence Adj_{full} du deuxième modèle, on initialise $Adj_{full} = Adj_{red}$ puis, pour toute paire de couleurs c et c' associées à un même sommet de G, on ajoute un arc entre c et c' dans Adj_{red}. On essaie ensuite de déterminer la complexité du problème LIST-COLORED GRAPH MOTIF selon la classe de graphes à laquelle appartient Adj_{red} (resp. Adj_{full}). Des résultats préliminaires intéressants ont été obtenus lors d'une visite chez Christian Komusiewicz en Allemagne en mai 2017 [1] et méritent d'être approfondis.

1. pour reprendre l'expression de Christian Komusiewicz (Philipps-Universität Marburg, Allemagne)

1. J'ai effectué deux séjours en Allemagne en 2017. Ces séjours ont été financés par le PHC PROCOPE numéro 37748TL entre la France et l'Allemagne.

Glossaire

arborescence *DAG* avec une racine unique et dont le degré entrant de chaque sommet est égal à 1. 16

arbre Graphe *connexe* et sans cycles. 15

arbre couvrant Un arbre couvrant d'un graphe $G = (V, E)$ est un arbre contenu dans G qui contient tous les sommets de V. 64

biparti Un graphe $G = (V, E)$ est biparti s'il existe V_1 et V_2 tels que $V_1 \cup V_2 = V$ et $V_1 \cap V_2 = \emptyset$. 15, 45

caterpillar Arbre qui devient un chemin si on enlève ses *feuilles*. 15, 46, 73, 120

clique Graphe $G = (V, E)$ tel que $E = \{(u, v) : \{u, v\} \in V\}$. 46

CNF Une formule ϕ du problème SAT est CNF (pour "Conjunctive Normal Form") si ϕ est une conjonction de clauses. 22

cograph Graphe $P4$-free, *i.e.* qui ne contient pas de chemin à 4 sommets comme *sous-graphe induit*. 46, 120

colorful Un graphe (resp. un ensemble de sommets) est colorful s'il ne contient pas deux sommets de même couleur. 15, 41, 47, 59

comb-graph Arbre de *degré* maximum 3 dans lequel il existe un chemin contenant tous les sommets de degré 3. 15, 46, 48, 52, 57, 66

composante connexe Une composante connexe de G est un *sous-graphe induit* et *connexe* $G' \subseteq G$ tel qu'il n'existe pas d'autre sous-graphe induit et connexe $G'' \subseteq G$ avec $G' \subset G''$. 14

connexe Un graphe $G = (V, E)$ (resp. un ensemble de sommets $V' \in V$) est connexe s'il existe un chemin entre chaque paire de sommets de V (resp. V') dans G . 14, 119

couleur difficile Une couleur $c \in \mathcal{C}$ est difficile si le degré entrant de c est strictement supérieur à 1 dans $\mathcal{H}$. 81

DAG Graphe orienté *connexe* et sans circuits. 16, 28, 58, 61, 119

degré Soit $G = (V, E)$ un graphe. Pour tout $v \in V$, le degré $d(v)$ de v se définit comme suit : $d(v) = |\{(u, v) \in E : u \in V\}|$. 45, 67, 119

diamètre Plus grande *distance* entre deux sommets d'un graphe. 46

distance La distance entre deux sommets u et v d'un graphe G est la longueur du plus court chemin entre u et v dans G. 15, 65, 119

feuille Sommet de degré 1 dans un arbre. Si cet arbre est enraciné, la racine n'est pas une feuille. 15, 46, 70, 119, 120

fully-colorful Un sous-graphe fully-colorful de G est un sous-graphe de G contenant *exactement* une occurrence de chaque couleur de G. 83, 102

hauteur Plus grande *distance* de la racine d'un arbre à un autre sommet . 15, 66, 70, 120

indépendant Soit $G = (V, E)$ un graphe. Un ensemble de sommets $V' \subseteq V$ est indépendant s'il n'existe pas $(u, v) \in E$ tels que u et v appartiennent à V'. 14

lobster Arbre qui devient un *caterpillar* si on enlève ses *feuilles*. 46, 120

ordre topologique Un ordre topologique d'un graphe orienté $G = (V, A)$ est un ordre total sur l'ensemble des sommets de V, dans lequel $u \in V$ précède $v \in V$ s'il n'existe pas de chemin de v vers u dans G. Si G admet un ordre topologique de ses sommets, alors G est un DAG. 16, 66, 91

PTAS La classe PTAS (pour "Polynomial-Time Approximation Scheme") contient tous les problèmes qui admettent un algorithme d'approximation (appelé algorithme PTAS) de ratio $(1 + \epsilon)$, pour tout $\epsilon > 0$.. 72

sous-graphe induit Soit $G = (V, E)$ et $V' \subseteq V$. Le sous-graphe induit de G par V' se note $G[V'] = (V', E')$, où $E' = \{(u, v) \in E : \{u, v\} \subseteq V'\}$. 14, 46, 119

split graph Graphe dont l'ensemble des sommets peut être partitionné en deux sous-ensembles V_1 et V_2 tels que $G[V_1]$ est une clique et V_2 est un ensemble de sommets indépendants. 46, 120

superstar Arbre de *hauteur* 2. Une superstar est un cas particulier de *lobster*. 15, 46, 48, 57, 70

threshold graph Graphe qui est à la fois un *split graph* et un *cographe*. 46

treewidth Paramètre d'un graphe dont la définition exacte (qui est donnée page 16) est trop longue pour entrer dans ce glossaire. 16, 45, 79, 81

version non-orientée Soit $G = (V, A)$ un graphe orienté. La version non-orientée de G est le graphe non-orienté $U(G) = (V, E)$ dans lequel chaque arc $(u, v) \in A$ est remplacé par une arête $(u, v) \in E$. 15, 81, 99, 100, 102

Bibliography

[1] E. Alm and A. P. Arkin. Biological networks. *Current opinion in structural biology*, 13(2):193–202, 2003. 42, 43

[2] N. Alon, R. Yuster, and U. Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995. 26, 27, 32, 33

[3] A. M. Ambalath, R. Balasundaram, C. R. H., V. Koppula, N. Misra, G. Philip, and M. S. Ramanujan. On the Kernelization Complexity of Colorful Motifs. In V. Raman and S. Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2010. 46, 48, 52

[4] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. 106

[5] N. Atias and R. Sharan. Comparative analysis of protein networks: hard problems, practical solutions. *Commun. ACM*, 55(5):88–97, 2012. 39

[6] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012. 12, 70, 72

[7] R. Bellman. Dynamic Programming Treatment of the Travelling Salesman Problem. *J. ACM*, 9(1):61–63, 1962. 30

[8] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997. 106

[9] N. Betzler, M. R. Fellows, C. Komusiewicz, and R. Niedermeier. Parameterized Algorithms and Hardness Results for Some Graph Motif Problems. In P. Ferragina and G. M. Landau, editors, *Combinatorial Pattern Matching, 19th Annual Symposium, CPM 2008, Pisa, Italy, June 18-20, 2008, Proceedings*, volume 5029 of *Lecture Notes in Computer Science*, pages 31–43. Springer, 2008. 49, 50, 51, 52

[10] N. Betzler, R. van Bevern, M. R. Fellows, C. Komusiewicz, and R. Niedermeier. Parameterized Algorithmics for Finding Connected Motifs in Biological Networks. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 8(5):1296–1308, 2011. 52, 54

[11] A. Björklund. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. 30

[12] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. 30, 33

[13] A. Björklund, P. Kaski, L. Kowalik, and J. Lauri. Engineering Motif Search for Large Graphs. In U. Brandes and D. Eppstein, editors, *Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments, ALENEX 2015, San Diego, CA, USA, January 5, 2015*, pages 104–118. SIAM, 2015. 55

[14] A. Björklund, P. Kaski, and L. Kowalik. Constrained Multilinear Detection and Generalized Graph Motifs. *Algorithmica*, pages 1–21, 2015. 42, 51

[15] G. Blin, F. Sikora, and S. Vialette. GraMoFoNe: a Cytoscape plugin for querying motifs without topology in Protein-Protein Interactions networks. In H. Al-Mubaid, editor, *2nd International Conference on Bioinformatics and Computational Biology (BICoB'10)*, pages 38–43. International Society for Computers and their Applications (ISCA), 2010. 54

[16] S. Böcker and K. Dührkop. Fragmentation trees reloaded. *J. Cheminformatics*, 8(1):5:1–5:26, 2016. 60, 106

[17] S. Böcker and Z. Lipták. A Fast and Simple Algorithm for the Money Changing Problem. *Algorithmica*, 48(4):413–432, 2007. 58

[18] S. Böcker and F. Rasche. Towards *de novo* identification of metabolites by analyzing tandem mass spectra. In *ECCB'08 Proceedings, Seventh European Conference on Computational Biology, 22-26 September 2008, Cagliari, Italy*, pages 49–55, 2008. 10, 11, 12, 58, 59, 63, 70, 80, 94, 106, 112

[19] H. Bodlaender. On Linear Time Minor Tests with Depth-First Search. *Journal of Algorithms*, 14(1):1 – 23, 1993. 33

[20] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. 35

[21] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization Lower Bounds by Cross-Composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. 35, 95, 104

[22] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. 35

[23] E. Bonnet and F. Sikora. The Graph Motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, pages –, 2016. 46, 48

[24] V. Bonnici, F. Busato, G. Micale, N. Bombieri, A. Pulvirenti, and R. Giugno. APPAGATO: an approximate parallel and stochastic graph querying tool for biological networks. 54

[25] S. Bruckner, F. Hüffner, R. M. Karp, R. Shamir, and R. Sharan. TORQUE: topology-free querying of protein interaction networks. *Nucleic Acids Research*, 37(Web-Server-Issue):106–108, 2009. 54

[26] S. Bruckner, F. Hüffner, R. M. Karp, R. Shamir, and R. Sharan. Topology-Free Querying of Protein Interaction Networks. *Journal of Computational Biology*, 17(3):237–252, 2010. 43, 49, 50, 52, 54

[27] F. J. Bruggeman and H. V. Westerhoff. The nature of systems biology. *TRENDS in Microbiology*, 15(1):45–50, 2007. 9

[28] J. Chen, S. Lu, S. Sze, and F. Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 298–307, 2007. 28, 33

[29] Y.-J. Chu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965. 84, 95

[30] S. A. Cook. The Complexity of Theorem-Proving Procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. 40

[31] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On Problems as Hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. 51

[32] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. *Discrete Applied Mathematics*, 160(15):2131–2141, 2012. 35, 46, 48, 51, 52

[33] D. P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Mathematics*, 30(3):289–293, 1980. 24

[34] T. Dandekar, S. Schuster, B. Snel, M. Huynen, and P. Bork. Pathway alignment: application to the comparative analysis of glycolytic enzymes. *Biochemical Journal*, 343(Pt 1):115, 1999. 40

[35] B. Das, M. K. Enduri, N. Misra, and I. V. Reddy. On Structural Parameterizations of Graph Motif and Chromatic Number. In D. R. Gaur and N. S. Narayanaswamy, editors, *Algorithms and Discrete Applied Mathematics - Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16-18, 2017, Proceedings*, volume 10156 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2017. 46

[36] L. F. De Figueiredo, S. Schuster, C. Kaleta, and D. A. Fell. Can sugars be produced from fatty acids? a test case for pathway analysis tools. *Bioinformatics*, 24(22):2615–2621, 2008. 39

[37] T. De Lange. Shelterin: the protein complex that shapes and safeguards human telomeres. *Genes & development*, 19(18):2100–2110, 2005. 9

[38] M. Deng, F. Sun, and T. Chen. Assessment of the reliability of protein-protein interactions and protein function prediction. In *Biocomputing 2003*, pages 140–151. World Scientific, 2002. 10, 42, 43

[39] R. Dondi, G. Fertin, and S. Vialette. Maximum Motif Problem in Vertex-Colored Graphs. In G. Kucherov and E. Ukkonen, editors, *Combinatorial Pattern Matching, 20th Annual Symposium, CPM 2009, Lille, France, June 22-24, 2009, Proceedings*, volume 5577 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2009. 63

[40] R. Dondi, G. Fertin, and S. Vialette. Complexity issues in vertex-colored graph pattern matching. *J. Discrete Algorithms*, 9(1):82–99, 2011. 47, 48, 49, 65, 66

[41] R. Dondi, G. Fertin, and S. Vialette. Finding approximate and constrained motifs in graphs. *Theor. Comput. Sci.*, 483:10–21, 2013. 42, 106

[42] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012. 11

[43] K. Dührkop, F. Hufsky, and S. Böcker. Molecular formula identification using isotope pattern analysis and calculation of fragmentation trees. *Mass Spectrometry*, 3(Special Issue 2):S0037–S0037, 2014. 60

[44] K. Dührkop, M. A. Lataretu, W. T. J. White, and S. Böcker. Heuristic algorithms for the Maximum Colorful Subtree problem. *CoRR*, abs/1801.07456, 2018. 59

[45] K. Dührkop, M. A. Lataretu, W. T. J. White, and S. Böcker. Heuristic Algorithms for the Maximum Colorful Subtree Problem. In L. Parida and E. Ukkonen, editors, *18th International Workshop on Algorithms in Bioinformatics, WABI 2018, August 20-22, 2018, Helsinki, Finland*, volume 113 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. 106

[46] K. Dührkop, H. Shen, M. Meusel, J. Rousu, and S. Böcker. Searching molecular structure databases with tandem mass spectra using CSI: FingerID. *Proceedings of the National Academy of Sciences*, 112(41):12580–12585, 2015. 60

[47] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965. 87

[48] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967. 84, 95

[49] L. Euler. Solutio Problematis ad geometriam situs pertinentis. 8:128–140, 01 1736. 13

[50] P. Fahad. *Dynamic Programming using Representative Families*. PhD thesis, Homi Bhabha National Institute, 2015. 32

[51] M. R. Fellows, G. Fertin, D. Hermelin, and S. Vialette. Upper and lower bounds for finding connected motifs in vertex-colored graphs. *J. Comput. Syst. Sci.*, 77(4):799–811, 2011. 11, 45, 46, 49, 51

[52] A. R. Fernie, R. N. Trethewey, A. J. Krotzky, and L. Willmitzer. Metabolite profiling: from diagnostics to systems biology. *Nature reviews molecular cell biology*, 5(9):763, 2004. 57, 58

[53] G. Fertin, J. Fradin, and G. Jean. Algorithmic Aspects of the Maximum Colorful Arborescence Problem. In T. V. Gopal, G. Jäger, and S. Steila, editors, *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 216–230, 2017. 57, 79

[54] G. Fertin, J. Fradin, and C. Komusiewicz. On the Maximum Colorful Arborescence Problem and Color Hierarchy Graph Structure. In G. Navarro, D. Sankoff, and B. Zhu, editors, *Annual Symposium on Combinatorial Pattern Matching, CPM 2018, July 2-4, 2018 - Qingdao, China*, volume 105 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. 79

[55] G. Fertin and C. Komusiewicz. Graph Motif Problems Parameterized by Dual. In *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM'16), Tel Aviv, June 2016*, 2016. 52, 53, 54, 88

[56] F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient Computation of Representative Sets with Applications in Parameterized and Exact Algorithms. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 142–151. SIAM, 2014. 32, 33

[57] C. V. Forst and K. Schulten. Phylogenetic analysis of metabolic pathways. *Journal of Molecular evolution*, 52(6):471–489, 2001. 40

[58] R. Ganian. Twin-Cover: Beyond Vertex Cover in Parameterized Algorithmics. In D. Marx and P. Rossmanith, editors, *Parameterized and Exact Computation - 6th International Symposium, IPEC 2011, Saarbrücken, Germany, September 6-8, 2011. Revised Selected Papers*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2011. 46

[59] R. Ganian. Improving Vertex Cover as a Graph Parameter. *Discrete Mathematics & Theoretical Computer Science*, 17(2):77–100, 2015. 46

[60] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*. W. H. Freeman, 1979. 24, 48

[61] L. Giot, J. S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y. Hao, C. Ooi, B. Godwin, E. Vitols, et al. A protein interaction map of Drosophila melanogaster. *science*, 2003. 42

[62] S. Guillemot and F. Sikora. Finding and counting vertex-colored subtrees. *Algorithmica*, 65(4):828–844, 2013. 51, 52

[63] J.-D. J. Han. Understanding biological functions through molecular networks. *Cell research*, 18(2):224, 2008. 9

[64] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized Algorithmics and Computational Experiments for Finding 2-Clubs. *J. Graph Algorithms Appl.*, 19(1):155–190, 2015. 52

[65] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962. 30

[66] F. Hufsky, K. Dührkop, F. Rasche, M. Chimani, and S. Böcker. Fast alignment of fragmentation trees. *Bioinformatics*, 28(12):265–273, 2012. 60

[67] R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. 22

[68] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41, 2001. 38

[69] R. M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. 21, 30, 95, 104

[70] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, et al. PubChem substance and compound databases. *Nucleic acids research*, 44(D1):D1202–D1213, 2015. 60

[71] G. W. Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):S59, 2009. 39

[72] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Divide-and-Color. In *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, pages 58–67, 2006. 28, 33

[73] I. Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2008. 29, 33

[74] I. Koutis. Constrained multilinear detection for faster functional motif discovery. *Inf. Process. Lett.*, 112(22):889–892, 2012. 51

[75] I. Koutis and R. Williams. Limits and Applications of Group Algebras for Parameterized Problems. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 653–664. Springer, 2009. 33

[76] I. Koutis and R. Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. 33

[77] O. Kuchaiev and N. Pržulj. Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics*, 27(10):1390–1396, 2011. 39

[78] V. Lacroix. *Identification de motifs dans les réseaux métaboliques. (Motif identification in metabolic networks)*. PhD thesis, Claude Bernard University Lyon 1, France, 2007. 39

[79] V. Lacroix, C. G. Fernandes, and M. Sagot. Motif Search in Graphs: Application to Metabolic Networks. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(4):360–368, 2006. 10, 11, 37, 40, 44, 48, 54, 115

[80] M. Lampis. A kernel of order $2\ k - c \log k$ for Vertex Cover. *Inf. Process. Lett.*, 111(23-24):1089–1091, 2011. 35

[81] R. L. Last, A. D. Jones, and Y. Shachar-Hill. Innovations: Towards the plant metabolome and beyond. *Nature Reviews Molecular Cell Biology*, 8(2):167, 2007. 57

[82] J. W.-H. Li and J. C. Vederas. Drug discovery and natural products: end of an era or an endless frontier? *Science*, 325(5937):161–165, 2009. 57

[83] S.-H. Li and X.-J. Li. Huntingtin–protein interactions and the pathogenesis of Huntington's disease. *TRENDS in Genetics*, 20(3):146–154, 2004. 38

[84] N. Malod-Dognin and N. Przulj. L-GRAAL: Lagrangian graphlet-based network aligner. *Bioinformatics*, 31(13):2182–2189, 2015. 40

[85] D. W. Matula and L. L. Beck. Smallest-Last Ordering and clustering and Graph Coloring Algorithms. *J. ACM*, 30(3):417–427, 1983. 107

[86] B. Monien. How to Find Long Paths Efficiently. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 239 – 254. North-Holland, 1985. 33

[87] D. L. Nelson, A. L. Lehninger, and M. M. Cox. *Lehninger principles of biochemistry*. Macmillan, 2008. 39

[88] S. Neumann and S. Böcker. Computational mass spectrometry for metabolomics: identification of metabolites and small molecules. *Analytical and bioanalytical chemistry*, 398(7-8):2779–2788, 2010. 57

[89] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 11, 24

[90] T. Nishioka, T. Kasama, T. Kinumi, H. Makabe, F. Matsuda, D. Miura, M. Miyashita, T. Nakamura, K. Tanaka, and A. Yamamoto. Winners of CASMI2013: automated tools and challenge data. *Mass Spectrometry*, 3(Special Issue 2):S0039–S0039, 2014. 60

[91] H. Oberacher, M. Pavlic, K. Libiseller, B. Schubert, M. Sulyok, R. Schuhmacher, E. Csaszar, and H. C. Köfeler. On the inter-instrument and inter-laboratory transferability of a tandem mass spectral reference library: 1. Results of an Austrian multicenter study. *Journal of mass spectrometry*, 44(4):485–493, 2009. 58

[92] 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(20):4021–4028, 2000. 40

[93] S. Panni and S. E. Rombo. Searching for repetitions in biological networks: methods, resources and tools. *Briefings in bioinformatics*, 16(1):118–136, 2013. 40

[94] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. 23

[95] A. G. Pedersen, P. Baldi, Y. Chauvin, and S. Brunak. The biology of eukaryotic promoter prediction—a review. *Computers & chemistry*, 23(3-4):191–207, 1999. 9

[96] R. Y. Pinter, O. Rokhlenko, E. Yeger-Lotem, and M. Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics*, 21(16):3401–3408, 2005. 40

[97] R. Y. Pinter, H. Shachnai, and M. Zehavi. Improved Parameterized Algorithms for Network Query Problems. *Parameterized And Exact Computation 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, page 294, 2014. 51

[98] R. Y. Pinter, H. Shachnai, and M. Zehavi. Partial Information Network Queries. *J. Discrete Algorithms*, 31:129–145, 2015. 50

[99] R. Y. Pinter, H. Shachnai, and M. Zehavi. Deterministic parameterized algorithms for the Graph Motif problem, journal = Discrete Applied Mathematics. 213:162–178, 2016. 51

[100] R. Y. Pinter and M. Zehavi. Algorithms for topology-free and alignment network queries. *J. Discrete Algorithms*, 27:29–53, 2014. 51

[101] F. Rasche, K. Scheubert, F. Hufsky, T. Zichner, M. Kai, A. Svatos, and S. Böcker. Identifying the unknowns by aligning fragmentation trees. *Analytical chemistry*, 84(7):3417–3426, 2012. 58, 60

[102] F. Rasche, A. Svatos, R. K. Maddula, C. Böttcher, and S. Böcker. Computing fragmentation trees from tandem mass spectrometry data. *Analytical Chemistry*, 83(4):1243–1251, 2011. 58, 60, 116

[103] I. Rauf, F. Rasche, F. Nicolas, and S. Böcker. Finding Maximum Colorful Subtrees in Practice. *Journal of Computational Biology*, 20(4):311–321, 2013. 63, 65

[104] R. Rizzi and F. Sikora. Some Results on More Flexible Versions of Graph Motif. *Theory Comput. Syst.*, 56(4):612–629, 2015. 42, 48, 67

[105] C. A. Ross and S. J. Tabrizi. Huntington's disease: from molecular pathogenesis to clinical treatment. *The Lancet Neurology*, 10(1):83–98, 2011. 38

[106] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173, 2005. 42

[107] D. P. Rubert, E. Araujo, and M. A. Stefanes. SIMBio: Searching and inferring colorful motifs in biological networks. In *Bioinformatics and Bioengineering (BIBE), 2015 IEEE 15th International Conference on*, pages 1–6. IEEE, 2015. 54

[108] B. M. Schmidt, D. M. Ribnicky, P. E. Lipsky, and I. Raskin. Revisiting the ancient concept of botanical therapeutics. *Nature chemical biology*, 3(7):360, 2007. 57

[109] B. Schwikowski, P. Uetz, and S. Fields. A network of protein–protein interactions in yeast. *Nature biotechnology*, 18(12):1257, 2000. 42

[110] E. L. Schymanski, C. Ruttkies, M. Krauss, C. Brouard, T. Kind, K. Dührkop, F. Allen, A. Vaniya, D. Verdegem, S. Böcker, et al. Critical assessment of small molecule identification 2016: automated methods. *Journal of cheminformatics*, 9(1):22, 2017. 60

[111] J. Scott, T. Ideker, R. M. Karp, and R. Sharan. Efficient Algorithms for Detecting Signaling Pathways in Protein Interaction Networks. *Journal of Computational Biology*, 13(2):133–144, 2006. 49, 80

[112] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003. 54

[113] R. Sharan, S. Suthram, R. M. 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 United States of America*, 102(6):1974–1979, 2005. 40

[114] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. H. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, 2005. 42

[115] R. Tautenhahn, K. Cho, W. Uritboonthai, Z. Zhu, G. J. Patti, and G. Siuzdak. An accelerated workflow for untargeted metabolomics using the METLIN database. *Nature biotechnology*, 30(9):826, 2012. 58

[116] P. J. Thornalley. The glyoxalase system: new developments towards functional characterization of a metabolic pathway fundamental to biological life. *Biochemical Journal*, 269(1):1, 1990. 10

[117] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399, 2002. 42, 43

[118] E. C. Webb et al. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* Number Ed. 6. Academic Press, 1992. 39, 40

[119] W. T. J. White, S. Beyer, K. Dührkop, M. Chimani, and S. Böcker. Speedy Colorful Subtrees. In D. Xu, D. Du, and D. Du, editors, *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, volume 9198 of *Lecture Notes in Computer Science*, pages 310–322. Springer, 2015. 8, 59, 106, 107, 109, 110, 112

[120] R. Williams. Finding paths of length k in $\mathcal{O}^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. 30, 33

[121] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011. 12

[122] D. S. Wishart, C. Knox, A. C. Guo, R. Eisner, N. Young, B. Gautam, D. D. Hau, N. Psychogios, E. Dong, S. Bouatra, R. Mandal, I. Sinelnikov, J. Xia, L. Jia, J. A. Cruz, E. Lim, C. A. Sobsey, S. Shrivastava, P. Huang, P. Liu, L. Fang, J. Peng, R. Fradette, D. Cheng, D. Tzur, M. Clements, A. Lewis, A. De Souza, A. Zuniga, M. Dawe, Y. Xiong, D. Clive, R. Greiner, A. Nazyrova, R. Shaykhutdinov, L. Li, H. J. Vogel, and I. Forsythe. HMDB: a knowledgebase for the human metabolome. *Nucleic Acids Research*, 37:D603–D610, 2009. 58

[123] D. Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3(1):103–128, 2007. 65, 69

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Julien FRADIN

Graphes complexes en biologie : problèmes, algorithmes et évaluations

Complex Graphs in Biology: problems, algorithms and evaluations

Résumé

Afin de mieux comprendre le fonctionnement d'un système biologique, il est nécessaire d'étudier les interactions entre les différentes entités qui le composent. Pour cela, on peut modéliser ces interactions biologiques sous la forme de graphes. Pour certains de ces graphes, les sommets sont colorés afin d'apporter une information supplémentaire sur l'entité qui leur est associée. Dans ce cadre, une problématique courante consiste à y rechercher un sous-graphe d'intérêt appelé motif. Dans la première partie de ce manuscrit, on présente un état de l'art d'un point de vue algorithmique sur le problème GRAPH MOTIF, qui consiste à rechercher des motifs dits fonctionnels dans ce type de graphes.

La modélisation de systèmes biologiques sous forme de graphes peut également être appliquée en spectrométrie de masse. Ainsi, on introduit le problème MAXIMUM COLORFUL ARBORESCENCE (MCA) dans le but de déterminer *de novo* la formule moléculaire de métabolites inconnus. Dans la deuxième partie de ce manuscrit, on réalise une étude algorithmique du problème MCA. Alors que MCA est algorithmiquement difficile à résoudre même dans des classes de graphes très contraintes, notre modélisation nous permet notamment d'obtenir de nouveaux algorithmes d'approximation dans ces mêmes classes, ainsi que de déterminer une nouvelle classe de graphes dans laquelle MCA se résout en temps polynomial. On montre également des résultats de complexité paramétrée pour ce problème, que l'on compare ensuite à ceux de la littérature sur des instances issues de données biologiques.

Mots clés

Bioinformatique, Graphes, Complexité paramétrée, Approximation, Algorithmes.

Abstract

In order to better understand how a biological system works, it is necessary to study the interactions between the different entities that compose it. To this aim, these biological interactions can be modelled in the form of graphs. In some of these graphs, the vertices are colored in order to provide additional information on the entity which is associated with them. In this context, a common problem consists in searching for a subgraph of interest, called a motif, in these graphs. In the first part of this manuscript, we present a state of the art from an algorithmic point of view of the GRAPH MOTIF problem, which consists in searching for so-called functional motifs in vertex-colored graphs.

The modeling of biological systems in graphs form can also be applied in mass spectrometry. Thus, we introduce the MAXIMUM COLORFUL ARBORESCENCE problem (MCA) in order to *de novo* determine the molecular formula of unknown metabolites. In the second part of this manuscript, we carry out an algorithmic study of the MCA problem. While MCA is algorithmically difficult to solve even in very constrained graph classes, our modeling allows us to obtain new approximation algorithms in these same classes, as well as to determine a new graph class in which MCA is solved in polynomial time. Parameterized complexity results for this problem are also shown, which are then compared to those in the literature on instances from biological data.

Key Words

Bioinformatics, Graphs, Parameterized complexity, Approximation, Algorithms.