UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Décomposition de graphes: longueur arborescente et jeux de poursuite

Thomas DISSAUX

Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S)
UMR7271 Université Côte d'Azur CNRS

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur

Dirigée par : Nicolas NISSE, Directeur de recherche, Université Côte d'Azur, CNRS, I3S, Sophia-Antipolis, France

Soutenue le : 25 september 2023

Devant le jury, composé de :

Christophe CRESPELLE (président du jury), Professeur, Université Côte d'Azur, I3S, CNRS, France

Guillaume DUCOFFE, Associate Professor, Faculty of Mathematics and Informatics, University of Bucharest, Roumanie

Cyril GAVOILLE, Professeur, LaBRI, Université de Bordeaux, France

Arnaud MALAPERT, Maître de conférences, Université Côte d'Azur, I3S, CNRS, France

Dimitrios M. THILIKOS, Directeur de recherche, LIRMM, Université de Montpellier, CNRS, Montpellier, France

DÉCOMPOSITION DE GRAPHES: LONGUEUR ARBORESCENTE ET JEUX DE POURSUITE

Graph decompositions: Treelength and pursuit-evasion games

Thomas DISSAUX

Jury :

Rapporteurs

Cyril GAVOILLE, Professeur, LaBRI, Université de Bordeaux, France
Dimitrios M. THILIKOS, Directeur de recherche, LIRMM, Université de Montpellier, CNRS, Montpellier, France

Examinateurs

Christophe CRESPELLE (président du jury), Professeur, Université Côte d'Azur, I3S, CNRS, France
Guillaume DUCOFFE, Associate Professor, Faculty of Mathematics and Informatics, University of Bucharest, Roumanie
Arnaud MALAPERT, Maître de conférences, Université Côte d'Azur, I3S, CNRS, France

Directeur de thèse

Nicolas NISSE, Directeur de recherche, Université Côte d'Azur, CNRS, I3S, Sophia-Antipolis, France

---

Université Côte d'Azur

Thomas DISSAUX

Décomposition de graphes: longueur arborescente et jeux de poursuite

viii+134 p.

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10.

Impression : output.tex – 29/9/2023 – 9:24

Révision pour la classe : these-ISSS.cls,v 2.10 2020/06/24 14:16:37 mpelleau

Décomposition de graphes: longueur arborescente et jeux de poursuite

Résumé

Les décompositions de graphes sont un outil permettant de représenter un graphe en plusieurs parties, appelées sacs, et structurées comme un arbre ou un chemin suivant si ce sont des décompositions arborescentes ou linéaires. Ces décompositions permettent de résoudre certains problèmes NP-difficiles en temps linéaire, si la taille maximum des sacs (i.e. la « largeur/width ») est bornée. Cela a motivé les travaux de ces 30 dernières années sur la largeur arborescente d'un graphe G (la plus petite largeur des décompositions arborescente de G). Cependant, il reste encore beaucoup de questions ouvertes comme la complexité du calcul de la largeur arborescente des graphes planaires. Pour pouvoir répondre à cette question, il peut être intéressant d'étudier une autre mesure des décompositions, la longueur. Cette mesure correspond au diamètre maximum des sacs d'une décomposition, et il a été prouvé qu'il existe une relation entre la longueur arborescente et la largeur arborescente dans les graphes planaires.

Nous nous intéressons donc dans le chapitre 2 à la longueur arborescente de classes de graphes planaires simples, comme les graphes série-parallèles. Nous explicitons une liste infinie de sous-graphes isométriques interdits pour les graphes série-parallèles de longueur arborescente 2. Grâce à cette liste, il est alors possible en temps polynomial de tester si un graphe série-parallèle a une longueur arborescente au plus 2 et, dans le cas d'une réponse positive, de calculer une décomposition arborescente optimale.

Nous nous intéressons aussi au cas de la longueur linéaire dans le chapitre 3. Nous nous focalisons sur les classes des arbres et des cycles pour lesquelles nous caractérisons la longueur linéaire. Nous nous intéressons aussi à la longueur linéaire des graphes planaires extérieurs et concevons un algorithme d'approximation de facteur additif 1.

Finalement, dans le chapitre 4, nous nous intéressons à une variante algorithmique des décompositions linéaires des graphes, par le biais d'un jeu de poursuite-évasion, le jeu *des Chasseurs et du Lapin*. Dans ce jeu, un groupe de chasseurs traque un lapin invisible qui est forcé de bouger à chaque étape sur une position voisine. Nous nous intéressons au nombre minimum $h(G)$ de chasseurs qui peuvent attraper le lapin quoi qu'il fasse sur un graphe G. Ce jeu a notamment été étudié dans le cas des graphes bipartis (grilles, arbres, hypercubes...) mais reste ouvert dans beaucoup de classes de graphes et notamment dans les arbres. Une notion très utile pour le calcul de stratégie dans les jeux de poursuite-évasion est la notion de monotonie. Nous définissons une variante monotone du jeu des chasseurs et du lapin, nous permettant entre autres, de prouver que, dans cette variante, le nombre minimum $mh(G)$ de chasseurs diffère d'au plus un de la largeur linéaire du graphe G. Ce résultat a d'importantes conséquences, comme le fait que le calcul de $mh(G)$ est NP-difficile. Nous caractérisons aussi $mh(G)$ pour plusieurs classes de graphes, comme les graphes scindés, les graphes d'intervalles, les arbres et les cographes. Nous étudions la différence entre mh et h dans ces classes de graphes et, en particulier, nous montrons qu'il peut exister une différence arbitrairement grande entre mh et h dans les arbres.

Mots-clés : Décomposition arborescente, Décomposition linéaire, Longueur arborescente, Longueur linéaire, Graphe planaire, Jeu des Chasseurs et du Lapin.

Graph decompositions: Treelength and pursuit-evasion games

Abstract

Graph decompositions are a powerful tool aimed to divide a graph into several parts, called bags, connected in a tree-like or a path-like fashion, depending on whether we consider a tree-decomposition or a path-decomposition. They can be used to solve some NP-hard problems in linear time, as long as the maximum size of the bags (i.e. the width) is bounded. This has motivated the study of treewidth (minimum width out of all tree-decompositions of a graph) over the past 30 years. Nevertheless, there are still a lot of open questions, such as the computational complexity of treewidth in planar graphs. To answer this question, it could be interesting to study the length of these decompositions. This measure corresponds to the maximum diameter among the bags of a decomposition, and it was shown that there is a relationship between treelength and treewidth in planar graphs.

In chapter 2, we study the treelength of several simple classes of planar graphs, such as series-parallel graphs. We give an infinite list of forbidden isometric subgraphs for series-parallel graphs of treelength 2. This list is then used to decide in polynomial time if a series-parallel graph has treelength at most 2 and, in case of a positive answer, to output a decomposition of length at most 2.

In chapter 3, we focus on the pathlength, and we give a characterization for trees and cycles. We also study the pathlength of outerplanar graphs for which we design an approximation algorithm with additive ratio 1.

Finally, in chapter 4, we study the *Hunters and Rabbit* game, an algorithmic variant of path-decomposition defined as a pursuit-evasion game. In this game, a group of hunters hunts an invisible rabbit which is forced to move to a neighboring position at every step. We are interested in the minimum number $h(G)$ of hunters needed to catch the rabbit, independently of its moves along a graph G. This game has already been studied for some bipartite graphs (meshes, trees, hypercubes...). However, it remains open questions in a lot of graph classes, notably in trees. A very useful notion for the computation of strategies in pursuit-evasion games is the notion of monotonicity. We define a monotone variant of the Hunters and Rabbit game, for which we prove that the minimum number $mh(G)$ of hunters differs by at most one from the pathwidth. This has important consequences, such as the fact that computing $mh(G)$ is NP-hard. We also characterize $mh(G)$ for several graph classes, such as split graphs, interval graphs, trees, and cographs. For these classes, we study the difference between mh and h and, in particular, we show that it is arbitrarily large in trees.

Keywords: Tree-decomposition, Path-decomposition, Treelength, Pathlength, Planar graph, Hunters and Rabbit game.

Table des matières

CHAPITRE 1

Introduction

Dans cette thèse, nous nous intéressons aux problèmes de la *théorie des graphes*. Plus précisément, nous nous intéressons aux problèmes de représentation de graphes en décompositions (arborescentes ou linéaires). L'idée est de décomposer les sommets et arêtes qui composent ces graphes en plusieurs parties, appelées *sacs*. Ces différents sacs sont interconnectés de telle sorte qu'ils induisent un arbre (décomposition arborescente) ou un chemin (décomposition linéaire). Bien sûr, décomposer un graphe en un seul sac est très facile. Par conséquent, nous nous intéressons à décomposer un graphe tout en minimisant une certaine mesure des décompositions. La mesure la plus connue des décompositions est la *largeur* et correspond à la taille maximum des sacs d'une décomposition. La largeur arborescente $tw(G)$ d'un graphe G est la plus petite largeur des décompositions arborescentes de G.

La largeur arborescente des graphes a été extrêmement étudiée, car il existe des applications très intéressantes pour ce paramètre. Les décompositions arborescentes de largeur bornée ont été historiquement très utiles pour la théorie des mineurs de graphes où elles ont été introduites par Robertson et Seymour (Robertson & Seymour, 1986a) il y a plus de 35 ans. Elles sont aussi très utiles dans la conception d'algorithmes paramétrés. En effet, beaucoup de problèmes NP-difficiles sont faciles à résoudre sur les graphes de largeur arborescente bornée, et il existe des graphes représentant des interactions réelles ayant une faible largeur arborescente.

Même si la largeur arborescente a été beaucoup étudiée, il reste des questions ouvertes concernant son calcul. Nous pouvons prendre pour exemple la complexité du calcul de la largeur arborescente dans les graphes planaires qui est toujours ouverte depuis plus de 35 ans. Pour répondre à cette question, de nouveaux outils devraient être nécessaires. La *longueur arborescente*, une autre mesure liée aux décompositions arborescentes, correspondant au diamètre maximum des sacs, pourrait bien en être la clef. Cette mesure paraît plus complexe à calculer de façon exacte, mais plus simple à approcher. En effet, décider si un graphe G quelconque a une longueur arborescente au plus 2 est un problème NP-complet alors que décider si un graphe G quelconque a une largeur arborescente au plus k (fixé) peut être fait en temps polynomial. À l'inverse, Le meilleur algorithme d'approximation pour la longueur arborescente a un ratio de 3 alors que le meilleur algorithme d'approximation pour la largeur arborescente a un ratio de $\sqrt{\log tw(G)}$. De plus, il existe un rapport entre ces deux mesures dans les graphes planaires. Étudier la complexité du calcul de la longueur arborescente dans les graphes planaires pourrait donc nous aider pour étudier la complexité du calcul de la largeur arborescente des graphes planaires.

Au-delà de nous aider pour des questions ouvertes concernant la largeur arborescente, la longueur arborescente a aussi des applications intéressantes, comme des algorithmes paramétrés par la longueur arborescente. La longueur arborescente a été beaucoup moins étudiée et donc peu de résultats sont connus sur le calcul de ce paramètre dans des classes de graphes simples. Le but de cette thèse est d'étoffer les connaissances sur la longueur arborescente. Ces connaissances pour-

raient servir, pour les applications, pour résoudre la complexité du calcul de ce paramètre dans les graphes planaires, ou encore étudier plus spécifiquement la différence entre la largeur arborescente et la longueur arborescente dans des classes de graphes spécifiques, comme des sous-classes des graphes planaires.

Pour comprendre un peu mieux la relation entre la largeur arborescente et la longueur arborescente, nous commençons cette étude par des sous-classes des graphes planaires. Par définition, la largeur et la longueur arborescente des arbres sont égales à un. De plus, lors de l'introduction de la longueur arborescente, il a été prouvé que la longueur arborescente d'un cycle contenant n sommets est de $\lceil \frac{n}{3} \rceil$. Le résultat précédent a été immédiatement généralisé pour les graphes planaires extérieurs ayant une longueur arborescente $\lceil \frac{is(G)}{3} \rceil$ où $is(G)$ est la longueur (taille) d'un plus long cycle isométrique. La prochaine classe de graphes simples à étudier était donc la classe des graphes série-parallèles. Nous avons obtenu des résultats à ce sujet, mais le calcul de la longueur arborescente des graphes série-parallèles semble beaucoup plus compliqué que le calcul de la longueur arborescente des graphes planaires extérieurs. Il reste notamment des questions ouvertes concernant les graphes série-parallèles de longueur arborescente $k \geq 3$.

Cette difficulté nous a poussé à considérer l'étude d'une variante de la longueur arborescente, la *longueur linéaire*. La longueur arborescente d'un graphe étant bornée par la longueur linéaire, les applications dans les graphes de longueur arborescente bornée, sont aussi des applications dans les graphes de longueur linéaire bornée. De plus, la longueur linéaire possède aussi des applications distinctes des applications de la longueur arborescente. La longueur linéaire est encore moins étudiée que la longueur arborescente. Concrètement, le seul résultat connu pour la longueur linéaire est sa complexité, NP-Difficile. À notre connaissance, aucune étude de classes de graphes n'a été faite pour la longueur linéaire (à part les arbres de longueur linéaire 1). Nous nous sommes donc intéressés au calcul de la longueur linéaire de sous-classes de graphes simples des graphes planaires. Cette fois-ci, nous commençons par caractériser la longueur linéaire des arbres et des cycles. Cela nous a donc amené à étudier les graphes planaires extérieurs. Nous avons obtenu un algorithme d'approximation de facteur additif $+1$. Nous n'avons malheureusement pas encore réussi à généraliser certains de nos lemmes, ce qui nous permettrait de prouver qu'il existe un algorithme exact polynomial (si cela est possible).

En parallèle, nous nous sommes intéressés à un jeu assez atypique des jeux de recherche. Ces jeux de recherche sont liés aux décompositions arborescentes et aux décompositions linéaires, car certaines variantes du jeu DES GENDARMES ET DU VOLEUR sont équivalentes à la largeur arborescente et à la largeur linéaire. Le jeu (qui nous intéresse) est le jeu DES CHASSEURS ET DU LAPIN. Ce jeu a principalement été étudié dans les arbres et les graphes bipartis. Deux questions importante sont restées ouvertes, "est-il possible de concevoir un algorithme polynomial calculant le nombre de chasseurs nécessaires pour trouver un lapin sur un arbre T" et "est-il possible de concevoir un algorithme polynomial calculant le nombre de chasseurs nécessaires pour trouver un lapin sur un graphe G". Dans le but de concevoir un algorithme polynomial pour les arbres, nous avons défini la notion de monotonie. Cela nous a notamment permis de concevoir un algorithme polynomial pour la variante monotone dans les arbres, et de prouver que la variante monotone pouvait être arbitrairement éloignée du jeu original DES CHASSEURS ET DU LAPIN.

La suite de cette introduction est dédiée aux définitions, applications, propriétés des décompositions (arborescentes et linéaires), et, plus généralement, à l'état de l'art concernant ces différentes notions.

1.1 Largeur arborescente et largeur linéaire

Avant de définir la largeur arborescente et la largeur linéaire d'un graphe, définissons formellement les décompositions arborescentes et les décompositions linéaires de G. Notons que *les décompositions arborescentes* des graphes ont été initialement introduites par Halin (Halin, 1976), puis redécouvertes dans le cadre de la théorie des mineurs de graphes par Robertson et Seymour (Robertson & Seymour, 1986a).

Définition 1.1.1 (Décomposition arborescente). Une *décomposition arborescente* d'un graphe $G = (V, E)$ est une paire $D = (T, \mathcal{X}) = \{X_t \mid t \in V(T)\}$ telle que T est un arbre, et $\mathcal{X}$ est un ensemble de sous-ensembles de sommets (appelés *sacs*) de G, correspondants aux nœuds de T tels que :

1. $\bigcup_{t \in V(T)} X_t = V(G)$, c'est-à-dire, chaque sommet de G est contenu dans au moins un sac de la décomposition D de G ;
2. Pour chaque arête $\{u, v\} \in E(G)$, il existe $t \in V(T)$ tel que $u, v \in X_t$, c'est-à-dire, les extrémités de chaque arête de G sont contenues toutes les deux dans au moins un même sac ;
3. Pour chaque sommet $v \in V(G)$, l'ensemble des sacs le contenant, $\{t \in V(T) \mid v \in X_t\}$, induit un sous-arbre de T.

Définition 1.1.2 (Décomposition linéaire). Une *décomposition linéaire* d'un graphe G est une décomposition arborescente $D = (T, \mathcal{X})$ de G telle que T est un chemin. Nous notons cette décomposition linéaire par $D = (X_1, \ldots, X_p)$ où $\mathcal{X} = \{X_1, \ldots, X_p\}$ et $\{X_i, X_{i+1}\} \in E(T)$ pour tout $1 \leq i < p$.

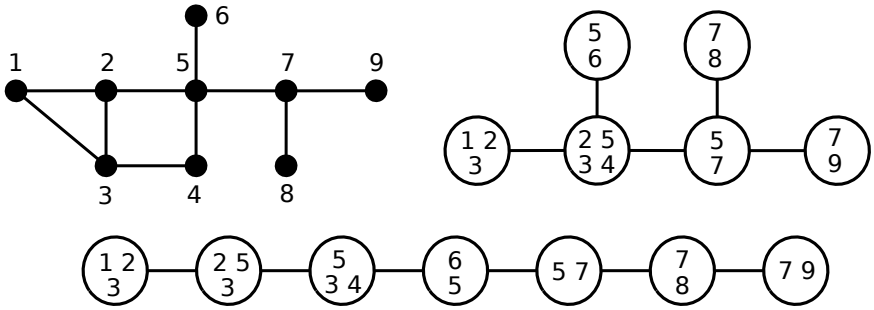


Figure 1.1 – Exemple de décomposition arborescente $(T, \mathcal{X})$ (à droite) de largeur 3 ($tw(G) = 2$) de G (à gauche) et d'une décomposition linéaire L (en bas) de largeur 2 ($pw(G) = 2$) de G.

La largeur d'une décomposition arborescente $D = (T, \mathcal{X})$ d'un graphe G est égale à la taille maximale des sacs de D (moins un), c'est-à-dire, $w(D) = \max_{X \in \mathcal{X}} |X| - 1$. La largeur (width) d'une décomposition linéaire $D = (X_1, \ldots, X_p)$ est donc $w(D) = \max_{1 \leq i \leq p} |X_i| - 1$. Rappelons que décomposer un graphe en une seule partie est très simple. Le but est donc de trouver des décompositions minimisant leurs largeurs. La largeur arborescente $tw(G)$ (treewidth) d'un graphe G est la plus petite largeur des décompositions arborescentes de G. La largeur linéaire $pw(G)$ (pathwidth) d'un graphe G est la plus petite largeur des décompositions linéaires de G. Nous dirons qu'une décomposition arborescente (resp. linéaire) de G est optimale s'il n'existe pas de

décompositions arborescentes (resp. linéaire) de G de plus petite largeur, c'est-à-dire, $w(D) = tw(G)$ (resp. $w(D) = pw(G)$). Notons que $tw(G) \leq pw(G)$ par définition.

Intuitivement, la largeur arborescente mesure à quel point un graphe G est proche d'être un arbre. Les arbres T sont les graphes connexes de largeur arborescente 1. La largeur linéaire mesure à quel point un graphe G est proche d'être un chemin. Les chemins P sont les graphes connexes de largeur linéaire 1.

1.1.1 Applications

La largeur arborescente et la largeur linéaire d'un graphe G ont deux types d'applications principales. Le premier type est leur utilité dans la théorie des mineurs de graphes. Le deuxième type d'applications est la conception d'algorithmes pour des problèmes NP-difficiles paramétrés par la largeur arborescente ou linéaire.

1.1.1.1 Théorie des Mineurs de graphes

Les décompositions arborescentes ont joué un rôle central dans la théorie des mineurs de graphes. La théorie des mineurs de graphes a eu beaucoup d'impacts sur la théorie des graphes, et même sur d'autres théories. Un *mineur* d'un graphe G est n'importe quel graphe H pouvant être obtenu à partir de G en supprimant des sommets et/ou des arêtes et/ou en contractant des arêtes, c'est-à-dire, H est obtenu à partir d'un sous-graphe H' de G en contractant des arêtes de H'. Un graphe G est dit *sans mineur* H, où H est un graphe, si G ne contient pas H comme mineur. Une famille (une classe) $\mathcal{F}$ de graphes est dite *close par mineur* si tout mineur d'un graphe de cette famille F appartient aussi à cette famille. Une des utilités des mineurs est la possibilité de caractériser des classes de graphes sous forme de mineurs *interdits*, c'est-à-dire, sous forme d'ensembles de graphes ne pouvant être contenus comme mineur par cette classe de graphes. Par exemple, il est facile de se convaincre qu'un graphe connexe est un arbre si et seulement si il ne contient pas K_3, la clique avec 3 sommets, comme mineur. Un graphe est *planaire extérieure* si et seulement si il ne contient pas K_4 et $K_{2,3}$ comme mineur (Chartrand & Harary, 1967). Ensuite, un graphe est série-parallèle (cf. définition 1.3.1) si et seulement si il ne contient pas K_4 comme mineur (Duffin, 1965 ; Dirac, 1952). La caractérisation des graphes planaires est sans doute la plus connue de ces caractérisations. Un graphe est planaire si et seulement si il ne contient pas $K_{3,3}$ et K_5 comme mineur (Wagner, 1937 ; Kuratowski, 1930). Plus généralement, nous obtenons que :

Théorème 1.1.1. *(Robertson & Seymour, 2004) Toute classe de graphes close par mineur peut être caractérisée par une liste finie de mineurs interdits.*

Intuitivement, si une classe est close par mineur, alors nous pouvons simplement lister l'ensemble des graphes minimaux n'étant pas contenus dans cette classe. Une des grandes conséquences de ces caractérisations est que :

Théorème 1.1.2. *Toute propriété de graphe close par mineur peut être testée en temps polynomial.*

En fait, ce théorème vient du Théorème 1.1.1, mais aussi du fait qu'il est possible, et ce, en temps polynomial ($\mathcal{O}(n^3)$), de tester si un graphe G contient un graphe H (fixé) comme mineur (Robertson & Seymour, 1995). Malheureusement, bien que cet algorithme testant si un graphe G contient un graphe H comme mineur soit polynomial, il existe des constantes larges cachées dans le $\mathcal{O}$. De plus, certaines de ces constantes dépendent du graphe H fixé. Rappelons de

plus que, bien que la liste des mineurs interdits (minimaux par mineur) soit finie pour une classe de graphes close par mineur, elle n'est pas forcément connue, potentiellement longue, et/ou compliquée à construire, etc. Par exemple, il a été prouvé que la liste des mineurs interdits pour les graphes de linear-width au plus 2 contenait 57 graphes (Thilikos, 2000), ou même 103 pour les graphes plongeables sur le plan projectif (Archdeacon, 1981). Il existe même des propriétés closes par mineur dont on ne connaît pas l'ensemble de mineurs interdits. Pour conclure, la théorie de mineurs est très importante puisqu'elle démontre, en théorie, l'existence d'algorithmes polynomiaux permettant de décider si un graphe G contient comme mineur un graphe H fixé, et donc de tester une propriété de graphes close par mineur, même si en pratique, ces algorithmes n'existent pas forcément ou sont inutilisables.

Notons que ce ne sont pas les seuls résultats obtenus par Robertson et Seymour. À travers leurs nombreux papiers sur les mineurs de graphes, beaucoup de résultats ont été prouvés. Certains seront précisés plus tard. Même si nous ne l'avons pas précisé, les décompositions arborescentes ont été un outil très utile, voir indispensable, pour le développement de cette théorie.

1.1.1.2 Paradigme de la programmation dynamique et algorithmes paramétrés

La largeur arborescente. L'intuition derrière les algorithmes paramétrés par la largeur arborescente ou linéaire est la stratégie de "diviser pour mieux régner". Plus précisément, l'idée générale pour résoudre un problème NP-difficile en temps polynomial sur une certaine instance G de largeur arborescente bornée est de calculer progressivement des solutions partielles de ce problème, c'est-à-dire, calculer la solution pour un sous-graphe induit par un sac B ainsi que tous ses descendants dans la décomposition à partir des solutions des sous-graphes induits par chacun de ses enfants et de leurs descendants respectifs. Des solutions seront calculées pour plusieurs sous-graphes de G, en commençant par les "feuilles" de la décomposition et en finissant par la "racine" de la décomposition, menant finalement au calcul d'une solution pour l'instance G. Il faut comprendre ici que la complexité du calcul d'un problème NP-difficile dépend principalement de la largeur de la décomposition arborescente utilisée. Il est donc très important de pouvoir calculer des décompositions arborescentes de petites largeurs.

Cette stratégie n'est malheureusement pas utilisable pour tous les problèmes NP-difficiles. Il est difficile de dire pour quels problèmes cette stratégie est utilisable. Mais grâce à Courcelle, nous savons au moins que tous les problèmes pouvant être exprimés en *logique monadique de second ordre (MSO_2)* admettent des algorithmes linéaires dans les graphes de largeur arborescente bornée.

Théorème 1.1.3. *(Courcelle & Mosbah, 1993) Toute propriété de graphes définissable en logique MSO_2 peut être décidée en temps linéaire dans les graphes de largeur arborescente bornée.*

Par exemple, les problèmes de la clique maximum, de coloration minimum, d'ensemble maximum de sommets indépendants, de triangulations minimum, d'ensemble de sommets dominants minimum, de cycle hamiltonien et d'arbre de Steiner minimum sont tous des problèmes définissables en MSO_2 alors qu'ils sont NP-difficiles en général (Arnborg & Proskurowski, 1989).

Notons que bien que le Théorème 1.1.3 certifie l'existence d'algorithmes linéaires pour des problèmes sur un graphe G, les algorithmes n'ont pas forcément une complexité en temps intéressante en pratique. En effet, les algorithmes sont construits par un automate. Même si la complexité en temps des algorithmes est toujours linéaire par rapport à la taille de l'entrée (nombre de sommets du graphe) mais est exponentielle par rapport à la largeur arborescente du graphe,

cette complexité en temps dépend de la formule ϕ en MSO_2 définissant la propriété cherchée, c'est-à-dire, la complexité en temps des algorithmes est $f(tw(G), \phi)|V(G)|$. Cette généralisation, pour n'importe quelle formule ϕ, a forcément un coup ($f(tw(G), \phi) = 2^{2^{\cdot^{\cdot^{\cdot^{tw(G)+\ell}}}}}$ où ℓ est la taille de ϕ), justifiant l'intérêt de définir plus spécifiquement des algorithmes de programmation dynamique pour chacun de ces problèmes. Par exemple, l'Algorithme linéaire, dédié à calculer un arbre Steiner minimum dans les graphes de largeur arborescente bornée, a une meilleure complexité que l'algorithme du méta-théorème de Courcelle applicable à n'importe quelle formule en MSO_2 (Chimani, Mutzel, & Zey, 2012). Dans (Bodlaender, Cygan, Kratsch, & Nederlof, 2015), des algorithmes sont conçus pour le problème d'arbre de Steiner, du voyageur de commerce (traveling salesman problem), des chemins de taille k, d'ensemble de sommets à retirer pour rendre le graphe acyclique (feedback vertex set), du nombre de cycles hamiltoniens... Les décompositions arborescentes jouent aussi un rôle crucial dans la conception d'algorithmes sous-exponentiels dans le contexte de la bi-dimensionnalité (Demaine, Fomin, Hajiaghayi, & Thilikos, 2004 ; Demaine & Hajiaghayi, 2008).

Notons que tous les algorithmes précédemment cités sont résolubles à paramètre fixé ("Fixed Parameter Tractable", FPT), c'est-à-dire, leur complexité en temps est $f(tw(G))|V(G)|^{\mathcal{O}(1)}$. Pour plus de précision sur les techniques utilisées pour la conception d'algorithme FPT, voir (Cygan et al., 2015a ; Diestel, 2012). Notons qu'il est aussi possible d'obtenir des algorithmes paramétrés par la largeur arborescente (non FPT) pour beaucoup de problèmes. Par exemple, décider si un graphe H est isomorphique à un sous-graphe (induit) de G est un problème NP-complet dans les graphes de largeur arborescente borné, mais admet des algorithmes polynomiaux sous certaines restrictions sur H comme le fait qu'il soit de degré borné (Matoušek & Thomas, 1992).

Comme précisé précédemment, ces algorithmes sont polynomiaux (voir linéaires) dans les graphes de largeur arborescente bornée. Notons que beaucoup de graphes ont une largeur arborescente bornée. Par exemple, les classes de graphes classiques pour lesquelles la largeur arborescente est bornée sont les arbres, les graphes série-parallèles, les graphes planaires extérieurs, les graphes Halin, les graphes cordaux (si la taille de leur clique maximum est fixée), les graphes k-planaires extérieurs (si k est fixé), *etc* (Bodlaender, 1988). Notons aussi qu'il existe des graphes de largeur arborescente bornée provenant d'applications réelles. Par exemple, il a été montré que 9712 composés chimiques de la base de données LIGAND ont une largeur arborescente au plus 4 dont seulement un ayant une largeur arborescente 4 (Yamaguchi & Aoki-Kinoshita, 2014). Un autre exemple est le graphe de flot de contrôle d'un programme Java ou C qui a une largeur arborescente respectivement au plus 6 (Gustedt, Mæhle, & Telle, 2002 ; Thorup, 1998).

Ce ne sont pas pour autant les seuls problèmes pour lesquels cette stratégie peut être utilisée. Pour définir certains de ces autres problèmes, le *genre* d'un graphe G est le plus petit entier k tel que G est plongeable dans la surface orientable de genre k. Il a été prouvé, comme pour le théorème 1.1.3, que les problèmes définissables en logique $CMSO_2$ (permettant aussi de compter le nombre d'éléments d'un ensemble) respectant une certaine propriété de couvrement sont résolubles en temps polynomial dans les graphes de genre borné (Bodlaender et al., 2016).

Largeur linéaire. Rappelons que $pw(G) \geq tw(G)$ pour tout graphe G. Par conséquent, tout graphe de largeur linéaire bornée est un graphe de largeur arborescente bornée. Cela implique que toutes les applications pour la largeur arborescente sont aussi des applications pour la largeur linéaire. Par exemple, le théorème de Courcelle 1.1.3 s'applique aux graphes de largeur linéaire bornée.

D'autres applications existent. Par exemple, il en existe pour la conception de *VLSI layout* puisque le problème "Gate matrix layout" est équivalent au problème calculant la largeur linéaire (Fellows & Langston, 1994). Il en existe aussi en traitement automatique des langues naturelles (natural languages processing). Cela est dû à l'équivalence entre la largeur linéaire et l'étroitesse ("narrowness") d'un graphe (Kornai & Tuza, 1992). La largeur linéaire a aussi des applications en biologie numérique. Par exemple, trouver un super-graphe H proprement coloré d'un graphe G k-colorable tel que H est un graphe d'intervalles est un problème utilisé pour modéliser les chevauchements de clones d'ADN ("perfect phylogeny" problem). De plus, ce problème est relié au calcul de la largeur linéaire où les sommets de chaque paire de sommets d'un sac ont des couleurs distinctes (Dinneen, 1995).

1.1.2 Complexité

Largeur arborescente. Le calcul de la largeur arborescente d'un graphe G quelconque est NP-complet (Arnborg, Corneil, & Proskurowski, 1987). Rappelons aussi que la complexité du calcul de la largeur arborescente d'un graphe planaire G est ouverte.

De plus, le calcul de la largeur arborescente est *FPT* par rapport à la taille de la solution, c'est-à-dire, décider si $tw(G) \leq k$ peut être résolu en temps $2^{\mathcal{O}(k^3)}n$ (Bodlaender & Kloks, 1996). Malheureusement, l'algorithme ne peut pas être utilisé en pratique, car la complexité de cet algorithme est super-exponentielle en k et à cause de la grande constante cachée dans le "grand O". Récemment, ce résultat a été amélioré pour obtenir une complexité en temps $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$ (Korhonen & Lokshtanov, 2022).

Il est aussi possible de concevoir des algorithmes d'approximation (polynomiaux) pour la largeur arborescente. Il en existe beaucoup (Amir, 2001 ; Feige, Hajiaghayi, & Lee, 2008 ; Bodlaender, Gilbert, Hafsteinsson, & Kloks, 1995). Ces algorithmes reposent principalement sur des algorithmes d'approximation pour le calcul d'ensembles de sommets séparateurs parallèles, et le meilleur connu (celui avec le plus petit ratio) certifie de calculer une décomposition arborescente de largeur au plus $O(\sqrt{\log tw(G)}tw(G))$ (Feige et al., 2008). Notons d'ailleurs qu'il est impossible (sauf si $P = NP$) de concevoir un algorithme d'approximation de ratio constant pour la largeur arborescente (Bodlaender et al., 1995) pour des graphes quelconques. Par contre, en contraignant le problème aux graphes de nombre astéroïde $an(G)$ borné, il existe alors un algorithme d'approximation de ratio $8an(G)$ (Bouchitté, Kratsch, Müller, & Todinca, 2004). Des algorithmes d'approximation pour la largeur arborescente ont aussi été conçus dans les graphes planaires, comme les algorithmes de (Kammer & Tholey, 2016) reposant sur une représentation d'un graphe comme une carte topographique, où l'altitude d'une position est associé à la distance minimale entre v, le sommet à cette position, et un sommet appartenant à la face extérieur. Notons que ces algorithmes sont conçus pour des graphes planaires pondérés et ont un rapport de 15 et 12 par rapport à la solution optimale. De plus, ces algorithmes ont été améliorés pour les graphes planaires non pondérés pour obtenir un algorithme d'approximation calculant une décomposition de largeur au plus $9tw(G) + 9$ (Kammer, 2012). Il existe aussi des algorithmes d'approximation de ratio 1.5 pour les graphes planaires (Seymour & Thomas, 1994). Notons que cet algorithme ne construit pas de décomposition arborescente de largeur bornée par $1.5tw(G)$. Plus tard, cet algorithme sera adapté (Demaine, Hajiaghayi, Nishimura, Ragde, & Thilikos, 2004) pour construire une décomposition arborescente de largeur au plus $1.5tw(G)+1$ pour les graphes sans apex (apex-free graphs). Quand k est fixé, il est aussi possible pour un graphe G de décider si $tw(G) > k$ ou de calculer une décomposition arborescente de G avec une largeur d'au plus $2k + 1$ en temps

$2^{\mathcal{O}(k)}n$ (Korhonen, 2021). Ce résultat a aussi été amélioré pour un rapport $(1+\epsilon)k$ en temps $k^{\mathcal{O}(k/\epsilon)}n^4$ pour $\epsilon \in]0,1[$ (Korhonen & Lokshtanov, 2022).

1.1.2.1 Largeur linéaire

Le calcul de la largeur linéaire d'un graphe G quelconque est NP-complet (Arnborg et al., 1987). De plus, la complexité du calcul de la largeur linéaire d'un graphe planaire G est NP-complet (Monien & Sudborough, 1988).

Le calcul de la largeur linéaire est aussi *FPT* par rapport à la taille de la solution, c'est-à-dire, décider si $pw(G) \leq k$ peut être résolu en temps $2^{\mathcal{O}(k^3)}n$ (Bodlaender & Kloks, 1996). Malheureusement, comme dans le cas de la largeur arborescente, l'algorithme ne peut pas être utilisé en pratique. Récemment, ce résultat a été amélioré pour obtenir une complexité en temps $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$ (Korhonen & Lokshtanov, 2022).

Il existe aussi un algorithme polynomial permettant de calculer une décomposition linéaire de G optimale (et donc $pw(G)$) pour tout graphe G à partir d'une décomposition arborescente de largeur t (fixé) (Bodlaender & Kloks, 1996). Ce résultat sera utilisé pour obtenir certains résultats concernant la largeur linéaire dans la section 1.3.

Concernant les algorithmes d'approximation, le meilleur algorithme connu a un ratio de $\mathcal{O}(\sqrt{\log(tw(G))} \times \log(tw(G)))$ (Groenland, Joret, Nadara, & Walczak, 2023).

Comme nous l'avons vu, la largeur arborescente et la largeur linéaire sont des paramètres très utiles. Cependant, ces paramètres ne sont pas toujours simples à calculer en pratique. Cela a motivé la définition de nombreuses autres mesures pour les décompositions arborescentes et linéaires.

1.2 Longueur arborescente et longueur linéaire

Un autre paramètre pour mesurer les décompositions arborescentes et linéaires a été défini. Il s'agit de la *longueur* (length) d'une décomposition arborescente (ou linéaire) $D = (T, \mathcal{X})$, correspondant au *diamètre* maximal de ses sacs dans G, c'est-à-dire, $\ell(D) = \max_{X \in \mathcal{X}} \max_{u,v \in X} dist_G(u,v)$ où $dist_G(u,v)$ est le nombre d'arêtes d'un plus court chemin entre u et v dans G. Comme pour la largeur, la *longueur arborescente* (treelength) d'un graphe G, notée $t\ell(G)$, est la longueur minimale de ses décompositions arborescentes (Dourisboure & Gavoille, 2007). De la même façon, la longueur linéaire (pathlength) d'un graphe G, notée $p\ell(G)$, est la longueur minimale de ses décompositions linéaires (Dourisboure & Gavoille, 2007).

Nous qualifierons aussi une décomposition arborescente (resp. linéaire) d'un graphe G d'*optimale* si sa longueur est égale à $t\ell(G)$ (resp. $p\ell(G)$). Notons que par définition, $p\ell(G) \geq t\ell(G)$.

1.2.1 Applications

La longueur arborescente a également des intérêts algorithmiques. Par exemple, le *Problème du Voyageur de Commerce* admet un FPTAS (schéma d'approximation entièrement en temps polynomial) dans les graphes de longueur arborescente bornée (Krauthgamer & Lee, 2006) ; le calcul de la *dimension métrique* est FPT par la longueur arborescente et le degré maximum (Belmonte, Fomin, Golovach, & Ramanujan, 2017) ; des schémas de routage compacts efficaces (efficient compact routing schemes) et des sous-arbres couvrants additifs (sparse additive spanners) de

congestion faible peuvent être construits dans la classe des graphes de longueur arborescente bornée (Dourisboure & Gavoille, 2007 ; Dragan & Köhler, 2014 ; Kosowski, Li, Nisse, & Suchan, 2015), *etc*. Notons que les intérêts algorithmiques de la largeur et la longueur des décompositions s'intersectent, mais différent.

Rappelons que, puisque $p\ell(G) \geq t\ell(G)$, tout graphe de longueur linéaire bornée a une longueur arborescente bornée. Par conséquent, les applications de la longueur arborescente sont aussi des applications de la longueur linéaire.

La longueur linéaire, quant à elle, permet de capturer des propriétés métriques des graphes. Par exemple, le problème de plongement avec faible distorsion de graphes dans des espaces métriques peut être approché lorsque la longueur linéaire est bornée (Dragan, Köhler, & Leitert, 2017) et ce problème a des applications en vision par ordinateur (Tenenbaum, de Silva, & Langford, 2000), en chimie et en biologie numérique (Indyk, 2001), pour les protocoles distribués (Herlihy, Kuhn, Tirthapura, & Wattenhofer, 2006), *etc*.

1.2.2 Complexité

Le calcul de la longueur arborescente (linéaire) d'un graphe G à n sommets est NP-complet (Lokshtanov, 2010 ; Ducoffe, Legay, & Nisse, 2020).

Les auteurs de (Lokshtanov, 2010 ; Ducoffe et al., 2020) ont même prouvé que décider si la longueur arborescente (resp. linéaire) d'un graphe est au plus 2 est NP-complet. Cela implique notamment que la longueur arborescente et la longueur linéaire ne sont pas FPT par rapport à la taille de la solution, ce qui est différent de la largeur arborescente et de la largeur linéaire qui, elles, le sont (Bodlaender & Kloks, 1996). Intuitivement, la largeur arborescente (linéaire) semble donc plus simple à calculer de façon exacte que la longueur arborescente.

Approcher la longueur arborescente (linéaire) d'un graphe G est beaucoup plus simple que d'approcher la largeur arborescente (linéaire). En effet, il existe plusieurs algorithmes d'approximation pour la longueur arborescente certifiant de calculer une décomposition arborescente de longueur au plus $3t\ell(G) + \mathcal{O}(1)$: l'algorithme $LexM$ calculant une décomposition à partir d'une triangulation du graphe G ; l'algorithme $BFS - Layering$ calculant une décomposition arborescente à partir d'un "layering-tree" (arbre enraciné en un nœud R correspondant à un sommet r de G tel que tout nœud X de l'arbre correspond a un sous-ensemble de sommets v de G tel que $dist_G(r, v) = dist_T(R, X)$) ; et l'algorithme $disk - tree$ présenté dans (Dourisboure & Gavoille, 2007). Notons que ce dernier algorithme repose sur le fait que pour une décomposition arborescente de longueur k, chaque sac est inclus dans le disque de rayon k par rapport à n'importe quel sommet du sac. L'idée est donc de construire séquentiellement une décomposition en ajoutant les disques de rayon k centré en un sommet dont les voisins ne sont pas tous dans la décomposition. Bien sûr, ces disques et leurs sommets doivent respecter quelques propriétés pour être sûr de construire une décomposition arborescente. De ce fait, cet algorithme ne réussit pas forcément à calculer une décomposition arborescente, mais si c'est le cas, alors la décomposition a une largeur au plus 2 fois la longueur arborescente du graphe. Malheureusement, il est seulement connu que l'algorithme termine si $k \geq 3t\ell(G) - 2$. Notons que l'algorithme $disk - tree$ est conjecturé avoir un ratio de 2 (c'est-à-dire, il est conjecturé que l'algorithme termine dans tous les cas, pour $k \geq tl(G)$).

Pour la longueur linéaire, il existe un algorithme d'approximation de ratio 2 (Dragan et al., 2017). Rappelons que les meilleurs algorithmes connus pour la largeur arborescente et la largeur linéaire ont respectivement un ratio de $\mathcal{O}(\sqrt{\log(tw(G))})$ et de $\mathcal{O}(\sqrt{\log(tw(G))} \times \log(n))$ (Feige

et al., 2008). Intuitivement, la longueur arborescente et la longueur linéaire sont donc plus simples à approcher que la largeur arborescente et la largeur linéaire.

D'ailleurs, il a été prouvé qu'il n'existe pas d'algorithme d'approximation de ratio $c < \frac{3}{2}$ pour la longueur arborescente et pour la longueur linéaire (sauf si P=NP) (Lokshtanov, 2010 ; Ducoffe et al., 2020).

1.3 Largeurs et longueurs (arborescentes et linéaires) de classes de graphes planaires

Avant de faire un résumé des résultats sur des classes de graphes restreintes, rappelons quelques définitions et propriétés bien connues des décompositions arborescentes et des décompositions linéaires.

Un graphe G est dit *connexe* si pour toute paire de sommets a, b dans $V(G)$ il existe un chemin entre a et b dans G. Un sous-ensemble S de $V(G)$ est un séparateur de G si et seulement si $G - S$ est non connexe. Un séparateur S de G est une clique-séparatrice si et seulement si S induit une clique (un graphe dans lequel toute paire de sommets a, b de G sont adjacents). Un graphe G est dit *premier* si et seulement si il n'existe aucune clique séparatrice dans G. Un *sous-graphe* $H = (V(H) \subseteq V, E(H) \subseteq E \cap (V(H) \times V(H)))$ de G est *isométrique* si $dist_H(u, v) = dist_G(u, v)$ pour tout $u, v \in V(H)$, c'est-à-dire, si H *préserve* les distances de G. Rappelons que $is(G)$ correspond à la longueur d'un plus long cycle isométrique dans G.

Rappelons à présent des propriétés bien connues des décompositions. Commençons par citer les différentes façons de formuler la troisième propriété des décompositions (cf. définition 1.1.1).

Remarque 1.3.1 – [Propriété 3^{bis} des décompositions arborescentes] Soit G un graphe quelconque et, soit $D = (T, \mathcal{X})$ une décomposition arborescente de G. Pour toute paire de sacs X et X' dans $\mathcal{X}$, l'intersection de X et X' est contenu dans tous les sacs sur le chemin entre X et X' dans T.

Remarque 1.3.2 – [Propriété 3^{bis} des décompositions linéaires] Soit G un graphe quelconque et soit $L = (X_1, \ldots, X_p)$ une décomposition linéaire de G. Pour tout $1 \leq i \leq z \leq j \leq p$, $X_i \cap X_j \subseteq X_z$.

Il existe encore une troisième façon de formuler la troisième propriété des décompositions :

Remarque 1.3.3 – [Propriété 3^{ter} des décompositions arborescentes] Soit G un graphe quelconque et, soit $D = (T, \mathcal{X})$ une décomposition arborescente de G. Pour toute arête $e = \{X, X'\} \in E(T)$ de T, soient T_1 et T_2 les deux composantes connexes de $T - e$. Alors, $X \cap X'$ est un séparateur de $\bigcup_{q \in V(T_1)} X_q \setminus (X \cap X')$ et $\bigcup_{q \in V(T_2)} X_q \setminus (X \cap X')$ s'ils sont non vides.

Remarque 1.3.4 – [Propriété 3^{ter} des décompositions linéaires] Soit G un graphe quelconque et soit $L = (X_1, \ldots, X_p)$ une décomposition linéaire de G. Pour tout $1 \leq i < p$, $X_i \cap X_{i+1}$ est un séparateur de $\bigcup_{1 \leq j \leq i} X_j \setminus (X_i \cap X_{i+1})$ et $\bigcup_{i < j \leq p} X_j \setminus (X_i \cap X_{i+1})$ s'ils sont non vides.

Des Remarques 1.3.3 et 1.3.4, nous obtenons les corollaires suivant sur les cliques K contenu dans G :

Corollaire 1.3.5. *Soit G un graphe et soit $D = (T, \mathcal{X})$ une décomposition arborescente de G. Pour toute clique K de G, il existe un sac X de D tel que $K \subseteq X$. De plus, la largeur arborescente $tw(K_n)$ de la clique contenant n sommets est égale à $n - 1$, et la longueur arborescente $t\ell(K_n)$ de la clique contenant n sommets est égale à 1.*

Corollaire 1.3.6. *Soit G un graphe et soit $L = (X_1, \dots, X_p)$ une décomposition linéaire de G. Pour toute clique K de G, il existe $1 \leq i \leq p$ tel que $K \subseteq X_i$. De plus, la largeur linéaire $pw(K_n)$ de la clique contenant n sommets est égale à $n-1$, et la longueur linéaire $p\ell(K_n)$ de la clique contenant n sommets est égale à 1.*

Nous pouvons donc nous restreindre au calcul de la largeur (longueur) arborescente des graphes premiers. En effet, à partir des décompositions arborescentes optimales pour deux composantes premières, nous pouvons obtenir une décomposition pour ces deux composantes en liant (en ajoutant une arête entre) deux sacs, contenant la clique-séparatrice K (séparant les deux composantes premières), de chaque décomposition.

Lemme 1.3.7. *Soit G un graphe et soit S une clique-séparatrice de G séparant $C \subseteq V(G)$ de $C' \subseteq V(G)$ telle que (C, S, C') est une partition de $V(G)$. Alors, $tw(G) = \max(tw(C \cup S), tw(C' \cup S))$ et $t\ell(G) = \max(t\ell(C \cup S), t\ell(C' \cup S))$.*

Malheureusement, ce n'est pas le cas pour la largeur linéaire et la longueur linéaire. En revanche, le calcul de la largeur arborescente (resp. de la largeur linéaire, de la longueur arborescente ou de la longueur linéaire) d'un graphe G peut se limiter au calcul de la largeur arborescente (resp. de la largeur linéaire, de la longueur arborescente ou de la longueur linéaire) de ses composantes connexes :

Lemme 1.3.8. *Soit G un graphe non connexe et soient $C_1 \dots, C_q$ ses composantes connexes. Alors, $tw(G) = \max(tw(C_1), \dots tw(C_q))$, $t\ell(G) = \max(t\ell(C_1), \dots t\ell(C_q))$, $pw(G) = \max(pw(C_1), \dots pw(C_q))$ et $p\ell(G) = \max(p\ell(C_1), \dots p\ell(C_q))$.*

Grâce aux lemmes précédents, nous ne considérerons, dans le chapitre 2 dédié à l'étude de la longueur arborescente, que les graphes connexes et premiers (nous le rappellerons au début du chapitre). De plus, nous ne considérerons, dans le chapitre 3 dédié à l'étude de la longueur linéaire, que les graphes connexes (nous le rappellerons au début du chapitre).

Notons aussi que la largeur arborescente (resp. linéaire) d'un graphe G est close par mineur (Bodlaender, 1998), c'est-à-dire, il n'existe pas de mineur H de G tel que $tw(H) > tw(G)$ (resp. $pw(H) > pw(G)$). Ce n'est malheureusement pas le cas pour la longueur arborescente (resp. linéaire) puisqu'il existe des graphes G et des mineurs H de G tels que $t\ell(H) > t\ell(G)$ (resp. $p\ell(H) > p\ell(G)$). En particulier, il existe même des graphes G et des sous graphes H de G tels que $t\ell(H) > t\ell(G)$ (resp. $p\ell(H) > p\ell(G)$) et donc la longueur arborescente n'est même pas close par sous-graphe. L'exemple le plus simple est un graphe H de longueur arborescente $k \geq 3$ et le graphe G obtenu à partir de H en rajoutant un sommet adjacent à tous les sommets de H. Clairement, $t\ell(G) \leq 2$ (puisque pour tout graphe G, $t\ell(G) \leq diam(G)$), alors que $t\ell(H) = k \geq 3$.

Par contre, la longueur arborescente est close par sous-graphe isométrique :

Lemme 1.3.9. *(Dourisboure & Gavoille, 2007) Pour tout graphe G et pour tout sous-graphe isométrique H de G, $t\ell(H) \leq t\ell(G)$ et $p\ell(H) \leq p\ell(G)$.*

Démonstration. Soit D une décomposition arborescente (resp. $L = (X_1, \dots, X_p)$ une décomposition linéaire) de G de longueur $t\ell(G)$ (resp. $p\ell(G)$). En supprimant les sommets de $V(G) \setminus V(H)$ des sacs de D (resp. de L), nous obtenons une décomposition arborescente D' (resp. linéaire L') de H. Puisque H est isométrique à G, nous obtenons que $\ell(D') \leq \ell(D)$ (resp. $\ell(L') \leq \ell(L)$) et donc $t\ell(H) \leq t\ell(G)$ (resp. $p\ell(H) \leq p\ell(G)$). □

Pour la suite, notons qu'à notre connaissance, il n'existe aucun résultat sur la longueur linéaire de classes de graphes simples.

1.3.1 Chemins et arbres

Par définition de tous ces paramètres arborescents ou linéaires, il devrait être clair que les chemins P ont une largeur linéaire et une longueur linéaire de 1 (pareil pour les paramètres arborescents). De plus, les graphes de largeur linéaire 1 sont exactement les chemins.

Les arbres T ont une largeur arborescente et une longueur arborescente de 1. De plus, les graphes de largeur arborescente 1 sont exactement les arbres.

Puisque les cliques de taille n ont une longueur arborescente et une longueur linéaire de 1 (cf. Lemmes 1.3.6 et 1.3.5), nous obtenons en fait que les graphes d'intervalles sont les graphes de longueur linéaire 1 et que les graphes cordaux sont les graphes de longueur arborescente 1.

Cependant, le calcul de la largeur linéaire et de la longueur linéaire des arbres est complexe. (Parsons, 1978) caractérise les arbres de largeur linéaire k. Pour décrire cette caractérisation, définissons les branches d'un sommet $v \in V(T)$ dans T comme les composantes connexes de $T \setminus v$. Rappelons les cas de bases, un graphe ne contenant qu'un sommet a une largeur linéaire égale à 0 et qu'un graphe connexe contenant deux sommets a une largeur linéaire égale à 1.

Théorème 1.3.10. *(Parsons, 1978)[Le théorème de Parsons] Soit $T = (V, E)$ un arbre. Pour tout $k \in \mathbb{N}^*$, $pw(T) \geq k + 1$ si et seulement si il existe un sommet v tel qu'il existe trois branches B_1, B_2 et B_3 de v tel que $pw(B_i) \geq k$ pour tout $i = 1, 2, 3$.*

Cette caractérisation permet de calculer la largeur linéaire des arbres T en les enracinant en un sommet v et en utilisant un algorithme de programmation dynamique calculant la largeur linéaire (et quelques autres informations) pour les sous-arbres de T induit par un sommet x et ses descendants (les sommets dont le chemin entre eux et v passe par x) dans T à partir de la largeur linéaire (et quelques autres informations) des sous-arbres de T induits respectivement par un enfant de x et de ses descendants.

Théorème 1.3.11. *(Ellis, Sudborough, & Turner, 1994) Il existe un algorithme calculant une décomposition linéaire optimale pour tout arbre T en temps $\mathcal{O}(|V(T)| \log(|V(T)|))$.*

Notons que le plus petit arbre ayant une largeur linéaire au moins 2 est le graphe *araignée* avec 3 pattes de longueur au moins 2, c'est-à-dire, le graphe obtenu à partir d'un graphe étoile à 3 feuilles, où chaque arête est subdivisée (l'opération qui ajoute un sommet au milieu de l'arête et donc, la divisant en deux) une fois.

1.3.2 Cycles et graphes planaires extérieurs

La prochaine classe qui va nous intéresser est celle des cycles.

Lemme 1.3.12. *(Dourisboure & Gavoille, 2007) Soit $C = (v_1, \dots v_n)$ un cycle où $n > 2$. Alors $tw(C) = pw(C) = 2$ and $t\ell(C) = \lceil \frac{n}{3} \rceil$.*

Avec le Lemme 1.3.7, le résultat suivant peut être obtenu pour les paramètres arborescents des graphes planaires extérieurs. Rappelons que les graphes *planaires extérieurs* sont des graphes pouvant être plongés dans le plan, de telle sorte que chaque sommet de G appartient à la face extérieure de ce plongement planaire.

Théorème 1.3.13. *(Dourisboure & Gavoille, 2007) Soit G un graphe planaire extérieur tel que $is(G) = k$. Alors, $tw(G) = 2$ et $t\ell(G) = \lceil \frac{k}{3} \rceil$. De plus, une décomposition arborescente optimale (pour la largeur ou la longueur) peut être calculée en temps linéaire.*

Malheureusement, puisqu'il n'existe pas d'adaptation du Lemme 1.3.7 pour la largeur linéaire, nous n'avons pas de formule explicitant la largeur linéaire des graphes planaires extérieurs. Par contre, il est quand même possible de calculer la largeur linéaire de tout graphe G planaire extérieur, mais la complexité du meilleur algorithme connu est élevée :

Lemme 1.3.14. *(Bodlaender & Kloks, 1996) Il existe un algorithme calculant la largeur linéaire des graphes planaires extérieurs en temps $O(n^{11})$.*

D'un autre côté, il existe un algorithme d'approximation de ratio 2 pour le calcul de la largeur linéaire des graphes planaires extérieurs de complexité en temps $O(n \log(n))$ (Bodlaender & Fomin, 2002 ; Coudert, Huc, & Sereni, 2007). Cet algorithme est basé sur la relation entre la largeur linéaire d'un graphe planaire extérieur et la largeur linéaire de son *dual faible*. Notons que dans le cas particulier des graphes planaires extérieurs, leur dual faible est une forêt. L'algorithme d'approximation utilise donc l'algorithme du Théorème 1.3.11 pour calculer une décomposition linéaire optimale de son dual faible et en déduit une décomposition linéaire pour le graphe planaire extérieur.

1.3.3 Graphes série-parallèles

La prochaine classe (toujours planaire) qui nous intéresse est celle des graphes série-parallèles.

Définition 1.3.1 (Graphe série-parallèle). Un graphe (s,t)-*série-parallèle* est un graphe (avec deux sommets distingués s et t) défini récursivement comme suit. Une arête $\{s,t\}$ est un graphe (s,t)-série-parallèle. De plus, étant donnés un graphe (s_1,t_1)-série-parallèle G_1 et un graphe (s_2,t_2)-série-parallèle G_2, un graphe (s,t)-série-parallèle G peut être obtenu à partir de l'union disjointe de G_1 et G_2 en identifiant l'un ou l'autre :

composition en série : t_1 et s_2 (auquel cas $s = s_1$, et $t = t_2$) soit,

composition parallèle : s_1 et s_2 d'une part, et t_1 et t_2 d'autre part (auquel cas $s = s_1 = s_2$, et $t = t_1 = t_2$).

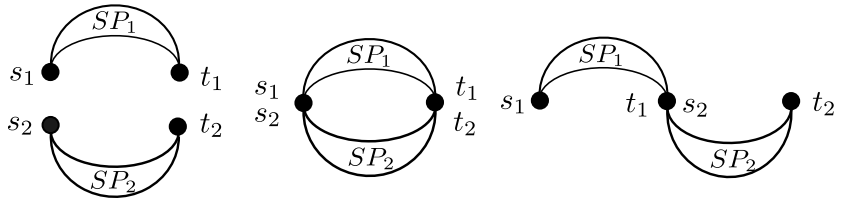


Figure 1.2 – La composition en série (droite) et la composition en parallèle (milieu) de deux graphes série-parallèles SP_1 et SP_2 (gauche).

Un graphe $G = (V,E)$ est *série-parallèle* s'il existe deux sommets $s,t \in V$ tels que G est un graphe (s,t)-série-parallèle. Il est bien connu qu'un graphe ne contient par K_4 comme mineur, ou de manière équivalente a une largeur arborescente d'au plus 2, si et seulement si ses composantes

2-connexes sont série-parallèles (Robertson & Seymour, 1991). Notons que les graphes planaires extérieurs sont précisément les graphes ne contenant pas K_4 et $K_{2,3}$ comme mineur et donc les graphes planaires extérieurs 2-connexes sont inclus dans la classe des graphes série-parallèles.

Théorème 1.3.15. *(Robertson & Seymour, 1991) Soit G un graphe série-parallèle. Alors, $tw(G) \leq 2$.*

Il est aussi possible de calculer en temps polynomial la largeur linéaire des graphes séries-parallèles, puisque leur largeur arborescente est bornée (Bodlaender & Kloks, 1996).

Malheureusement, jusqu'à cette thèse, rien n'est connu pour la longueur arborescente et la longueur linéaire des graphes série-parallèles.

1.3.4 Graphes planaires

Une autre classe très intéressante pour les décompositions arborescentes et linéaires est celle des grilles.

Théorème 1.3.16. *(Dourisboure & Gavoille, 2007 ; Bodlaender, 1998) Soit G une grille de taille $(n \times m)$ pour $n, m > 1$. Si $m \neq n$ ou s'ils sont pairs, alors $t\ell(G) = \min(n, m)$, sinon $t\ell(G) = n - 1$. Dans tous les cas, $tw(G) = pw(G) = \min(n, m)$.*

L'intérêt de cette classe de graphes est d'autant plus important grâce au résultat suivant. Intuitivement, pour tout graphe G ayant une largeur arborescente supérieure à $f(k)$ pour une certaine fonction f, il existe une grille de taille $(k \times k)$ contenu comme mineur dans G. Ce théorème a d'abord été prouvé par Robertson et Seymour (Robertson & Seymour, 1986b), puis la fonction f a été séquentiellement améliorée. Concrètement, il a été d'abord prouvé que $f(k) = 20^{2r^5}$ dans les graphes quelconque et $f(k) = 6k-5$ pour les graphes planaires (Robertson, Seymour, & Thomas, 1994). Ensuite, (Kawarabayashi & Kobayashi, 2012) ont prouvé que $f(k) = 2^{\mathcal{O}(k^2 \log(k))}$. (Leaf & Seymour, 2015) ont prouvé que $f(k) = k^{8k^2}$. (Chekuri & Chuzhoy, 2016) donne la première borne polynomiale, c'est-à-dire, $f(k) = k^{98} polylog(k)$. (Chuzhoy, 2016) améliore et simplifie les preuves, lui permettant d'obtenir que $f(k) = k^{19} polylog(k)$. Le résultat le plus récent à ce sujet est :

Théorème 1.3.17. *(Chuzhoy & Tan, 2021) Soit G un graphe quelconque, soit $k \in \mathcal{N}$ et soient $c_1, c_2 > 0$. Si $tw(G) \geq c_1 k^9 log^{c_2}(k)$, alors G contient une grille de taille $(k \times k)$ comme mineur.*

Ce théorème de mineur de grille est très utile, car il a notamment été utilisé pour répondre à un des problèmes fondamentaux qui ont mené à l'élaboration de la théorie des mineurs, c'est-à-dire, le problème de calculer k chemins disjoints, ou pour les applications algorithmiques liées à la bi-dimensionnalité. De plus, la borne supérieure décrite dans le prochain paragraphe sur la largeur arborescente (par rapport à la longueur arborescente) a été obtenue grâce à une variante du théorème de mineur de grille (Demaine, Hajiaghayi, & Thilikos, 2006).

Concernant les graphes planaires, il existe une relation entre la largeur arborescente et la longueur arborescente. Ce n'est pas très intuitif, puisque pour les classes de graphes (très simples) des cycles et des cliques, ces deux paramètres diffèrent de façon extrême. Rappelons que les cliques de taille n ont largeur arborescente $n-1$ et longueur arborescente 1 et, par ailleurs, que les cycles ont largeur arborescente 2 et longueur arborescente $\lceil \frac{n}{3} \rceil$. En fait, ces classes de graphes sont en quelque sorte les cas extrêmes. Plus les cliques maximum d'un graphe sont grandes, plus ce graphe a une largeur arborescente importante (ce n'est pas le cas pour la longueur). Au contraire, plus les

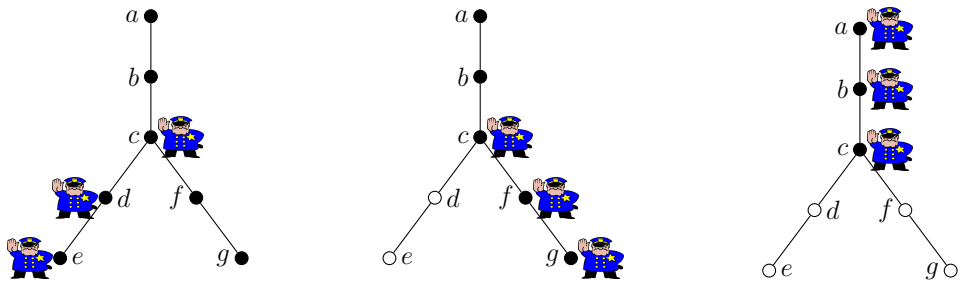
cycles isométriques maximum d'un graphe sont longs, plus ce graphe a une longueur arborescente importante (ce n'est pas le cas pour la largeur arborescente). De ce fait, si nous interdisons les grosses cliques et les gros cycles isométriques, la différence entre ces deux paramètres s'amoindrit. Plus précisément, rappelons qu'une décomposition arborescente de largeur au plus k, peut être vu comme un ensemble de séparateurs parallèles contenant au plus k sommets (Parra & Scheffler, 1997). Dans (Coudert, Ducoffe, & Nisse, 2016), il a été prouvé que toutes paires de sommets d'un de ces séparateurs sont à distance au plus $\lfloor \frac{is(G)}{2} \rfloor$, impliquant que le diamètre de chacun de ces séparateurs S est au plus $\lfloor \frac{is(G)}{2} \rfloor \cdot (|S| - 1)$ et donc $t\ell(G) \leq \lfloor \frac{is(G)}{2} \rfloor tw(G)$. Cela implique que tout algorithme d'approximation pour la largeur arborescente donne une borne non triviale pour la longueur arborescente. De plus, ils ont aussi prouvé que pour les graphes de genre g borné tel que $tw(G) > 48g^2 + 48g$, $tw(G) \leq 72\sqrt{2}(g+1)^{\frac{3}{2}} \cdot tl(G) + \mathcal{O}(g^2)$. Cela implique un algorithme d'approximation de ratio $\mathcal{O}(q \cdot is(G) \cdot (g+1)^{\frac{3}{2}})$ pour la largeur arborescente à partir d'un algorithme d'approximation de ratio q pour la longueur arborescente. Notons qu'un tel algorithme ne retourne pas de décomposition arborescente de largeur bornée.

Un autre résultat intéressant est que dans tout graphe G planaire, et pour toute décomposition arborescente de longueur ℓ (calculable en temps polynomial grâce aux algorithmes d'approximation de ratio 3), il est possible de calculer en temps polynomial une décomposition arborescente de largeur au plus 9ℓ pour G (Dieng & Gavoille, 2009). En résumé, le calcul de décompositions arborescentes de "petite" longueur implique de "bons" algorithmes d'approximation de la largeur arborescente des graphes planaires. Plus précisément, puisque $t\ell(G) \leq \lfloor \frac{is(G)}{2} \rfloor tw(G)$ pour tout graphe G planaire, nous obtenons qu'il existe un algorithme d'approximation de ratio $\frac{27 \cdot is(G)}{2}$ pour la largeur arborescente dans les graphes planaires.

1.4 Le Jeu des Gendarmes et du Voleur

Il existe une façon totalement différente de présenter les décompositions de graphes. Les décompositions peuvent être représentées comme une stratégie à suivre par des agents pour gagner à un certain jeu. Plus précisément, un tel jeu est le jeu DES GENDARMES ET DU VOLEUR. Ce jeu est très connu et tombe dans la catégorie des jeux de *recherche* (graphs searching en anglais) dans les graphes (Breisch, 1967 ; Parsons, 1978). Plus précisément, ces jeux se jouent à deux joueurs, l'un incarnant une équipe de chercheurs, les *agents*, alors que l'autre joueur incarne l'entité cherchée par cette équipe, le *fugitif*. Le but pour les agents est bien entendu de capturer le fugitif, alors que le but du fugitif est d'échapper indéfiniment aux agents. Toutes les entités occupent (selon les jeux) soit des sommets, soit des arêtes du graphe. Le déroulement du jeu se déroule, soit sous forme de tours, en alternant les mouvements du fugitif et ceux d'un ou plusieurs agents, ou de façon simultanée. Par exemple, dans sa forme classique, toutes les entités occupent des sommets, et si un agent se trouve sur le même sommet que le fugitif (et que ce dernier ne peut pas s'enfuir), alors l'agent capture le fugitif, ce qui met fin au jeu. Bien sûr, ces jeux ont des contraintes supplémentaires selon la variante du jeu en question. Dans certains cas, le fugitif peut se déplacer très rapidement, être invisible, être forcé de bouger, et/ou les mouvements des agents sont restreints, et/ou il existe des conditions particulières pour que les agents puissent capturer le fugitif, etc.

Notons que ce genre de jeu peut se jouer avec autant de chercheurs que l'on souhaite. Il est donc possible d'utiliser autant de chercheur que de sommets du graphe, ce qui permet à coup sûr de capturer le fugitif. De ce fait, nous essayons pour chacun de ces jeux de minimiser le nombre de chercheurs utilisés. De plus, la stratégie du fugitif peut énormément influer sur la stratégie des



(a) $S_1 = \{c, d, e\}$ (b) $S_2 = \{c, f, g\}$ (c) $S_3 = \{a, b, c\}$

Figure 1.3 – Exemple d'un arbre et d'une stratégie de capture $\mathcal{S} = (S_1, S_2, S_3)$ gagnante optimale utilisant 3 gendarmes pour la variante d'échappement-sommet. Chaque sous-figure représente la situation pendant le tour correspondant. Pendant un tour i, un sommet est noir si le fugitif peut être sur ce sommet et blanc si le fugitif ne peut pas être sur ce sommet.

chercheurs. De ce fait, nous considérons souvent que ce jeu ne se joue qu'à un joueur, le joueur "chercheur" qui doit trouver une stratégie utilisant un certain nombre d'agents et certifiant que le fugitif perd contre cette stratégie, qu'importe la stratégie qu'il utilise. Intuitivement, cela revient à considérer que l'ensemble des sommets et/ou des arêtes du graphe sont "contaminés" au début du jeu, et que le groupe de chercheurs doit en fait "nettoyer" le graphe. Ces notions de contamination et de nettoyage seront très utilisées dans la suite.

Ces types de jeux ont souvent des applications concrètes dans la vie réelle, comme rechercher une personne perdue dans une grotte ou tout simplement attraper un voleur, etc. De plus, il existe des applications en intelligence artificielle (Isaza, Lu, Bulitko, & Greiner, 2008), dans la planification de mouvement de robots (Chung, Hollinger, & Isler, 2011), dans les problèmes de satisfaction de contraintes, dans la théorie des bases de données (Gottlob, Leone, & Scarcello, 2003), et dans le calcul distribué (Nisse, 2019). De plus, ces jeux sont souvent reliés à des propriétés structurelles des graphes, c'est-à-dire, le nombre de chercheurs nécessaires pour capturer un fugitif sur un graphe G pour une variante du jeu GENDARMES ET DU VOLEUR est égale à un paramètre $p(G)$ de G (plus/moins une constante). C'est le cas pour la largeur arborescente (Seymour & Thomas, 1993), la largeur linéaire (Kinnersley, 1992 ; Kirousis & Papadimitriou, 1986), la profondeur arborescente (treedepth) (Giannopoulou, Hunter, & Thilikos, 2012), la largeur hyper-arborescente (hyper-treewidth) (Adler, 2004), le rang cyclique (cycle rank) (Giannopoulou et al., 2012), et la largeur arborescente dirigée (directed treewidth) (Johnson, Robertson, Seymour, & Thomas, 2001). Au moins pour la largeur arborescente et la largeur linéaire, ces équivalences sont montrées grâce à des stratégies particulières, les stratégies *monotones* (Bienstock & Seymour, 1991 ; Seymour & Thomas, 1993 ; Mazoit & Nisse, 2008 ; Ilcinkas, Nisse, & Soguet, 2009). Ces stratégies certifient que le fugitif ne peut jamais atteindre un sommet précédemment nettoyé, c'est-à-dire, qu'il n'existe aucun sommet "recontaminé" pendant le déroulement de la stratégie des chercheurs, et ce, qu'importe la stratégie de survie utilisée par le fugitif.

Plus précisément, la première étape, pour montrer l'équivalence entre la largeur arborescente (resp. la largeur linéaire) et le nombre d'agents nécessaires dans la variante où le fugitif est visible (resp. invisible), est de montrer que la recontamination n'aide pas dans cette variante ("recontamination does not help to search a graph") (LaPaugh, 1993 ; Seymour & Thomas, 1993 ; Kirousis & Papadimitriou, 1986), c'est-à-dire, que toute stratégie non-monotone gagnante peut être trans-

formée en une stratégie monotone gagnante utilisant le même nombre de chercheurs. Ensuite, les auteurs de (Seymour & Thomas, 1993) ont montré l'équivalence entre le nombre de chercheurs monotone de la variante, où le fugitif est visible, et la largeur arborescente. De la même façon, Kinnersley a montré l'équivalence entre le nombre de chercheurs monotone de la variante, où le fugitif est invisible, et la largeur linéaire (Kinnersley, 1992).

Comme précisé précédemment, il existe beaucoup de variantes de ces jeux, mais seulement deux d'entre elles nous intéressent particulièrement puisqu'elles sont équivalentes aux largeurs des décompositions de graphes. Puisque la variante pour la largeur linéaire est plus intuitive, nous commençons par celle-ci.

1.4.1 Jeu correspondant aux décompositions linéaires.

Une des premières variantes du jeu DES GENDARMES ET DU VOLEUR est appelée la variante *d'échappement-sommet*. Dans cette variante, le joueur "chercheur" peut soit placer un policier (chercheur) sur un sommet du graphe, soit retirer un policier du graphe, c'est-à-dire, on parle intuitivement de saut des chercheurs (en s'aidant d'un hélicoptère par exemple). Le joueur fugitif est "un peu plus puissant" puisque qu'il pourra se déplacer à une vitesse infinie et est invisible pour le joueur "chercheur". De plus, le jeu se déroule de façon simultanée pour les deux joueurs. La condition pour qu'un policier attrape le fugitif est de se trouver sur le même sommet que lui. Remarquons que quand le fugitif se déplace, il peut se déplacer sur n'importe quel sommet pour lequel il existe un chemin entre lui et la position courante du fugitif tel qu'il n'y a aucun chercheur sur les sommets du chemin. En d'autre terme, un policier présent sur un sommet v empêche le fugitif de se déplacer en passant par v, c'est-à-dire, il *garde* le sommet. Pour illustrer cette variante, supposons que nous sommes sur un chemin $P = (v_1, \ldots, v_p)$, et qu'à un certain tour, pour un certain $1 \leq i < p$, l'ensemble des sommets contaminés, c'est-à-dire, où le fugitif peut se trouver, sont $v_{i+1}, \ldots, v_p$ et qu'un policier se trouve sur v_i. Premièrement, le joueur chercheur ne peut pas savoir où se trouve précisément le fugitif, mais il sait que, grâce à la stratégie qu'il a utilisée jusqu'à maintenant, le fugitif ne peut pas être sur les sommets nettoyés $(v_1, \ldots, v_i)$. De plus, s'il veut bouger le chercheur de v_i à v_{i+1}, le fugitif, avec une rapidité arbitrairement grande, peut parcourir le chemin pour arriver à v_1 avant que le policier n'ait pu se replacer sur v_{i+1}. Cela implique notamment que pour un chemin de taille au moins 2, il faut au moins deux chercheurs, pour être sûr de capturer le fugitif. Nous appellerons le *nombre de capture-sommet*, noté par $ns(G)$, le nombre minimum k tel que k chercheurs ont une stratégie de capture gagnante contre toutes stratégies de survie du fugitif. Nous avons donc que $ns(P) \geq 2$ pour tout chemin P avec au moins 2 sommets.

Comme précisé précédemment, on peut définir des stratégies dites *monotones*, n'autorisant jamais la zone des sommets contaminés d'augmenter. Le plus petit nombre tel qu'il existe une telle stratégie sur un graphe G est le *nombre de capture-sommet monotone*, noté par $mns(G)$. Remarquons que d'un point de vue algorithmique, il est beaucoup plus simple de concevoir des stratégies monotones que des stratégies non monotones. De plus, le nombre de capture-sommet monotone et non monotone sont égaux (LaPaugh, 1993). De ce fait, nous ne considérons jamais de stratégie dite non-monotone. Cette égalité est une étape très importante reliant le nombre de capture-sommet à la largeur linéaire. En effet, vous pouvez imaginer une décomposition linéaire $L = (X_1, \ldots, X_p)$ comme une stratégie monotone à suivre, utilisant $w(L) + 1$ agents et vice-versa. Pour montrer cette équivalence, définissons une stratégie monotone pour les agents dans un graphe G. Intuitivement, une stratégie correspond à une séquence de positions pour les agents

telle que, entre deux ensembles consécutifs de positions, on peut soit rajouter des agents sur des nouveaux sommets, soit on peut retirer les agents de certains sommets. De plus, la stratégie doit être monotone, c'est-à-dire, si un sommet v est dans deux ensembles de positions, alors v est aussi contenu dans tous les ensembles de positions entre ces deux ensembles dans la séquence. Formellement, soit $\mathcal{S} = (S_1, \ldots, S_p)$ une séquence de sous-ensembles de sommets de G telle que $S_i \subseteq V(G)$ pour tout $1 \leq i \leq p$. $\mathcal{S}$ est une stratégie pour les agents si $S_i \cup X = S_{i+1}$ où $X \cap S_i = \emptyset$ ou si $S_i \setminus X = S_{i+1}$ où $X \subseteq S_i$. Intuitivement, entre deux étapes de la stratégie, nous pouvons soit rajouter des agents sur de nouveaux sommets, soit retirer un partie des agents qui était déjà placés sur des sommets. De plus, $\mathcal{S}$ est monotone si pour tout $1 \leq i \leq z \leq j \leq p$, $S_i \cap S_j \subseteq S_z$. Finalement, $\mathcal{S}$ est gagnante s'il n'existe pas de séquence $\mathcal{R} = (v_1, \ldots, v_p)$ de sommets dans G telle que pour tout $1 \leq i \leq p$, $v_i \in V \setminus S_i$ et pour tout $1 \leq i < p$, il existe un chemin entre v_i et v_{i+1} ne contenant aucun sommets de $X_i \cap X_{i+1}$, c'est-à-dire, si le fugitif ne peut pas éviter les agents pendant toute la stratégie $\mathcal{S}$.

Lemme 1.4.1. *(Kinnersley, 1992 ; Kirousis & Papadimitriou, 1986) Pour tout graphe G, $mns(G) = pw(G) + 1$.*

Démonstration. Soit $\mathcal{L} = (X_1, \ldots, X_p)$ une décomposition linéaire de G de largeur k. Soit $S_1 = X_1$. Évidement, le fugitif ne se trouve pas sur des sommets de S_1 quand les agents sont sur S_1, sinon il perd.

Supposons à présent par induction, que pour $1 \leq i < p$, nous avons construit une stratégie $\mathcal{S} = (S_1, \ldots, S_{i'})$ telle que le fugitif ne peut pas être sur les sommets $\bigcup_{1 \leq h \leq i'} S_h$ quand les agents occupent les sommets de $S_{i'} = X_i$ et prouvons qu'il existe une stratégie $\mathcal{S} = (S_1, \ldots, S_{i'}, S_{i'+1}, S_{i'+2})$ telle que le fugitif n'est pas sur les sommets de $\bigcup_{1 \leq h \leq i'+2} S_h$ quand les agents occupent les sommets de $S_{i'+2} = X_{i+1}$.

Soient $A = X_i \setminus X_{i+1}$ et $B = X_{i+1} \setminus X_i$, c'est-à-dire, A correspond aux sommets sur lesquels se trouve un agent qui doit être retiré et B correspond aux sommets sur lesquels nous devons rajouter un agent. Soient $S_{i'+1} = S_{i'} \setminus A$ et $S_{i'+2} = S_{i'+1} \cup B$. Rappelons que par hypothèse d'induction, nous avons construit une stratégie $\mathcal{S} = (S_1, \ldots, S_{i'})$ telle que le fugitif ne peut pas être sur les sommets $\bigcup_{1 \leq h \leq i'} S_h$ quand les agents occupent les sommets de $S_{i'}$. Rappelons aussi qu'il n'y a aucun chemin d'un sommet de $\bigcup_{i+1 \leq q \leq p} X_q \setminus (X_i)$ à un sommet de $\bigcup_{1 \leq q \leq i} S_q \setminus X_{i+1}$ évitant les sommets de $X_i \cap X_{i+1}$ par la $3^{ème}$ propriété des décompositions linéaires. De ce fait, $\mathcal{S} = (S_1, \ldots, S_{i'}, S_{i'+1})$ est une stratégie telle que le fugitif n'est pas sur les sommets $\bigcup_{1 \leq h \leq i'+1} S_h$ quand les agents occupent les sommets de $S_{i'+1}$. Autrement dit, quand les agents sont sur $S_{i'+1}$, le fugitif se trouve sur un sommets de $\bigcup_{i+1 \leq h \leq p} X_h \setminus X_i$. De la même façon que précédemment, le fugitif ne peut pas se déplacer d'un sommet de $\bigcup_{i+1 \leq h \leq p} X_h \setminus X_i$ à un sommet de $\bigcup_{1 \leq h \leq i} X_h \setminus X_{i+1}$ en évitant les sommets de $X_i \cap X_{i+1}$. Par conséquent, $\mathcal{S} = (S_1, \ldots, S_{i'+2})$ est une stratégie telle que le fugitif n'est pas sur $\bigcup_{1 \leq h \leq i+1} X_h$ (il ne peut pas être sur B, sinon il est capturé à l'étape $i'+2$ de la stratégie). Le fait que $\mathcal{S} = (S_1, \ldots, S_{i'+2})$ est une stratégie monotone suit de la $3^{ème}$ propriété des décompositions (pour tout $1 \leq i \leq z \leq j \leq p$, $X_i \cap X_j \subseteq X_z$) et de la construction de $\mathcal{S}$. Le fait que $\mathcal{S} = (S_1, \ldots, S_{i'+2})$ utilise au plus $k+1$ agents suit du fait que pour tout $1 \leq i \leq p$, $|X_i| \leq k+1$. Nous pouvons donc conclure qu'il existe une stratégie $\mathcal{S} = (S_1, \ldots, S_q)$ monotone et gagnante dans G utilisant $k+1$ agents (puisque $|X_i| \leq k+1$ pour tout $1 \leq i \leq p$). Il est possible de prouver que $\mathcal{L} = (S_1, \ldots, S_q)$ est une décomposition linéaire de G de largeur k de façon similaire. □

Grâce à ces stratégies monotones (c'est-à-dire, au fait que $mns(G) = ns(G)$ pour tout graphe G), il a donc été montré que cette variante du jeu DES GENDARMES ET DU VOLEUR est équi-

valente à la largeur linéaire, c'est-à-dire, pour tout graphe G, $ns(G) = pw(G) + 1$ (Kinnersley, 1992).

De façon équivalente, nous pourrions définir une nouvelle mesure pour ce jeu, étant la distance maximale entre deux policiers pendant une stratégie. Cette nouvelle mesure pour une stratégie monotone et optimal serait alors égale à la longueur de la décomposition linéaire correspondante à cette stratégie. Notons que nous ne savons pas si un tel jeu est équivalent à sa variante monotone.

1.4.2 Jeu correspondant aux décompositions arborescentes.

Passons à présent à la variante du jeu DES GENDARMES ET DU VOLEUR dont les stratégies de capture correspondent à des décompositions arborescentes. Ce jeu est exactement le même que le précédent à l'exception que le fugitif est cette fois visible. Nous noterons le nombre de capture-sommet visible d'un graphe G par $vns(G)$. Le nombre de capture-sommet visible monotone d'un graphe G est noté par $mvns(G)$, et, comme précédemment, est égal à $vns(G)$ (Seymour & Thomas, 1993). De plus, il est montré dans (Seymour & Thomas, 1993) que $mvns(G) = tw(G) + 1$. Intuitivement, une stratégie de capture peut être obtenue à partir d'une décomposition arborescente. Il suffit, pour le premier tour, de placer les policiers sur tous les sommets de n'importe quel sac X de la décomposition $(T, \mathcal{X})$. Remarquons alors que le fugitif ne peut se trouver sur ces sommets, sinon il est directement capturé. De plus, puisque le fugitif est visible, les chercheurs savent dans quel sous-graphe, induit par les sommets contenus dans les sacs d'une des composantes connexes de $T - X$, le fugitif se trouve. Supposons que le fugitif se trouve dans G_1 le sous-graphe induit par les sacs de T_1 où $T_1, \ldots, T_p$ sont les composantes connexes de $T - X$. Notons alors par X' le voisin de X dans T_1. Nous pouvons alors retirer les chercheurs des sommets de $X \setminus X'$ sans permettre au fugitif de recontaminer des sommets de $X \setminus X'$ ou des sommets des sacs dans $T_2, \ldots, T_p$. Répéter ces opérations jusqu'à ce que les chercheurs soient sur les sommets d'un sac étant une feuille de la décomposition, revient donc, intuitivement, à pousser le fugitif dans un coin (le sac feuille) du graphe pour finalement le capturer.

Au-delà de ces deux jeux équivalents à la largeur arborescente et à la largeur linéaire, il existe beaucoup de variantes des jeux de recherche dans les graphes. Pour donner quelques exemples, il existe le jeu de l'endiguement (containment en anglais) où les chercheurs se trouvent sur les arêtes du graphe alors que le fugitif se trouve sur un sommet du graphe. Le but pour les chercheurs dans cette variante est d'occuper toutes les arêtes incidentes à la position du fugitif (Crytser, Komarov, & Mackey, 2020). Une autre variante est la recherche exclusive de graphe (exclusive graph searching) où il n'est pas autorisé pour le joueur "chercheur" de positionner deux chercheurs sur le même sommet (Blin, Burman, & Nisse, 2017 ; Markou, Nisse, & Pérennes, 2017). Il existe aussi une variante où le fugitif peut éliminer un chercheur du graphe (Bonato et al., 2013). Il en existe encore plein d'autres (Bonato & Nowakowski, 2011).

1.5 Le jeu des Chasseurs et du Lapin

Dans cette thèse, nous allons nous intéresser à une variante assez originale du jeu DES GENDARMES ET DU VOLEUR, le jeu DES CHASSEURS ET DU LAPIN (Britnell & Wildon, 2013 ; Haslegrave, 2014).

1.5.1 Définitions

Ce jeu se déroule toujours sur un graphe G et un entier k correspondant au nombre de chasseurs autorisés est fixé. Le lapin est invisible (caché derrière des buissons représentés par les sommets du graphe) et l'équipe de chasseurs essaie de capturer le lapin en lui tirant dessus (avec des fléchettes tranquillisantes). Comme les précédents jeux, ce jeu se déroule tour par tour. Nous rajoutons un tour 0 correspondant à l'initialisation du jeu, où le lapin décide de se positionner sur un sommet r_0 du graphe. Excepté ce tour 0, un tour correspond, tout d'abord, au sous-ensemble S_i des sommets de $V(G)$ sur lesquels les chasseurs tirent simultanément. Si un chasseur tire sur le lapin pendant un tour i, c'est-à-dire, $r_{i-1} \in S_i$ le jeu est terminé et le joueur "chasseur" gagne. Sinon, le lapin est effrayé par la salve de tirs S_i et doit se déplacer, de sa position courante r_{i-1}, vers un sommet adjacent $r_i \in N(r_{i-1})$. Notons que le lapin reste invisible aux yeux des chasseurs pendant ce déplacement. De plus, il n'y a pas de limite de nombre de tours, c'est-à-dire, le lapin doit survivre jusqu'à ce que les chasseurs arrêtent de tirer (jusqu'à la fin de la stratégie des chasseurs). Plus précisément, une *trajectoire* pour le lapin est une séquence de sommets $R = (r_0, \ldots, r_p)$ telle que $\{r_{i-1}, r_i\} \in E(G)$ pour tout $1 \leq i \leq p$. Une *stratégie de chasse* $S = (S_1, \ldots, S_p)$ pour les chasseurs est une séquence de sous-ensembles de sommets de G.

Puisque le lapin est invisible, il est difficile (pour le joueur "chercheurs") de connaitre la trajectoire qu'il suit. De ce fait, pour que les chasseurs gagnent, il faut trouver une stratégie certifiant que le lapin est capturé, qu'importe la stratégie qu'il utilise. Nous considérons donc, informellement, un jeu à un joueur, le joueur "chasseur", dont le but est d'exécuter une séquence de salves d'au plus k tirs chacune, de telle sorte qu'à la fin de la séquence, pour n'importe quelle stratégie utilisée par le lapin pour survivre, il existera toujours au moins un tour durant lequel il est capturé par un chasseur. Plus précisément, une stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_p)$ est *gagnante* si pour toute trajectoire du lapin $R = (r_0, \ldots, r_p)$ et il existe $1 \leq i \leq p$ tel que $r_{i-1} \in S_i$. Dans le cas contraire, nous dirons qu'une trajectoire du lapin $R = (r_0, \ldots, r_p)$ est gagnante contre la stratégie de chasse $\mathcal{S} = (S_1, \ldots S_p)$ si pour tout $1 \leq i \leq p$, $r_{i-1} \notin S_i$. Soit $h(G)$ le *nombre minimum de chasseurs* pouvant gagner sur le graphe G, c'est-à-dire, tel qu'il existe au moins une stratégie de chasse gagnante utilisant au plus $h(G)$ chasseurs et qu'il n'existe pas de stratégie de chasse gagnante utilisant moins de $h(G)$ chasseurs.

Comme pour le jeu DES GENDARMES ET DU VOLEUR, si $k = |V(G)|$, alors le problème est évident, c'est-à-dire, la stratégie de chasse $S = (S_1 = V(G))$ utilisant k chasseurs est gagnante. Mais, cette fois-ci, la règle sur le déplacement obligatoire du lapin pendant chaque tour implique que, quand $k = |V(G)| - 1$, il suffit de tirer sur n'importe quel ensemble de $|V(G)| - 1$ sommets du graphe G deux fois consécutivement. Plus précisément, pour tout $v \in V(G)$, $S = (S_1 = V(G) \setminus v, S_2 = V(G) \setminus v)$ est une stratégie gagnante utilisant $|V(G)| - 1$ chasseurs. Par conséquent, $h(G) \leq |V(G)| - 1$. Nous dirons que le lapin a une *stratégie de survie* $\mathcal{R}$ *contre* $k \geq 1$ *chasseurs* dans G si pour toute stratégie de chasse $\mathcal{S}$ utilisant k chasseurs dans G, il existe une trajectoire $\mathcal{R}(\mathcal{S})$ dans G gagnante contre $\mathcal{S}$. Autrement dit, le lapin suit une trajectoire dépendant des prochaines salves de tirs des chasseurs. Notons que si le lapin a une stratégie de survie contre $k \geq 1$ chasseurs dans G, alors $h(G) > k$.

Ce jeu est très proche de la variante du jeu DES GENDARMES ET DU VOLEUR correspondant à la largeur linéaire. Les principales différences étant, l'obligation pour le lapin de bouger à chaque tour sur un sommet adjacent à sa position courante, la vitesse du lapin et le fait que le lapin peut se rendre sur un sommet qui vient d'être tiré, c'est-à-dire, les chasseurs n'ont pas le rôle de "garde" des gendarmes. Il est néanmoins possible d'imiter cette fonction de garde en tirant plusieurs fois

consécutivement sur un sommet v. Dans ce cas, si le lapin décide de se rendre sur le sommet v avant le dernier tir consécutif, il sera immédiatement capturé.

Les premières apparitions de ce jeu furent dans des concours ou des forums (probleme 6^* dans (Fedorov, Levy, Kovaldzhi, & Yashchenko, 2011), au SWERC 2010 ou encore sur MathOverflow, etc.). Plus tard, deux articles scientifiques (Britnell & Wildon, 2013 ; Haslegrave, 2014) définissent ce jeu. Notons que dans tous ces cas, le jeu n'est défini qu'avec un chasseur, c'est-à-dire, $k = 1$. Ils s'intéressent principalement au plus petit arbre T sur lequel il n'y a aucune stratégie de chasse gagnante pour un chasseur. Cet arbre (le plus petit nécessitant au moins deux chasseurs) est le graphe *araignée* avec 3 pattes de longueur au moins 3, c'est-à-dire, le graphe obtenu à partir d'un graphe étoile avec 3 feuilles, où chaque arête est subdivisée deux fois.

Lemme 1.5.1. *Soit T un arbre. $h(T) \leq 1$ si et seulement si T ne contient pas de graphe araignée S avec 3 pattes de tailles au moins 3 comme sous-graphe, c'est-à-dire, si T contient un chemin P tel que pour tout sommet $v \in V(T)$, $\min_{u \in V(P)} dist(u, v) \leq 2$.*

Ce résultat implique notamment que pour tout chemin P_n de taille n, $h(P_n) \leq 1$. Puisque les jeux de recherche dans les graphes tendent à localiser le joueur "cherché", nous dirons que le graphe ne contenant qu'un sommet a un nombre de chasseurs 0, c'est-à-dire, il n'est pas nécessaire de le trouver et de le capturer, puisque nous savons exactement où il se trouve. Ce cas de base est défini de cette façon pour nous aider pour nos futurs résultats.

Certains (SWERC and (Haslegrave, 2014)) considèrent les stratégies de chasse les plus rapides utilisant un chasseur, c'est-à-dire, nécessitant le moins de tours possible, et déterminent le nombre minimal q de tours tel qu'il existe une stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_q)$ gagnante utilisant un chasseur. Dans un arbre T, un sommet v est une *brindille* si $d(v) > 1$ et au moins $d(v) - 1$ sommets de $N(v)$ sont des feuilles.

Lemme 1.5.2. *Soit T un arbre avec $n \geq 3$ sommets tel qu'il ne contient pas de graphe araignée S avec 3 pattes de tailles au moins 3 comme sous-graphe. Si T est un graphe étoile, alors il existe une stratégie de chasse gagnante utilisant 1 chasseurs durant 2 tours. Sinon, il existe une stratégie de chasse gagnante utilisant 1 chasseur durant $2n + 2t - 2l - 4$ où t est le nombre de brindilles de T et l est le nombre de feuilles de T.*

La version généralisée, pour $k > 1$, fut ensuite introduite dans (Abramovskaya, Fomin, Golovach, & Pilipczuk, 2016). Grâce au Lemme 1.5.1, nous savons quels arbres ont un nombre de chasseurs 1. Pour autant, les stratégies gagnantes utilisant au plus 1 chasseur dans ces arbres ne sont pas triviales. L'idée pour être capable de décrire de telles stratégies est de faire une supposition sur les sommets de départ pour le lapin. Dans ce but, il existe une variante de ce jeu dans lequel les positions du lapin pour le tour 0 sont restreintes. Cette variante est très intéressante puisque, sous certaines conditions (respectées par les arbres et les chemins), elle est équivalente au jeu original.

1.5.2 Cas des graphes bipartis

Rappelons qu'un graphe G est biparti s'il existe une partition des sommets de G en deux ensembles indépendants V_r et V_w ($V_r \cup V_w = V(G)$), c'est-à-dire, tel que pour toute paire u, v de sommets dans V_r (resp. dans V_w), $\{u, v\} \notin E(G)$. Nous parlerons de sommets rouges pour les sommets de V_r et de sommets blancs pour les sommets de V_w. Nous noterons $G = (V_r \cup V_w, E)$ un graphe biparti où V_r et V_w sont les ensembles de sommets indépendants. La variante consiste

à restreindre les positions initiales possibles pour le lapin. Nous considérons, dans cette variante rouge, que le lapin doit commencer sur un sommet de V_r, les sommets rouges (cf. Figure 1.4).

Le nombre minimum de chasseurs nécessaires pour gagner dans cette variante rouge sur un graphe biparti $G = (V_r \cup V_w, E)$, noté $h_{V_r}(G)$ est égale au nombre minimum de chasseurs nécessaires pour gagner dans la version classique du jeu sur G. Ce résultat est assez intuitif si nous remarquons d'abord la propriété suivante.

Remarque 1.5.3 – Soient $G = (V_r \cup V_w, E)$ un graphe biparti et $\mathcal{R} = (r_0, r_1, \ldots, r_\ell)$ une trajectoire suivie par le lapin commençant en V_r dans G. Alors, pour chaque $1 \leq i \leq \lceil \ell/2 \rceil$, $r_{2i-2} \in V_r$ et (si $2i \leq \ell$) $r_{2i-1} \in V_w$.

Intuitivement, pour la version classique du jeu, il est possible de supposer dans un premier temps que le lapin a commencé sur V_r, puis, après avoir appliqué une stratégie gagnante de la variante rouge, soit le lapin a commencé sur V_r, et donc a été capturé, soit le lapin a en faite commencé sur V_w. Il suffit donc, d'attendre (en tirant n'importe où) que le lapin se trouve sur un sommet de V_r (en sachant qu'il a commencé sur V_w), pour appliquer une nouvelle fois la stratégie gagnante de la variante rouge.

Lemme 1.5.4. *(Abramovskaya et al., 2016) Pour tout graphe biparti $G = (V_r \cup V_w, E)$, $h(G) = h_{V_r}(G) = h_{V_w}(G)$.*

Démonstration. Par définition, $\max\{h_{V_r}(G), h_{V_w}(G)\} \leq h(G)$. Pour montrer que $h(G) \leq h_{V_r}(G)$ (resp., $h(G) \leq h_{V_w}(G)$), considérons $\mathcal{S}_r = (S_1, \ldots, S_\ell)$, une stratégie de chasse gagnante dans G quand le lapin commencent sur les sommets rouges V_r (resp., par rapport à V_w). Si ℓ est impair, alors $(S_1, \ldots, S_\ell, S_1, \ldots, S_\ell)$ est une stratégie de chasse gagnante par rapport à $V(G)$, et sinon, $(S_1, \ldots, S_\ell, \{u\}, S_1, \ldots, S_\ell)$ (où u est n'importe quel sommet de $V(G)$) est une stratégie de chasse gagnante quand le lapin commence sur $V(G)$. □

Notons que, dans la majeure partie du chapitre 4, nous considérerons seulement les stratégies de chasse quand le lapin commence sur V, excepté dans la section 4.4. Plus précisément, dans la section 4.4, nous étudierons le jeu DES CHASSEURS ET DU LAPIN dans les arbres lorsque le lapin doit commencer sur un sommet rouge (de V_r).

1.5.3 Propriétés et Résultats

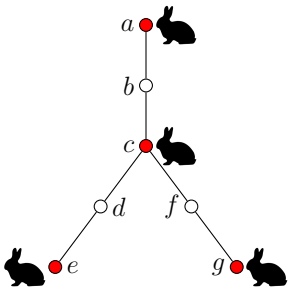
Concernant les propriétés classiques du jeu, remarquons tout d'abord qu'il est possible de se restreindre au calcul du nombre de chasseurs pour les composantes connexes d'un graphe :

Lemme 1.5.5. *(Bolkema & Groothuis, 2019) Pour tout graphe G non connexe où $C_1, \ldots C_p$ sont ses composantes connexes, $h(G) = \max_{1 \leq i \leq p} h(C_i)$.*

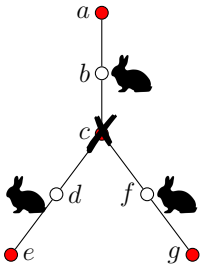
De plus, ce nombre $h(G)$ est clos par sous-graphe.

Lemme 1.5.6. *(Abramovskaya et al., 2016) Pour tout graphe G, tout sous-graphe H de G, $h(H) \leq h(G)$*

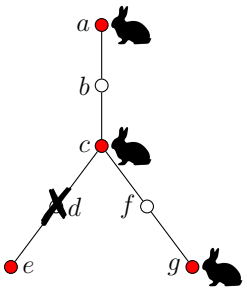
Par contre, avant cette thèse, il n'était pas connu si ce paramètre est clos par mineur ou non.



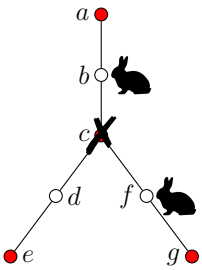
(a) $Z_0 = \{a, c, e, g\}$



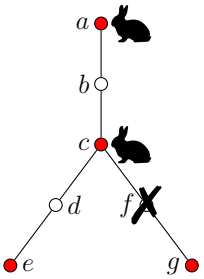
(b) $S_1 = \{c\}$, $Z_1 = \{b, d, f\}$



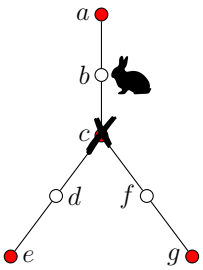
(c) $S_2 = \{d\}$, $Z_2 = \{a, c, g\}$



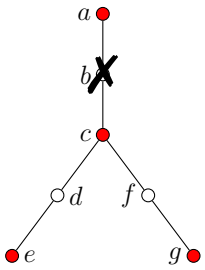
(d) $S_3 = \{c\}$, $Z_3 = \{b, f\}$



(e) $S_4 = \{f\}$, $Z_4 = \{c, a\}$



(f) $S_5 = \{c\}$, $Z_5 = \{b\}$



(g) $S_6 = \{b\}$, $Z_6 = \emptyset$

Figure 1.4 – Exemple d'un graphe biparti (où $V_r = \{a, c, e, g\}$ correspond à la partie rouge de la bipartition, illustrée par les sommets rouges dans les figures) et d'une stratégie de chasse gagnante quand le lapin commence sur V_r. Chaque sous-figure représente la situation à la fin du tour correspondant. Au tour 0, le lapin occupe n'importe quel sommet dans $\{a, c, e, g\}$. Ensuite, au tour 1, le chasseur tire sur le sommet c (représenté par la croix sur le sommet correspondant de la sous-figure (b)) et le lapin se déplace vers l'un des sommets de $\{b, d, f\}$. Le jeu se poursuit jusqu'à la fin du tour 6 (sous-figure (g)), où le chasseur est sûr d'avoir capturé le lapin (s'il a commencé sur V_r) qu'importe la stratégie qu'il a utilisée. Formellement, nous avons $\mathcal{S} = (\{c\}, \{d\}, \{c\}, \{f\}, \{c\}, \{b\})$ and $\mathcal{Z}(\mathcal{S}) = (\{a, c, e, g\}, \{b, d, f\}, \{a, c\}, \{b\}, \{a, c\}, \{b\}, \emptyset)$.

Bornes inférieures. Il existe plusieurs bornes inférieures sur le nombre minimum de chasseurs comme le degré minimum et la dégénérescence. Ces bornes reposent sur l'intuition qu'il doit exister une stratégie de survie pour le lapin.

Lemme 1.5.7. *(Bolkema & Groothuis, 2019) Pour tout graphe G, $h(G) \geq \delta$ où δ est le degré minimum de G, c'est-à-dire, $\delta = \min_{v \in V(G)} d(v) = \min_{v \in V(G)} |\{u \in V(G) \mid \{u, v\} \in E\}|$.*

Idée de Preuve. Supposons qu'il existe une stratégie utilisant moins que δ chasseurs. À toute étape de la stratégie (et ce depuis le premier tour), il existe toujours un voisin w de la position courante du lapin v qui n'est pas tiré au prochain tour (puisque v a au moins δ voisins et au plus $\delta - 1$ sont tirés au prochain tour). Cette propriété nous permet donc de définir une trajectoire pour le lapin gagnante contre toute stratégie de chasse utilisant $k < \delta$ chasseurs. ■

Par le Lemme 1.5.6, nous obtenons une généralisation du lemme précédent pour la dégénérescence d'un graphe. Rappelons que la dégénérescence d'un graphe G est le plus petit q tel qu'il existe un sous-graphe de G de degré minimum q.

Lemme 1.5.8. *(Bolkema & Groothuis, 2019) Pour tout graphe G, $h(G) \geq k$ où k est la dégénérescence de G*

Une autre façon d'imaginer des bornes inférieures consiste à regarder jusqu'à quel point nous pouvons restreindre les positions du lapin à un certain temps t par rapport au nombre de chasseurs utilisés. Plus précisément, définissons $mun_G(k) = \min\{|N(W)| \mid W \subseteq V(G), |W| = k\}$. Intuitivement, après la $i^{ème}$ salve de tir et avant le mouvement du lapin, s'il existe k positions possibles pour le lapin, alors, il y aura au moins $mun_G(k)$ positions possibles pour le lapin après son déplacement. Définissons ensuite $m_0 = \max_k\{mun_G(k) - k\}$. Intuitivement, m_0 est le nombre minimum de positions possibles pour le lapin quand il joue contre une stratégie utilisant au plus m_0 chasseurs.

Lemme 1.5.9. *(Bolkema & Groothuis, 2019) Pour tout graphe connexe G, $h(G) > m_0$.*

Ce jeu ayant été défini récemment, il existe encore peu de résultats, et ce même dans le cas des arbres. Par exemple, il n'est pas connu comment calculer le nombre de chasseurs minimum dans les arbres. Cependant, des bornes inférieures ont été calculées pour ce nombre de chasseurs dans les arbres. Par exemple :

Lemme 1.5.10. *(Abramovskaya et al., 2016) Il existe un arbre T contenant $2^{\mathcal{O}(k \log(k))}$ sommets et tel que $h(T) \geq k$.*

Lemme 1.5.11. *(Gruslys & Méroueh, 2015) Il existe un arbre (obtenu à partir d'un arbre binaire complet de taille k en subdivisant chaque arête une fois) T contenant n sommets tel que $h(T) \geq (\frac{1}{4} - \epsilon) \log_2(n)$.*

Bornes supérieures. Rappelons que les règles de la variante des Gendarmes et du Voleur équivalente à la largeur linéaire sont à la fois proches et différentes de celles du jeu des Chasseurs et du Lapin. Plus précisément, les seules différences sont la vitesse du fugitif, le fait que le fugitif soit forcé de bouger (dans le jeu des chasseurs et du lapin), et que les chasseurs sont en dehors du graphe pour le jeu des chasseurs et du lapin alors qu'ils sont sur le graphe pour la variante jeu des Gendarmes et du Voleur (ce qui leur permet de garder des sommets). Bien que ces jeux diffèrent, toute stratégie de capture gagnante pour cette variante du jeu des Gendarmes du Voleur est aussi une stratégie de chasse gagnante dans le jeu des Chasseurs et du Lapin. De ce fait :

Lemme 1.5.12. *(Abramovskaya et al., 2016) Pour tout graphe G, $h(G) \leq pw(G) + 1$.*

De plus, il existe des graphes G tels que $h(G) = pw(G) + 1$. Cette relation nous permet notamment de borner le nombre de chasseurs des arbres, puisque la largeur linéaire des arbres est au plus $\mathcal{O}(\log(n))$ (Parsons, 1978).

Quelques classes de graphes bipartis ont été étudiées en commençant par les arbres. Il a été prouvé que $h(T) \leq \mathcal{O}(\log(n))$ pour tout arbre T (Abramovskaya et al., 2016). Notons qu'un résultat plus fort a été prouvé dans (Gruslys & Méroueh, 2015) :

Lemme 1.5.13. *(Gruslys & Méroueh, 2015) Pour tout arbre T contenant n sommets, $h(T) \leq \lceil \frac{1}{2} \log_2(n) \rceil$.*

Ce résultat implique notamment qu'il existe des arbres tels que le nombre de chasseurs est plus petit que la largeur linéaire puisqu'il existe un arbre contenant n sommets de largeur linéaire $log(n)$.

Il a aussi été prouvé que la valeur de $h(G)$ pour les grilles de taille $n \times m$ est égale à $\lfloor \frac{\min\{n,m\}}{2} \rfloor + 1$ (Abramovskaya et al., 2016). Ensuite, il a aussi été prouvé que $h(Q^n) = 1 + \Sigma_{i=0}^{n-2} \binom{i}{\lfloor i/2 \rfloor}$ (Bolkema & Groothuis, 2019), où Q^n est l'hypercube de dimension n.

Pour conclure, les résultats connus sur ce jeu sont principalement des résultats pour les graphes bipartis. Même si de bonnes bornes ont été trouvées pour les arbres, la complexité du calcul du nombre de chasseurs pour les arbres est toujours ouverte (c'est aussi le cas pour les graphes quelconques). Définir une propriété de monotonie pourrait nous aider pour répondre à cette question. Il serait aussi intéressant de calculer le nombre de chasseurs pour des classes de graphes non biparties. La classe des graphes scindés serait donc un bon candidat puisque ce sont des graphes pouvant être partitionnés en deux ensembles, l'un étant un ensemble indépendant et l'autre étant un ensemble complètement connexe (une clique).

1.6 Contributions de la thèse

La longueur est un paramètre intéressant des décompositions, que ce soit pour ses applications ou ses liens avec la largeur des décompositions. Le but de cette thèse est d'enrichir nos connaissances sur ce paramètre, que ce soit pour la longueur arborescente ou linéaire. Nous nous intéressons dans le chapitre 2 à la longueur arborescente. Ensuite, dans le chapitre 3, nous nous intéressons à la longueur linéaire. Finalement, dans le chapitre 4, nous étudierons le jeu DES CHASSEURS ET DU LAPIN.

Décomposition Arborescente. Dans le chapitre 2, nous nous concentrons sur le calcul de la longueur arborescente dans les graphes série-parallèles. Ces résultats ont été obtenus en collaboration avec Nicolas Nisse, Simon Nivelle et Guillaume Ducoffe (Dissaux, Ducoffe, Nisse, & Nivelle, 2021b, 2021c, 2021a). La section 2.1 est consacrée aux définitions formelles des principaux concepts utilisés dans ce chapitre, comme la représentation des graphes série-parallèles en décomposition en oreilles imbriquées. Dans la section 2.2, nous considérons les graphes *melons*. Ce sont les graphes série-parallèles G obtenus en identifiant les extrémités de p chemins deux-à-deux disjoints intérieurement $(P_i)_{i \leq p}$ de longueur respective ℓ_i (avec $\ell_1 \geq \ldots \geq \ell_p$). Nous montrons que, pour tout graphe melon G, $t\ell(G) = \min\{\lceil \frac{lc(G)}{3} \rceil, \max\{\lceil \frac{is(G)}{3} \rceil, \ell_p\}\}$ où $is(G)$ (resp., $lc(G)$) est la taille d'un plus grand cycle isométrique (resp., d'un plus grand cycle) dans G. De plus, nous présentons un exemple de graphe série-parallèle dont sa longueur arborescente

est différente de la taille de ses plus grands cycles (isométriques ou non). Ce graphe nous permet de définir les graphes *Dumbo* qui sont des graphes série-parallèles dont leur plus grand cycle isométrique est de taille 6 et ayant une longueur arborescente au moins 3. Avec les cycles isométriques de taille au moins 7, c'est la seule famille de graphes ne pouvant pas être contenu comme sous-graphes isométriques dans les graphes série-parallèles de longueur arborescente 2. Dans la section 2.4, cette caractérisation, en termes de sous-graphes isométriques interdits, nous permet de décider en temps polynomial si un graphe série-parallèle a une longueur arborescente au plus 2. Nous concevons aussi un algorithme d'approximation de rapport $\frac{3}{2}$ pour la longueur arborescente des graphes série-parallèles dans la section 2.3. Enfin, nous concluons dans la Section 2.5 en discutant comment notre caractérisation pourrait être généralisée pour calculer la longueur arborescente des graphes série-parallèles.

Décomposition Linéaire. Dans le chapitre 3, nous nous concentrons sur le calcul de la longueur linéaire de classes de graphes simples. Ces résultats ont été obtenus en collaboration avec Nicolas Nisse (Dissaux & Nisse, 2022c, 2022b, 2022a). Plus précisément, nous commençons par les arbres et les cycles dans la section 3.2. Nous concevons tout d'abord un algorithme calculant en temps polynomial la longueur linéaire des arbres. Ensuite, nous prouvons que la longueur linéaire des cycles à n sommets est égale à $\lfloor \frac{n}{2} \rfloor$. Ensuite, notre but est de caractériser la longueur linéaire des graphes planaires extérieurs. Nous réussissons à concevoir un algorithme d'approximation de facteur additif $+1$, c'est-à-dire, calculant une décomposition de longueur au plus $p\ell(G)+1$. Nous donnons aussi un exemple de graphe planaire extérieur pour lequel notre algorithme calcule une décomposition de longueur $p\ell(G)+1$.

Le jeu des Chasseurs et du Lapin. Finalement, nous étudions une variante du jeu DES GENDARMES ET DU VOLEUR, le jeu DES CHASSEURS ET DU LAPIN. Ces travaux ont été réalisés en collaboration avec Nicolas Nisse, Foivos Fioravantes et Harmender Galhawat (Dissaux, Fioravantes, Galhawat, & Nisse, 2023). Plus précisément, dans la section 4.2, nous commençons par nous intéresser à la définition (non triviale) d'une propriété de monotonie de ce jeu dans le but de nous aider à concevoir des stratégies de chasse. Dans la section 4.2.1, nous justifions la définition de cette propriété de monotonie, et nous enchaînons sur des propriétés impliquées par de telles stratégies de chasse monotones. Nous prouvons que $pw(G) \leq mh(G) \leq pw(G)+1$ dans la section 4.2.2. Cela implique notamment que le calcul de $mh(G)$ est un problème NP-complet. Nous nous intéressons ensuite, dans la section 4.3, à des classes de graphes simples, comme les graphes scindés, les graphes d'intervalles et les cographes. Nous caractérisons le nombre de chasseurs monotone et non-monotone dans ces classes. Nous adaptons aussi l'algorithme du calcul de la largeur linéaire des arbres (Parsons, 1978) pour calculer le nombre de chasseurs monotone des arbres. Nous nous intéressons ensuite à la différence entre ces deux paramètres (nombre de chasseurs monotone et non-monotone) et montrons qu'il existe des arbres pour lesquels la différence est arbitrairement grande, c'est-à-dire, qu'autoriser la recontamination permet de diminuer (significativement) le nombre de chasseurs nécessaires. Finalement, nous montrons que le calcul du nombre de chasseurs monotone ou non monotone admet un noyau polynomial par rapport au nombre de sommets minimum couvrant les arêtes d'un graphe G, $vc(G)$.

Nous concluons dans la section 5 en présentant des perspectives sur les futurs travaux.

Mes publications

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021a, septembre). Longueur Arborescente des Graphes Série-Parallèles. In *ALGOTEL 2021 - 23èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications.* La Rochelle, France. Consulté sur https://hal.science/hal-03217731

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021b). Treelength of series-parallel graphs. In C. E. Ferreira, O. Lee, & F. K. Miyazawa (Eds.), *Proceedings of the XI latin and american algorithms, graphs and optimization symposium, LAGOS 2021, online event / são paulo, brazil, may 2021* (Vol. 195, pp. 30–38). Elsevier. Consulté sur https://doi.org/10.1016/j.procs.2021.11.008 doi: 10.1016/j.procs.2021.11.008

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021c). Treelength of series-parallel graphs. *Discrete Applied Mathematics (accepté).*

Dissaux, T., Fioravantes, F., Galhawat, H., & Nisse, N. (2023, février). *Further results on the Hunters and Rabbit game through monotonicity* (Rapport technique). Inria - Sophia Antipolis. (soumis à une conférence internationale). Consulté sur https://hal.science/hal-03995642

Dissaux, T., & Nisse, N. (2022a). Longueur linéaire des graphes planaires extérieurs. In *ICGT 2022 - The 11th International Colloquium on Graph Theory and combinatorics.* Montpellier, France.

Dissaux, T., & Nisse, N. (2022b, mai). Longueur linéaire des graphes planaires extérieurs. In *AlgoTel 2022 - 24èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications.* Saint-Rémy-Lès-Chevreuse, France. Consulté sur https://hal.science/hal-03655647

Dissaux, T., & Nisse, N. (2022c). Pathlength of outerplanar graphs. In A. Castañeda & F. Rodríguez-Henríquez (Eds.), *LATIN 2022 : Theoretical informatics - 15th latin american symposium, guanajuato, mexico, november 7-11, 2022, proceedings* (Vol. 13568, pp. 172–187). Springer. (soumis à une revue internationale). Consulté sur https://doi.org/10.1007/978-3-031-20624-5_11 doi: 10.1007/978-3-031-20624-5_11

CHAPITRE 2

La longueur arborescente / Treelength

Dans ce chapitre, nous nous concentrons sur l'étude de la longueur arborescente. Ces résultats ont été obtenus en collaboration avec Nicolas Nisse, Simon Nivelle et Guillaume Ducoffe (Dissaux et al., 2021b, 2021c, 2021a). Rappelons qu'il existe une relation entre la largeur arborescente et la longueur arborescente dans les graphes planaires. De plus, nous sommes intéressés par la complexité du calcul de la longueur arborescente dans les graphes planaires. Pour comprendre un peu mieux le comportement de ce paramètre dans les graphes planaires, nous enquêtons sur des sous-classes des graphes planaires. Plus précisément, étant donné que nous connaissons déjà la longueur arborescente des graphes planaires extérieurs et des grilles, nous étudions la longueur arborescente des graphes série-parallèles. Malheureusement, cette classe de graphes ne parait pas si simple pour le calcul de la longueur arborescente. Nous commençons donc par étudier une version très simple des graphes série-parallèles, les graphes melons. Cette classe nous permet notamment d'identifier une famille de sous-graphes isométriques interdits pour les graphes série-parallèles de longueur 2. Nous explicitons ensuite la liste des obstructions (en tant que sous-graphes isométriques interdits) pour les graphes série-parallèles. Cela implique l'existence d'un algorithme polynomial décidant si oui ou non un graphe série-parallèle a une longueur arborescente au plus 2 (et calculant une décomposition de longueur 2 si possible). Au passage, nous concevons aussi un algorithme d'approximation de ratio $\frac{3}{2}$ pour la longueur arborescente des graphes série-parallèles. Finalement, nous donnons quelques arguments de pourquoi notre approche (en tant que sous-graphes isométriques interdits) ne semble pas pouvoir être généralisée pour $k > 2$.

2.1 Préliminaires

Dans ce chapitre, nous ne considérons que les graphes simples, non dirigés et non pondérés (sans boucles ni arêtes parallèles). Nous noterons en général par $G = (V, E)$ les graphes que nous considérons où V est l'ensemble de sommets de G et E est l'ensemble d'arête de G. Lorsqu'il ne sera pas précisé, n correspondra toujours au nombre $|V|$ de sommets de G.

Dans ce qui suit, toute arête $\{x, y\}$ est aussi considérée comme l'ensemble des deux sommets x et y. En particulier, nous disons que $X \subseteq V$ contient une arête e si $e \subseteq X$. Pour tout $v \in V$, notons $N_G(v) = \{w \in V(G) \mid \{v, w\} \in E(G)\}$ le voisinage de v dans G et $N[v] = N(v) \cup \{v\}$ son *voisinage fermé*. Étant donné $S \subseteq V$, notons $N(S) = \{v \in V \setminus S \mid \exists u \in S, \{u, v\} \in E\}$

l'ensemble des sommets de $G \setminus S$ adjacents à un sommet de S. La *distance* $dist_G(u, v)$ dans $G = (V, E)$ entre deux sommets $u, v \in V$ est égale à la *longueur* minimale (nombre d'arêtes) d'un chemin reliant u et v dans G (l'indice G est omis lorsqu'il n'y a pas d'ambiguïté), et $P_G(u, v)$ désigne un plus court chemin entre u et v. Le diamètre de G est la distance maximale entre ses sommets, c'est-à-dire, $\max_{u,v \in V} dist_G(u, v)$. Étant donné $S \subseteq V$, notons $G[S] = (S, E \cap (S \times S))$ le sous-graphe de G induit par les sommets de S et notons par $G \setminus S$ le sous-graphe $G[V \setminus S]$. Un *sous-graphe* $H = (V(H) \subseteq V, E(H) \subseteq E \cap (V(H) \times V(H)))$ de G est *isométrique* si $dist_H(u, v) = dist_G(u, v)$ pour tout $u, v \in V(H)$, c'est-à-dire, si H *préserve* les distances de G. Notons par $is(G)$ la taille du plus grand cycle isométrique dans G.

Décompositions Arborescentes. Une *décomposition arborescente* d'un graphe $G = (V, E)$ est une paire $D = (T, \mathcal{X} = \{X_t \mid t \in V(T)\})$ telle que T est un arbre, et $\mathcal{X}$ est un ensemble de sous-ensembles (appelés *sacs*) de sommets de G, indexant les nœuds de T tels que :

1. $\bigcup_{t \in V(T)} X_t = V(G)$;
2. Pour toute arête $e \in E(G)$, il existe $t \in V(T)$ tel que $e \subseteq X_t$;
3. pour chaque sommet $v \in V(G)$, l'ensemble $\{t \in V(T) \mid v \in X_t\}$ induit un sous-arbre de T.

La *largeur* $w(D)$ de $D = (T, \mathcal{X})$ est égale à la plus grande taille des sacs de $(T, \mathcal{X})$ moins un, c'est-à-dire, $w(D) = \max_{t \in V(T)} |X_t| - 1$. La *largeur arborescente* $tw(G)$ de G est la largeur minimale des décompositions arborescentes (réduites) de G. La *longueur* $\ell(D)$ de $D = (T, \mathcal{X})$ est égale au diamètre maximum (dans G) de ses sacs, c'est-à-dire, $\ell(D) = \max_{t \in V(T)} \ell(X_t) = \max_{t \in V(T)} \max_{u,v \in X_t} dist_G(u, v)$. La *longueur arborescente* $t\ell(G)$ de G est la longueur minimale des décompositions arborescentes (réduites) de G. Suivant le paramètre considéré, nous dirons qu'une décomposition arborescente de G de longueur $t\ell(G)$ (resp. de largeur $tw(G)$) est dite *optimale*. Une décomposition arborescente est *réduite* si aucun sac n'est contenu dans un autre. Il est facile de vérifier que tout graphe admet une décomposition arborescente réduite optimale.

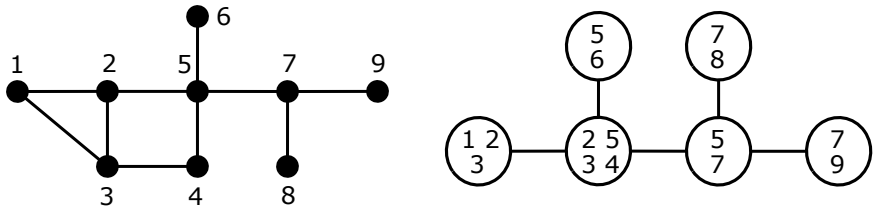


Figure 2.1 – Exemple de décomposition arborescente $(T, \mathcal{X})$ (à droite) de longueur minimum ($t\ell(G) = 2$) de G (à gauche) où le sac $\{2, 3, 4, 5\}$ a longueur 2 et tous les autres sacs ont longueur 1.

Un graphe $G = (V, E)$ est *connexe* si, pour tout $u, v \in V$, il existe un chemin entre u et v dans G. Par le Lemme 1.3.8, nous pouvons nous restreindre à l'étude de la longueur arborescente des graphes connexes. L'ensemble S est une *clique séparatrice* de G si $G - S$ (obtenu à partir de G en enlevant les sommets de S) n'est pas connexe, et si le sous-graphe $G[S]$ induit par S dans G est un graphe complet, c'est-à-dire, une clique. Un graphe G est appelé *premier* s'il n'admet aucune clique séparatrice. Rappelons que le calcul de la largeur (resp. longueur) arborescente peut se restreindre à ses composantes premières (cf. Lemme 1.3.7). Notons que le calcul des composantes

premières de tout graphe planaire peut être effectué en temps linéaire (Coudert & Ducoffe, 2018). Par conséquent, nous ne considérons que des graphes 2-*connexes*, c'est-à-dire, des graphes sans séparateurs de taille un (le Lemme 2.4.6 considère même des graphes sans cliques séparatrices de taille 2, c'est-à-dire, sans arêtes-séparatrices).

Rappelons que, pour tout sous-graphe isométrique H de G, $t\ell(H) \leq t\ell(G)$ (cf. Lemme 1.3.9). Ce résultat sera très souvent utilisé dans la suite de ce chapitre. En particulier, vu que la longueur arborescente d'un cycle contenant n sommets égale $\lceil \frac{n}{3} \rceil$, nous obtenons que $t\ell(G) \geq \lceil \frac{is(G)}{3} \rceil$. Rappelons que la $3^{ème}$ propriété des décompositions arborescente peut être reformulée de deux autres façons différentes (Remarque 1.3.1 et Remarque 1.3.3). L'une d'elle (Remarque 1.3.3) sera utilisée implicitement dans nos preuves. Plus précisément, nous utiliserons souvent l'argument que l'intersection Z de deux sacs X et Y adjacent ($\{X, Y\} = e \in E(T)$) dans la décomposition arborescente $(T, \mathcal{X})$ est un séparateur de $\bigcup_{q \in V(T_1)} X_q \setminus (X \cap X')$ et $\bigcup_{q \in V(T_2)} X_q \setminus (X \cap X')$ s'ils sont non vides, où T_1 et T_2 sont les deux composantes connexes de $T - e$.

Graphes série-parallèles. Rappelons que les graphes série-parallèles sont les graphes pouvant être construits récursivement à partir d'une arête, en réalisant des compositions en parallèle et des compositions en série de deux graphes série-parallèles (cf. Définition 1.3.1).

Notons que, dans n'importe quel graphe série-parallèle G, un plus grand cycle isométrique (et donc $is(G)$) peut être calculé en temps linéaire par un simple algorithme de programmation dynamique (en utilisant une séquence récursive de compositions permettant de construire G pouvant elle-même être obtenue en temps linéaire (Valdes, Tarjan, & Lawler, 1982)).

Décompositions en oreilles. Une *décomposition* en oreilles d'un graphe $G = (V, E)$ est une partition $(E_0, \ldots, E_p)$ de E telle que E_0 induit un cycle dans G et, pour chaque $1 \leq i \leq p$, E_i induit un chemin entre deux sommets a_i et b_i dans G. De plus, $V(E_i) \cap V(G_{i-1}) = \{a_i, b_i\}$ où G_{i-1} est le sous-graphe induit par $\bigcup_{j \leq i-1} V(E_j)$ (c'est-à-dire, le chemin induit par E_i est intérieurement disjoint de $V(E_0), \ldots, V(E_{i-1})$). On dit que a_i et b_i sont les *sommets d'attachement* de E_i dans G_{i-1} (on note que $\{a_i, b_i\}$ est un séparateur de G_i). Il est bien connu qu'un graphe admet une décomposition en oreilles si, et seulement si, il est 2-connexe (Diestel, 2012).

Les oreilles (d'une décomposition en oreilles) sont *imbriquées* si de plus, pour chaque $1 \leq i \leq i' \leq p$:

- les sommets d'attachement a_i et b_i de E_i apparaissent dans une oreille précédente E_j, avec $j < i$, c'est-à-dire, il existe $j < i$ tel que $a_i, b_i \in V(E_j)$, auquel cas, on dit que E_i est *attachée* à E_j. Nous noterons aussi par j_i le plus petit indice $0 \leq j < i$ tel que E_i est attachée à E_j, et
- si deux oreilles E_i et $E_{i'}$ sont toutes deux attachées à une certaine oreille E_j, alors le chemin $P_{E_j}(a_i, b_i)$, entre a_i et b_i dans E_j, contient (pas nécessairement proprement) $P_{E_j}(a_{i'}, b_{i'})$, ou vice versa, ou $P_{E_j}(a_i, b_i)$ et $P_{E_j}(a_{i'}, b_{i'})$ sont intérieurement sommet-disjoints. C'est-à-dire que deux oreilles "ne se croisent pas".

Un graphe est un graphe série-parallèle 2-connexe si et seulement si il admet une décomposition en oreilles imbriquées (Eppstein, 1992). Notons que leur définition de décomposition en oreilles est un peu différente de la nôtre. Nous avons notamment ajouté que la première oreille est un cycle plutôt qu'un chemin, ce qui explique l'ajout de la propriété de 2-connexité. Il est facile de prouver que l'on peut supposer que E_0 est un plus grand cycle isométrique de G et que, pour tout indice $1 \leq i \leq p$, $|E(E_i)| \geq |E(P_{E_{j_i}}(a_i, b_i))|$, c'est-à-dire, G_i est un sous-graphe isométrique de G pour tout $0 \leq i \leq p$. Une décomposition en oreilles imbriquées satisfaisant ces conditions est appelée *isométrique*.

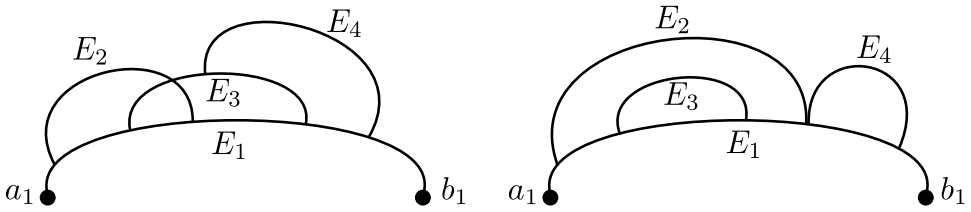


Figure 2.2 – Exemple d'oreilles non imbriquées (à gauche) et imbriquées (à droite).

Lemme 2.1.1. *Pour tout graphe série-parallèle 2-connexe G, une décomposition isométrique en oreilles imbriquées à partir d'un plus grand cycle isométrique de G peut être calculée en temps quadratique.*

Démonstration. Tout d'abord, notons que, par la remarque ci-dessus, un plus grand cycle isométrique C de G (et sa longueur) peut être calculé en temps linéaire. Ensuite, une décomposition isométrique en oreilles imbriquées de G peut être calculée en temps polynomial comme suit :

— étape E_0 : Notons un plus grand cycle isométrique C de G par E_0. Notons le sous-graphe $G[V(E_0)]$ induit par E_0 par G_0.

— étape E_i, pour tout $1 \leq i \leq p$: Soient $C_1, \ldots, C_k$ les k composantes connexes de $G - G_{i-1}$ (calculable en temps linéaire). Pour n'importe quel indice $1 \leq j \leq k$, on considère le sous-graphe C_j^* induit par $V(C_j) \cup V(N_{G_{i-1} \cup C_j}(C_j))$. Notons que $|V(N_{G_{i-1} \cup C_j}(C_j))| = 2$ (sinon G contient une clique K_4 comme mineur) et notons ces deux sommets par $\{a_i, b_i\}$. De plus, C_j^* est un graphe séries-parallèle et donc, il est possible de calculer un arbre de décomposition de C_j^* en temps linéaire. Définissons E_i comme un plus court chemin entre a_i et b_i dans C_j^* et $G_i = G[V(G_{i-1}) \cup V(E_i)]$ (calculable en temps linéaire).

Notons que, puisque nous avons défini E_i comme un plus court chemin dans C^*, il est impossible qu'il existe une oreille $E_{i'}$ attaché à E_i tel que $|E(E_{i'})| < |E(P_{E_i}(a_{i'}, b_{i'}))|$. □

Comme précisé précédemment, ce chapitre sera divisé en trois parties. La première, la sous-section 2.2, consistera en l'étude de la longueur arborescente d'une sous-classe, la plus simple possible, des graphes série-parallèles. Et bien que cette sous-classe soit simple, la caractérisation de sa longueur arborescente n'est pas triviale pour autant. Ensuite, nous donnerons un algorithme d'approximation plutôt simple de ratio $\frac{3}{2}$ dans la section 2.3. Finalement, le résultat principal, un algorithme polynomial décidant si la longueur arborescente d'un graphe série-parallèle est inférieur ou égale à 2, est présenté dans la dernière partie, la section 2.4.

2.2 Dans les graphes série-parallèles, les graphes melons

Cette section est consacrée à la sous-classe la plus simple (incluant les cycles) des graphes série-parallèles 2-connexes, que nous appelons les graphes *melons*. Un graphe *melon* est tout graphe $G = (P_1, \ldots, P_p)$ obtenu à partir de deux sommets x et y en ajoutant $p \geq 2$ chemins intérieurement sommet-disjoints $P_1, \ldots, P_p$ entre x et y. Dans la suite, on considère que $\ell_i = |E(P_i)|$ est la longueur de P_i pour chaque $1 \leq i \leq p$ et, sans perte de généralité, on suppose que $\ell_1 \geq \ldots \geq \ell_p > 0$. Notons qu'un plus grand cycle isométrique de G est constitué de P_1 et de P_p et donc $is(G) = \ell_1 + \ell_p$ et qu'un plus grand cycle est constitué de P_1 et de P_2 et donc contient

$\ell_1 + \ell_2$ sommets et arêtes. Nous noterons la taille d'un plus grand cycle dans G par $lc(G)$. Pour tout graphe melon, $lc(G) = \ell_1 + \ell_2 \geq is(G) = \ell_1 + \ell_p$.

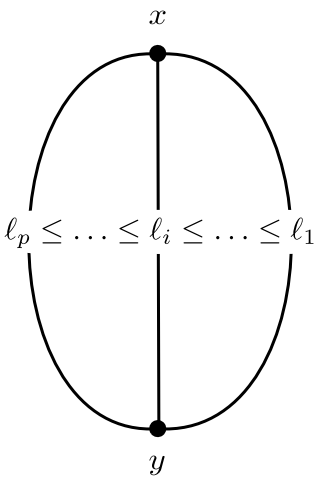


Figure 2.3 – Un graphe melon $G = (P_1, \ldots, P_p)$.

Théorème 2.2.1. *Pour tout graphe melon* $G = (P_1, \ldots, P_p)$, $t\ell(G) = \min\{\lceil \frac{lc(G)}{3} \rceil; \max\{\lceil \frac{is(G)}{3} \rceil, \ell_p\}\}$.

Démonstration. Montrons d'abord les bornes supérieures dans chacun des trois cas : $\ell_p = |E(P_p)| \leq \lceil \frac{is(G)}{3} \rceil$ (dans ce cas, nous cherchons à prouver que $t\ell(G) = \lceil \frac{is(G)}{3} \rceil$); $\lceil \frac{is(G)}{3} \rceil \leq \ell_p \leq \lceil \frac{lc(G)}{3} \rceil$ (auquel cas $t\ell(G) = \ell_p$); et $\lceil \frac{lc(G)}{3} \rceil \leq \ell_p$ (auquel cas $t\ell(G) = \lceil \frac{lc(G)}{3} \rceil$).

— Premièrement, supposons que $\ell_p \leq \lceil \frac{\ell_1+\ell_p}{3} \rceil$. Définissons $I_1 = \{1 \leq i < p \mid \ell_i > \lceil \frac{\ell_1+\ell_p}{3} \rceil - \ell_p\}$ et $I_2 = \{1, \ldots, p-1\} \backslash I_1$. Notons que, pour tout indice $i \in I_2$, $\ell_i = |E(P_i)| \leq \lceil \frac{\ell_1+\ell_p}{3} \rceil$.
Pour tout indice $i \in I_1$, définissons z_i, le sommet de P_i tel que le sous-chemin P'_i de P_i, de x à z_i, ait une longueur $\lceil \frac{\ell_1+\ell_p}{3} \rceil - \ell_p$ et ne passe pas par y (éventuellement $z_i = x$). Le chemin $P_p \cup P'_i$ allant de y à z_i et passant par x a pour longueur $\lceil \frac{is(G)}{3} \rceil$. Pour tout $i \in I_1$, définissons $P''_i = (P_i \backslash P'_i) \cup z_i$ et notons par γ_i le sommet central de P''_i, c'est-à-dire, tel que $d_{P''_i}(\gamma_i, z_i) = \lfloor \frac{|E(P''_i|}{2} \rfloor$. Soit Q_i (resp. Q'_i) le sous chemin de P''_i allant de γ_i à z_i (resp., à y). Notons que $|E(Q_i)| \leq |E(Q'_i)| = \lceil \frac{\ell_i - (\lceil \frac{\ell_1+\ell_p}{3} \rceil - \ell_p)}{2} \rceil = \lceil \frac{\ell_p+\ell_i-\lceil \frac{\ell_1+\ell_p}{3} \rceil}{2} \rceil \leq \lceil \frac{\ell_p+\ell_1-\lceil \frac{\ell_1+\ell_p}{3} \rceil}{2} \rceil = \lceil \frac{\ell_1+\ell_p}{3} \rceil$.
Pour tout $i \in I_2$, définissons $z_i = x$, c'est-à-dire, $V(P'_i) = \{z_i\}$ et $P''_i = P_i$. Q_i et Q'_i sont définis de la même manière que précédemment. Par définition de I_2, $|E(Q_i)| \leq |E(Q'_i)| \leq |E(P_i)| = \ell_i \leq \lceil \frac{\ell_1+\ell_p}{3} \rceil$.
Construisons une décomposition arborescente comme suit. Commençons par un sac $X_0 = V(P_p)$. Pour chaque $1 \leq i < p$, ajoutons un sac $X^1_i = X_0 \cup V(P'_i)$ adjacent à X_0, puis un sac $C_i = \{z_i, \gamma_i, y\}$ adjacent à X^1_i, et finalement, deux sacs $X^2_i = V(Q_i)$ et $X^3_i = V(Q'_i)$ tous deux adjacents à C_i. D'après le paragraphe précèdent, il s'agit d'une décomposition arborescente de longueur $\lceil \frac{\ell_1+\ell_p}{3} \rceil$ (voir la figure 2.4).

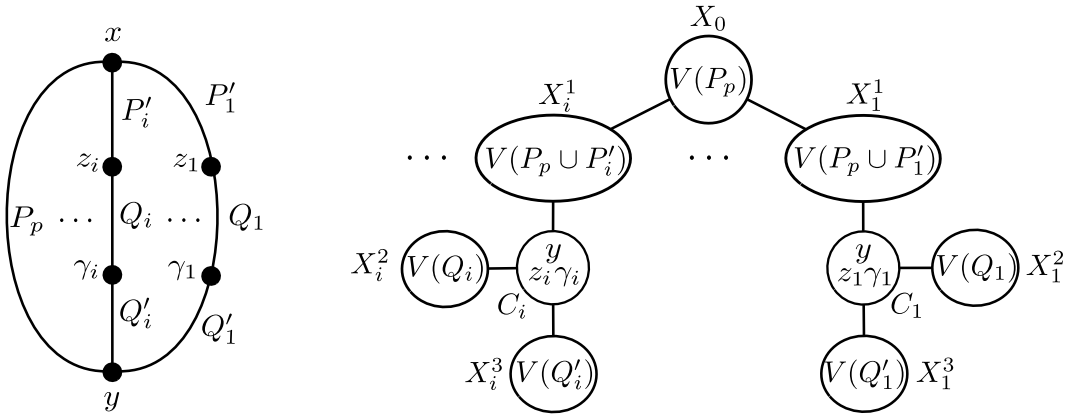


Figure 2.4 – Le graphe $G = (P_1, \ldots, P_p)$ (gauche) et une décomposition arborescente de G de longueur $\lceil \frac{\ell_1+\ell_p}{3} \rceil$ (droite) lorsque $\ell_p \leq \lceil \frac{\ell_1+\ell_p}{3} \rceil$.

— Supposons que $\lceil \frac{\ell_1+\ell_p}{3} \rceil \leq \ell_p \leq \lceil \frac{\ell_1+\ell_2}{3} \rceil$. La décomposition arborescente s'obtient comme dans le cas précédent à la seule différence que $z_i = x$ pour chaque $1 \leq i \leq p$ (c'est-à-dire, $P'_i = \{z_i\}$ pour chaque i). Cette fois-ci, il s'agit d'une décomposition arborescente de G de longueur ℓ_p (voir figure 2.5).

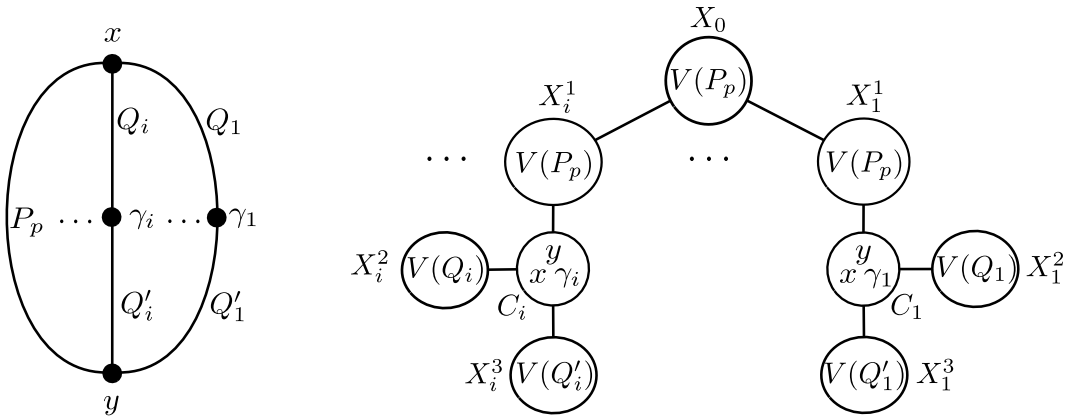


Figure 2.5 – Le graphe $G = (P_1, \ldots, P_p)$ (à gauche) et une décomposition arborescente de G de longueur ℓ_p (à droite) lorsque $\lceil \frac{\ell_1+\ell_p}{3} \rceil \leq \ell_p \leq \lceil \frac{\ell_1+\ell_2}{3} \rceil$.

— Enfin, considérons le cas où $\lceil \frac{\ell_1+\ell_2}{3} \rceil < \ell_p$. Pour chaque $1 \leq i \leq p$, définissons γ_i le sommet de P_i à la distance $\lceil \frac{\ell_1+\ell_2}{3} \rceil < \ell_p$ de x et définissons P'_i comme le plus court chemin de x à γ_i contenu dans P_i. Pour chaque $1 \leq i, j \leq p$, $dist_G(\gamma_i, \gamma_j) \leq \lceil \frac{\ell_1+\ell_2}{3} \rceil$ (via le plus court chemin passant par y). Soit Q le sous-arbre induit par $\{\gamma_1, \ldots, \gamma_p\}$ et la composante connexe de $G - \{\gamma_1, \ldots, \gamma_p\}$ qui contient y.
Construisons une décomposition arborescente comme suit. Commençons par un sac $X_0 = \{x, \gamma_1, \ldots, \gamma_p\}$. Pour chaque $1 \leq i \leq p$, ajoutons un sac $X_i = V(P'_i)$ adjacent à X_0. Enfin, on ajoute un sac $X_{p+1} = V(Q)$ adjacent à X_0. Il s'agit d'une décomposition arborescente de longueur $\lceil \frac{\ell_1+\ell_2}{3} \rceil$ (voir Figure 2.6).

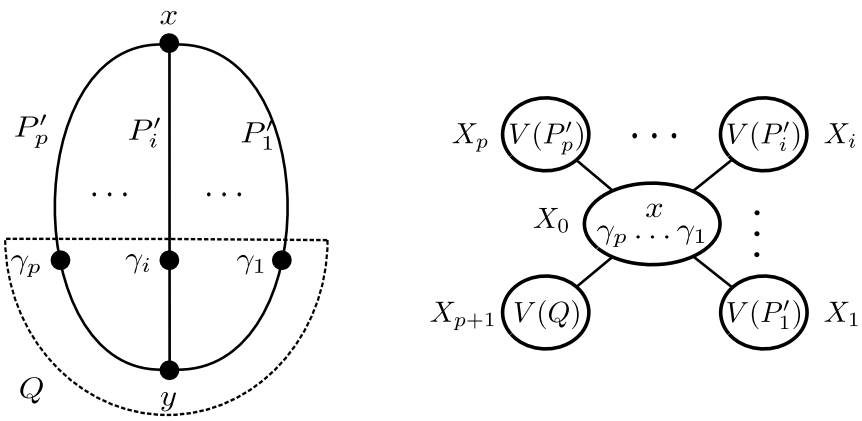


Figure 2.6 – Le graphe $G = (P_1, \ldots, P_p)$ (gauche) et une décomposition arborescente de G de longueur $\lceil \frac{\ell_1+\ell_2}{3} \rceil$ (droite) lorsque $\lceil \frac{\ell_1+\ell_2}{3} \rceil < \ell_p$.

Maintenant, prouvons les bornes inférieures. Par les Lemmes 1.3.9 et 1.3.12, dans tous les cas, $t\ell(G) \geq \lceil \frac{is(G)}{3} \rceil = \lceil \frac{\ell_1+\ell_p}{3} \rceil$. Par conséquent, dans le cas où $\ell_p \leq \lceil \frac{\ell_1+\ell_p}{3} \rceil$, $t\ell(G) = \lceil \frac{\ell_1+\ell_p}{3} \rceil$. Pour conclure la preuve, il reste à prouver que, si $\ell_p > \lceil \frac{\ell_1+\ell_p}{3} \rceil$, alors $t\ell(G) \geq \min\{\ell_p, \lceil \frac{\ell_1+\ell_2}{3} \rceil\}$. Pour arriver à une contradiction, supposons que $t\ell(G) < k$ pour $k = \min\{\ell_p, \lceil \frac{\ell_1+\ell_2}{3} \rceil\}$ et considérons une décomposition arborescente $D = (T, \mathcal{X})$ de G ayant une longueur minimale, c'est-à-dire, une décomposition arborescente optimale. Soit α (resp. β) le sommet à distance k de x sur $P_1 - y$ (resp. sur $P_2 - y$). Notons que α et β sont bien définis puisque soit $\ell_1 \geq \ell_2 \geq \ell_p > \lceil \frac{\ell_1+\ell_2}{3} \rceil \geq k$ ou $\lceil \frac{\ell_1+\ell_2}{3} \rceil \geq \ell_p > \lceil \frac{\ell_1+\ell_p}{3} \rceil$ et donc $\ell_1 \geq \ell_2 > \ell_p \geq k$. Puisque $k = \min\{\ell_p, \lceil \frac{\ell_1+\ell_2}{3} \rceil\}$, $dist_G(\alpha, \beta) \geq k$ et donc, aucun sac de $(T, \mathcal{B})$ ne peut contenir au moins deux sommets de $\{x, \alpha, \beta\}$. Nous pouvons donc définir B_x, B_α et B_β, trois sacs contenant respectivement x, α et β. Il y a plusieurs cas à considérer.

— Premièrement, supposons que B_x est sur le chemin de T entre B_α et B_β. Par conséquent, α et β doivent être dans des composantes connexes différentes de $G - B_x$. Autrement dit, B_x doit contenir un sommet du chemin de α à β passant par y (et non par x). Or, chaque sommet de ce chemin est à une distance supérieure ou égale à k de x, ce qui contredit que B_x est sur le chemin de T entre B_α et B_β.

— Deuxièmement, supposons que B_α est sur le chemin de T entre B_x et B_β. Par conséquent, x et β doivent être dans des composantes connexes différentes de $G - B_\alpha$. Autrement dit, B_α doit contenir un sommet du chemin de x à β ne passant pas par y. Chaque sommet de ce chemin est à une distance supérieure ou égale à k de α, ce qui contredit que B_α est sur le chemin de T entre B_x et B_β. Remarquons que la même affirmation vaut si B_β est entre B_x et B_α, c'est-à-dire, ces deux cas sont symétriques. Donc, B_β ne peut être entre B_x et B_α dans T.

— Enfin, il doit exister un sac B tel que B_x, B_α et B_β sont chacun dans une composante connexe distincte de $T - B$. L'ensemble B doit donc séparer x, α et β. Par conséquent, B doit contenir un sommet dans chacun des trois chemins de x à α (sans passer par y), de x à β (sans passer par y) et de α à β passant par y (et sans passer par x). Puisque le cycle $P_1 \cup P_2$ contenant ces trois sommets a pour longueur au moins $3k \leq \ell_1 + \ell_2$ et $k \leq \ell_p$,

au moins deux de ces trois sommets sont à une distance supérieure ou égale à k, ce qui contredit que $t\ell(G) < k = \min\{\ell_p, \lceil \frac{\ell_1+\ell_2}{3} \rceil\}$.

□

Le résultat ci-dessus (impliquant des graphes série-parallèles 2-connexes contenant au plus 2 sommets de degré supérieur à 2) et la famille suivante de graphes série-parallèles (avec seulement quatre sommets de degré supérieur à 2) nous donnent l'impression que la longueur arborescente des graphes série-parallèles ne peut pas être exprimée par une formule "simple" (comme dans le cas des graphes planaires extérieurs). Soit $p \in \mathbb{N}^*$. Définissons G_p, le graphe obtenu à partir d'un cycle de longueur $12p$ et notons a, b, c, d quatre sommets distincts de celui-ci tels que $dist(a, b) = dist(c, d) = 4p$ et $dist(a, d) = dist(b, c) = 2p$. Notons aussi par f le sommet à distance p de b et c dans $P_G(b, c)$. Ensuite, on ajoute un chemin de longueur $8p$ de a à b et un chemin de longueur $8p$ de c à d ayant respectivement e et g au milieu de ce chemin. Notons que $is(G_p) = 12p$, que le plus grand cycle de G (non isométrique) a une longueur de $20p$, que tous les autres cycles (non isométriques) ont une longueur de $16p$ et que ses chemins maximaux avec des sommets internes de degré 2 ont une longueur de $2p$, $4p$ ou $8p$. Par des arguments similaires à ceux de la preuve précédente, on peut montrer que $t\ell(G_p) = 5p$ ce qui ne semble pas directement lié aux invariants mentionnés précédemment.

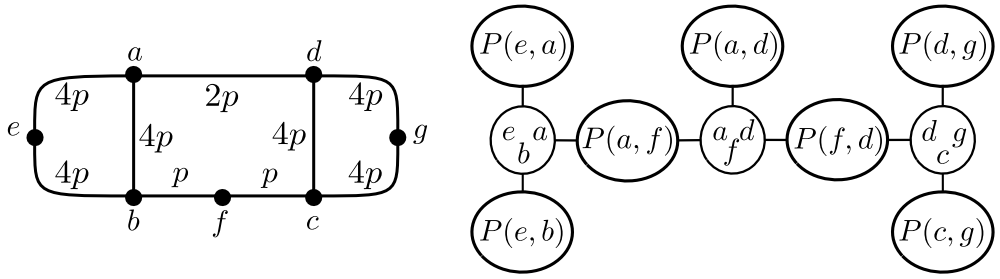


Figure 2.7 – Le graphe G_p (à gauche) et une décomposition arborescente de longueur $5p$ (à droite).

Lemme 2.2.2. *Pour tout $p \in \mathbb{N}^*$, $t\ell(G_p) = 5p$.*

Démonstration. Construisons une décomposition arborescente comme suit (voir la figure 2.7). Commençons par un sac $X_0 = \{a, b, e\}$. Ajoutons deux sacs adjacents à X_0 contenant respectivement le plus court chemin entre a et e, et le plus court chemin entre b et e. Ajoutons ensuite un sac X_1, adjacent à X_0, contenant le plus court chemin entre a et b, et le plus court chemin entre b et f (le diamètre de ce sac est de $5p$). Ajoutons le sac $X_2 = \{a, d, f\}$ adjacent à X_1. Ajoutons un sac, adjacent à X_2, contenant le plus court chemin de a à d. Ajoutons le sac X_3, adjacent à X_2, contenant le plus court chemin de d à f (ce sac a aussi un diamètre de $5p$). Ensuite, on ajoute le sac $X_4 = \{d, c, g\}$ adjacent à X_3. Enfin, on ajoute les sacs, adjacents à X_4, contenant respectivement le plus court chemin entre d et g, et entre c et g. Cette décomposition arborescente a une longueur de $5p$. Pour prouver la borne inférieure, considérons les sommets a, f et g. Dans toute décomposition arborescente de longueur inférieure à $5p$, aucun sac ne peut contenir au moins deux de ces sommets. Soient B_a, B_f et B_g certains sacs contenant respectivement a, f et g dans une telle décomposition (que nous supposons exister pour arriver à une contradiction). Il y a plusieurs cas à considérer.

— Premièrement, supposons que B_f est sur le chemin de T entre B_a et B_g. Par conséquent, a et g doivent être dans des composantes connexes différentes de $G_p - B_f$. Autrement dit, B_f doit contenir un sommet du plus court chemin de a à g passant par d. Or, chaque sommet de ce chemin est à une distance au moins $5p$ de f, ce qui contredit que B_f est sur le chemin de T entre B_a et B_g.
— Deuxièmement, supposons que B_a est sur le chemin de T entre B_f et B_g. Par conséquent, f et g doivent être dans des composantes connexes différentes de $G_p - B_a$. Autrement dit, B_a doit contenir un sommet du plus court chemin de g à f passant par c. Or, chaque sommet de ce chemin est à une distance au moins $5p$ de a, ce qui est une contradiction.
— Ensuite, supposons que B_g est sur le chemin de T entre B_f et B_a. Par conséquent, f et a doivent être dans des composantes connectées différentes de $G_p - B_g$. Autrement dit, B_g doit contenir un sommet du plus court chemin de a à f passant par b. Or, chaque sommet de ce chemin est à une distance au moins $5p$ de g, ce qui contredit que B_g est sur le chemin de T entre B_f et B_a.
— Enfin, il doit exister B_0 tel que B_a, B_g et B_f sont chacun dans des composantes connectées distinctes de $T - B_0$ (car les précédents cas nous ont amené à une contradiction). Par conséquent, l'ensemble B_0 doit séparer a, f et g. Il y a plusieurs cas à considérer.
  — Supposons d'abord que B_0 contient un sommet v du plus court chemin entre b et f (pour séparer a de f). Le sac B_0 doit également contenir un sommet u du plus court chemin entre g et a (contenant d). Si $dist_{G_p}(v,u) < 5p$, alors $dist_{G_p}(a,u) \leq p$. Enfin, B_0 doit contenir un sommet sur le plus court chemin de g à f (en passant par c) qui sont tous à une distance au moins $5p$ de u, ce qui contredit que le sommet v entre a et f (passant par b) contenu dans B_0 est entre b et f.
  — Au contraire, B_0 doit contenir un sommet v du chemin de b à a dans G_p (passant par e). Le sac B_0 doit aussi contenir un sommet u du chemin entre a et g (passant par d) et un sommet w du chemin de f à g (passant par c). Notons que si $dist_{G_p}(u,w) < 5p$, alors $dist_{G_p}(a,u) > p$. Par conséquent, v doit être dans le chemin entre a et e (sinon $dist_{G_p}(u,v) > 5p$). Cela implique que $dist_{G_p}(v,w) > 5p$, ce qui contredit qu'il existe une décomposition arborescente de longueur inférieure à $5p$.

□

2.3 Algorithme d'approximation

Cette section montre que, même si l'on ne sait toujours pas si le calcul de la longueur arborescence des graphes série-parallèles peut être effectué en temps polynomial, il existe un algorithme d'approximation efficace utilisant les décompositions isométriques en oreilles imbriquées.

Théorème 2.3.1. *Pour tout graphe série-parallèle G, une décomposition arborescente de G de longueur au plus $\frac{3}{2} \cdot t\ell(G)$ peut être calculée en temps quadratique.*

Démonstration. D'après le Lemme 1.3.8, il suffit de considérer des graphes 2-connexes. Soit G un graphe série-parallèle 2-connexe. Il découle des Lemmes 1.3.9 et 1.3.12 que $t\ell(G) \geq \frac{is(G)}{3}$. Voyons comment calculer une décomposition arborescente de longueur au plus $\frac{is(G)}{2}$, ce qui nous permettra de conclure notre preuve. Intuitivement, chaque sac sera constitué d'un sous-graphe d'un cycle isométrique, et donc, pour chaque paire de sommets, x et y, dans un sac de la décomposition, ils appartiendront tous deux à un cycle isométrique C, et donc, $dist_G(x,y) =$

$dist_C(x,y) \leq \frac{|E(C)|}{2} \leq \frac{is(G)}{2}$. Considérons une décomposition isométrique en oreilles imbriquées $\mathcal{E} = (E_0, \ldots, E_p)$ commençant par un plus grand cycle isométrique E_0 pour G (elle existe et peut être calculée en temps quadratique par le Lemme 2.1.1). Pour tout $1 \leq i \leq p$, notons par a_i et b_i les extrémités de E_i. Construisons la décomposition comme suit. Commençons par un sac contenant $V(E_0)$. Puis, pour chaque $1 \leq i \leq p$, ajoutons un sac constitué de $V(E_i)$ adjacent au sac contenant $V(E_{j_i})$ où $0 \leq j_i < i$ est l'indice minimal tel que E_{j_i} contient a_i et b_i, les deux extrémités de E_i. Puisque $\mathcal{E}$ est une décomposition isométrique en oreilles imbriquées, G_{i-1} est un sous-graphe isométrique de G, et donc, pour toute paire de sommets $x, y \in V(G_{i-1})$, $d_G(x,y) = d_{G_{i-1}}(x,y)$. Cela implique que le cycle C' induit par $V(E_i)$ et un plus court chemin dans G_{i-1} entre a_i et b_i est un cycle isométrique dans G_{i-1}. De plus, G_i est aussi un sous-graphe isométrique de G (parce que $\mathcal{E}$ est isométrique), ce qui implique que C' est isométrique dans G.

Par conséquent, la longueur de la décomposition arborescente est au plus $\frac{is(G)}{2} \leq \frac{3}{2} \cdot \frac{is(G)}{3} \leq \frac{3}{2} \cdot t\ell(G)$. Puisque $\mathcal{E}$ et $is(G)$ peuvent être calculés en temps polynomial (voir le Lemme 2.1.1), il existe un algorithme d'approximation $\frac{3}{2}$ pour calculer la longueur arborescente d'un graphe série-parallèle. □

2.4 Caractérisation des graphes série-parallèles de longueur arborescente 2

Avant d'énoncer notre théorème principal, un dernier ingrédient est nécessaire, à savoir les graphes *Dumbo*. Un *Graphe Dumbo* est un graphe construit comme suit (voir Figure 2.8). Commencez par un cycle $C_0 = (v_0, \ldots, v_5)$ d'ordre 6, et ajoutez un chemin R de longueur (nombre d'arêtes) au moins 3 et au plus 4 entre v_0 et v_2, et un chemin L de longueur au moins 3 et au plus 4 entre v_3 et v_5. Notons qu'un graphe Dumbo D est un graphe série-parallèle avec $is(D) = 6$.

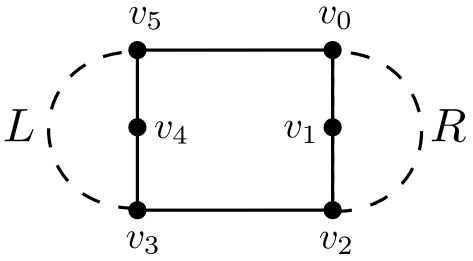


Figure 2.8 – Un graphe Dumbo de longueur arborescente 3 avec deux oreilles L et R telles que $3 \leq |E(L)| \leq 4$ et $3 \leq |E(R)| \leq 4$.

Cette section est consacrée à la preuve du théorème suivant qui s'appuie fortement sur les décompositions isométriques en oreilles imbriquées des graphes série-parallèles.

Théorème 2.4.1. *Pour tout graphe série-parallèle G, $t\ell(G) \leq 2$ si et seulement si $is(G) \leq 6$ et G ne contient pas un graphe Dumbo comme sous-graphe isométrique.*

De plus, il existe un algorithme polynomial qui soit calcule une décomposition arborescente de longueur au plus 2 de G, soit présente un certificat que $t\ell(G) > 2$ (un grand cycle isométrique ou un sous-graphe Dumbo isométrique).

La partie “seulement si" du Théorème 2.4.1 découle des Lemmes 1.3.9 et 1.3.12 ($t\ell(G) \geq \lceil \frac{is(G)}{3} \rceil$) et du Lemme 2.4.3 ($t\ell(G) \geq t\ell(D) \geq 3$ si G contient un graphe Dumbo D comme sous-graphe isométrique) dont la preuve utilise le Lemme 2.4.2.

Lemme 2.4.2. *Soient G un graphe et C un cycle isométrique quelconque de longueur ℓ dans G. Dans toute décomposition arborescente $(T, \mathcal{X})$ de G de longueur au plus $\lceil \frac{\ell}{3} \rceil$, il existe un sac $X \in \mathcal{X}$ contenant trois sommets $a, b, c \in V(C)$ tels que $\lceil \frac{\ell}{3} \rceil = dist(a, b) \geq dist(a, c) \geq \lfloor \frac{\ell}{3} \rfloor$ et $dist(a, c) \geq dist(c, b) \geq \lfloor \frac{\ell}{3} \rfloor - 1$.*

Démonstration. Soit $(T, \mathcal{X})$, n'importe quelle décomposition arborescente de G de longueur au plus $\lceil \frac{\ell}{3} \rceil$. Notons que, par les Lemmes 1.3.9 et 1.3.12, la longueur de $(T, \mathcal{X})$ est égale à $\lceil \frac{\ell}{3} \rceil$. Puisque chaque arête doit apparaître dans un sac, il doit y avoir des sacs contenant au moins deux sommets de C. Pour chaque sac $X \in \mathcal{X}$ avec $|X \cap V(C)| \geq 2$, définissons $d(X) = \max_{u,v \in X \cap V(C)} dist(u, v)$. Notons par X un sac maximisant $d(X)$ et notons par a et b deux sommets de $X \cap V(C)$ tels que $dist(a, b) = d(X)$. Puisque $d(X) \leq \ell(X) \leq \ell(T, \mathcal{X})$, alors $dist(a, b) \leq \lceil \frac{\ell}{3} \rceil$. Soit P, le chemin dans C entre a et b de longueur $\ell - dist(a, b)$, et notons par c le sommet dans P tel que $0 \leq dist(a, c) - dist(b, c) \leq 1$. Par définition, $dist(a, c) \geq dist(b, c) \geq \lfloor \frac{\ell}{3} \rfloor$. Si $c \in X$, alors, par les hypothèses précédentes, $dist(a, b) \geq dist(a, c) \geq dist(b, c)$. Remarquons également que, $dist(a, b) \geq \lceil \frac{\ell}{3} \rceil$ (resp. $dist(b, c) \leq \lfloor \frac{\ell}{3} \rfloor$), sinon il y a une contradiction avec la longueur de C, c'est-à-dire, $dist(a, b) + dist(a, c) + dist(b, c) < \ell$ (resp. $dist(a, b) + dist(a, c) + dist(b, c) > \ell$). Ainsi, $\lceil \frac{\ell}{3} \rceil = dist(a, b) \geq dist(a, c) \geq dist(b, c) = \lfloor \frac{\ell}{3} \rfloor$ et donc, a, b, c et X sont les sommets et le sac satisfaisant l'énoncé du lemme.

Pour arriver à une contradiction, supposons qu'aucun sac ne contient a, b et c. Notons donc par Y, le sac contenant c qui est le plus proche de X dans T, et par X' le sac contenant a et b qui est le plus proche de Y. Dans le cas où X' et Y sont adjacents dans T, notons par $Z = X' \cap Y$ leur intersection. Dans le cas contraire, notons par Z, l'intersection entre X' et X'', où X'' est le voisin de X' dans le chemin entre X' et Y dans T. Notons que $c \notin Z$ et qu'au moins un des deux sommets a et b n'est pas dans Z (sinon, cela contredirait soit le fait que X' est le plus proche de Y, soit que Y est le plus proche de X, ou qu'aucun sac ne contient les sommets a, b et c).

Supposons en premier que b n'est pas dans Z. Puisque $(T, \mathcal{X})$ est une décomposition arborescente et que Z est l'intersection de deux sacs adjacents dans T, Z doit séparer b et c. Par conséquent, il existe un sommet u entre b et c dans P qui appartient à Z. Puisque $dist(a, u) \leq dist(a, b) = d(X) = d(X')$ par maximalité de $d(X)$, nous avons $dist(a, u) = dist(a, c) + dist(c, u)$, c'est-à-dire, le chemin entre a et u passe par c. Nous pouvons en déduire que $dist(a, c) < \lceil \frac{\ell}{3} \rceil$, sinon $dist(a, u) \geq \lceil \frac{\ell}{3} \rceil + 1$, ce qui contredit la longueur de $(T, \mathcal{X})$. Par conséquent, $dist(a, c) = dist(a, b) = \lfloor \frac{\ell}{3} \rfloor$, et donc, $\ell \equiv 1 \pmod 3$. La seule valeur possible pour $dist(c, u)$, afin que $(T, \mathcal{X})$ soit une décomposition de longueur $\lceil \frac{\ell}{3} \rceil$, est donc 1, c'est-à-dire, $dist(a, u) = \lfloor \frac{\ell}{3} \rfloor + 1 = \lceil \frac{\ell}{3} \rceil$ and $dist(b, u) = \lfloor \frac{\ell}{3} \rfloor - 1$. a, b, u and X' sont donc des sommets et un sac satisfaisant l'énoncé du lemme.

Il nous reste donc à considérer le cas où a n'est pas dans Z. Puisque $(T, \mathcal{X})$ est une décomposition arborescente et que Z est l'intersection de deux sacs adjacents dans T, Z doit séparer a et c. Par conséquent, il existe un sommet u entre a et c dans P qui appartient à Z. Puisque $dist(b, u) \leq dist(a, b) = d(X) = d(X')$ par maximalité de $d(X)$, nous avons $dist(b, u) = dist(b, c) + dist(b, u)$, c'est-à-dire, le chemin entre a et u passe par c. Nous avons donc que $dist(u, b) > dist(b, c) = \lfloor \frac{\ell}{3} \rfloor$, ce qui implique que $dist(u, b) = \lceil \frac{\ell}{3} \rceil$ puisque $dist(u, b) \leq d(X') \leq d(X) \leq \lceil \frac{\ell}{3} \rceil$. Autrement dit, $dist(u, c) = 1$ et donc $dist(u, b) = dist(b, c) + 1$ et

$dist(a,u) = dist(a,c) - 1$. Puisque $\lceil \frac{\ell}{3} \rceil \geq dist(a,c) \geq \lfloor \frac{\ell}{3} \rfloor$, $\lfloor \frac{\ell}{3} \rfloor \geq dist(a,u) \geq \lfloor \frac{\ell}{3} \rfloor - 1$. a, b, u et X' sont donc les sommets et le sac satisfaisant l'énoncé du lemme. □

Lemme 2.4.3. *Soit G un graphe connexe. Si G est un graphe Dumbo, alors $t\ell(D) = 3$.*

Démonstration. Notons tout d'abord que la décomposition arborescente composée de trois sacs, X_1 adjacent à X_2, lui-même adjacent à X_3, tels que X_1 contient le chemin L, X_2 contient le cycle C et X_3 contient le chemin R, est une décomposition arborescente de G de longueur 3. Par conséquent, pour prouver que $t\ell(G) = 3$, il ne reste qu'à montrer que $t\ell(G) > 2$, c'est-à-dire, il n'y a pas de décomposition arborescente de G avec une longueur au plus 2.

Pour arriver à une contradiction, supposons que G est un graphe Dumbo admettant une décomposition arborescente $(T, \mathcal{X})$ de longueur au plus 2. Par le Lemme 2.4.2, il doit exister un sac $X \in \mathcal{X}$ contenant $\{v_0, v_2, v_4\}$ ou $\{v_1, v_3, v_5\}$. Par symétrie, supposons que $\{v_0, v_2, v_4\} \subseteq X$. Soit z un sommet de $L - \{v_5, v_3\}$ tel que $|dist(z, v_5) - dist(z, v_3)| \leq 1$. Notons que $dist(z, v_5), dist(z, v_3) \geq 1$ et $\max\{dist(z, v_5), dist(z, v_3)\} \geq 2$. De plus, tout chemin de z vers v_0, v_2 ou v_4 passe par v_3 ou v_5. Notons également qu'aucun sac ne contient $\{v_0, v_2, v_4, z\}$ puisque z est à une distance au moins 3 de certains sommets de v_0, v_2, v_4.

Notons par Y, le sac contenant z le plus proche de X, et notons X' le sac contenant v_0, v_2, v_4 le plus proche de Y. Si $X'Y \in E$, notons par Z l'intersection entre X' et Y, sinon, notons par Z' le voisin de X' sur le chemin entre X' et Y dans T, et Z comme l'intersection de Z' et X'. Notons que $z \notin Z$ et qu'au moins un des v_0, v_2 et v_4 n'est pas dans Z (sinon, cela contredirait soit le fait que X' est le plus proche de Y, que Y est le plus proche de X ou qu'aucun sac ne contient tous les sommets v_0, v_2, v_4 et z). Soit $W = \{v_0, v_2, v_4\} \setminus Z$. Puisque Z est l'intersection de deux sacs adjacents de T, Z doit séparer chaque $w \in W$ de z. Rappelons que $Z \cap \{v_1, v_3, v_5\} = \emptyset$, sinon, $X' \cap \{v_1, v_3, v_5\} \neq \emptyset$, contredisant la longueur de la décomposition arborescente T (c'est-à-dire, il existe deux sommets à distance 3 tous les deux contenus dans X'). Il y a plusieurs cas à considérer selon les sommets de v_0, v_2 et v_4 qui ne sont pas dans Z :

- Si v_2 appartient à Z, alors $W \subseteq \{v_0, v_4\}$. Par conséquent, il doit exister un sommet u dans Z tel que u appartient au sous chemin entre z et v_5 de L. Puisque $z, v_5 \notin Z$, $u \neq z$ et $u \neq v_5$, et donc, $dist(u, v_3) \geq 2$. Cela implique que $dist(u, v_2) \geq 3$, contredisant la longueur de la décomposition arborescente T. Nous pouvons donc supposer que v_2 n'appartient pas à Z.
- Si v_0 appartient à Z, alors $W \subseteq \{v_2, v_4\}$. Par conséquent, il doit exister un sommet v dans Z tel que v appartient au sous chemin entre z et v_3 de L. Puisque $z, v_3 \notin Z$, $v \neq z$ et $v \neq v_3$, et donc, $dist(v, v_5) \geq 2$. Cela implique que $dist(v, v_0) \geq 3$, contredisant la longueur de la décomposition arborescente T. Nous pouvons donc supposer que v_0 n'appartient pas à Z.
- Enfin, puisque $v_0, v_2 \notin Z$, $v_0, v_2 \in W$. Donc, il doit y avoir u dans le sous chemin z-v_5 de L qui est dans Z et il doit y avoir v dans le sous chemin z-v_3 de L qui est dans Z. Puisque $z, v_5 \notin Z$, $v, u \notin \{z, v_5\}$, ce qui implique que $dist(v, v_0) \geq 3$ et $dist(u, v_2) \geq 3$. Rappelons que Z est l'intersection de X' avec un autre sac et que X' contient $\{v_0, v_2, v_4\}$. X' contient donc deux sommets à distance 3, une contradiction avec la longueur de la décomposition arborescente T. Par conséquent, il n'existe pas de décomposition arborescente de longueur 2 pour G et donc $t\ell(G) > 2$.

□

Notons que le lemme précédent implique qu'un graphe de longueur arborescente au plus 2 ne peut contenir un graphe Dumbo comme sous graphe isométrique grâce au Lemme 1.3.9 :

Corollaire 2.4.4. *Pour tout graphe G, si G contient un graphe Dumbo comme sous-graphe isométrique, alors $t\ell(G) > 2$.*

La partie “si" du Théorème 2.4.1 découle du Lemme 2.4.6 dont la preuve décrit l'algorithme. Il calcule une décomposition arborescente, d'un graphe simple série-parallèle premier G, de longueur 2 ou retourne un certificat que $t\ell(G) > 2$, c'est-à-dire soit un graphe Dumbo, soit un cycle d'ordre au moins 7, contenu comme un sous-graphe isométrique dans G. Le lemme 2.4.5 sera utilisé dans la preuve du lemme 2.4.6 pour traiter le cas des oreilles de longueur 2.

Lemme 2.4.5. *Soient G un graphe série-parallèle 2-connexes quelconque sans clique-séparatrices et $\mathcal{E} = (E_i)_{0\leq i\leq p}$ une décomposition isométrique en oreilles imbriquées de G. Soit $(T', \mathcal{X}')$ une décomposition arborescente, de longueur au moins égale à 2, du sous-graphe G_j de G induit par $E_0, \ldots, E_j$. Soit E_i tel que $1 \leq j_i \leq j < i \leq p$ (où j_i est le plus petit indice tel que E_{j_i} contient les deux extrémités de E_i) et $|E_i| = 2$, c'est-à-dire, E_i est une oreille de longueur 2 qui n'est pas dans G_j, mais dont les deux extrémités sont dans G_j. Alors, il existe une décomposition arborescente $(T, \mathcal{X})$ de $G_j \cup E_i$ avec la même longueur et telle que, pour chaque $B' \in \mathcal{X}'$, il existe $B \in \mathcal{X}$ tel que $B' \subseteq B$.*

Démonstration. Remarquons que, par hypothèse, les deux extrémités de E_i appartiennent à G_j puisqu'elles appartiennent à E_{j_i}. Supposons d'abord que les extrémités de E_i sont dans un même sac B de $(T', \mathcal{X}')$. Alors, la décomposition arborescente obtenue à partir de $(T', \mathcal{X}')$ en ajoutant un sac $V(E_i)$ adjacent à B satisfait l'énoncé du lemme.

Considérons maintenant le cas où aucun sac de $(T', \mathcal{X}')$ ne contient les deux extrémités a_i et b_i de E_i. Soient $X \in \mathcal{X}'$ et $Y \in \mathcal{X}'$ tels que $a_i \in X$, $b_i \in Y$ et la distance dans T entre deux tels sacs est minimale.

Notons que, puisque G n'a pas d'arête-séparatrice et puisque les oreilles sont ajoutées dans l'ordre isométrique (c'est-à-dire, $2 \leq |E(P_{E_{j_i}}(a_i, b_i)| \leq |E(E_i)| = 2$), a_i et b_i doivent avoir des voisins communs dans G_j. Notons également que, puisque G est série-parallèle (en particulier, les oreilles sont imbriquées) sans clique-séparatrices (c'est-à-dire, il n'y a pas d'oreille attachée à deux sommets adjacents), alors chaque voisin commun w de a_i et b_i satisfait $N(w) = \{a_i, b_i\}$.

Puisque $(T', \mathcal{X}')$ est une décomposition arborescente, chaque sac W sur le chemin X-Y dans T' doit séparer $X \setminus Y$ de $Y \setminus X$. En particulier, $N_{G_j}(a_i) \cap N_{G_j}(b_i) \subseteq W$. De même, $N_{G_j}(a_i) \cap N_{G_j}(b_i) \subseteq X$ et $N_{G_j}(a_i) \cap N_{G_j}(b_i) \subseteq Y$. Soit v le voisin commun de a_i et b_i dans E_i. Alors, l'ajout de v dans chaque sac W sur le chemin X-Y dans T' (incluant X et Y) donne la décomposition désirée. En particulier, pour chaque $v' \in W$, $dist_G(v', w) = dist_G(v', v)$ où w est un sommet quelconque dans $N_{G_j}(a_i) \cap N_{G_j}(b_i)$ (car, $N(w) = N(v)$), et donc la décomposition arborescente obtenue a la même longueur que $(T', \mathcal{X}')$. □

Quelques notations sont encore nécessaires. Soit G un graphe série-parallèle 2-connexe avec une décomposition isométrique en oreilles imbriquées $\mathcal{E} = (E_i)_{0\leq i\leq p}$ telle que E_0 est un plus grand cycle isométrique de G. Rappelons que a_i et b_i désignent les extrémités de E_i ($a_i, b_i \in V(G_{i-1})$). Notons $\ell_i = |E(E_i)|$ et $d_i = dist_{G_{i-1}}(a_i, b_i)$. Puisque $\mathcal{E}$ est isométrique, $d_i \leq \ell_i$ pour tout $1 \leq i \leq p$. Enfin, pour tout sous-graphe H de G induit par $\bigcup_{i'\leq j\leq i} V(E_j)$, notons $Att(H) \subseteq V(H)$ l'ensemble des sommets de H qui sont les sommets d'attachement (a_k et b_k) d'une certaine oreille E_k avec $k > i$.

Lemme 2.4.6. *Soit G un graphe série-parallèle (simple) premier quelconque tel que $is(G) \leq 6$. Si G ne contient pas de graphe de Dumbo comme sous-graphe isométrique, alors $t\ell(G) \leq 2$.*

Démonstration. Supposons que G n'est pas un graphe cordal, auquel cas le résultat est trivial (rappelons que $t\ell(G) = 1$ si et seulement si G est cordal, ce qui peut être décidé en temps linéaire). Par conséquent, nous pouvons supposer que $t\ell(G) \geq 2$.

Soit G n'importe quel graphe série-parallèle premier (c'est-à-dire, sans clique-séparatrice), avec $is(G) \leq 6$, et sans graphe de Dumbo comme sous-graphe isométrique. Soit $\mathcal{E} = (E_i)_{0 \leq i \leq p}$ une décomposition isométrique en oreilles imbriquées de G où E_0 est un plus grand cycle isométrique de G. Notons que $\mathcal{E}$ ne contient aucune oreille de longueur un puisque G est simple.

Nous allons construire une suite $\mathcal{E}_1 \subset \mathcal{E}_2 \subset \ldots \subset \mathcal{E}_{p'} = \mathcal{E}$ tel que $E_0 \in \mathcal{E}_1$ et, pour chaque $1 \leq i \leq p'$,

1. $G_i = G[\bigcup_{E' \in \mathcal{E}_i} V(E')]$ est un sous-graphe isométrique série-parallèle de G avec $\mathcal{E}_i$ comme décomposition en oreilles ;
2. Il n'y a aucune oreille de longueur deux attachée à G_i, c'est-à-dire, chaque oreille de $\mathcal{E}$ pas encore dans G_i avec les deux extrémités dans G_i a une longueur d'au moins 3 ;
3. G_i admet une décomposition arborescente $(T^i, \mathcal{X}^i)$ de longueur 2 ;
4. Pour toutes oreilles $E_j \in \mathcal{E} \setminus \mathcal{E}_i$ attaché à G_i, il existe $t \in V(T^i)$ tel que $\{a_j, b_j\} \subseteq X^i_t \in \mathcal{X}^i$, c'est-à-dire, chaque oreille qui n'est pas encore dans G_i et dont les deux extrémités sont dans G_i (donc de longueur au moins égale à 3) a ses deux extrémités dans un sac quelconque de $(T^i, \mathcal{X}^i)$.

La preuve se fait par induction sur $1 \leq i \leq p'$. Le cas de base consiste à construire $\mathcal{E}_1$. Commençons par construire $\mathcal{E}_1$. Il existe plusieurs cas selon la taille ℓ_0 de E_0. Notons que $\ell_0 > 3$, car sinon, G est cordal. Par conséquent, $4 \leq \ell_0 \leq 6$.

— Supposons en premier que $E_0 = (a, b, c, d)$ a une longueur de 4. Rappelons que puisque G est premier, pour tout $E_j \in \mathcal{E}$, $d_j > 1$ et $\ell_j > 1$. De plus, puisque les oreilles de $\mathcal{E}$ sont imbriquées, pour deux oreilles quelconques E_q et E'_q dans $\mathcal{E}$, soit $P_{E_{j_q}}(a_q, b_q) \subseteq P_{E_{j_{q'}}}(a_{q'}, b_{q'})$, soit $P_{E_{j_{q'}}}(a_{q'}, b_{q'}) \subseteq P_{E_{j_q}}(a_q, b_q)$, soit elles sont disjointes. Ainsi, si on suppose qu'il existe une oreille dans $\mathcal{E} \setminus E_0$ ayant a et c comme sommets d'attachement, alors pour n'importe quelle autre oreille dans $\mathcal{E} \setminus E_0$ attachée à E_0, a et c sont ses sommets d'attachement. Par conséquent, par symétrie, $Att(E_0) = \{a, c\}$ (si $Att(E_0) = \emptyset$, alors $G = E_0$ et le résultat est trivial).

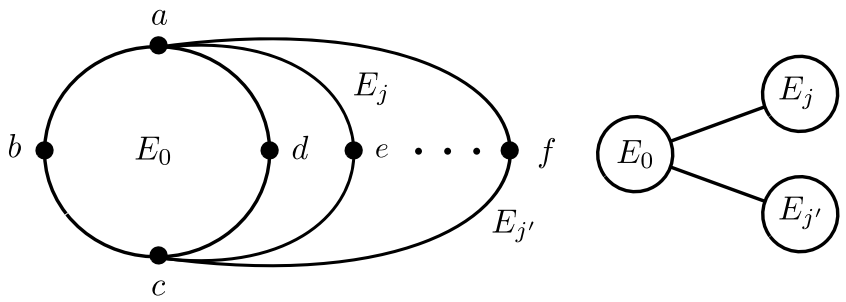


Figure 2.9 – Graphe (gauche) et décomposition arborescente (droite) avec $\ell_0 = 4$.

Définissons $\mathcal{E}_1$ par E_0 et l'ensemble de toutes les oreilles de longueur deux attachées à a et c. Alors, $(T^1, \mathcal{X}^1)$ est la décomposition arborescente avec un sac "central $\{a, b, c, d\}$, avec un sac voisin E_j pour chaque oreille $E_j \in \mathcal{E}_1 \setminus \{E_0\}$ (voir figure 2.9). Clairement, $(T^1, \mathcal{X}^1)$ est une décomposition arborescente de G^1 avec une longueur de 2. Enfin, comme

les oreilles sont imbriquées et qu'il n'y a pas de clique-séparatrice, chaque oreille de $\mathcal{E}\backslash\mathcal{E}_1$ avec des sommets de rattachement dans G_1 doit avoir a et c comme sommets d'attachement. Si une telle oreille dans $\mathcal{E}\backslash\mathcal{E}_1$ existe, alors sa longueur est d'au moins 3 ce qui contredirait le fait que E_0 est un plus grand cycle isométrique. Par conséquent, une telle oreille n'existe pas et $G_1 = G$.

— Si $E_0 = (a, b, c, d, e)$ a une longueur de 5 alors, par symétrie, $Att(E_0) \subseteq \{a, c, d\}$ (voir Figure 2.10). En effet, si $Att(E_0) = \emptyset$, alors $G = E_0$ et le résultat est trivial. Sinon, puisque G ne contient aucune clique-séparatrice, toutes les oreilles attachées à E_0 sont attachées respectivement à deux sommets à distance 2. De plus, comme les oreilles sont imbriquées, et par symétrie, soit les oreilles sont attachées à a et c ou à a et d. Notons que toutes ces oreilles ont une taille inférieure à 4, sinon E_0 n'est pas un plus grand cycle isométrique.

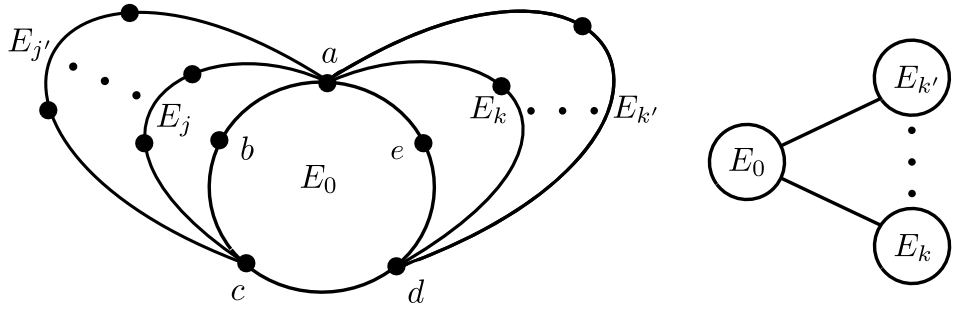


Figure 2.10 – Graphe (gauche) et décomposition arborescente (droite) avec $\ell_0 = 5$ (E_k et $E_{k'}$ sont contenus dans un sac puisqu'ils ont une longueur de 2. E_j et $E_{j'}$ ne sont pas contenus dans un sac puisqu'ils ont une longueur 3).

Définissons $\mathcal{E}_1$ par E_0 et l'ensemble de toutes les oreilles de longueur deux attachées à E_0. Alors, $(T^1, \mathcal{X}^1)$ est la décomposition arborescente avec un sac central $\{a, b, c, d, e\}$, avec un sac voisin E_j pour chaque oreille $E_j \in \mathcal{E}_1 \setminus \{E_0\}$ (voir figure 2.10). Clairement, $(T^1, \mathcal{X}^1)$ est une décomposition arborescente de G^1 avec une longueur de 2. Enfin, chaque oreille dans $\mathcal{E} \setminus \mathcal{E}_1$ attachée à G_1 a ses sommets d'attachement dans E_0, car $\mathcal{E}$ est une décomposition en oreilles imbriquées et G est premier. Plus précisément, dans le cas contraire, puisqu'une oreille ne peut pas avoir de sommets d'attachement adjacents (pas de clique-séparatrice), il y aurait une oreille $E_j \in \mathcal{E} \setminus \mathcal{E}_1$ et une oreille $E_k \in \mathcal{E}_1 \setminus \{E_0\}$ (disons avec des sommets d'attachement a et c) avec $a_j \in E_k \setminus \{a, c\}$ et $b_j \notin \{a, c\}$. Cela impliquerait que G contient un K_4 comme mineur, une contradiction. Donc, pour toutes oreilles dans $\mathcal{E} \setminus \mathcal{E}_1$, ses sommets d'attachement sont dans $\{a, c, d\}$ et elles ont une longueur d'au moins 3, sinon elles sont déjà dans $\mathcal{E}_1$. Les hypothèses d'induction sont donc satisfaites.

— Considérons ensuite le cas où $E_0 = (a, b, c, d, e, f)$ a une longueur de 6. S'il existe une oreille attachée à deux sommets à distance de 3 l'un de l'autre, alors cette oreille a une longueur de 3 puisque E_0 est un plus grand cycle isométrique. De plus, toutes ces oreilles ont les mêmes sommets d'attachement puisque les oreilles sont imbriquées (sinon, il y aurait un mineur de K_4).
Sans perte de généralité, supposons que a et d sont les sommets d'attachement de toutes les oreilles (s'il y en a) attachées à deux sommets à une distance de 3 dans E_0. Définissons

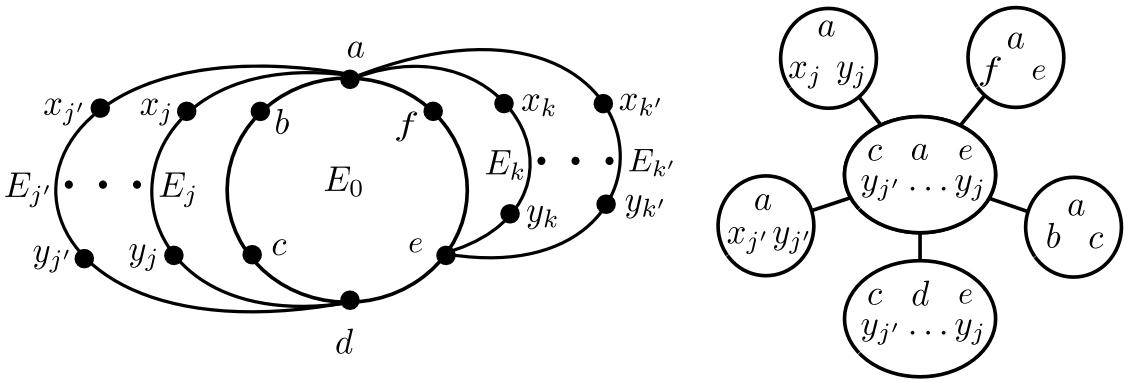


Figure 2.11 – Graphe (gauche) et décomposition arborescente (droite) avec $\ell_0 = 6$.

$\mathcal{E}'_1$ comme étant composé de E_0 et de toutes les oreilles $E_j = (a_j = a, x_j, y_j, b_j = d)$ attachées à a et d. Soit X une oreille de $\mathcal{E} \setminus \mathcal{E}'_1$ avec une longueur au moins 3 et telle que ses sommets d'attachement, x et y, sont dans $G' = G[\bigcup_{E' \in \mathcal{E}'_1} V(E')]$. Rappelons que x et y ne peuvent pas être adjacent puisque G ne contient aucune clique-séparatrice. De plus, si $\mathcal{E}'_1 \neq \{E_0\}$, alors $\{x, y\} \cap \{a, d\} \neq \emptyset$ puisque les oreilles de $\mathcal{E}$ sont imbriquées. Par symétrie, nous pouvons donc supposer que $x = a$. Dans ce cas, $y \neq d$ (sinon $X \in \mathcal{E}'_1$). De ce fait, et puisque G n'a pas de clique-séparatrice, $y \in \{c, e\} \cup \bigcup_{E_j \in \mathcal{E}'_1} \{y_j\}$. Puisque X existe, montrons qu'il n'y a pas d'autre oreille de longueur 3 attaché à G' avec des sommets différents de ceux préciser pour l'oreille X. Dans ce but, définissons X' comme n'importe quel autre oreille de $\mathcal{E} \setminus \mathcal{E}'_1$ de taille 4 et telle que X' est attaché à G', c'est-à-dire, ses sommets d'attachements x' et y' sont dans G'. Pour arriver à une contradiction, supposons que $x' = d$. Par les mêmes arguments (par définition de $\mathcal{E}'_1$ et parce que G est premier), nous pouvons prouver que $y' \in \{b, f\} \cup \bigcup_{E_j \in \mathcal{E}'_1} \{x_j\}$. Dans ce cas, soit X et X' ne sont pas imbriquées (si elles sont attachées à la même oreille de G'), ou X, X', et le cycle contenant x, y, x' et y' induisent un graphe Dumbo contenu comme sous graphe isométrique de G, une contradiction. Pour conclure notre analyse des sommets d'attachement, par symétrie, toutes les oreilles de taille 3 qui sont attachées à G' ont a et un sommet de $B = \{c, e\} \cup \bigcup_{E_j \in \mathcal{E}'_1} \{y_j\}$ comme sommets d'attachement (voir Figure 2.11). Soit $(T', \mathcal{X}')$ la décomposition arborescente avec un sac "central" $C = B \cup \{a\}$ avec un sac voisin $\{a, x_j, y_j\}$ pour chaque oreille $E_j \in \mathcal{E}'_1 \setminus \{E_0\}$, un sac voisin $\{a, b, c\}$, un sac voisin $\{a, f, e\}$ et un sac voisin $\{d\} \cup B$. Alors, $(T', \mathcal{X}')$ est clairement une décomposition arborescente de G' de longueur 2 telle que toutes les oreilles de longueur au moins 3 attachées à G' ont leurs sommets d'attachement dans C. Soit F, l'ensemble de toutes les oreilles de longueur 2 attachées à G'. Soit $\mathcal{E}_1 = \mathcal{E}'_1 \cup F$. Par le Lemme 2.4.5, à partir de $(T', \mathcal{X}')$, on peut obtenir une décomposition arborescente $(T^1, \mathcal{X}^1)$ de G_1 de longueur 2 telle que chaque sac dans $\mathcal{X}'$ est contenu dans un sac de $\mathcal{X}^1$ (voir Figure 2.11).
Notons que pour chaque oreille de $\mathcal{E} \setminus \mathcal{E}_1$ qui est attachée à G^1, elle est attachée soit à G', soit à une oreille de F puisque $\mathcal{E}$ est imbriqué. Si une oreille de $\mathcal{E} \setminus \mathcal{E}_1$ est attachée à F, alors elle est attachée à deux sommets adjacents, ce qui contredit que G est premier. Ainsi, chaque oreille dans $\mathcal{E} \setminus \mathcal{E}_1$ qui est attachée à G_1, est alors attachée à G', et donc par construction de $(T', \mathcal{X}')$, il existe au moins un sac (le sac C) qui contient ses extrémités.

Maintenant, prouvons par induction sur $1 \leq i < p'$ que nous pouvons construire une décomposition d'oreille $\mathcal{E}_{i+1}$ à partir de $\mathcal{E}_i$ avec toutes les propriétés désirées. Soit E_j, l'oreille la plus courte qui n'est pas dans $\mathcal{E}_i$ avec des sommets d'attachement $\{a_j, b_j\} \in V(G_i)$. Puisque G n'a pas de clique-séparatrices et que, par hypothèse d'induction, G_i a une décomposition arborescente $(T^i, \mathcal{X}^i)$ de longueur 2 avec un sac contenant a_j et b_j, on note que $d_j = dist_G(a_j, b_j) = dist_{G_i}(a_j, b_j) = 2$. De plus, comme $is(G) = 6$ et qu'il n'y a pas d'oreille de longueur 2 attachée à G_i, la longueur ℓ_j de E_j est telle que $3 \leq \ell_j \leq 4$. Il existe donc deux cas selon la longueur de E_j.

— Si $E_j = (a_j, x, y, b_j)$ a une longueur de 3, alors, par symétrie, $Att(G_i \cup E_j) \cap V(E_j) \subseteq \{a_j, y, b_j\}$. En effet, puisque G n'a pas de clique-séparatrices, aucune oreille ne peut être attachée à deux sommets adjacents. De plus, puisque toutes les oreilles de $\mathcal{E}$ sont imbriquées, premièrement, il n'existe pas deux oreilles, l'une attachée à a_j et y et l'autre attachée à x et b_j, et deuxièmement, il n'y a pas d'oreille attachée à un sommet de $V(E_j)\backslash\{a_j, b_j\}$ et à un sommet de $V(G_i)\backslash\{a_j, b_j\}$ (voir figure 2.12). Soit $\mathcal{E}'_{i+1}$ l'ensemble composé des oreilles de $\mathcal{E}_i$ et de E_j. Soient $G' = G[\bigcup_{E' \in \mathcal{E}'_{i+1}} V(E')]$ et $(T', \mathcal{X}')$ la décomposition arborescente construite à partir de $(T^i, \mathcal{X}^i)$ avec un sac $B = \{a_j, x, y, b_j\}$ connecté à un sac de $(T^i, \mathcal{X}^i)$ contenant a_j et b_j. Alors, $(T', \mathcal{X}')$ est clairement une décomposition arborescente de G' de longueur 2. Soit F l'ensemble de toutes les oreilles de longueur 2 attachées à G' (notons que, en raison de l'hypothèse d'induction et du fait que la décomposition initiale de l'oreille est isométrique, toutes ces oreilles sont attachées à a_j et y). Soit $\mathcal{E}_{i+1} = \mathcal{E}'_{i+1} \cup F$. Par le Lemme 2.4.5, à partir de $(T', \mathcal{X}')$, on peut obtenir une décomposition arborescente $(T^{i+1}, \mathcal{X}^{i+1})$ de G_{i+1} de longueur 2 telle que chaque sac de $\mathcal{X}'$ est contenu dans un sac de $\mathcal{X}^{i+1}$ (voir Figure 2.12). Clairement, s'il y a une oreille attachée au sommet de degré 2 d'une oreille E_f de F, alors par définition d'une décomposition en oreilles imbriquées, son deuxième point d'extrémité est un sommet dans E_f, ce qui contredit le fait que G n'a pas de clique-séparatrice. Nous pouvons déduire que pour chaque E_m attaché à G_{i+1} il existe $t \in V(T^{i+1})$ tel que $\{a_m, b_m\} \subseteq X_t^{i+1}$.

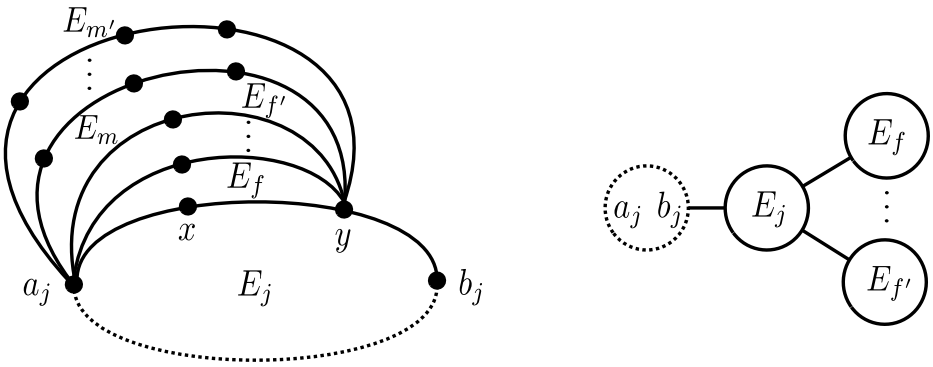


Figure 2.12 – Graphe (gauche) et décomposition arborescente (droite) de G_{i+1} avec $\ell_j = 3$.

— Maintenant, supposons que $E_j = (a_j, x, y, z, b_j)$ a une longueur de 4. Il existe plusieurs cas selon les sommets de E_j qui sont des sommets d'attachement pour d'autres oreilles E_l dans $\mathcal{E}\backslash(\mathcal{E}_i \cup \{E_j\})$ attachées à E_j. Parce que G n'a pas de clique-séparatrice, que $\mathcal{E}$ est une décomposition isométrique en oreilles imbriquées, et par symétrie, nous avons les possibilités suivantes :

— Si $Att(E_j) \subseteq \{a_j, y, b_j\}$ (voir Figure 2.13), alors que $\mathcal{E}'_{i+1}$ est constitué de $\mathcal{E}_i$ et E_j. Soit $(T', \mathcal{X}')$ la décomposition arborescente de $G' = G[\bigcup_{E \in \mathcal{E}'_{i+1}} V(E)]$ construite à partir de $(T^i, \mathcal{X}^i)$ comme suit. Notons par B un sac quelconque de $(T^i, \mathcal{X}^i)$ contenant à la fois a_j et b_j (qui existe par l'hypothèse d'induction). Ajoutons le sac $\{a_j, y, b_j\}$ adjacent à B et aux sacs $\{a_j, x, y\}$ et $\{y, z, b_j\}$. Puisque $(T^i, \mathcal{X}^i)$ est une décomposition arborescente de G_i de longueur 2, alors $(T', \mathcal{X}')$ est aussi une décomposition arborescente de G' de longueur 2. Soit F l'ensemble des oreilles de longueur 2 attachées à E_j et soit $\mathcal{E}_{i+1}$ constitué de $\mathcal{E}'_{i+1}$ et de F. Par le Lemme 2.4.5, on peut obtenir de $(T', \mathcal{X}')$ une décomposition arborescente $(T^{i+1}, \mathcal{X}^{i+1})$ de longueur 2 de G_{i+1}. Enfin, $(T^{i+1}, \mathcal{X}^{i+1})$ satisfait aux propriétés souhaitées (en particulier parce que G n'a pas de séparateur d'arêtes, chaque oreille attachée à G_{i+1} a ses sommets d'attachement dans un sac de $(T^{i+1}, \mathcal{X}^{i+1})$).

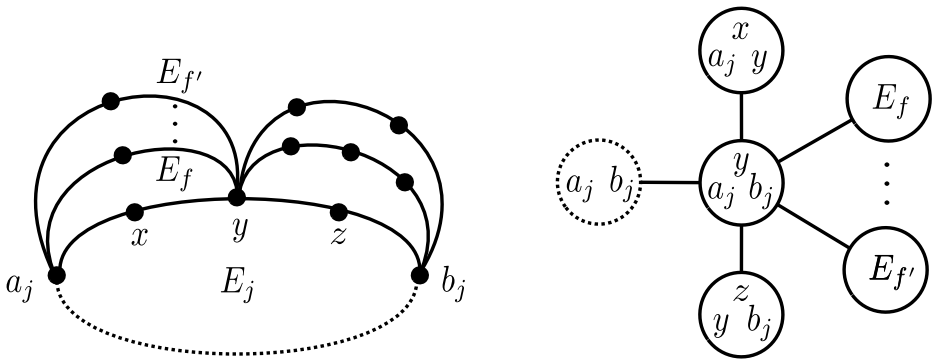


Figure 2.13 – Graphe (gauche) et décomposition arborescente (droite) de G_{i+1} avec $\ell_j = 4$ et $Att(E_j) \subseteq \{a_j, y, b_j\}$.

— Maintenant, supposons, par symétrie, qu'il existe une oreille E' attachée à a_j et z. Notons qu'une telle oreille est de longueur 3 puisque $\mathcal{E}$ est une décomposition isométrique d'oreilles imbriquées et qu'aucun cycle isométrique n'a une longueur supérieure à 6. Définissons $\mathcal{E}'$ comme l'ensemble de toutes les oreilles $E_{j'} = (a_j = a_{j'}, x_{j'}, y_{j'}, b_{j'} = z) \notin \mathcal{E}_i$ de longueur 3 attachées à a_j et z (en particulier, E' est une telle oreille). Définissons $\mathcal{E}'_{i+1}$ consistant en $\mathcal{E}_i \cup E_j \cup \mathcal{E}'$ (voir Figure 2.14).
Montrons d'abord qu'aucune oreille $E_q \in \mathcal{E} \setminus \mathcal{E}'_{i+1}$ de longueur au moins 3 n'est attachée à $x_{j'}$ et $z = b_{j'}$ pour un j' tels que $E_{j'} \in \mathcal{E}'$ (resp. pour x et z). Pour arriver à une contradiction, supposons qu'une telle oreille E_q existe. Rappelons que, par l'hypothèse d'induction, a_j et b_j doivent appartenir à un même sac de $(T^i, \mathcal{X}^i)$ de longueur 2 et que, du fait de l'absence de clique-séparatrice, $a_j b_j \notin E(G)$. Par conséquent, $dist_G(a_j, b_j) = dist_{G_i}(a_j, b_j) = 2$. Soit E_ℓ la première (c'est-à-dire, avec ℓ minimisé) oreille de G_i contenant à la fois a_j et b_j (une telle oreille doit exister puisque E_j ne peut être rattaché qu'aux sommets d'une certaine oreille précédente).
  — Si $E_\ell = E_0$, alors le sous-graphe induit par $V(E_0) \cup V(E_{j'}) \cup V(E_q)$ (resp. $V(E_0) \cup V(E_j) \cup V(E_q)$) est un graphe de Dumbo isométrique, une contradiction.
  — Sinon (if $\ell \neq 0$), notons par a_ℓ and b_ℓ les extrémités de E_ℓ, et notons par G^* le sous-graphe induit par les sommets des oreilles dans $\{E_m \in \mathcal{E}^i \mid m < \ell\}$. Notons que G^* est un sous-graphe isométrique de G_i. Sans perte de généralité, $a_\ell \notin \{a_j, b_j\}$ (sinon cela contredirait le fait que E_ℓ est la première oreille dans laquelle a_j et

b_j apparaissent). Soit P tout plus court chemin a_ℓ-b_ℓ dans G^*. Puisque a_ℓ et b_ℓ ne sont pas adjacents (sinon il y aurait une arête-séparatrice dans G), P a une longueur d'au moins 2. Alors, le sous-graphe induit par $V(P) \cup V(E_\ell) \cup V(E_{j'}) \cup V(E_q)$ (resp. $V(P) \cup V(E_\ell) \cup V(E_j) \cup V(E_q)$) est un graphe isométrique de Dumbo, une contradiction.

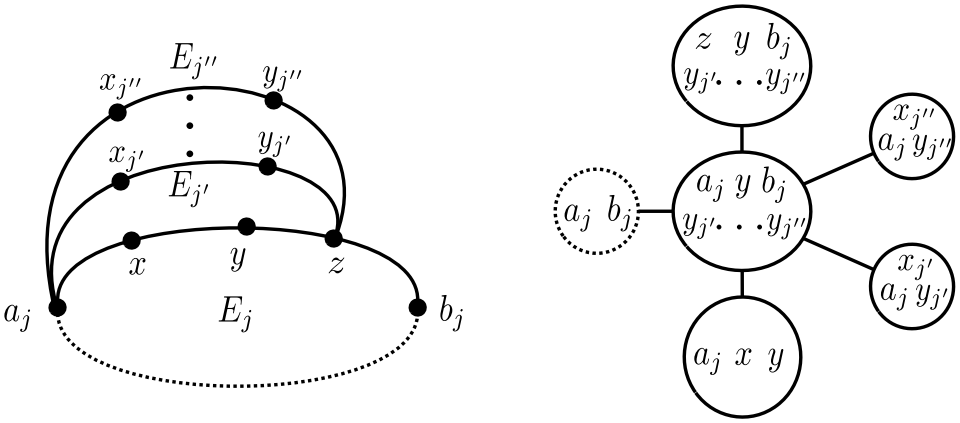


Figure 2.14 – Cas où E_j a une longueur de 4 et où il y a au moins une oreille attachée à a_j et z.

Soit B un sac quelconque de $(T^i, \mathcal{X}^i)$ contenant à la fois a_j et b_j (qui existe par l'hypothèse d'induction). Soient $B' = \{a_j, b_j, y\} \bigcup_{E_{j'} \in \mathcal{E}'} \{y_{j'}\}$, $B_{j'} = \{a_j, x_{j'}, y_{j'}\}$ pour tout j' tel que $E_{j'} \in \mathcal{E}'$, $B'' = \{b_j, z, y\} \bigcup_{E_{j'} \in \mathcal{E}'} \{y_{j'}\}$, et $B_j = \{a_j, x, y\}$.
Soit $(T', \mathcal{X}')$ la décomposition arborescente de $G' = G[\bigcup_{E \in \mathcal{E}'_{i+1}} V(E)]$ construite à partir de $(T^i, \mathcal{X}^i)$ en ajoutant le sac B' adjacent à B, à B'', à B_j, et à $B_{j'}$ pour tous les j' tels que $E_{j'} \in \mathcal{E}'$. On peut montrer que $(T', \mathcal{X}')$ est une décomposition arborescente de G', de longueur 2, telle que chaque oreille de longueur au moins 3 attachée à G' a ses deux sommets d'attachement dans un certain sac de $(T', \mathcal{X}')$. Soit F l'ensemble des oreilles de longueur 2 attachées à une oreille quelconque dans $\mathcal{E}' \cup E_j$ et soit $\mathcal{E}_{i+1}$ constitué de $\mathcal{E}'_{i+1}$ et F. Par le Lemme 2.4.5, on peut obtenir de $(T', \mathcal{X}')$ une décomposition arborescente $(T^{i+1}, \mathcal{X}^{i+1})$ de longueur 2 de G_{i+1}.
Enfin, $(T^{i+1}, \mathcal{X}^{i+1})$ satisfait aux propriétés souhaitées (en particulier parce que G n'a pas de séparateur d'arêtes, chaque oreille attachée à G_{i+1} a ses sommets d'attachement dans un sac de $(T^{i+1}, \mathcal{X}^{i+1})$).

Ceci conclut la preuve. Notons que cette preuve est constructive et fournit un algorithme en temps polynomial qui renvoie soit un graphe Dumbo isométrique, soit une décomposition arborescente de longueur 2. □

2.5 Poursuite des travaux

Ce travail présente la première caractérisation de la longueur arborescente d'une classe de graphes en termes de sous-graphes isométriques interdits. En particulier, nous montrons que décider si la longueur arborescente d'un graphe série-parallèle est au plus de 2 peut être fait en temps polynomial alors que ce problème est NP-complet dans les graphes généraux. Notre approche semble difficile à généraliser à des valeurs plus grandes de la longueur arborescente. En

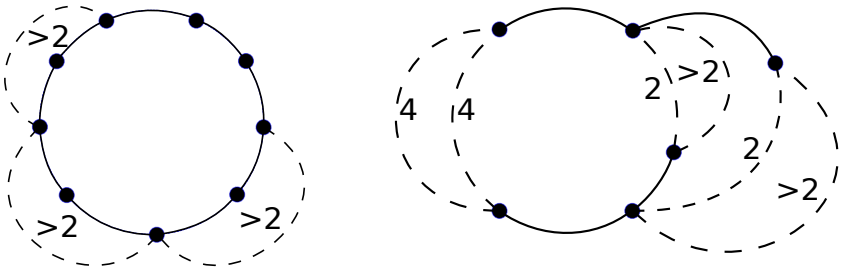


Figure 2.15 – Deux familles de graphes interdits comme sous-graphe isométrique dans les graphes série-parallèles de longueur arborescente 3. Les lignes pointillées représentent des chemins dont leurs longueurs sont données. Les autres lignes sont juste des arêtes.

effet, pour une longueur arborescente de 3, nous avons déjà identifié environ 15 familles de sous-graphes isométriques interdits. Toutes ces familles sont de légères variations des graphes Dumbo mais nous ne savons toujours pas comment les décrire de manière synthétique.

Intuitivement, le graphe Dumbo est construit à partir d'un cycle isométrique de taille 6, car c'est le plus petit cycle isométrique tel que nous ne pouvons pas mettre tous les sommets dans un sac pour obtenir une décomposition arborescente de longueur 2 de ce cycle. Autrement dit, c'est le plus petit cycle isométrique tel qu'il est nécessaire de sélectionner 3 sommets à distance au plus 2 l'un de l'autre pour partitionner notre cycle en 3 partie (chemin) de longueur au plus 2. La première différence pour les obstructions isométriques des graphes série-parallèles de longueur arborescente 3, est que le plus petit cycle isométrique nécessitant la sélection de 3 sommets à distance au plus 3 l'un de l'autre, est le cycle de taille 8. Nous avons donc deux cycles différents (taille 8 et 9) comme base pour définir de nouvelles familles d'obstruction (cf. la famille de gauche dans la Figure 2.15).

Ensuite, pour obtenir le graphe Dumbo, il faut ajouter une oreille attachée à ce cycle, telle que le plus court chemin dans le cycle entre les extrémités de l'oreille contient un des sommets de ces tuples de 3 sommets partitionnant le cycle. De plus il faut aussi que la longueur de l'oreille soit supérieure à la taille du plus court chemin entre les extrémités de l'oreille. La deuxième différence est donc que le cycle de taille 6 n'avait que deux tuples de 3 sommets qui partitionnent le cycle, alors que le cycle de taille 9 en a trois.

Une des dernières raisons de l'augmentation du nombre de familles, est que rajouter des oreilles de même taille que le plus court chemin entre leur extrémité dans le cycle, permet de multiplier le nombre d'oreilles que nous pouvons possiblement ajouter sur notre cycle. Cela se transcrit intuitivement en considérant que nous pouvons prendre des graphes melons de longueur arborescente 3 comme base (cf. la famille de droite dans la Figure 2.15).

Toutes ces raisons nous laissent donc penser que le nombre de familles va croître rapidement par rapport à la longueur arborescente maximale dont elles sont censées être une obstruction.

L'étape suivante consiste donc à trouver un algorithme qui calcule en temps polynomial la longueur des arbres des graphes série-parallèles (ou à prouver que c'est un problème NP-difficile).

CHAPITRE 3

Longueur linéaire / Pathlength

Dans ce chapitre, nous nous concentrons sur l'étude de la longueur linéaire. Ces travaux ont été réalisés en collaboration avec Nicolas Nisse (Dissaux & Nisse, 2022c, 2022b, 2022a). Rappelons que peu de résultats sont connus à part la complexité de calcul de la longueur linéaire (NP-complet) (Ducoffe et al., 2020). Nous commençons donc par des classes de graphes basiques (pour la longueur linéaire), c'est-à-dire, par les arbres et les cycles dans la section 3.2. Plus précisément, nous présentons un algorithme polynomial permettant de calculer la longueur linéaire des arbres. Ensuite, nous donnons la valeur de la longueur linéaire de tout cycle C_n de taille n. Finalement, nous concevons un algorithme d'approximation pour les graphes planaires extérieurs inspiré de celui pour les arbres, puisque le dual faible d'un graphe planaire extérieur est une forêt. Cet algorithme d'approximation a un facteur additif $+1$, c'est-à-dire, il calcule une décomposition linéaire de longueur au plus $p\ell(G) + 1$. Nous donnons aussi un exemple de graphe planaire extérieur pour lequel notre algorithme calcule une décomposition de longueur $p\ell(G) + 1$.

3.1 Préliminaires

Rappelons que, dans ce chapitre, nous ne considérons que les graphes simples, non dirigés et non pondérés (sans boucles ni arêtes parallèles). Nous noterons en général par $G = (V, E)$ les graphes que nous considérons où V est l'ensemble de sommets de G et E est l'ensemble d'arêtes de G. Lorsqu'il ne sera pas précisé, n correspondra toujours au nombre $|V|$ de sommets de G.

Décompositions Linéaires. Une *décomposition linéaire* de $G = (V, E)$ est une séquence $D = (X_1, \ldots, X_p)$ de sous-ensembles de sommets, appelés *sacs*, tels que :

1. $\bigcup_{i \leq p} X_i = V$;
2. Pour tout $e \in E$, il existe $i \leq p$ avec $e \subseteq X_i$;
3. Pour tout $1 \leq i \leq j \leq q \leq p$, $X_i \cap X_q \subseteq X_j$.

La *longueur* $\ell(D)$ de D est le diamètre maximum (dans G) de ses sacs, c'est-à-dire, $\ell(D) = \max_{i \leq p} \ell(X_i) = \max_{i \leq p} \max_{u,v \in X_i} dist_G(u, v)$. La *longueur linéaire* $p\ell(G)$ de G est la longueur minimale de ses décompositions linéaires. Une décomposition linéaire de G de longueur $p\ell(G)$ est dite *optimale*.

Une décomposition linéaire est *réduite* si aucun sac n'est contenu dans un autre. Il est facile de vérifier que tout graphe admet une décomposition linéaire réduite optimale.

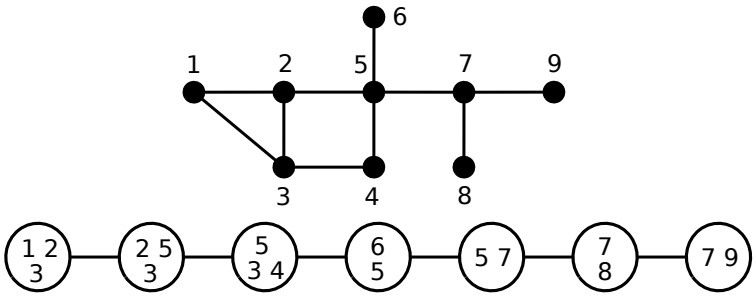


Figure 3.1 – Exemple de décomposition linéaire L (en bas) de largeur minimum ($pw(G) = 2$) de G (en haut).

Un graphe $G = (V, E)$ est *connexe* si, pour tout $u, v \in V$, il existe un chemin entre u et v dans G. Par le Lemme 1.3.8, nous pouvons nous restreindre à l'étude de la longueur linéaire des graphes connexes.

Rappelons que, pour tout sous-graphe isométrique H de G, $p\ell(H) \leq p\ell(G)$ 1.3.9. Ce résultat sera très souvent utilisé dans la suite de ce chapitre.

Rappelons que la $3^{\text{ème}}$ propriété des décompositions linéaires peut être reformulée de deux autres façons différentes (Remarque 1.3.2 et Remarque 1.3.4). L'une d'elle (Remarque 1.3.4) sera utilisée implicitement dans nos preuves. Plus précisément, nous utiliserons souvent l'argument que pour tout $1 \leq i < p$, $X_i \cap X_{i+1}$ est un séparateur de $\bigcup_{1 \leq j \leq i} X_j \setminus (X_i \cap X_{i+1})$ et $\bigcup_{i \leq j \leq p} X_j \setminus (X_i \cap X_{i+1})$ s'ils sont non vides.

Définition des décompositions linéaires via un ordre sur les sommets. Une autre façon (équivalente) de définir les décompositions linéaires d'un graphe $G = (V, E)$ consiste à ordonner les sommets de V. Cette représentation sera particulièrement utile dans la section 3.3. Soit $\mathcal{O} = (v_1, \ldots, v_n)$ un ordre (c'est-à-dire, une permutation) de V. Soit $D(\mathcal{O}) = (X_1, \ldots, X_{2n-2})$ définie comme suit. Soit $X_1 = \{v_1\}$ et, pour chaque $2 \leq i < n$, soit $X_{2i-2} = X_{2i-3} \cup \{v_i\}$, soit $S_i \subseteq X_{2i-2}$ l'ensemble des sommets $x \in X_{2i-2}$ tels que $N[x] \subseteq \bigcup_{j \leq 2i-2} X_j$ (c'est-à-dire, tous les sommets de X_{2i-2} dont chaque voisin appartient déjà à un sac X_j pour n'importe quel $j \leq 2i-2$) et soit $X_{2i-1} = X_{2i-2} \setminus S_i$. En particulier, notons que $S_i \subseteq N[v_i]$ et il peut être vide. Enfin, posons $X_{2n-2} = X_{2n-3} \cup \{v_n\}$. L'affirmation suivante est directe.

Assertion 3.1.0.1 – Soit $\mathcal{O}$ un ordre quelconque de l'ensemble de sommets V d'un graphe $G = (V, E)$. Alors, $D(\mathcal{O})$ est une décomposition linéaire (non réduite) de G.

Étant donné un ordre $\mathcal{O} = (v_1, \ldots, v_n)$ et $1 \leq i \leq j \leq n$, notons $\mathcal{O}[v_i, v_j] = (v_i, \ldots, v_j)$.

Étant donné deux séquences $D = (X_1, \ldots, X_p)$ et D' de sous-ensembles de V et $S \subseteq V$, on définit $D \cup S = (X_1 \cup S, \ldots, X_p \cup S)$ (si $S = \{v\}$, on écrira $D \cup v$ au lieu de $D \cup \{v\}$). Définissons $D \cap S$ et $D \setminus S$ de manière similaire (dans ces cas, les sacs devenant vide sont supprimés). Enfin, définissons $D \odot D'$ comme la séquence obtenue par concaténation de D et D'. L'affirmation suivante est directe (et bien connue).

Assertion 3.1.0.2 – Soient D une décomposition linéaire de $G = (V, E)$ et $S \subseteq V$.

1. Alors, $D' = D \cap S$ (resp., $D' = D \setminus S$) est une décomposition linéaire de $G[V \cap S]$ (resp., de $G \setminus S$). De plus, si $G[V \cap S]$ (resp., $G \setminus S$) est un sous-graphe isométrique de G, alors $\ell(D') \leq \ell(D)$.

2. Soit D' une décomposition linéaire de $G[V \setminus S]$. Alors, $D' \cup S$ est une décomposition linéaire de G.
3. Soit $V = A \cup B$ avec $A \cap B = S$ et S séparant $A \setminus S$ et $B \setminus S$ (il n'existe pas d'arête $\{u, v\} \in E$ avec $u \in A \setminus S$ et $v \in B \setminus S$), soit D_1 une décomposition linéaire de $G[A]$ avec le dernier sac contenant S, et soit D_2 une décomposition linéaire de $G[B]$ avec le premier sac contenant S. Alors, $D_1 \odot D_2$ est une décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\}$.

3.2 Longueur linéaire des arbres et des cycles

Commençons par les arbres, la sous-classe la plus "facile" des graphes planaires. Notons qu'intuitivement, l'algorithme présenté dans la section 3.3, qui calcule la longueur linéaire des graphes planaires extérieurs, suit des idées similaires à celles de l'algorithme que nous présentons pour les arbres.

Soient $k \in \mathbb{N}$ et S_k l'arbre obtenu à partir d'une étoile à trois feuilles en subdivisant chaque arête k fois, c'est-à-dire, en remplaçant chaque arête par un chemin de $k+1$ arêtes.

Lemme 3.2.1. *(Dourisboure & Gavoille, 2007) Pour tout $k \in \mathbb{N}$, $p\ell(S_k) = k + 1$.*

Notez que $p\ell(S_k) = \frac{|V(S_k)|-1}{3}$. Cela diffère de la largeur linéaire qui est bornée par $O(\log n)$ pour tout arbre à n sommets.

Notons que ce résultat nous aidera à prouver que la décomposition linéaire définie dans la preuve suivante est optimale.

Théorème 3.2.2. *La longueur linéaire d'un arbre quelconque et une décomposition linéaire optimale peuvent être calculées en temps linéaire.*

Démonstration. Soit T un arbre quelconque. Si T est un chemin, le résultat est évident (la longueur linéaire est égale à un). On peut donc supposer que T a au moins trois feuilles (sommets de degré 1).

Soit P_0 le plus long chemin de T et notons par x et y ses extrémités (qui sont des feuilles puisque P_0 est un plus long chemin). Soit z une feuille de T maximisant sa distance à P_0 (c'est-à-dire, pour toute feuille f de T, $dist(f, P_0) = \min_{v \in V(P_0)} dist(f, v) \leq dist(z, P_0)$). Soit v^* la projection de z sur P_0, c'est-à-dire, v^* est le sommet de P_0 minimisant sa distance à z. Notons que $dist(v^*, z) \leq \min\{dist(v^*, x), dist(v^*, y)\}$ (sinon, P_0 ne serait pas un plus long chemin). Soit T' le sous-arbre à inclusion minimale de T contenant x, y et z. Notons que T' contient S_k comme sous-graphe isométrique pour $k = dist(v^*, z) - 1$, et donc, S_k est un sous-graphe isométrique de T. Alors, par les Lemmes 3.2.1 et 1.3.9, $p\ell(T) \geq p\ell(S_k) = k + 1$.

Nous montrons maintenant que $p\ell(T) = k + 1$ en construisant une décomposition linéaire D de T de longueur $k + 1$. Let $P_0 = (x = v_0, v_1, \ldots, v_t = y)$. Décrivons un algorithme itératif construisant D. Commençons avec D contenant uniquement le sac $\{v_0, v_1\}$.

Pour i allant de 1 à $t - 1$, faisons ce qui suit. Soit T^i la composante connexe de $T \setminus E(P_0)$ (c'est-à-dire, en supprimant les arêtes de P_0, mais en conservant ses sommets) qui contient v_i. Soit F^i la séquence des feuilles de T^i à l'exception de v_i (si v_i est une feuille de T^i) ordonnée en suivant n'importe quel algorithme DFS de T^i à partir de v_i. Notons que, par définition de z, pour tout $f \in F^i$, $dist(f, v_i) \leq k + 1$ (notons cette propriété par $(**)$). Soit $\mathcal{Q} = (Q_1, \ldots, Q_{|F^i|})$ la séquence des chemins allant de v_i à f, pour tout $f \in F^i$, où l'ordre des chemins suit l'ordre des

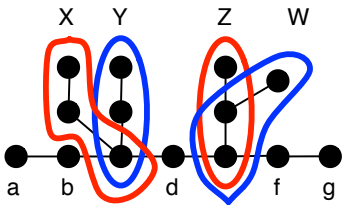


Figure 3.2 – Exemple d'arbre T (avec $p\ell(T) = 2$) dont la décomposition linéaire obtenue grâce au théorème 3.2.2 pourrait être $(\{ab\}, \{bc\}, X, Y, \{cd\}, \{de\}, Z, W, \{ef\}, \{fg\})$.

feuilles dans F^i. Pour q allant de 1 à $|F^i|$, ajouter $V(Q_q)$ comme prochain sac de D, puis ajouter le sac $\{v_i, v_{i+1}\}$ à D.

Soit $D = (X_1, \ldots, X_h)$ le résultat de l'algorithme ci-dessus. Tout d'abord, par la propriété $(**)$, pour chaque $1 \leq i \leq h$, $diam(X_i) = \max_{u,v \in X_i} dist_G(u, v) \leq k+1$. Il reste donc à prouver que D est une décomposition linéaire de T. Par construction, il est clair que chaque sommet et chaque arête de T appartiennent à au moins un sac de D. De plus, pour chaque $1 \leq j \leq t$, toujours par construction, si $v_j \in X_a \cap X_b$, alors $v_j \in X_c$ pour chaque $a \leq c \leq b$. Enfin, pour tout $v \in V(T) \setminus V(P_0)$, notons par $1 \leq i < t$ l'indice tel que $v \in V(T^i)$ et supposons que v appartient à $X_a \cap X_b$ pour $1 \leq a < b \leq h$. Par construction, il existe deux feuilles f_a et f_b de $T^i \setminus \{v_i\}$ telles que v appartient à P_a et P_b, le chemin entre v_i et f_a et le chemin entre v_i et f_b respectivement, $X_a = V(P_a)$, et $X_b = V(P_b)$. De plus, $f_a < f_b$ dans l'ordre de F^i. Maintenant, pour tout $a \leq c \leq b$, le sac X_c correspond au chemin P_c entre v_i et une feuille f_c de $T^i \setminus \{v_i\}$ telle que $f_a \leq f_c \leq f_b$ dans l'ordre de F^i. Puisque les feuilles de T^i ont été ordonnées en suivant n'importe quel algorithme DFS de T^i à partir de v_i, le chemin P_c doit aussi contenir v et donc $v \in X_c = V(P_c)$. Par conséquent, D est une décomposition linéaire de T de longueur au plus $k + 1$. Un exemple d'une telle décomposition est donné dans la Figure 3.2.

Le fait que l'algorithme ci-dessus puisse être exécuté en temps linéaire provient du fait que P_0 peut être calculé en temps linéaire (le diamètre d'un arbre peut être calculé à l'aide de 2 algorithme BFS, le premier trouve un ensemble contenant des extrémités possibles de P_0, et le deuxième, lancé à partir d'une des extrémités possibles de P_0, trouve une extrémité associée) et qu'il suffit d'exécuter un algorithme DFS à partir de v_i pour chaque T^i pour calculer l'ordre de Q pour chaque T^i. □

Remarque. La preuve ci-dessus montre en fait que, pour tout arbre T, $p\ell(T)$ est égal à l'excentricité minimal d'un plus court chemin ($+1$ si T est un chemin). Notons qu'il était déjà connu que le plus court chemin d'excentricité minimale des arbres peut être calculé en temps linéaire (Dragan & Leitert, 2016).

Nous allons maintenant nous concentrer sur les cycles.

Théorème 3.2.3. *Soit C_n un cycle de longueur n. Alors, $p\ell(C_n) = \lfloor \frac{n}{2} \rfloor$.*

Démonstration. Tout d'abord, la décomposition linéaire consistant en un unique sac contenant $V(C_n)$ a pour longueur $\lfloor \frac{n}{2} \rfloor$, donc, $p\ell(C_n) \leq \lfloor \frac{n}{2} \rfloor$. Montrons qu'il s'agit d'une décomposition linéaire optimale.

Pour arriver à une contradiction, supposons que C_n admette une décomposition linéaire réduite $D = (X_1, \ldots, X_p)$ de longueur $k < \lfloor \frac{n}{2} \rfloor$. Tout d'abord, montrons que nous pouvons supposer

que X_1 induit un sous-graphe connexe (plus précisément, un sous-chemin). Supposons que ce n'est pas le cas. Puisque D est réduit, il existe un sommet dans X_1 qui n'est pas dans chaque X_i pour $2 \leq i \leq p$. Soit v un tel sommet. Puisque v est seulement dans X_1, les deux voisins de v appartiennent aussi à X_1. Soit A la composante connectée de $G[X_1]$ contenant v. Notons que A est un chemin, sinon $V(A) = V(C_n)$, ce qui contredit que $k < \lfloor \frac{n}{2} \rfloor$. Soient x et y les deux extrémités de A et soit $A' = A \setminus \{x, y\}$. Alors, $D' = (A, X_1 \setminus A', X_2 \setminus A', \ldots, X_p \setminus A')$ est une décomposition linéaire de longueur au plus k et dont le premier sac induit un chemin. Si nous supprimons les sacs redondants de D' (notons que A n'est contenu dans aucun autre sac), on peut supposer que D' est réduit.

Nous pouvons donc noter par $D = (X_1, \ldots, X_p)$, une décomposition linéaire réduite de longueur $k < \lfloor \frac{n}{2} \rfloor$ telle que X_1 induit un chemin. Nous supposons en outre que, parmi ces décompositions, D maximise $|X_1|$. Nous montrons ensuite que $|X_1| = k+1$. Soient x et y les deux extrémités du chemin induit par X_1. Notons que $|X_1| - 1 = dist(x, y) \leq k$. Pour arriver à une contradiction, supposons que $dist(x, y) < k$ (c'est-à-dire, $|X_1| < k + 1$). Notons par x' et y' les voisins respectivement de x et y qui ne sont pas dans X_1. Soient X_j et $X_{j'}$ les premiers sacs (les plus proches de X_1) qui contiennent respectivement y' et x'. Notons que $y \in X_i$ pour tout $1 \leq i \leq j$ (resp. $x \in X_{i'}$ pour tout $1 \leq i' \leq j'$), sinon il n'y aurait pas de sac contenant l'arête $\{y, y'\}$ (resp. l'arête $\{x, x'\}$). Sans perte de généralité, supposons que $j \leq j'$. Par conséquent, X_j contient x, y et y'. Notons que $dist(x, y') = dist(x, y) + 1 \leq k$. Alors, $D' = (X_1 \cup \{y'\}, \ldots, X_{j-1} \cup \{y'\}, X_j, \ldots, X_p)$ est une décomposition linéaire dont le premier sac induit un chemin de longueur $|X_1|$. Nous allons montrer que D' a une longueur maximale de k, ce qui contredit la maximalité de $|X_1|$ dans D.

Pour arriver à une contradiction, supposons que D' a une longueur strictement supérieure à k. Il existe donc $1 < h < j$ et $w \in X_h$ tel que $dist(w, y') > k$. Puisque $y \in X_h$, alors $dist(w, y) \leq k$. Notons également que $dist(w, y') \leq dist(w, y) + 1 \leq k + 1$ et donc $dist(w, y') = k + 1$. Puisque $dist(w, y) \geq dist(w, y') - 1 = k$, alors w est l'unique sommet à distance k de y et à distance $k + 1$ de y'. Soit w' le voisin de w tel que $dist(w', y) = k + 1$. Notons que w' ne peut pas être dans un sac contenant y puisqu'ils sont à distance $k + 1$. Par conséquent, il n'est pas dans X_i pour $1 \leq i \leq j$ (puisque ces sacs contiennent y). Notons que w' et w doivent être dans un sac commun de D (puisqu'ils sont adjacents). Par conséquent, il existe un sac X_l avec $j < l \leq p$ qui contient w et w'. Puisque $w \in X_h \cap X_l$, alors $w \in X_j$ ce qui contredit le fait que D a une longueur au plus k (puisque $dist(w, y') > k$). Par conséquent, D' est une décomposition linéaire de longueur k et il y a donc une contradiction avec la maximalité de $|X_1|$ dans D. L'hypothèse que $dist(x, y) < k$ est donc fausse, c'est-à-dire, le premier sac de D induit un chemin de longueur k (puisque D a longueur k).

Notons par x' et y' les voisins de x et y qui ne sont pas dans X_1. Soient X_j et $X_{j'}$ les premiers sacs (les plus proches de X_1) qui contiennent respectivement y' et x'. Notons que $y \in X_i$ pour tout $1 \leq i \leq j$ (resp. $x \in X_i$ pour tout $1 \leq i \leq j'$). Supposons que $j \leq j'$. Par conséquent, X_j contient x, y et y', ce qui contredit la longueur de D. Nous pouvons donc conclure, qu'il n'y a pas de décomposition linéaire de C_n de longueur au plus $\lfloor \frac{n}{2} \rfloor - 1$, c'est-à-dire, $p\ell(C_n) \geq \lfloor \frac{n}{2} \rfloor$, et donc $p\ell(C_n) = \lfloor \frac{n}{2} \rfloor$. □

3.3 Longueur linéaire des graphes planaires extérieurs

Cette section est consacrée à notre résultat principal : un algorithme calculant, en temps polynomial, une décomposition linéaire de tout graphe planaire extérieur simple (sans arêtes parallèles ni boucles) G d'une longueur au plus $p\ell(G) + 1$.

Un graphe $G = (V, E)$ est *planaire extérieur* s'il peut être plongé dans le plan sans croisement d'arêtes et tel que tous les sommets se trouvent sur la face extérieure (la face non bornée). Une arête d'un graphe planaire extérieur est qualifiée d'*interne* si elle ne se trouve pas sur la face extérieure (du plongement du graphe). Notons que, puisque nous ne considérons que des graphes simples, le fait qu'une arête soit interne ou non ne dépend pas du plongement du graphe. Soient $E_{int} \subseteq E$ l'ensemble des arêtes internes et $E_{out} = E \setminus E_{int}$ l'ensemble des arêtes *externes*. Notons que toute arête interne $e = \{u, v\} \in E_{int}$ d'un graphe planaire externe $G = (V, E)$ est un séparateur (c'est-à-dire, $G \setminus V(e)$ a plusieurs composantes connexes). Plus précisément, il existe toujours exactement deux composantes C et C' telles que $N(C) = N(C') = \{u, v\}$ (s'il y en a moins de 2, alors e n'est pas une arête interne et sinon, G n'est pas un graphe planaire extérieur puisqu'il contient $K_{2,3}$ comme mineur).

Décomposition linéaire avec un premier et un dernier éléments fixés. Soit $G = (V, E)$ un graphe connexe et soit $v \in V$ (resp., $e \in E$). Une décomposition linéaire $D = (X_1, \ldots, X_p)$ de G *commence* à partir de v (resp., à partir de e) si $v \in X_1$ (resp., si $e \subseteq X_1$). De même, D *finit* avec v (resp., avec e) si $v \in X_p$ (resp., si $e \subseteq X_p$). Si $x, y \in V \cup E$, une décomposition linéaire de G commençant par x et finissant par y est appelée une *$\{x, y\}$-décomposition linéaire*. Définissons $p\ell(G, x, y)$, la longueur minimale de toutes les $\{x, y\}$-décompositions linéaires de G. Une $\{x, y\}$-décomposition linéaire est une $\{x, y\}$-décomposition linéaire *optimale* si sa longueur est $p\ell(G, x, y)$. Il est clair que toute décomposition $(X_1, \ldots, X_p)$ correspond à une décomposition $(X_p, \ldots, X_1)$ de même longueur, et donc :

Assertion 3.3.0.1 – Pour tout graphe connexe $G = (V, E)$ et $x, y \in E \cup V$, $p\ell(G, x, y) = p\ell(G, y, x)$.

L'affirmation suivante vient directement de la définition des décompositions linéaires et du fait qu'il existe toujours une décomposition linéaire réduite optimale (notons que, pour toute décomposition linéaire réduite D d'un graphe connexe, il existe $x, y \in E$ telles que D commence par x et finit par y).

Assertion 3.3.0.2 – Pour tout graphe connexe $G = (V, E)$, $p\ell(G) = \min_{x,y \in E} p\ell(G, x, y)$.

Pour notre objectif, nous devons raffiner l'affirmation ci-dessus comme suit.

Lemme 3.3.1. *Pour tout graphe connexe $G = (V, E)$, $p\ell(G) = \min_{x,y \in E_{out}} p\ell(G, x, y)$.*

Démonstration. Soient $x, y \in E$ telles que $p\ell(G) = p\ell(G, x, y)$ et telles que la "distance" entre x et y est maximisée (où la distance doit être comprise ici comme le nombre de faces internes qui doivent être traversées pour aller de x à y). Soit $D = (X_1, \ldots, X_p)$ une $\{x, y\}$-décomposition linéaire optimale de G (donc D est une décomposition linéaire optimale de G).

Pour arriver à une contradiction, et sans perte de généralité, supposons que $x \in E_{int}$. Soit C une composante connexe de $G \setminus x$ qui ne contient pas y. Soient $D' = D \cap (C \cup x))$ (notons que $\ell(D') \leq \ell(D)$ par l'Assertion 3.1.0.2 et parce que $G \cap (C \cup x)$ est un sous-graphe isométrique

de G) et $D_2 = D \setminus C$ (notons que $\ell(D_2) \leq \ell(D)$ par l'Assertion 3.1.0.2 et parce que $G \setminus C$ est un sous-graphe isométrique de G). Soit D'' la décomposition linéaire réduite de $G[C \cup x]$ obtenue à partir de D' (c'est-à-dire, en supprimant les sacs qui sont contenus dans d'autres). Puisque D'' est réduite, le dernier sac de D'' doit contenir une arête $x' \neq x$ de $G[C \cup x]$. Enfin, supposons que D_1 soit la décomposition linéaire de $G[C \cup x]$ obtenue à partir de D'' en inversant l'ordre des sacs. Alors, $D_1 \odot D_2$ est une $\{x', y\}$-décomposition linéaire de G de longueur au plus $p\ell(G)$ (par l'Assertion 3.1.0.2 et puisque D_1 et D_2 contient x respectivement dans son dernier sac et dans son premier sac), ce qui contredit la maximalité de la distance entre x et y. Par conséquent, si x et y maximisent la "distance" entre eux, alors x et y doivent être dans E_{out}. □

Par le Lemme 3.3.1, le calcul de $p\ell(G)$ peut être restreint aux $O(n^2)$ calculs de $p\ell(G, x, y)$ pour toute paire d'arêtes $x, y \in E_{out}$ fixées (puisque G est planaire, $|E| = O(n)$). La majeure partie de ce qui suit est consacrée à cette tâche. Par conséquent, dans les sections 3.3.1 à 3.3.2, $x, y \in V \cup E_{out}$ seront fixés. La section 3.3.1 est consacrée aux cas "faciles". En fonction de x et y, il sera parfois possible de réduire le problème à des instances plus petites (puis de procéder par induction) et, dans d'autres cas (en gros, lorsque $x = y$ et que $G \setminus x$ est connexe), nous présentons un algorithme glouton qui calcule une $\{x, y\}$-décomposition linéaire optimale de G. La section 3.3.2 est consacrée au cas restant : en gros, lorsque x et y sont distincts, ne sont pas inclus l'un dans l'autre, ne séparent pas G et appartiennent à une même face interne (bornée) de G. Dans ce dernier cas, nous montrons que nous pouvons restreindre notre attention à des décompositions linéaires particulières de G et qu'une telle décomposition optimale peut être calculée en temps polynomial par programmation dynamique. Enfin, la section 3.3.3 énonce formellement notre résultat principal et décrit notre algorithme principal qui utilise les résultats des sections 3.3.1 à 3.3.2 pour calculer une décomposition linéaire d'un graphe planaire extérieur connexe de longueur au plus $p\ell(G) + 1$.

3.3.1 Récursion et l'algorithme glouton

Soit $G = (V, E)$ un graphe extérieur planaire simple et connexe et soient $x, y \in E_{out} \cup V$. Dans ce qui suit, rappelons qu'une arête est vue comme un ensemble de deux sommets. Dans cette section, nous montrons d'abord que si $G \setminus x$ n'est pas connexe (Section 3.3.1.1) ou si x et y sont "séparés" (voir la définition formelle dans la Section 3.3.1.2), le problème du calcul d'une $\{x, y\}$-décomposition linéaire optimale de G peut être réduit à des problèmes similaires dans des instances plus petites. Ensuite, dans la section 3.3.1.3, nous montrons que, si aucune des deux propriétés précédentes n'est satisfaite et que $x = y$ ou $x \in y$ ou $y \in x$, alors une $\{x, y\}$-décomposition linéaire optimale de G peut être calculée en temps linéaire par un algorithme glouton.

3.3.1.1 Cas où $G \setminus x$ n'est pas connexe.

Supposons tout d'abord que $G \setminus x$ n'est pas connexe. Nous avons divisé ce cas en plusieurs lemmes qui se ressemblent beaucoup (les preuves semblent redondantes). Nous n'avons pas réussi à les factoriser tout en gardant une lisibilité raisonnable de cette partie. En particulier, nous espérons que la division en différents lemmes sera utile pour convaincre que tous les cas sont réellement considérés.

Nous commençons par considérer le cas où $x = y$:

Lemme 3.3.2. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe et soit $x = y \in E_{out} \cup V$ tel que $G \setminus x$ n'est pas connexe.*

Soit C une composante connexe de $G \setminus x$. Soit D_1 une $\{x, x\}$-décomposition optimale de $G[C \cup x]$ et soit D_2 une $\{x, x\}$-décomposition optimale de $G \setminus C$. Alors, $D_1 \odot D_2$ est une $\{x, x\}$-décomposition linéaire optimale de G.

Démonstration. Soit D une $\{x, x\}$-décomposition linéaire optimale de G. Alors, puisque $C \cup \{x\}$ et $G \setminus C$ sont des sous-graphes isométriques de G et par l'Assertion 3.1.0.2, $D \cap (C \cup \{x\})$ est une décomposition linéaire de $G[C \cup x]$ de longueur au plus $\ell(D)$ et $D \setminus C$ est une décomposition linéaire de $G \setminus C$ de longueur au plus $\ell(D)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, x)$.

Réciproquement, par le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors $D_1 \odot D_2$ est une $\{x, x\}$-décomposition linéaire de G de longueur $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, x)$, c'est-à-dire, une $\{x, x\}$-décomposition linéaire optimale de G. □

Par conséquent, dans la partie restante de cette section (c'est-à-dire le cas où $G \setminus x$ n'est pas connexe), nous pouvons en outre supposer que $x \neq y$.

Dans les quatre lemmes suivants, nous considérons les cas où $x = \{u, v\} \in E_{out}$, $G \setminus x$ n'est pas connexe et $x \neq y$ ($y \in V \cup E_{out}$). Notons que, puisque $x = \{u, v\}$ n'est pas une arête interne, le fait que $G \setminus x$ ne soit pas connexe implique que l'un des sommets de $\{u, v\}$ (ou les deux) est un sommet-séparateur. Sans perte de généralité, nous supposerons, dans le reste de cette section, que $G \setminus u$ n'est pas connexe, et : dans les Lemmes 3.3.3 et 3.3.4, nous supposons que $x \cap y = \{u\}$, et dans les Lemmes 3.3.5 et 3.3.6, nous considérons les cas où $u \notin x \cap y$.

Lemme 3.3.3. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soit $x = \{u, v\} \in E_{out}$ de sorte que $G \setminus u$ n'est pas connexe et $\{u\} = x \cap y$. Soit C la composante connexe de $G \setminus u$ qui contient v et supposons que $y \setminus u$ n'est pas dans C.*

Soit D_1 une décomposition optimale $\{x, u\}$ de $G[C \cup x]$ et soit D_2 une décomposition optimale $\{u, y\}$ de $G \setminus C$. Notons que $D_1 \odot D_2$ est une décomposition optimale $\{x, y\}$ de G.

Démonstration. Soit D une décomposition linéaire optimale de G (notons que u appartient à tous les sacs de D). Par le premier énoncé de l'Assertion 3.1.0.2 et parce que $C \cup x$ et $G \setminus C$ sont des sous-graphes isométriques de G, $D_1 = D \cap (C \cup x)$ est une $\{x, u\}$-décomposition linéaire de $G[C \cup x]$ de longueur au plus $p\ell(G, x, y)$ et $D_2 = D \setminus C$ est une $\{u, y\}$-décomposition linéaire de $G \setminus C$ de longueur au plus $p\ell(G, x, y)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$

Réciproquement, d'après le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$ décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$. □

Lemme 3.3.4. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soit $x = \{u, v\} \in E_{out}$ de sorte que $G \setminus u$ n'est pas connexe et $\{u\} = x \cap y$. Supposons que $y \setminus u$ est dans la composante connexe C' de $G \setminus u$ qui contient v et que C soit toute composante connexe de $G \setminus u$ qui ne contient pas v.*

Si D_1 est une $\{x, x\}$-décomposition optimale de $G[C \cup x]$ et si D_2 est une $\{x, y\}$-décomposition optimale de $G \setminus C$, alors $D_1 \odot D_2$ est une $\{x, y\}$-décomposition optimale de G.

Démonstration. Soit D une $\{x, y\}$-décomposition linéaire optimale de G (notons que u appartient à tous les sacs de D). Par l'Assertion 3.1.0.2 et parce que $C \cup u$ et $G \setminus C$ sont des sous-graphes isométriques de G, $D_1 = (D \cap (C \cup u)) \cup v$ est une $\{x, x\}$-décomposition linéaire de $G[C \cup x]$ de longueur au plus $p\ell(G, x, y)$ (nous pouvons ajouter v puisque tous les sacs de D contiennent un sommet de C', c'est-à-dire, un sommet entre v et $y \cap C'$) et $D_2 = D \setminus C$ est une $\{x, y\}$-décomposition linéaire de $G \setminus C$ de longueur au plus $p\ell(G, x, y)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$.

Réciproquement, par le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$. □

Lemme 3.3.5. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soit $x = \{u, v\} \in E_{out}$ tel que $G \setminus u$ n'est pas connexe et $u \notin x \cap y$. Soit C la composante de $G \setminus u$ qui contient v et supposons que y n'est pas dans C.*

Si D_1 est une $\{x, u\}$-décomposition linéaire optimale de $G[C \cup x]$ et D_2 est une $\{u, y\}$-décomposition linéaire optimale de $G \setminus C$, alors $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire optimale de G.

Démonstration. Soit D une décomposition linéaire optimale de G. Alors, par l'Assertion 3.1.0.2 et parce que $C \cup u$ et $G \setminus C$ sont des sous-graphes isométriques de G, $D_1 = (D \cap (C \cup u)) \cup u$ est une $\{x, u\}$-décomposition linéaire de $G[C \cup x]$ de longueur au plus $p\ell(G, x, y)$ (ceci est vrai puisque chaque sac de D contient u ou un sommet de la composante de $G \setminus u$ qui contient y) et $D_2 = D \setminus C$ est une $\{u, y\}$-décomposition linéaire de $G \setminus C$ de longueur au plus $p\ell(G, x, y)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$.

Réciproquement, par le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$. □

Lemme 3.3.6. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soit $x = \{u, v\} \in E_{out}$ tel que $G \setminus u$ n'est pas connexe et $u \notin x \cap y$. Supposons que y est contenu dans la composante C' de $G \setminus u$ qui contient v. Soit C n'importe quelle composante connexe de $G \setminus u$ qui ne contient pas y.*

Si D_1 est une $\{x, x\}$-décomposition optimale de $G[C \cup x]$ et D_2 est une $\{x, y\}$-décomposition optimale de $G \setminus C$, alors $D_1 \odot D_2$ est une $\{x, y\}$-décomposition optimale de G.

Démonstration. Soit D une décomposition linéaire optimale de G. Alors, par l'Assertion 3.1.0.2 et parce que $C \cup u$ et $G \setminus C$ sont des sous-graphes isométriques de G, $D_1 = (D \cap (C \cup u)) \cup x$ est une $\{x, x\}$-décomposition linéaire de $G[C \cup x]$ de longueur au plus $p\ell(G, x, y)$ (ceci est vrai puisque chaque sac de D contient un sommet de C' entre v et $y \cap C'$ et un sommet de C' entre u et $y \cap C'$) et $D_2 = D \setminus C$ est une $\{x, y\}$-décomposition linéaire de $G \setminus C$ de longueur au plus $p\ell(G, x, y)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$.

Réciproquement, d'après le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$. □

Le cas où $x \in V$, $y \in E_{out}$ et $x \in y$ peut être traité de façon similaire en inversant la décomposition linéaire (Assertion 3.3.0.1).

Il ne reste que le cas où $x, y \in V$, $x \neq y$ et $G \setminus x$ n'est pas connexe.

Lemme 3.3.7. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in V$ tels que $G \setminus x$ n'est pas connexe et $y \neq x$. Soit C n'importe quelle composante connexe de $G \setminus x$ qui ne contient pas y.*

Si D_1 est une $\{x, x\}$-décomposition optimale de $G[C \cup x]$ et D_2 est une $\{x, y\}$-décomposition optimale de $G \setminus C$, alors $D_1 \odot D_2$ est une $\{x, y\}$-décomposition optimale de G.

Démonstration. Soit D une $\{x, y\}$-décomposition linéaire optimale de G. Alors, par l'Assertion 3.1.0.2 et parce que $C \cup x$ et $G \setminus C$ sont des sous-graphes isométriques de G, $D_1 = D \cap (C \cup x) \cup x$ est une $\{x, x\}$-décomposition linéaire de $G[C \cup x]$ de longueur au plus $p\ell(G, x, y)$ (ceci est vrai puisque chaque sac de D contient x ou un sommet de $V \setminus C$ puisque y est dans le dernier sac) et $D_2 = D \setminus C$ est une $\{x, y\}$-décomposition linéaire de $G \setminus C$ de longueur au plus $p\ell(G, x, y)$. Nous avons donc prouvé que $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$.

Réciproquement, par le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus $\max\{\ell(D_1), \ell(D_2)\} \leq p\ell(G, x, y)$. □

À partir de maintenant, on peut supposer que $G \setminus x$ est connexe. Pour les mêmes raisons et par l'Assertion 3.3.0.1 (et la phrase qui précède la précédente assertion), on peut aussi supposer que $G \setminus y$ est connexe.

3.3.1.2 Cas où $G \setminus x$ et $G \setminus y$ sont connexes, et x, y sont "séparés".

Soit $G = (V, E)$ un graphe extérieur planaire simple et connexe, et soient $x, y \in E_{out} \cup V$. Dans cette section, nous supposons toujours que $G \setminus x$ et $G \setminus y$ sont connexes. On dit que x et y sont *séparés* dans G s'il existe $z \in V \cup E$ tel que $x' = x \setminus z \neq \emptyset$, $y' = y \setminus z \neq \emptyset$ et que tous les chemins de x' à y' intersectent z. Notons que ceci requiert que $x \neq y$ et que ni $x \in y$, ni $y \in x$, mais x et y peuvent s'intersecter (dans ce cas, z est ou contient $x \cap y$). Notons également que, si $z \in E$ et est minimal par inclusion pour la propriété de séparation de x et y (c'est-à-dire qu'aucune extrémité de z ne sépare x et y), alors $z \in E_{int}$.

Lemme 3.3.8. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in E_{out} \cup V$ telles que $G \setminus x$ et $G \setminus y$ sont connexes, et telles que x et y sont séparés par $z \in V \cup E_{int}$. De plus, supposons que z soit minimale par inclusion pour cette propriété.*

Soit C^x la composante connectée de $G \setminus z$ contenant (ou intersectant) x et soit $C^y = V \setminus C^x$.

Soient D_1 une $\{x, z\}$-décomposition linéaire optimale de $G[C^x \cup z]$ et D_2 une $\{z, y\}$-décomposition linéaire optimale de $G[C^y]$. Alors, $D_1 \odot D_2$ est une décomposition linéaire optimale de $G[C^y]$.

Démonstration. Soit $D = (X_1, \ldots, X_p)$ une $\{x, y\}$-décomposition linéaire optimale de G. Soit $1 \leq i \leq p$ le plus petit entier tel que $z \in X_i$ (resp., tel que $z \subseteq X_i$ si z est une arête) et soit $i \leq j \leq p$ le plus grand entier tel que $z \in X_j$ (resp., tel que $z \subseteq X_j$ si z est une arête). Soit $D_1 = (X_1 \cap (C^x \cup z), \ldots, X_j \cap (C^x \cup z), (X_{j+1} \cap (C^x \cup z)) \cup z, \ldots, (X_p \cap (C^x \cup z)) \cup z)$. Il est clair que D_1 est une $\{x, z\}$-décomposition linéaire de $G[C^x \cup z]$. De plus, puisque $y \in X_p$, tous les sacs X_q (pour $j < q \leq p$) doivent contenir un sommet de la composante connexe de $G \setminus z$ contenant y. Par conséquent, puisque $C^x \cup \{z\}$ est un sous-graphe isométrique de G, et par l'Assertion 3.1.0.2, $\ell(D_1) \leq \ell(D)$. De la même façon, définissons $D_2 = ((X_1 \setminus C) \cup z, \ldots, (X_{i-1} \setminus C) \cup z, X_i \setminus C, \ldots, X_p \setminus C)$. Alors, D_2 est une $\{z, y\}$-décomposition linéaire de $G \setminus C^x$. De plus, comme

$x \in X_1$, tous les sacs X_q (pour $1 \leq q < i$) doivent contenir un sommet de C^x. Par conséquent, puisque C^y est un sous-graphe isométrique de G, et par l'Assertion 3.1.0.2, $\ell(D_2) \leq \ell(D)$.

Réciproquement, par le dernier énoncé de l'Assertion 3.1.0.2, si D_1 et D_2 sont définis comme dans l'énoncé du lemme, alors il est clair que $D_1 \odot D_2$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus $\max \ell(D_1), \ell(D_2) \leq p\ell(G, x, y)$. □

3.3.1.3 Cas où $G \setminus x$ et $G \setminus y$ sont connexes, et $x = y$ ou $x \in y$ ou $y \in x$: algorithme glouton.

Dans le cas de cette section, nous montrons qu'une $\{x, y\}$-décomposition linéaire optimale peut être calculée en temps linéaire si $x = y$, $x \in y$ ou $y \in x$. Nous commençons par définir un processus récursif permettant de construire une $\{x, x\}$-décomposition linéaire particulière lorsque $x \in E_{out} \cup V$.

Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe et soit $x \in E_{out}$ tel que ni u ni v n'est un sommet-séparateur, ou soit $x \in V$ et n'est pas un sommet-séparateur. Une *décomposition linéaire gloutonne P de G basée sur* x est toute $\{x, x\}$-décomposition linéaire de G qui peut être obtenue récursivement (sur $|V|$) comme suit.

Algorithme Glouton récursif :

- Le premier cas de base est celui où G est un cycle $(v_1, \ldots, v_n)$ (sans perte de généralité, $x = \{v_1, v_n\}$ ou $x = v_1$). Dans ce cas, $D = (X_1, \ldots, X_{n-1})$ avec, pour chaque $1 \leq i \leq n-1$, $X_i = x \cup \{v_i, v_{i+1}\}$. Le deuxième cas de base est celui où G est une arête $\{u, v\}$ et $x = u$, auquel cas, $D = (\{u, v\})$.
- Autrement (G n'est ni un cycle ni une arête), si x est un sommet de degré un dans G, notons par w son voisin et par D' une décomposition linéaire gloutonne de $G \setminus x$ basée sur w. Alors, $D = (\{x, w\}) \odot (D' \cup x)$ est une décomposition linéaire gloutonne de G basée sur x.
- Autrement (G n'est ni un cycle, ni une arête et le degré de x est au moins 2), si x est un sommet-séparateur qui n'appartient à aucune face interne de G, notons par C n'importe quelle composante connexe de $G \setminus x$. Soit D_1 une décomposition linéaire gloutonne de $G[C \cup x]$ basée sur x et soit D_2 une décomposition linéaire gloutonne de $G \setminus C$ basée sur x. Alors, $D = D_1 \odot D_2$ est une décomposition linéaire gloutonne de G basée sur x.
- Autrement (G n'est pas un cycle, ni une arête, x a un degré au moins 2 et x ne sépare pas G en plusieurs composantes connexes), supposons que $(v_1, \ldots, v_q)$ soit une face interne quelconque contenant x telle que $x = \{v_1, v_q\}$ (resp. $x = v_1$, si x est un sommet).
  - S'il existe un sommet-séparateur v_j pour $2 \leq j < q$ (resp. $2 \leq j \leq q$), notons par C la composante connexe de $G \setminus v_j$ qui ne contient pas x. Soient $G_1 = G[C \cup v_j]$ et $G_2 = G[V \setminus C]$. Soit D_1 une décomposition linéaire gloutonne de G_1 basée sur v_j, soit $D_2 = (X_1, \ldots, X_p)$ une décomposition linéaire gloutonne de G_2 basée sur x et soit $1 \leq h \leq p$ le plus grand entier tel que $\{v_{j-1}, v_j\} \subseteq X_h$ (si $v_j = v_2$, $1 \leq h \leq p$ est le plus grand entier tel que $\{v_{j-1}, v_j\} \subseteq X_h$ moins 1). Alors, $D = (X_1, \ldots, X_h) \odot (D_1 \cup (X_h \cap X_{h+1})) \odot (X_{h+1}, \ldots, X_p)$ est une décomposition linéaire gloutonne de G basée sur x.
  - Sinon, il existe $1 \leq j < q$ (resp., $1 \leq j \leq q$) tel que $f = \{v_j, v_{j+1}\} \in E_{int}$ où ni v_j ni v_{j+1} est un sommet-séparateur. Tout d'abord, si $1 \leq j < q - 1$ (resp., $1 \leq j < q$) (le cas $j = q - 1$ lorsque x est une arête et le cas $j = q$ lorsque x est un sommet sont traités plus tard), notons par C et C' les deux composantes connexes de $G \setminus f$ et, sans

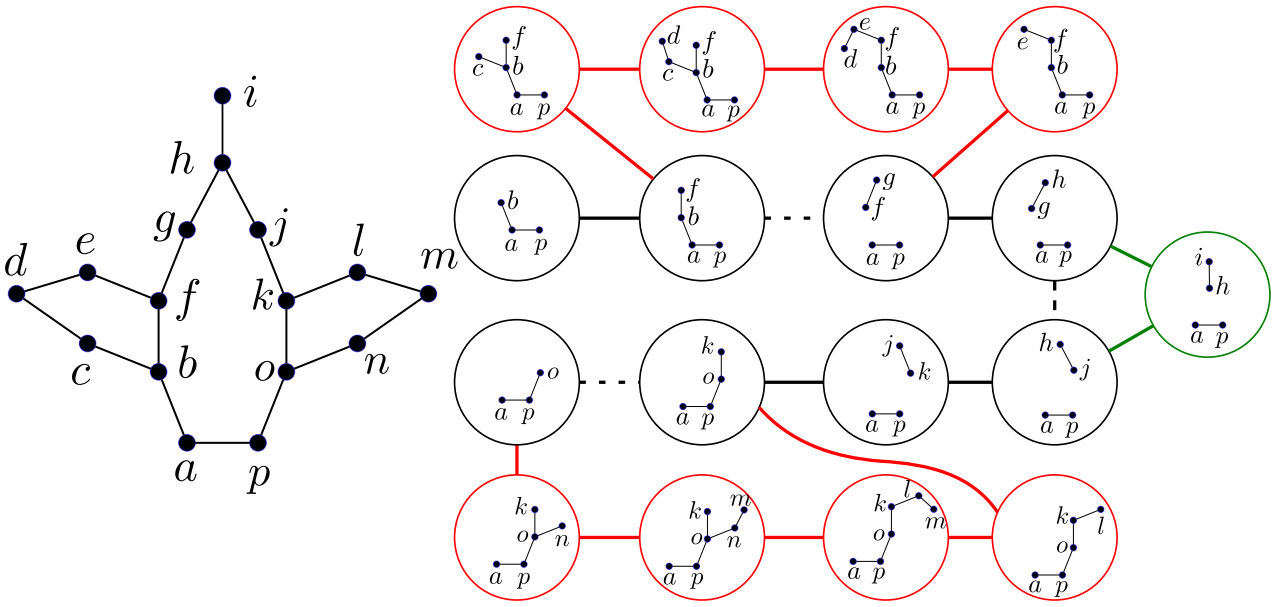


Figure 3.3 – Exemple de décomposition linéaire de G basée sur l'arête $\{a, p\}$. L'algorithme calcule d'abord la décomposition linéaire en noir. Les lignes discontinues représentent des arêtes de la décomposition linéaire noir qui sont supprimées pour insérer les autres décompositions. Ensuite, il insère la décomposition linéaire verte. Finalement, les décompositions linéaires rouges sont ajoutées séquentiellement.

perte de généralité, supposons que C intersecte x. Soit D_1 une décomposition linéaire gloutonne de $G[C' \cup f]$ basée sur f et soit $D_2 = (X_1, \ldots, X_p)$ une décomposition linéaire gloutonne de $G[C \cup f]$ basée sur x et soit $1 \leq h \leq p$ le plus petit entier tel que $f \subseteq X_h$. Alors, $D = (X_1, \ldots, X_h) \odot (D_1 \cup (X_h \cap X_{h+1})) \odot (X_{h+1}, \ldots, X_p)$ est une décomposition linéaire gloutonne de G basée sur x. Il reste donc le cas où $j = q - 1$ (resp., $j = q$). Soit C et C' les deux composantes connexes de $G \setminus f$ (resp., $G \setminus \{v_q, v_1\}$) et, sans perte de généralité, supposons que C intersecte x. Soit $D_1 = (X'_1, \ldots, X'_{p'})$ une décomposition gloutonne de $G[C' \cup f]$ basée sur f et soit $D_2 = (X_1, \ldots, X_p)$ une décomposition gloutonne de $G[C \cup f]$ basée sur x. Notons que X_{p-1} et X_p contiennent f et x par construction.
Alors, $D = (X_1, \ldots, X_{p-1}) \odot (D_1 \cup (X_{p-1} \cap X_p)) \odot (X_p)$ est une décomposition linéaire gloutonne de G basée sur x.

Notons que, dans le cas où $y \in x$, une $\{x, y\}$-décomposition linéaire optimale peut être calculée comme une $\{x, x\}$-décomposition linéaire $(X_1, \ldots, X_q)$ telle que, à la fin de la construction, nous supprimons $x \setminus y$ de $X_{i+1}, \ldots X_q$ où X_i est le premier sac tel que $N(x) \subseteq \bigcup_{1 \leq j \leq i} X_j$. Notons que par l'Assertion 3.3.0.1, le cas $x \in y$ est similaire.

Notons aussi qu'il n'existe pas de décomposition unique résultant du processus ci-dessus. Cependant, il est facile de montrer par induction (en particulier, chaque arête est considérée au maximum deux fois) que :

Lemme 3.3.9. *Toute séquence $D = (X_1, \ldots, X_p)$ renvoyée par l'algorithme Glouton est une $\{x, x\}$-décomposition linéaire de G. De plus, la complexité de l'algorithme Glouton est linéaire.*

De plus, pour tout $1 < i \leq p$, $|X_i \setminus X_{i-1}| \leq 1$ *(si* $i = 1$*, alors* $|X_1 \setminus x| \leq 1$*) et, si* $X_i \setminus X_{i-1} = \{u\}$*, alors* X_i *est un sous-ensemble de l'ensemble constitué de* u*, de l'un de ses voisins* u' *(avec* $\{u, u'\}$ *une arête extérieure), des sommets de l'arête interne* x *et de tous les sommets de chaque arête interne ou sommet qui sépare* u *de* x*.*

Démonstration. Le fait que l'algorithme renvoie une décomposition linéaire D se vérifie par induction sur le nombre d'opérations permettant de construire D. C'est clairement le cas si G est un cycle ou si G est une arête. Si x est de degré un (avec un voisin w), alors par induction, D' est une décomposition linéaire de $G \setminus x$ basé sur w et donc $D = (\{x, w\}) \odot (D' \cup x)$ est une décomposition linéaire. Si x est un sommet-séparateur (où C est n'importe quel composante connexe de $G \setminus x$), D_1 et D_2 sont des décompositions linéaires respectivement de $G[C \cup \{x\}]$ et de $G[G \setminus C]$ basées sur x et, puisque $(C \cup \{x\}) \cap (V \setminus C) = \{x\}$, alors $D_1 \odot D_2$ est une décomposition linéaire de G basée sur x. Considérons maintenant le cas où x est une arête extérieure $\{v_1, v_q\}$ (le cas où x est un sommet est similaire). S'il existe un sommet-séparateur v_j ($2 \leq j \leq q$) alors, par induction, D_1 est une décomposition linéaire de G_1 (chaque sommet et arête de G_1 apparaît dans un sac de D_1) où $G_1 = G[C \cup \{v_j\}]$ avec C correspondant à n'importe quelle composante connexe de $G \setminus v_j$ ne contenant pas x, et $D_2 = (X_1, \ldots, X_p)$ est une décomposition linéaire de G_2 (chaque sommet et arête de G_2 apparaît dans un sac de D_2) où $G_2 = G[V \setminus C]$. Alors, puisque $v_j \in X_h$, $(X_1, \ldots, X_h) \odot (D_1 \cup (X_h \cap X_{h+1})) \odot (X_{h+1}, \ldots, X_p)$ est une décomposition linéaire de G (la troisième propriété étant assurée puisque nous avons ajouté $X_h \cap X_{h+1}$ à tous les sacs de D_1). Le dernier cas peut être prouvé de façon similaire. De plus, par construction, il est clair que ce sont toutes des $\{x, x\}$-décompositions linéaires.

La preuve du second énoncé se fait aussi par induction sur le nombre d'opérations permettant de construire D. Pour le cas de base de l'induction, quand G correspond a un cycle ou une arête, alors le résultat est trivial. On peut donc supposer, par hypothèse d'induction, que pour tout sous-graphe G' de G, l'algorithme calcul une décomposition linéaire D' de G' respectant les conditions de l'énoncé. Si x est un sommet ayant un unique voisin w, rappelons que $D = (\{x, w\}) \odot (D' \cup x) = (X_1, \ldots, X_p)$. Par hypothèse d'induction, D' satisfait l'énoncé, et donc, $|X_i \setminus X_{i-1}| \leq 1$ pour tout $1 < i \leq p$. De plus, si $X_i \setminus X_{i-1} = \{u\}$, puisque chaque arête de sommet qui sépare u de w sépare aussi u de x, l'énoncé tient. Si x est un sommet-séparateur, alors l'énoncé tient par induction pour D_1 et D_2 et donc pour $D_1 \odot D_2$. Considérons enfin le dernier cas où $x = \{v_1, v_q\}$ (ou de façon similaire, x est un sommet) et $f = \{v_j, v_{j+1}\}$ est une arête-séparatrice (le cas où v_j est un sommet-séparateur est similaire). Soit $D = (X'_1, \ldots, X'_{p'}) = (X_1, \ldots, X_h) \odot (D_1 \cup (X_h \cap X_{h+1})) \odot (X_{h+1}, \ldots, X_p)$ définie comme dans l'algorithme (c'est-à-dire, $D_2 = (X_1, \ldots, X_p)$ est une décomposition linéaire gloutonne de $G[C \cup f]$ basée sur f où C et C' sont les composantes connexes de $G \setminus f$ avec $x \in V(C)$) et rappelons que D_1 est une décomposition linéaire basée sur f satisfaisant l'énoncé par hypothèse d'induction. Alors, le fait que $|X'_i \setminus X'_{i-1}| \leq 1$ pour tout $1 < i \leq p'$ tient clairement puisque $f \subseteq X_h$ (et le premier sac de D_1 contient au plus un sommet en plus de f). Considérons le cas où $X_i \setminus X_{i-1} = \{u\}$ et $u \in C'$. Puisque le résultat tient par induction pour D_2, alors $X_h \cap X_{h+1}$ contient f et tous les sommets de chaque arête ou sommet interne qui sépare f de x. Puisque, de plus, chaque arête ou sommet qui sépare u de f sépare aussi u de x, l'énoncé tient. Enfin, si $X_i \setminus X_{i-1} = \{u\}$ et $u \in C$, le résultat tient puisque D_2 satisfait la condition par induction. □

Une autre façon de voir la décomposition linéaire ci-dessus est la suivante. Soit $x = \{v_1, v_n\}$ (ou $x = v_1$ si x est un sommet) et soit $\mathcal{O} = (v_1, \ldots, v_n)$ un ordre DFS de V obtenu en partant de v_1, en suivant la face extérieure de G et en terminant par v_n (ou en revenant à v_1 si x est un

sommet). En d'autres termes, les sommets sont ordonnés dans l'ordre où ils sont rencontrés le long de la face externe en partant de v_1. Notons que, à l'exception de l'ordre inverse, cet ordre est unique si G est 2-connexe. Une décomposition linéaire de G basée sur x est donc similaire à n'importe quelle décomposition linéaire $D(\mathcal{O})$ construite comme décrit dans la section 3.1. En effet, ces décompositions ne sont pas complètement équivalentes. Plus précisément, dans les décompositions linéaires de G basée sur x, quand on ajoute un sommet v tel qu'il existe $\{v, u\} \in E_{int}$ avec $v <_{\mathcal{O}} u$, on ajoute u avant tous les sommets entre v et u dans $\mathcal{O}$. Ce n'est pas le cas pour les décompositions linéaires obtenues à partir de l'ordre $\mathcal{O}$ sur les sommets.

Théorème 3.3.10. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes et que $x = y$, $y \in x$, ou $x \in y$.*

Une $\{x, y\}$-décomposition linéaire optimale de G peut être calculée en temps linéaire (en temps $O(|E|)$).

Démonstration. Considérons d'abord le cas où $x = y$. Remarquons que, puisque x appartient à chaque sac de chaque $\{x, x\}$-décomposition linéaire de G, alors $p\ell(G, x, x) \geq \max_{w \in V} dist(w, x)$ (si $x = \{u, v\} \in E$, $dist(w, x) = \max\{dist(w, u), dist(w, v)\}$).

Soit $D = (X_1, \ldots, X_p)$ une séquence de sous-ensemble de sommet calculée par l'algorithme $Glouton$ avec G comme entrée. Par le Lemme 3.3.9, D est une $\{x, x\}$-décomposition linéaire de G calculée en temps linéaire. Il nous reste donc à prouver que D est optimal. Pour arriver à une contradiction, supposons, pour $1 < i \leq p$ tel que $|X_i \setminus X_{i-1}| > 0$, qu'il existe deux sommets v et v' dans $X_i \setminus x$ tels que $dist(v', v) > \max_{w \in V} dist(w, x)$. Notons que puisque $v, v' \notin x$, nous obtenons que $max_{w \in V} dist(w, x) \geq max(dist(v, x), dist(v', x)) \geq 1$. Par conséquent, v et v' ne peuvent pas être adjacents (v et v' ne peuvent pas être dans la même arête-séparatrice de u et x, et ils ne peuvent pas être u et u').

Pour l'instant, supposons que x est une arête. Ainsi, par le Lemme 3.3.9 et sans perte de généralité, nous pouvons supposer que $v \in S_1$, un séparateur de x et u (resp., ou $v \in \{u, u'\}$), et $v' \in S_2$, un séparateur de $x \setminus S_2$ et u tel que v n'est pas dans la composante connexe de $G \setminus S_2$ qui contient $x \setminus S_2$. Par conséquent, si S_2 est un sommet-séparateur, alors chaque chemin entre v et x contient v' et donc $dist(v, v') \leq dist(v, x)$, une contradiction. Sinon, S_2 est une arête-séparatrice (c'est-à-dire, S_2 dans E_{int}). Notons par v_x le sommet de x tel que $dist(v, x)$ est maximisée. Si $v' \in (v, v_x)$-chemin, alors $dist(v, v') \leq dist(v, v_x)$, une contradiction. Sinon, le seul sommet v^* dans $S_2 \setminus v'$ est dans chaque (v, v_x)-chemin et donc $dist(v, v') \leq dist(v, v^*) + 1 \leq dist(v, v_x)$ (même si $v^* \in x$), une contradiction. Notons qu'il reste le cas où x est un sommet. Notons que si nous pouvons supposer que $v \in S_1$, un séparateur de x et u (resp., $S_1 = N[u]$), et $v' \in S_2$, un séparateur de x et u tel que v n'est pas dans la composante connexe de $G \setminus S_2$ qui contient x, alors les mêmes arguments que précédemment s'appliquent (parce que $min_{s \in S_2} dist(s, x) \geq 1$), ce qui implique une contradiction. Sinon, v' et x sont adjacents et $G \setminus \{v', x\}$ n'est pas connexe (c'est-à-dire $\{v', x\} \in E_{int}$). Notons par $\mathcal{C}_{v'}$ l'ensemble des composantes connexes de $G \setminus v'$ ne contenant pas x (si une telle composante existe). Soient C_1 et C_2 les deux composantes connexes de $G \setminus (\{x, v'\} \bigcup_{C \in \mathcal{C}_{v'}} V(C))$. Supposons que $\{v_1 \ldots, v_q\} \subseteq C_1 \cup \{v', x\}$ tel que $x = v_1$ et $v' = v_q$. Notons que v' ne peut pas faire partie d'une autre arête-séparatrice, puisque nous avons déjà considéré ce cas. Par conséquent, $v \in C \in \mathcal{C}_{v'}$ (sinon, par le Lemme 3.3.9, il n'y a pas de sac contenant v et v', une contradiction) Rappelons que nous avons supposé que $dist(v, v') = k > \max_{w \in V} dist(w, x)$ pour arriver à une contradiction.

De plus, pour tout sac X contenant v', $X \cap C_1 = N_{C_1}(v')$, $X \cap C_2 = N_{C_2}(v')$. Cela implique donc que $v \in C \in \mathcal{C}_{v'}$. Notons que tout chemin (v, x) contient v', ce qui implique que $dist(v, v') \leq dist(v, x)$, ce qui est une contradiction.

Par conséquent, le diamètre $\max_{u,v \in X_i} dist_G(u, v)$ de X_i est au plus $\max_{v \in V(G)} dist(v, x)$. Par conséquent, D est une décomposition linéaire optimale de G.

Supposons maintenant que $x = \{u, v\}$ et $y = v$ (le cas où $x \in y$ est symétrique). Remarquons que, puisque v appartient à chaque sac de chaque $\{x, y\}$-décomposition linéaire de G, alors $p\ell(G, x, y) \geq \max_{w \in V} dist(w, v)$. Soit F la face interne contenant x (une telle face existe et est unique puisque $G \setminus x$ et $G \setminus y$ sont connexes, $\{u, v\} \in E_{out}$ et $G \neq x$) et soit $w \neq v$ le second voisin de u dans F. Soit C la composante connexe de $G \setminus \{u, w\}$ qui ne contient pas v et telle que $N(C) = \{u, w\}$ (éventuellement $C = \emptyset$). Notons que, pour tout $h \in C$, $dist_G(h, u) \leq dist_G(h, v)$. Rappelons qu'une $\{x, y\}$-décomposition linéaire D' de G calculé par l'algorithme *Glouton* est obtenu d'une $\{x, x\}$-décomposition linéaire $D = (X_1, \ldots, X_p)$ de G calculée par l'algorithme telle que $D' = (X_1, \ldots, X_i, X_{i+1} \setminus \{u\}, \ldots, X_p \setminus \{u\})$ où X_i est le premier sac tel que $N(u) \subseteq \bigcup_{1 \leq j \leq i} X_j$. Par conséquent, D' est une décomposition de longueur au plus $\max_{w \in V} dist(w, v)$.

□

3.3.2 Cas où x et y sont "autour" d'une même face

Dans cette section, nous considérons le dernier cas restant qui est beaucoup plus technique que les précédents. Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe à n sommets et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G (c'est-à-dire, ils ne sont pas séparés par une arête ou un sommet).

Dans ce cadre, nous montrons d'abord qu'il existe toujours une décomposition linéaire presque optimale de $\{x, y\}$ satisfaisant des propriétés spécifiques (d'abord, *contiguë*, puis *g-contiguë* (Section 3.3.2.1) et enfin *de Gauche-à-Droite g-contiguë* (Section 3.3.2.2), voir les définitions formelles plus bas). Nous présentons ensuite un algorithme de programmation dynamique permettant de calculer une telle $\{x, y\}$-décomposition linéaire optimale de G (Section 3.3.2.3). Nous avons d'abord besoin d'une notation supplémentaire présentée dans la Section 3.3.2.

Notation. Soient $x = \{x_1, x_2\}$ et $y = \{y_1, y_2\}$ (pour simplifier la présentation, nous supposons que $x, y \in E_{out}$, c'est-à-dire, si $x \in V$ et/ou $y \in V$, il suffit de fixer $x_1 = x_2$ et/ou $y_1 = y_2$). Soit F la face interne contenant x et y constituée de deux chemins intérieurement disjoints P_{up} entre x_1 et y_1 et P_{down} entre x_2 et y_2 (quand x et y sont des arêtes, ils peuvent partager un même sommet, auquel cas, nous supposons que $x_2 = y_2$ et que P_{down} est réduit à x_2).

Soit $\mathcal{C}$ l'ensemble des composantes connexes de $G \setminus F$. Soit $\mathcal{C}_{up}$ (resp., $\mathcal{C}_{down}$) l'ensemble des composantes connexes C de $G \setminus F$ telles que $N(C) \subseteq V(P_{up})$ (resp., $N(C) \subseteq V(P_{down})$). Pour toute composante $C \in \mathcal{C}_{up} \cup \mathcal{C}_{down}$, définissons $\bar{C} = C \cup N(C)$ et $s_C = N(C)$. Notons que, puisque G est un graphe planaire extérieur et que F est une face, alors s_C est soit une arête, soit un sommet de F (en abusant de la notation, s_C sera soit une arête, soit un sommet de F).

Lemme 3.3.11. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G.*

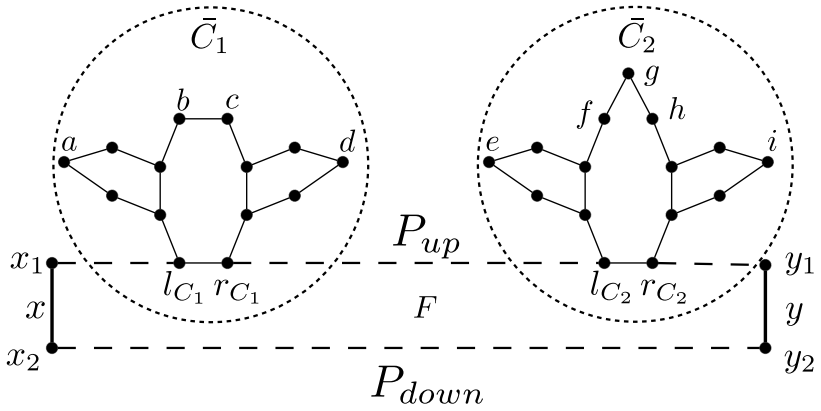


Figure 3.4 – Exemple de graphe G et des composantes $C_1, C_2 \in \mathcal{C}_{up}$ de $G \setminus F$. $\mathcal{M}_{C_1} = \{a, b, c, d\}$ et $\mathcal{M}_{C_2} = \{e, f, g, h, i\}$. C_2 est une composante convenable et $h^*_{C_2} = g$. Au contraire, C_1 n'est pas une composante convenable. Les lignes discontinues sont utilisées pour représenter des chemins.

Si $p\ell(G, x, y) \leq k$, alors il existe une $\{x, y\}$-décomposition linéaire $(X_1, \ldots, X_p)$ de G de longueur au plus k telle que, pour tout $C \in \mathcal{C}$, si $X_i \cap C \neq \emptyset$, alors $s_C \in X_i$ (ou $s_C \subseteq X_i$ si s_C est une arête).

Démonstration. Considérons une $\{x, y\}$-décomposition linéaire $D' = (X_1, \ldots, X_p)$ de G de longueur au plus k qui maximise le nombre de composantes de $C \in \mathcal{C}$ qui satisfont la propriété suivante : si $X_i \cap C \neq \emptyset$, alors $s_C \in X_i$.

Pour arriver à une contradiction, supposons qu'il existe une composante $C \in \mathcal{C}$ qui ne satisfait pas cette propriété. Supposons, sans perte de généralité, que $C \in \mathcal{C}_{up}$. Soit $1 \leq i \leq j \leq p$ le plus petit (respectivement le plus grand) entier tel que $C \cap X_i \neq \emptyset$ ($C \cap X_j \neq \emptyset$, resp.). Notons que, pour chaque $i \leq h \leq j$, $X_h \cap V(P_{down}) \neq \emptyset$, et, en outre, pour chaque sommet $u \in s_C$, $v \in C$ et $w \in V(P_{down})$, $dist(u, w) \leq dist(v, w)$ et $dist(v, u) \leq dist(v, w)$.

Alors, $(X_1, \ldots, X_{i-1}, X_i \cup s_C, \ldots, X_j \cup s_C, X_{j+1}, \ldots, X_p)$ est une $\{x, y\}$-décomposition linéaire de G de longueur au plus k, ce qui contredit la maximalité de D'. □

Soit C une composante de $G \setminus F$.

— Si s_C est un sommet, définissons $d_C = \max_{v \in C \cup s_C} dist(v, s_C)$ et notons par h^*_C un sommet tel que $dist(h^*_C, s_C) = d_C$.

— Si s_C est une arête, notons ses sommets par $\{l_C, r_C\}$, définissons $d_C = \max_{v \in C \cup s_C} \max\{dist(v, r_C), dist(v, l_C)\}$ et définissons $\mathcal{M}_C = \{v \in C \cup s_C \mid \max\{dist(v, r_C), dist(v, l_C)\} = d_C\}$. Notons que, pour tout $v \in C \cup s_C$, $dist(v, l_C) - 1 \leq dist(v, r_C) \leq dist(v, l_C) + 1$. S'il existe $v \in \mathcal{M}_C$ tel que $dist(v, r_C) = dist(v, l_C) = d_C$, alors notons par h^*_C un tel sommet. Dans le cas contraire, on note par h^*_C n'importe quel sommet de $\mathcal{M}_C$.

Si s_C est un sommet ou s'il existe un sommet $v \in \mathcal{M}_C$ avec $dist(v, r_C) = dist(v, l_C) = d_C$, nous qualifierons la composante C de *convenable*.

Assertion 3.3.11.1 – Soit C une composante connexe de $G \setminus F$ et soient $v \in C$ et $u \in G \setminus C$. Si C est convenable ou si $v \notin \mathcal{M}_C$, alors $dist(v, u) \leq dist(h^*_C, u)$. Sinon, $dist(v, u) \leq dist(h^*_C, u) + 1$.

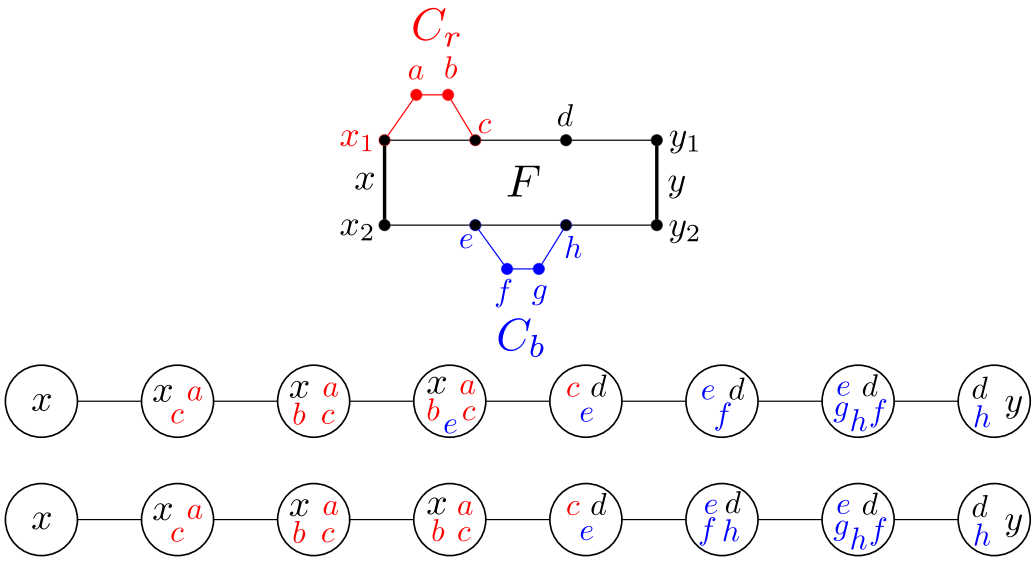


Figure 3.5 – Exemple de graphe G (en haut), d'une $\{x, y\}$-décomposition linéaire $D = (X_1, \ldots, X_8)$ non contiguë de G (au milieu), et d'une $\{x, y\}$-décomposition linéaire D' contiguë de G (en bas). La composante rouge n'est pas contiguë dans D car les sacs contenant des sommets de la composantes rouge (X_3 et X_4) ne contiennent pas le même sous-ensemble de sommets de F. De plus, la composante bleu C_b n'est pas contiguë dans D car un (X_6) des sacs contenant des sommets de la composante bleu (X_6 et X_7) ne contient pas s_{C_b} (h). Notons que dans D', $a_{C_r} = 3, b_{C_r} = 4, a_{C_b} = 6$ et $b_{C_b} = 7$.

3.3.2.1 Vers des décompositions g-contiguës

Soit $D = (X_1, \ldots, X_p)$ une $\{x, y\}$-décomposition linéaire de G et soit $C \in \mathcal{C}$. La composante C est dite *contigüe (par rapport à D)* s'il existe $1 \leq a_C \leq b_C \leq p$ tel que (1) $C \cap X_i = \emptyset$ si et seulement si $i \notin \{a_C, \ldots, b_C\}$, et $s_C \subseteq X_j$ pour tout $a_C \leq j \leq b_C$, et (2) il existe $R_C \subseteq V(F)$ tel que $X_i \setminus C = R_C$ pour tout $a_C \leq i \leq b_C$. Intuitivement, C est contiguë par rapport à D si une fois qu'un sommet de C a été introduit dans D, aucun sommet de $G \setminus C$ ne peut être introduit dans D avant que tous les sommets de C aient été introduits.

Une décomposition linéaire D est *contiguë* si chaque composante de $G \setminus F$ est contiguë par rapport à D.

Dans ce qui suit, nous montrons qu'il existe toujours une $\{x, y\}$-décomposition linéaire optimale (ou presque) de G qui est contiguë. Pour commencer, dans le cas ou toutes les composantes de $\mathcal{C}$ sont convenables, nous prouvons qu'il existe un $\{x, y\}$-décomposition linéaire optimale de G.

Théorème 3.3.12. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y appartiennent à la même face interne F de G. De plus, supposons que toutes les composantes de $\mathcal{C}$ sont convenables.*

Si $p\ell(G, x, y) \leq k$, alors il existe une $\{x, y\}$-décomposition linéaire contiguë D' de G de longueur au plus k. De plus, pour tout sac X de D', il existe au plus une composante C de $G \setminus F$ telle que $X \setminus F \subseteq C$.

Démonstration. Étant donné une $\{x,y\}$-décomposition linéaire $D = (X_1, \ldots, X_p)$ de G de longueur au plus k, notons par $\mathcal{Q}(D) \subseteq \mathcal{C}$ l'ensemble de composantes C de $\mathcal{C}$ telles qu'il existe $1 \leq a_C \leq b_C \leq p$ tel que :

1. $C \cap X_i \neq \emptyset$ si et seulement si $a_C \leq i \leq b_C$, et $s_C \subseteq X_j$ pour tout $a_C \leq j \leq b_C$, et ;
2. il existe $R_C \subseteq F \cup \bigcup_{C' \notin \mathcal{Q}(D)} C'$ tel que, pour tout $a_C \leq i \leq b_C$, $X_i \setminus C = R_C$.

Soit D une $\{x,y\}$-décomposition linéaire de G de longueur k. Alors, un tel ensemble $\mathcal{Q}(D)$ est bien défini (éventuellement, $\mathcal{Q}(D)$ est vide).

Considérons une $\{x,y\}$-décomposition linéaire $D' = (X_1, \ldots, X_p)$ de G de longueur au plus k, qui maximise $|\mathcal{Q}(D')|$. Si $\mathcal{Q}(D') = \mathcal{C}$, alors D' est la décomposition linéaire désirée.

Pour arriver à une contradiction, supposons que $\mathcal{C} \setminus \mathcal{Q}(D') \neq \emptyset$. Soit $C \in \mathcal{C} \setminus \mathcal{Q}(D')$ et soit $1 \leq i \leq p$ le plus petit indice tel que $h^*_C \in X_i$. Notons que, par le Lemme 3.3.11, nous pouvons supposer que $s_C \subseteq X_i$. Notons aussi qu'il n'existe pas de $C' \in \mathcal{Q}(D')$ tel que $a_{C'} < i \leq b_{C'}$ par la seconde propriété ci-dessus.

Soit $Y = (D' \cap C) \cup (s_C \cup (X_{i-1} \cap X_i) \setminus C)$. Par l'Assertion 3.3.11.1 et le fait que C est convenable, $\ell(Y) \leq k$.

Par conséquent, $D'' = (X_1 \setminus C, \ldots, X_{i-1} \setminus C) \odot Y \odot (X_i \setminus C, \ldots, X_p \setminus C)$ est une $\{x,y\}$-décomposition linéaire de G, de longueur au plus k, et telle que $\mathcal{Q}(D') \cup \{C\} \subseteq \mathcal{Q}(D'')$. Ceci contredit la maximalité de $|\mathcal{Q}(D')|$. □

Ensuite, nous considérons le cas général, où toutes les composantes de $G \setminus F$ ne sont pas forcément convenables. Nous montrons que cela (rendre les composantes non convenables contiguës) peut avoir un coût puisque la longueur de la décomposition contiguë de G obtenu peut augmenter de 1 par rapport à la longueur linéaire de G. Plus loin, nous montrons que cette augmentation ne peut être évitée dans certains cas, c'est-à-dire, il existe des graphes qui n'ont pas de $\{x,y\}$-décomposition linéaire contiguë de longueur $p\ell(G,x,y)$.

Théorème 3.3.13. *Soit $G = (V,E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$, et x et y se trouvent sur la même face interne F de G.*

Si $p\ell(G,x,y) \leq k$, alors il existe une $\{x,y\}$-décomposition linéaire contiguë D' de G de longueur au plus $k+1$. De plus, pour toute composante $C \in \mathcal{C}$ et tout indice $a_C \leq i \leq b_C$, si $\ell(X_i) = k+1$, alors il existe $h^*_C \in X_i \cap \mathcal{M}_C$ et $v \in X_i \setminus C \subseteq V(F)$ tels que $dist(h^*_C, v) = k+1$.

Démonstration. Soit $D = (X_1, \ldots, X_p)$, n'importe quelle $\{x,y\}$-décomposition linéaire de G de longueur au plus $k+1$, telle que, pour chaque $1 \leq i \leq p$, chaque $C \in \mathcal{C}$ et chaque $u, v \in C \cap X_i$, nous obtenons que $dist(u,v) \leq k$ (Propriété $(*)$).

Soit $\mathcal{Q}(D) \subseteq \mathcal{C}$ l'ensemble des composantes C de $\mathcal{C}$ telles qu'il existe $1 \leq a_C \leq b_C \leq p$ tel que :

1. $C \cap X_i \neq \emptyset$ si et seulement si $a_C \leq i \leq b_C$, et $s_C \subseteq X_j$ pour tout $a_C \leq j \leq b_C$, et ;
2. il existe $R_C \subseteq F \cup \bigcup_{C' \notin \mathcal{Q}(D)} C'$ tel que, pour tout indice $a_C \leq i \leq b_C$, $X_i \setminus C = R_C$, et ;
3. pour chaque $1 \leq j \leq p$ tel qu'il n'existe pas de $C \in \mathcal{Q}(D)$ avec $a_C \leq j \leq b_C$, alors $\ell(X_j) \leq k$. De plus, s'il existe $C, C' \in \mathcal{Q}(D)$ avec $b_C = j = a_{C'} - 1$, alors $\ell(X_j \cap X_{j+1}) \leq k$.

Notons que n'importe quelle $\{x,y\}$-décomposition linéaire D de G de longueur k satisfait trivialement la propriété $(*)$ et $\mathcal{Q}(D)$ est bien défini (mais possiblement vide).

Considérons une décomposition linéaire $D' = (X_1, \ldots, X_p)$ de G de longueur au plus $k+1$ satisfaisant la propriété $(*)$, et qui maximise $|\mathcal{Q}(D')|$. Si $\mathcal{Q}(D') = \mathcal{C}$, alors D' est la décomposition désirée.

Pour arriver à une contradiction, supposons que $\mathcal{C} \setminus \mathcal{Q}(D') \neq \emptyset$. Soient $C \in \mathcal{C} \setminus \mathcal{Q}(D')$ et $1 \leq i \leq p$ le plus petit indice tel que $h^*_C \in X_i$. Nous pouvons supposer, grâce au Lemme 3.3.11, que $s_C \subseteq X_i$. Notons aussi qu'il n'existe pas de $C' \in \mathcal{Q}(D')$ telle que $a_{C'} < i \leq b_{C'}$ par la deuxième propriété de la définition de $\mathcal{Q}(D')$.

Définissons $Y = (D' \cap C) \cup (s_C \cup (X_{i-1} \cap X_i) \setminus C)$. Par la propriété $(*)$, l'Assertion 3.3.11.1 et la troisième propriété de D' ci-dessus, $\ell(Y) \leq k+1$ (si C est convenable, on a même $\ell(Y) \leq k$). Plus précisément, ces différentes propriétés impliquent que s'il existe un sac X de Y tel que $\ell(X) = k+1$, alors il existe $h^*_C \in \mathcal{M}_C$ et $v \in X_{i-1} \cap X_i$ tels que $dist(h^*_C, v) = k+1$.

Par conséquent, $D'' = (X_1 \setminus C, \ldots, X_{i-1} \setminus C) \odot Y \odot (X_i \setminus C, \ldots, X_p \setminus C)$ est une $\{x,y\}$-décomposition linéaire de longueur au plus $k+1$ et satisfaisant la propriété $(*)$, et telle que $\mathcal{Q}(D') \cup \{C\} \subseteq \mathcal{Q}(D'')$. Ceci contredit la maximalité de $|\mathcal{Q}(D')|$. □

Malheureusement, le théorème précédent ne peut pas être amélioré, car il existe des graphes planaires extérieurs 2-connexes G et des arêtes $x, y \in E_{out}$, tels que toute $\{x,y\}$-décomposition linéaire contiguë a une longueur au moins égale à $p\ell(G,x,y)+1$.

Lemme 3.3.14. *Il existe des graphes planaires extérieurs 2-connexes G et des arêtes $x, y \in E_{out}$ tels que toute $\{x,y\}$-décomposition linéaire contiguë de G a une longueur au moins égale $p\ell(G,x,y)+1$.*

Démonstration. Soit G, le graphe représenté sur la Figure 3.6. Soit C l'unique composante de $G \setminus F$. Notons les sommets de s_C par l_C et r_C comme indiqué dans la Figure 3.6. Décrivons d'abord une $\{x,y\}$-décomposition linéaire D de G de longueur 10. Soient $X_1 = \{x\}$, $X_2 = C_l \cup C_m \cup L$ où L est le (x_2, r_c)-chemin dans F ne contenant pas y_1 et y_2. Soit $X_3 = C_m \cup down$ où $down$ est le (x_2, y_2)-chemin dans F ne contenant pas x_1 et y_1. Soit $X_4 = C_m \cup C_r \cup R$ où R est le chemin (y_2, l_c) dans F ne contenant pas x_1 et x_2. Soit $X_5 = \{y\}$. Remarquons que $D = (X_1, \ldots, X_5)$ est une $\{x,y\}$-décomposition linéaire de longueur 10. De plus, D n'est pas contiguë puisque X_2 et X_4 contiennent tous deux des sommets de C, mais $X_2 \setminus V(C) \neq X_4 \setminus V(C)$.

Appliquons le lemme 3.3.13 à D. Remarquons que dans la $\{x,y\}$-décomposition linéaire contiguë $\{x,y\}$ obtenue, chaque sommet de $V(C)$ est soit contenu dans un sac avec les sommets de L ou les sommets de R. Par conséquent, suivant le choix du sommet $h^*_c \in \mathcal{M}_C$, soit l et y_2 sont tous deux dans un sac, soit r et x_2 sont tous deux dans un sac. Dans tous les cas, il existe un sac contenant deux sommets à distance 11. □

Une décomposition linéaire $D = (X_1, \ldots, X_p)$ contiguë de G est dite *g-contiguë* si, pour tout $C \in \mathcal{C}$, $(D_{a_C} \cap \bar{C}, \ldots, D_{b_C} \cap \bar{C})$ est une décomposition linéaire gloutonne de $G[\bar{C}] = G[C \cup s_C]$ basée sur s_C (voir Section 3.3.1.3). Rappelons que ces décompositions linéaires gloutonnes sont des $\{s_C, s_C\}$-décompositions linéaires optimales de $G[\bar{C}]$.

Théorème 3.3.15. *Soit $G = (V,E)$ un graphe planaire extérieur simple et connexe et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G.*

Si $p\ell(G,x,y) \leq k$, alors il existe une $\{x,y\}$-décomposition linéaire g-contiguë $D = (X_1, \ldots, X_p)$ de G de longueur au plus $k+1$. De plus, pour tout $C \in \mathcal{C}$ et $a_C \leq i \leq b_C$, $\ell(X_i) = k+1$ seulement si $h^*_C \in X_i$ pour n'importe quel sommet $h^*_C \in \mathcal{M}_C$.

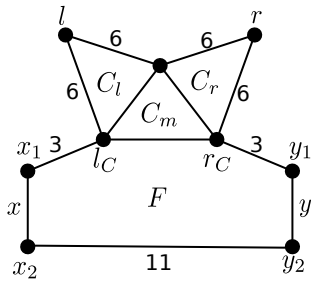


Figure 3.6 – Exemple d'un graphe G avec les arêtes $x, y \in E_{out}$ tels que toute $\{x, y\}$-décomposition linéaire contiguë de G a une longueur au moins égale à $p\ell(G, x, y) + 1$. Le poids des arêtes représente la longueur d'un chemin entre les deux extrémités des arêtes.

Démonstration. Par le Théorème 3.3.13, il existe une $\{x, y\}$-décomposition linéaire contiguë $D' = (X_1, \ldots, X_p)$ de G de longueur au plus $k + 1$.

Itérativement, pour chaque composante $C \in \mathcal{C}$, remplacer D' par $(X_1, \ldots, X_{a_C-1}) \odot \mathcal{G}(C) \cup (X_{a_C-1} \cap X_{b_C+1}) \odot (X_{b_C+1}, \ldots, X_p)$ où $\mathcal{G}(C)$ est une $\{s_C, s_C\}$-décomposition linéaire gloutonne optimale de $G[\bar{C}] = G[C \cup s_C]$ basée sur s_C. Cela reste une $\{x, y\}$-décomposition linéaire contiguë de G puisque, pour tout $a_C \leq i \leq b_C$, $s_C \in X_i$ et $X_i \setminus C \subseteq V(F)$. Notons aussi que, par l'Assertion 3.3.11.1, $\ell(D') \leq k + 1$.

De plus, par le Théorème 3.3.10 et le Théorème 3.3.13, deux sommets u et v dans un sac X_i ($a_C \leq i \leq b_C$ pour un certain $C \in \mathcal{C}$) peuvent être à distance $k + 1$ seulement si $u \in \mathcal{M}_C$ et $v \in V(F)$. □

3.3.2.2 Vers des décompositions de Gauche-à-Droite g-contiguës

Rappelons que nous considérons un graphe planaire extérieur simple et connexe à n sommets $G = (V, E)$, et des sommets et/ou arêtes $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G.

Pour le reste de cette section, notons les sommets de x par $\{x_1, x_2\}$ si x est une arête (ou $x = x_1 = x_2$ si x est un sommet) et notons les sommets de y de la même façon. Notons les deux chemins intérieurement disjoints, qui consistent en tous les sommets de la face F, par $P_{up} = (x_1 = u_1, \ldots, u_t = y_1)$ et par $P_{down} = (x_2 = d_1, \ldots, d_s = y_2)$. Notons qu'il est possible que $x_1 = y_1$ ou $x_2 = y_2$, mais pas les deux (dans ce qui suit, nous supposons que $x_1 \neq y_1$).

Pour tout $C \in \mathcal{C}_{up}$ (c'est-à-dire, telle que $s_C \in V(P_{up})$ ou $s_C \subseteq V(P_{up})$), notons par l_C (resp, r_C) le sommet de s_C le plus proche (resp. le plus éloigné) de x_1 dans P_{up} (ou $l_C = r_C = s_C$ si s_C est un sommet). De même, pour tout $C \in \mathcal{C}_{down}$ (c'est-à-dire, telle que $s_C \in V(P_{down})$ ou $s_C \subseteq V(P_{down})$), nous notons l_C (resp., r_C) le sommet de s_C le plus proche (resp. le plus éloigné) de x_2 dans P_{down} (ou $l_C = r_C = s_C$ si s_C est un sommet).

Une arête $e \in E(F) \setminus \{x, y\}$ est dite *triviale* s'il n'existe pas $C \in \mathcal{C}$ tel que $e = s_C$ (il est quand même possible qu'une extrémité de e soit un sommet-séparateur). Bien que les arêtes triviales ne soient liées à aucune composante de $G \setminus F$, nous devons les inclure dans l'analyse qui suit. Afin d'unifier les notations, définissons $\bar{\mathcal{C}} = \{\bar{C} = C \cup s_C \mid C \in \mathcal{C}\} \cup \{e \in E(F) \setminus \{x, y\} \mid$ e est une arête triviale$\}$. Intuitivement, chaque arête triviale $e \in E(F) \setminus \{x, y\}$ peut être considérée comme $e = s_C$ pour une composante fictive C vide, c'est-à-dire, $C = \emptyset$. De la même manière,

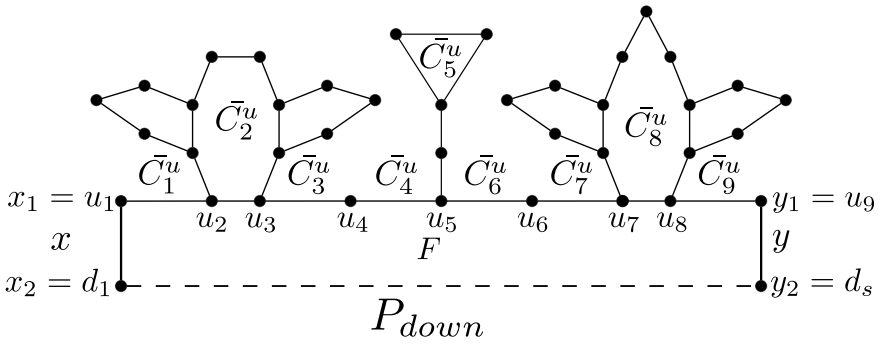


Figure 3.7 – Notation de la section 3.3.2.2

définissons $\bar{C}_{up} = \{\bar{C} = C \cup s_C \mid C \in \mathcal{C}_{up}\} \cup \{e \in E(P_{up}) \mid \text{e est une arête triviale}\}$ et $\bar{\mathcal{C}}_{down} = \{\bar{C} = C \cup s_C \mid C \in \mathcal{C}_{down}\} \cup \{e \in E(P_{down}) \mid \text{e est une arête triviale}\}$. Notons que $\bar{\mathcal{C}} = \bar{\mathcal{C}}_{up} \cup \bar{\mathcal{C}}_{down}$.

Soit $\mathcal{O}_{up} = (C^u_1, \ldots, C^u_{s'})$ n'importe quel ordre de $\bar{\mathcal{C}}_{up}$ tel que, si $l_{C^u_i}$ est strictement plus proche de x_1 dans P_{up} que $l_{C^u_j}$ et/ou si $r_{C^u_i}$ est strictement plus proche de x_1 dans P_{up} que $r_{C^u_j}$, alors $i < j$. De la même façon, supposons que $\mathcal{O}_{down} = (C^d_1, \ldots, C^d_{s'})$ soit un ordre quelconque de $\bar{\mathcal{C}}_{down}$ tel que, si $l_{C^u_i}$ est strictement plus proche de x_2 dans P_{down} que $l_{C^u_j}$ et/ou si $r_{C^u_i}$ est strictement plus proche de x_1 dans P_{up} que $r_{C^u_j}$, alors $i < j$. Intuitivement, nous ordonnons les composantes de $\mathcal{C}_{up}$ et les arêtes triviales de P_{up} de x_1 à y_1 (resp. de $\mathcal{C}_{down}$ et les arêtes triviales de P_{down} de x_2 à y_2), de "gauche à droite" ("de x à y"). Notons que pour deux composantes $C, C' \in \mathcal{C}$ telles que $s_C = s_{C'} \in V$, l'ordre relatif entre C et C' dans $\mathcal{O}_{up}$ (resp., dans $\mathcal{O}_{down}$ si $s_C \in V(P_{down})$) n'est pas pertinent, mais pour tous sommets $v \in C$, toutes les composantes connexes C' avec $s_C = v$ sont consécutives dans $\mathcal{O}_{up}$ (resp., dans $\mathcal{O}_{down}$).

Dans cette section, nous ne considérons que des $\{x, y\}$-décompositions linéaires $D = (X_1, \ldots, X_p)$ g-contiguës de G. En d'autres termes, pour toute composante $C \in \mathcal{C}$, il existe $1 \leq a_C \leq b_C \leq p$, et un intervalle $I_C = [a_C, b_C]$ tels que $X_i \cap C \neq \emptyset$ si et seulement si $i \in I_C$, $s_C \in X_i$ pour tout indice $i \in I_C$ et $X_i \setminus C \subseteq F$ pour tout $i \in I_C$. En particulier, $I_C \cap I_{C'} = \emptyset$ pour toutes les composantes $C, C' \in \mathcal{C}$ distinctes. On dit que C apparaît dans D dans le sac X_{a_C}. De plus, $(X_{a_C} \cap \bar{C}, \ldots, X_{b_C} \cap \bar{C})$ est une $\{s_C, s_C\}$-décomposition linéaire gloutonne de $G[\bar{C}]$ basée sur s_C. Rappelons également que nous pouvons supposer que la propriété du dernier énoncé du Théorème 3.3.15 est valide. De plus, pour tout $i \in I_C$, $X_i \setminus C \subseteq F$.

Par définition, D induit un ordre total $\mathcal{O}_D = (\bar{C}_1, \ldots, \bar{C}_b)$ sur $\bar{\mathcal{C}}$ tel que, pour tout $1 \leq i < j \leq b$, $\bar{C}_i$ apparaît dans D avant $\bar{C}_j$ (c'est-à-dire, $b_{C_i} < a_{C_j}$). Notre but est de considérer de telles $\{x, y\}$-décompositions linéaires D g-contiguës telles que les ordres totaux $\mathcal{O}_D$ qu'elles induisent satisfont une propriété supplémentaire définie ci-dessous.

Soit $H = H^1 \cup H^2$ un ensemble de base avec $H^1 \cap H^2 = \emptyset$. Soit $\mathcal{O} = (H_1, \ldots, H_q)$ un ordre total sur H, et $\mathcal{O}^i$ un ordre total de H^i pour $i \in \{1, 2\}$. Un préfixe $\mathcal{P} = (H_1, \ldots, H_{q'})$ $(q' \leq q)$ de H est *compatible* avec $\mathcal{O}^i$ si $\mathcal{P} \cap H^i$ est un préfixe de $\mathcal{O}^i$ pour $i \in \{1, 2\}$. Si $q' = q$, on dit que $\mathcal{O}$ est *compatible* avec $\mathcal{O}^1$ et $\mathcal{O}^2$.

En gros, une $\{x, y\}$-décomposition linéaire contiguë de G est dite GaD (*de gauche à droite*) si $\mathcal{O}_D$ est compatible avec $\mathcal{O}_{up}$ et $\mathcal{O}_{down}$. Plus précisément,

Définition 3.3.1 ($\{x, y\}$-décomposition linéaire GaD (de gauche à droite) g-contiguë). Une $\{x, y\}$-décomposition linéaire g-contiguë $D = (X_1, \ldots, X_p)$ de G est GaD si et seulement si

(1), pour toutes composantes $C_i^u, C_j^u \in \mathcal{O}_{up}$ (resp., $C_i^d, C_j^d \in \mathcal{O}_{down}$) avec $i < j$, $b_{C_i^u} < a_{C_j^u}$ (resp., avec $b_{C_i^d} < a_{C_j^d}$) et, de plus, (2), pour toute composante $C \in \bar{\mathcal{C}}_{up}$ (resp., $\bar{\mathcal{C}}_{down}$), et $i \in I_C$, $X_i \cap F = \{l_C, r_C, f_C\}$ où f_C est un sommet de $V(P_{down})$ (resp., de $V(P_{up})$).

Dans ce qui suit, nous transformerons itérativement une $\{x, y\}$-décomposition linéaire g-contiguë donnée de G en différentes décompositions linéaires. Au cours de ces transformations, la décomposition linéaire obtenue restera toujours une $\{x, y\}$-décomposition linéaire g-contiguë de G, mais sa longueur peut augmenter temporairement. Pour résoudre cette difficulté, nous définissons la *longueur faible*, notée $w\ell(D)$, d'une $\{x, y\}$-décomposition linéaire $D = (X_1, \ldots, X_p)$ d'un graphe planaire extérieur G (où x et y sont des arêtes extérieures d'une même face de G). La *longueur faible*, notée par $w\ell(X_i)$, d'un sac X_i ($1 \leq i \leq p$) est égale à $\max_{u \in X_i, v \in X_i \cap Y} dist(u, v)$ où $Y = V(P_{up})$ (resp., $Y = V(P_{down})$) si $a_C \leq i \leq b_C$ pour une composante $\bar{C} \in \bar{C}_{down}$ (resp., $C \in \bar{C}_{up}$). Comme la longueur, $w\ell(D) = \max_{i \leq p} w\ell(X_i)$.

Lemme 3.3.16. *Soit D, une $\{x, y\}$-décomposition linéaire GaD g-contiguë de G, satisfaisant le dernier énoncé du Théorème 3.3.15, et de longueur faible k. Alors, $\ell(D) \leq k$.*

Démonstration. Ceci est vrai grâce aux propriétés d'une décomposition linéaire GaD g-contiguë de G et par le dernier énoncé du Théorème 3.3.15. □

Le théorème suivant dit en gros que, à partir d'une $\{x, y\}$-décomposition linéaire g-contiguë de G, nous pouvons obtenir une $\{x, y\}$-décomposition linéaire GaD g-contiguë de G de même longueur.

Théorème 3.3.17. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe, et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G.*

Supposons qu'il existe une $\{x, y\}$-décomposition linéaire g-contiguë $D = (X_1, \ldots, X_p)$ de G de longueur k et telle que, pour tout $C \in \mathcal{C}$ et $a_C \leq i \leq b_C$, $\ell(X_i) = k$ seulement si $\mathcal{M}_C \cap X_i \neq \emptyset$.

Alors, il existe une $\{x, y\}$-décomposition linéaire GaD g-contiguë de G de longueur au plus k.

Démonstration. Soit $D = (X_1, \ldots, X_p)$ une $\{x, y\}$-décomposition linéaire g-contiguë de G de longueur faible k. Disons que D satisfait la propriété $(*)$ si, pour tout $C \in \mathcal{C}$ et $a_C \leq i \leq b_C$, $w\ell(X_i) = k$ seulement si $X_i \cap \mathcal{M}_C \neq \emptyset$.

Rappelons que $\mathcal{O}_{up}$ (resp., $\mathcal{O}_{down}$) ne sont pas définis de manière unique au sens où l'ordre entre deux composantes $C, C' \in \mathcal{C}_{up}$ (resp., $C, C' \in \mathcal{C}_{down}$) avec $s_C = s_{C'} \in V$ est arbitraire (mais toutes ces composantes sont consécutives dans $\mathcal{O}_{up}$ (resp., $\mathcal{O}_{down}$)).

Soit $D = (X_1, \ldots, X_p)$, une $\{x, y\}$-décomposition linéaire g-contiguë de G de longueur faible au plus égale à k satisfaisant la propriété $(*)$ et soient $\mathcal{O}_{up}$ et $\mathcal{O}_{down}$, des ordonnancements de $\bar{\mathcal{C}}_{up}$ et $\bar{\mathcal{C}}_{down}$ (tels que définis précédemment) maximisant $1 \leq h \leq p$ de telle sorte que $(X_1, \ldots, X_h)$ soit compatible avec $\mathcal{O}_{up}$ et $\mathcal{O}_{down}$. Notons que, si $h = p$, alors D est la décomposition linéaire souhaitée. Par conséquent, pour arriver à une contradiction, supposons que $h < p$.

Remarquons que, puisque D est une $\{x, y\}$-décomposition linéaire contiguë de G et que $(X_1, \ldots, X_h)$ est compatible avec $\mathcal{O}_{up}$ et $\mathcal{O}_{down}$, il existe $1 \leq i \leq s$ et $1 \leq j \leq t$ (rappelons que t et s sont le nombre de sommets de P_{up} et P_{down} respectivement) tels que $X_h \cap X_{h+1} = \{u_i, d_j\}$ (en faite, $X_h \cap X_{h+1} \subseteq F$, mais nous pouvons retirer $\{u_{i+1} \ldots, u_s, d_{j+1}, \ldots, d_t\}$ des sacs

$X_1, \dots X_h$ et $u_1, \dots u_{i-1}, d_1, \dots, d_{j-1}\}$ des sacs $X_{h+1}, \dots, X_p$ tout en gardant les propriétés de D).

Soit $\mathcal{O}_D = \mathcal{O} \odot (C_1, \dots, C_q)$ où $\mathcal{O}$ est le préfixe de $\mathcal{O}_D$ qui correspond aux composantes apparaissant dans $(X_1, \dots, X_h)$. Supposons que $C_1 \in \bar{\mathcal{C}}_{up}$. Soit $\mathcal{O}_{up} = \mathcal{O}' \odot (C'_1, \dots, C'_{q'})$ où $\mathcal{O}' = \mathcal{O} \cap \bar{\mathcal{C}}_{up}$. Par maximalité de h, $C_1 \neq C'_1$. Plus précisément, $C_1 = C'_z$ pour $1 < z \leq q'$ et $h + 1 = a_{C'_z}$. Il y a deux cas à considérer suivant si nous sommes capables ou non de contredire la maximalité de h.

1. Premièrement, supposons que, pour tout $1 \leq \alpha < z$ et pour tout $h^*_{C'_\alpha} \in \mathcal{M}_{C'_\alpha}$, $dist(h^*_{C'_\alpha}, d_j) \leq k$. Définissons alors, une nouvelle décomposition linéaire D' :

   $$D' = (X_1, \dots, X_h) \odot ((X_{a_{C'_1}}, \dots, X_{b_{C'_1}}) \cap \bar{C}'_1) \cup \{d_j\} \odot ((X_{a_{C'_2}}, \dots, X_{b_{C'_2}}) \cap \bar{C}'_2) \cup \{d_j\} \odot$$

   $$\dots \odot ((X_{a_{C'_{z-1}}}, \dots, X_{b_{C'_{z-1}}}) \cap \bar{C}'_{z-1}) \cup \{d_j\} \odot ((X_{h+1}, \dots, X_p) \setminus ((\bigcup_{1 \leq \alpha < z} \bar{C}'_\alpha) \setminus \{l_{C'_z}\})).$$

   Intuitivement, toutes les composantes situées entre (dans $\mathcal{O}_{up}$) la dernière composante de $\mathcal{O}$ et C_1 sont "déplacées" juste avant C_1 dans la décomposition (dans D, toutes ces composantes apparaissaient après C_1).

   Puisque D est une $\{x, y\}$-décomposition linéaire g-contiguë de G satisfaisant la propriété $(*)$, alors D' est une $\{x, y\}$-décomposition linéaire g-contiguë de G satisfaisant la propriété $(*)$. De plus, sa longueur faible est au plus k. En particulier, pour tout sac B de $((X_{a_{C'_\alpha}}, \dots, X_{b_{C'_\alpha}}) \cap \bar{C}'_\alpha)) \cup \{d_j\}$ pour $1 \leq \alpha < z$, $w\ell(B) \leq k$ parce que D satisfait la propriété $(*)$ et parce que, par hypothèse, $dist(h^*_{C'_\alpha}, d_j) \leq k$ pour tout $h^*_{C'_\alpha} \in \mathcal{M}_{C'_\alpha}$. Concernant n'importe quel autre sac X de la décomposition, il existe toujours un sac X' de D tel que $X \subseteq X'$. Par conséquent, nous avons effectivement que la longueur faible de D' est au plus k.

   Pour conclure ce cas, D' est une $\{x, y\}$-décomposition linéaire g-contiguë de G de longueur faible au plus k satisfaisant la propriété $(*)$ et avec un préfixe plus grand que D compatible avec $\mathcal{O}_{up}$ et $\mathcal{O}_{down}$, ce qui contredit la maximalité de h.

2. Contrairement à la première supposition, nous pouvons maintenant supposer que pour toute décomposition D définie comme ci-dessus et maximisant h (où $z := z(D)$, défini comme ci-dessus, dépend de D), il existe un entier $1 \leq \alpha < z(D)$ et un sommet $h^*_{C'_\alpha} \in \mathcal{M}_{C'_\alpha}$ tel que $dist(h^*_{C'_\alpha}, d_j) > k$ (sinon, nous retombons dans le cas précédent). Notons par $\alpha(D)$ le plus petit entier α pour la décomposition D tel qu'il existe un sommet $h^*_{C'_\alpha} \in \mathcal{M}_{C'_\alpha}$ tel que $dist(h^*_{C'_\alpha}, d_j) > k$.

   Notons aussi $1 < \alpha^*(D) \leq q$ tel que $C'_{\alpha(D)} = C_{\alpha^*(D)}$, c'est-à-dire, notons par $\alpha^*(D)$ l'indice de la composante C'_α dans le préfixe $\mathcal{O}$ de l'ordre d'apparition des composantes de $\mathcal{C}$ dans la décomposition D.

   Considérons une décomposition D (maximisant toujours h) mais qui cette fois minimise $\alpha^*(D)$. À partir de maintenant, nous notons l'entier $\alpha(D)$ (pour cette décomposition particulière D) par α et $\alpha^*(D)$ est noté par α^*.

   Soient $\alpha < \beta \leq z \leq \gamma \leq q'$ défini de telle sorte que $[\beta, \gamma]$ est l'intervalle maximal par inclusion (contenant z) tel que chaque composante C'_m avec $m \in [\beta, \gamma]$ apparaît avant C_{α^*} dans $\mathcal{O}_D$ (c'est-à-dire,, pour tout $m \in [\beta, \gamma]$, en fixant m' de telle sorte que $C'_m = C_{m'}$,

alors $1 \leq m' < \alpha^*$). Notons par $C_{l_{C'_\beta}}$ l'ensemble des composantes C de $\mathcal{C}$ telles que $l_{C'_\beta} \in s_C$, $C \notin I$ et $C \leq_D \alpha^*$, c'est-à-dire, les composantes qui apparaissent avant C_{α^*} dans D, qui ne seront pas déplacées, et qui contiennent $l_{C'_\beta}$ (notons que nous ne pouvons pas supprimer $l_{C'_\beta}$ des sacs associés aux composantes dans $C_{l_{C'_\beta}}$ sans briser la propriété de contiguïté). Notons que C_{α^*} peut être dans $C_{l_{C'_\beta}}$. Définissons aussi $C_{r_{C'_\gamma}}$, l'ensemble des composantes C de $\mathcal{C}$ telles que $r_{C'_\gamma} \in s_C$, $C \notin I$ et $C \leq_D \alpha^*$. Comme précédemment, il y a plusieurs cas à considérer sur les ensembles $C_{l_{C'_\beta}}$ et $C_{r_{C'_\gamma}}$. Chacun de ces cas définira une décomposition linéaire D' particulière, mais leur longueur faible sera toujours inférieure à k pour les mêmes raisons. Ces décompositions contrediront aussi une des hypothèses émises précédemment, nous permettant de conclure cette preuve. Puisque la décomposition la plus simple est la première, c'est-à-dire, la décomposition définie dans le cas $(2.a)$, nous présenterons les arguments justifiant que $w\ell(D'') \leq k$ et les arguments menant à une contradiction seulement pour ce cas $(2.a)$.

(a) Premièrement, supposons que $C_{l_{C'_\beta}} = C_{r_{C'_\beta}} = \emptyset$. Définissons alors D' :

$$D' = (X_1, \ldots, X_h) \odot ((X_{h+1}, \ldots, X_{a_{C_{\alpha^*}}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$(X_{a_{C_{\alpha^*}}}, \ldots, X_{b_{C_{\alpha^*}}}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$((X_{h+1} \ldots, X_{a_{C_{\alpha^*}}-1}) \cap \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m) \cup (X_{b_{C_{\alpha^*}}} \cap V(P_{down})) \odot (X_{b_{C_{\alpha^*}}+1}, \ldots, X_p).$$

Intuitivement, toutes les composantes $C'_\beta, \ldots, C'_\gamma$ (et en particulier, $C_1 = C'_z$) qui apparaissaient avant C_{α^*} dans D (mais qui sont censées être après la composante C_{α^*} dans $\mathcal{O}_{up}$) sont "déplacées" après C_{α^*} dans D.

Puisque D est une $\{x, y\}$-décomposition linéaire g-contiguë satisfaisant la propriété $(*)$, D' est aussi une $\{x, y\}$-décomposition linéaire g-contiguë de G satisfaisant la propriété $(*)$. En particulier, chaque arête de G appartient à un sac, car nous avons supposé que toutes les composantes intersectant $\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m$ n'apparaissent pas dans $(X_{h+1}, \ldots, X_{a_{C_{\alpha^*}}-1})$, c'est-à-dire, $C_{l_{C'_\beta}} = C_{r_{C'_\beta}} = \emptyset$. Il reste à prouver que sa longueur faible est au plus k.

Puisque la plupart des sacs de D' sont des sous-ensembles de sac de D, il nous suffit de prouver que chaque sac de $((X_{h+1} \ldots, X_{a_{C_{\alpha^*}}-1}) \cap \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m) \cup (X_{b_{C_{\alpha^*}}} \cap V(P_{down}))$ a une longueur faible au plus k pour certifier que $w\ell(D') \leq k$. Puisque nous considérons la longueur faible, nous devons prouver que, pour chaque $v \in \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m$ et chaque $w \in X_{b_{C_{\alpha^*}}} \cap V(P_{down})$, $dist(v, w) \leq k$. Rappelons que pour n'importe quel $w \in X_{b_{C_{\alpha^*}}} \cap V(P_{down})$, w et $h^*_{C'_\alpha}$ appartiennent tous deux à un sac de D, et donc $dist(h^*_{C'_\alpha}, w) \leq k$. Par conséquent, dans certain cas, il sera suffisant de montrer que $dist(v, w) \leq dist(h^*_{C'_\alpha}, w)$ pour prouver que $dist(v, w) < k$.

Notons que puisque tout sommet du (d_1, d_{j-1})-chemin a été retiré des sacs $X_{h+1}, \ldots, X_p$ de D, alors w est forcément entre d_j et y_2 dans P_{down}. De plus,

puisque $dist(h^*_{C'_\alpha}, d_j) > k$ et $dist(h^*_{C'_\alpha}, w) \leq k$, alors le plus court chemin entre w et $h^*_{C'_\alpha}$ passent par y_2.

Soit $\beta \leq m \leq \gamma$ tel que $v \in \bar{C'_m}$ et soit $h < \delta < a_{C'_\alpha}$ tel que $v \in X_\delta$ (dans D). Puisque D est une décomposition linéaire et que $d_j \in X_h$ et $w \in X_{a_{C'_\alpha}}$, il doit y avoir un sommet $w' \in X_\delta$ qui est dans le chemin entre d_j et w dans P_{down} (il est possible que $w' = w$ ou $w' = d_j$), et donc $dist(v, w') \leq k$ puisque D a une longueur faible au plus k. Si le plus court chemin entre v et w' passe par y_2, on obtient que $dist(v, w) \leq dist(v, w') \leq k$ et donc $w\ell(D') \leq k$. Sinon, w est strictement entre w' et y_2 (en particulier $w \neq w'$ sinon $dist(w, v) \leq k$) et le plus court chemin entre v et w' passe par x_2. Notons également que w' appartient à chaque sac de $X_{a_{C'_m}}, \ldots, X_{b_{C'_m}}$ et, en particulier, l'un de ces sacs contient un sommet $h^*_{C'_m}$, et donc $dist(h^*_{C'_m}, w') \leq k$.

En résumé, $dist(w', l_{C'_m}) + dist(l_{C'_m}, h^*_{C'_m}) = dist(h^*_{C'_m}, w') \leq k < dist(h^*_{C'_\alpha}, d_j) \leq dist(w', l_{C'_\alpha}) + dist(l_{C'_\alpha}, h^*_{C'_\alpha})$. Comme $l_{C'_\alpha}$ est entre x_2 et $l_{C'_m}$ dans P_{up} et que le plus court chemin entre $l_{C'_m}$ et w' passe par x_2, on obtient que $dist(w', l_{C'_m}) \geq dist(w', l_{C'_\alpha})$ et donc $dist(l_{C'_m}, h^*_{C'_m}) < dist(l_{C'_\alpha}, h^*_{C'_\alpha})$ puisque $dist(h^*_{C'_m}, w') < dist(h^*_{C'_\alpha}, d_j)$. Enfin, $k \geq dist(h^*_{C'_\alpha}, w) = dist(h^*_{C'_\alpha}, r_{C'_\alpha}) + dist(r_{C'_\alpha}, w) \geq dist(r_{C'_m}, h^*_{C'_m}) + dist(r_{C'_m}, w) = dist(h^*_{C'_m}, w)$. La dernière inégalité vient du fait que $r_{C'_m}$ est entre y_1 et $r_{C'_\alpha}$ dans P_{up} et que le plus court chemin entre $l_{C'_\alpha}$ et w passe par y_1 et y_2. On obtient donc que $dist(h^*_{C'_m}, w) \leq k$ et donc, $dist(v, w) \leq k$ par la propriété $(*)$.

Il reste à montrer que D' contredit la minimalité de α^*. Soit C la composante qui apparaît dans D' juste après X_h et traitons différemment le cas où $C \in \mathcal{C}_{up}$ du cas où $C \in \mathcal{C}_{down}$:

i. Premièrement, supposons que $C \in \mathcal{C}_{up}$.

   Si $C \leq C'_\alpha$ dans $\mathcal{O}_{up}$, alors par définition et par minimalité de α, D' correspond au cas (1) de la preuve du Théorème, c'est-à-dire que C et toutes les composantes précédents C dans $\mathcal{O}_{up}$ peuvent être déplacées juste après X_h dans D' (rappelons que toutes ces composantes peuvent être ajoutées avec d_j par définition de α).

   Sinon, par définition de α, nous obtenons que $\alpha(D') = \alpha(D) = \alpha$ et, donc $C_{\alpha^*(D')} = C_{\alpha^*(D)}$. Puisque $C_{\alpha^*(D)}$ est "plus proche" de X_h dans D' que dans D (puisque D' est obtenu à partir de D en déplaçant au moins C_1 après $C_{\alpha^*(D)}$), on obtient que $\alpha^*(D') < \alpha^*(D)$, ce qui contredit la minimalité de $\alpha^*(D)$.

ii. Si $C \in \mathcal{C}_{down}$, alors nous répétons le processus. Soit on tombe dans le cas (1), ce qui contredit la maximalité de h, soit il faut répéter une des quatre transformations $((a), (b), (c)$ or $(d))$ du cas (2). Cela conduit à une nouvelle décomposition D'' avec le même préfixe $(X_1, \ldots, X_h)$ et une composante C' qui apparaît juste après ce préfixe dans D''. Si $C' \in \mathcal{C}_{down}$, alors l'application du paragraphe précédent $(2.a.i)$ avec D' et D'' au lieu de D et D' conduit à une contradiction. Sinon, puisque le préfixe (et donc d_j) est le même, l'application du paragraphe ci-dessus $(2.a.i)$ pour $C' \in \mathcal{C}_{up}$ contredit également la minimalité de $\alpha^*(D)$ (c'est-à-dire que $\alpha^*(D'') < \alpha^*(D)$).

(b) Deuxièmement, supposons que $r_{C'_\alpha} = l_{C'_\beta}$ et qu'il existe une composante $C \notin \{C'_\beta, \ldots, C'_\gamma\}$ telle que $r_C = l_{C'_\beta}$ qui apparaît avant C'_α dans D (parce que $C \notin$

$\{C'_\beta, \ldots, C'_\gamma\}$, cela implique que $C < C'_\alpha$ dans $\mathcal{O}_{up}$). Soit $1 \leq \delta < \alpha$ le plus petit entier tel que C'_δ est une telle composante C. Définissons alors D' :

$$D' = (X_1, \ldots, X_h) \odot ((X_{h+1}, \ldots, X_{a_{C'_\delta}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$(((X_{a_{C'_\delta}}, \ldots, X_{a_{C_{\alpha^*}}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}\}) \odot$$

$$((X_{a_{C_{\alpha^*}}}, \ldots, X_{b_{C_{\alpha^*}}}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$((X_{h+1}, \ldots, X_{a_{C_{\alpha^*}}-1}) \cap \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m) \cup (X_{b_{C_{\alpha^*}}} \cap V(P_{down})) \odot (X_{b_{C_{\alpha^*}}+1}, \ldots, X_p).$$

Avec des arguments similaires au cas $(2.a)$, D' a une longueur faible au plus k et contredit la minimalité de α.

(c) Troisièmement, supposons que $C_{l_{C'_\beta}} \neq \emptyset$, et que $C_{r_{C'_\gamma}} = \emptyset$ (notons que le cas où $C_{l_{C'_\beta}} = \emptyset$, et $C_{r_{C'_\gamma}} \neq \emptyset$ est symétrique). Notons par C'_δ la première composante de D qui est aussi dans $C_{l_{C'_\beta}}$. Définissons alors D' :

$$D' = (X_1, \ldots, X_h) \odot ((X_{h+1}, \ldots, X_{a_{C'_\delta}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$(((X_{a_{C'_\delta}}, \ldots, X_{a_{C_{\alpha^*}}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}\}) \odot$$

$$(((X_{a_{C_{\alpha^*}}}, \ldots, X_{b_{C_{\alpha^*}}}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}\}) \odot$$

$$((X_{h+1}, \ldots, X_{a_{C_{\alpha^*}}-1}) \cap \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m) \cup (X_{b_{C_{\alpha^*}}} \cap V(P_{down})) \odot (X_{b_{C_{\alpha^*}}+1}, \ldots, X_p).$$

Avec des arguments similaires au cas $(2.a)$, D' a une longueur faible au plus k et contredit la minimalité de α.

(d) Quatrièmement, supposons que $C_{l_{C'_\beta}} \neq \emptyset$, et que $C_{r_{C'_\beta}} \neq \emptyset$. Désignons par C^l_δ la première composante de D qui est aussi dans $C_{l_{C'_\beta}}$. Désignons par C^r_δ la première composante de D qui est aussi dans $C_{r_{C'_\gamma}}$. Supposons que C^l_δ apparaisse avant C^r_δ dans D. Définissons alors D' :

$$D' = (X_1, \ldots, X_h) \odot ((X_{h+1}, \ldots, X_{a_{C^l_\delta}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \odot$$

$$(((X_{a_{C^l_\delta}}, \ldots, X_{a_{C^r_\delta}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}\}) \odot$$

$$(((X_{a_{C'_\delta}}, \ldots, X_{a_{C_{\alpha^*}}-1}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}, r_{C'_\gamma}\}) \odot$$

$$(((X_{a_{C_{\alpha^*}}}, \ldots, X_{b_{C_{\alpha^*}}}) \setminus (\bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m)) \cup \{l_{C'_\beta}, r_{C'_\gamma}\}) \odot$$

$$((X_{h+1} \ldots, X_{a_{C_{\alpha^*}}-1}) \cap \bigcup_{\beta \leq m \leq \gamma} \bar{C}'_m) \cup (X_{b_{C_{\alpha^*}}} \cap V(P_{down})) \odot (X_{b_{C_{\alpha^*}}+1}, \ldots, X_p).$$

Avec des arguments similaires au cas $(2.a)$, D' a une longueur faible au plus k et contredit la minimalité de α.

□

3.3.2.3 Calculer en temps polynomial des décompositions linéaires de Gauche-à-Droite g-contiguë optimales

Enfin, nous montrons qu'une $\{x, y\}$-décomposition linéaire de GaD g-contiguë optimale peut être calculée efficacement.

Théorème 3.3.18. *Soit $G = (V, E)$ un graphe planaire extérieur simple et connexe contenant n sommets et soient $x, y \in E_{out} \cup V$ tels que $G \setminus x$ et $G \setminus y$ sont connexes, $x \neq y$, $x \notin y$, $y \notin x$ et x et y se trouvent sur la même face interne F de G. Supposons aussi que $p\ell(G, x, y) \leq k$.*

Alors, une $\{x, y\}$-décomposition linéaire de GaD g-contiguë optimale de G de longueur au plus k peut être calculée en temps $O(n + k^2)$.

Démonstration. Tout d'abord, les composantes connexes C (et $\bar{C} = C \cup s_C$) de $G \setminus F$ peuvent être calculées en temps linéaire. Notons que ces composantes sont deux à deux arêtes-disjointes. Par le Théorème 3.3.10, en temps global $O(|E|)$, une décomposition linéaire optimale D_C basée sur s_C peut être calculée pour chacune des composantes $\bar{C}$.

Pour chaque sommet-séparateur $v \in V(F)$, notons par $\mathcal{C}_v$ l'ensemble des composantes connexes de $G \setminus v$ qui ne contiennent pas x (ni y) et notons par $C_v \in \mathcal{C}_v$ (notons que $v = s_{C_v}$) une composante qui maximise $dist(h^*_{C_v}, v)$. Soit G' obtenu à partir de G en supprimant, pour chaque sommet-séparateur $v \in V(F)$, toutes les composantes de $\mathcal{C}_v$ à l'exception de C_v. En d'autres termes, pour chaque sommet-séparateur v de F, il existe une seule composante de $G' \setminus v$ ne contenant ni x ni y.

Nous calculons d'abord une décomposition linéaire pour G' que nous adaptons ensuite en une décomposition linéaire de G.

Soient $(\bar{C}^u_1, \ldots, \bar{C}^u_q)$ les composantes C, telles que $s_C \in P_{up}$, ordonnées de "gauche à droite", c'est-à-dire, de x_1 à y_1, et soient $(\bar{C}^d_1, \ldots, \bar{C}^d_{q'})$ les composantes C telles que $s_C \in P_{down}$ ordonné de "gauche à droite", c'est-à-dire, de x_2 à y_2 (nous utilisons la même notation que dans la Section 3.3.2.2).

Notons que F induit un sous-graphe isométrique de G (et de G') et donc, par le Lemme 1.3.9 et Théorème 3.2.3, $|F| = O(k)$. Par conséquent, puisque chaque sommet-séparateur de G' correspond à une seule composante, nous obtenons que $q = O(k)$ et $q' = O(k)$.

Pour chaque $0 \leq i \leq q$, $0 \leq j \leq q'$, notons par $G[i,j]$ le sous-graphe de G induit par $x = \{x_1, x_2\}$, chaque composante $\bar{C}^u_{i'}$ avec $1 \leq i' \leq i$, chaque composante $\bar{C}^d_{j'}$ avec $1 \leq j' \leq j$ et en ajoutant une nouvelle arête $e_{i,j}$ entre $r_{\bar{C}^u_i}$ et $r_{\bar{C}^d_j}$ de longueur $dist_G(r_{\bar{C}^u_i}, r_{\bar{C}^d_j})$ (cette arête artificielle est utilisée pour conserver les mêmes distances que dans G et G'). Notons aussi par $D[i,j]$ une $\{x, e_{i,j}\}$-décomposition linéaire de GaD g-contiguë optimale de $G[i,j]$.

Soit $D_1 = D[i-1,j] \odot D_{C^u_i} \cup \{r_{C^d_j}\}$ et $D_2 = D[i,j-1] \odot D_{C^d_j} \cup \{r_{C^u_i}\}$. Soit $D' \in \{D_1, D_2\}$ la décomposition minimisant sa longueur. Alors, par définition des $\{x, e_{i,j}\}$-décompositions linéaires de GaD g-contiguës, D' est une $\{x, e_{i,j}\}$-décomposition linéaire de GaD g-contiguë de $G[i,j]$ optimale, c'est-à-dire, il n'y a pas d'autre $\{x, e_{i,j}\}$-décomposition linéaire D'' de GaD g-contiguë de $G[i,j]$ telle que $\ell(D'') < \ell(D')$. Par conséquent, pour chaque $0 \leq i \leq q$, $0 \leq j \leq q'$, $D[i,j]$ peut être calculé en temps constant à partir de $D[i-1,j]$ et $D[i,j-1]$. En effet, le calcul de la longueur de D_1 (resp., D_2) à partir de celle de $D[i-1,j]$ (resp., de $D[i,j-1]$) ne repose que sur la distance entre $h^*_{C^u_i}$ et $r_{C^d_j}$ (resp., entre $h^*_{C^d_j}$ et $r_{C^u_i}$) et ces distances peuvent toutes être pré-calculées en temps linéaire global (pour chaque composant C, un seul algorithme BFS à partir de s_C suffit pour déterminer h^*_C et sa distance à s_C).

Ainsi, une fois que les pré-traitements linéaires (calcul des décompositions gloutonnes et des distances) sont effectués, une $\{x, y\}$-décomposition linéaire $D[q, q']$ de GaD g-contiguë $D[q, q']$ de G' de longueur au plus k peut être calculée en temps $O(k^2)$ par programmation dynamique.

En conclusion, pour obtenir la décomposition désirée pour G à partir de $D[q, q']$, il suffit, pour chaque sommet-séparateur $v \in V(F)$ et chaque composante $C \in \mathcal{C}_v \setminus \{C_v\}$, d'insérer la décomposition gloutonne $D_C \cup x_v$ juste après la décomposition gloutonne D_{C_v} de la composante C_v dans $D[q, q']$ où $x_v \neq v$ est le (seul) sommet de F qui n'est pas dans $\bar{C}_v$ et qui apparaît dans les mêmes sacs que $\bar{C}_v$ dans $D[q, q']$ (par maximalité de $dist(h^*_{C_v}, v)$ lors de la définition de C_v, cela n'augmente pas la longueur de la décomposition). □

3.3.3 (+1)-Approximation dans les graphes planaires extérieurs

Nous sommes enfin prêts à prouver notre théorème principal.

Théorème 3.3.19. *Il existe un algorithme qui, pour tout graphe planaire extérieur connexe $G = (V, E)$ contenant n sommets, décide en temps $O(n^3(n + k^2))$ si $p\ell(G) > k$ ou renvoie une décomposition linéaire de G de longueur au plus $k + 1$.*

Démonstration. Pour tout $x, y \in E_{out} \cup V$ (éventuellement $x = y$), l'algorithme décide si $p\ell(G, x, y) > k$ ou calcule une $\{x, y\}$-décomposition linéaire de G de longueur au plus $k + 1$. Par le Lemme 3.3.1, si l'algorithme n'a pas calculé une $\{x, y\}$-décomposition linéaire de G de longueur au plus $k + 1$ pour n'importe quels $x, y \in E_{out} \cup V$, alors $p\ell(G) > k$.

Fixons $x, y \in E_{out} \cup V$ et supposons que $p\ell(G, x, y) \leq k$, nous présentons un algorithme qui calcule une $\{x, y\}$-décomposition linéaire de G de longueur au $k + 1$ en temps $O(n(n + k^2))$.

Si $x = y$, l'algorithme $Glouton$ (voir la section 3.3.1.3) calcule une $\{x, y\}$-décomposition linéaire gloutonne basée sur $x = y$ optimale de G en temps linéaire (par le Théorème 3.3.10) et nous avons terminé. Supposons donc que $x \neq y$.

Premièrement, en temps linéaire, les sommets-séparateurs et les arêtes internes séparant x et y sont calculés (cela peut être fait, par exemple, en utilisant des arbres SPQR (Hopcroft & Tarjan, 1973)). Soient $\{e_0 = x, e_1, \ldots, e_{q-1}, e_q = y\}$, où $e_i \in E_{int} \cup V$ pour chaque $0 < i < q$, l'ensemble de ces séparateurs dans l'ordre où ils sont rencontrés en allant de x à y. Pour chaque

$0 \leq i < q$, $e_i \neq e_{i+1}$ (mais ils peuvent s'intersecter), et soit, e_i et e_{i+1} partagent une même face interne F_i, soit, e_i et e_{i+1} sont des sommets et $\{e_i, e_{i+1}\} \in E$. Soit $\mathcal{C}$ l'ensemble des composantes connexes de $G \setminus e_i$ pour n'importe quel $0 \leq i \leq q$, qui ne contiennent ni x ni y. En d'autres termes, pour tout $C \in \mathcal{C}$, il existe $0 \leq i \leq q$ et $v \in e_i$ (ou $v = e_i$ si e_i est un sommet) tels que $N(C) = \{v\}$. Notons que cette décomposition en plusieurs composantes connexes est effectuée une fois pour toutes au début de l'exécution de l'algorithme. En particulier, elle n'est plus effectuée dans les appels récursifs décrits ci-dessous et donc s'additionne à la complexité de l'algorithme décrit ci-dessous.

Supposons d'abord que $e_1 \neq y$. Notons par C' la composante connexe de $G \setminus e_1$ contenant (ou intersectant si $e_1 \cap x \neq \emptyset$) x. Soit $G_y = G[V \setminus C']$, soit $\mathcal{C}_x \subseteq \mathcal{C}$ l'ensemble des composantes connexes de $G \setminus x$ qui ne contiennent pas y (elles ont déjà été calculées ci-dessus), soit $G_x = G[(C \cup e_1)]$ et soit $G'_x = G_x \setminus \bigcup_{C \in \mathcal{C}_x} V(C)]$ être le sous-graphe induit par les sommets de $C \cup e_1$ qui ne sont pas dans une composante de $\mathcal{C}_x$.

Notre algorithme calcule d'abord récursivement une $\{e_1, y\}$-décomposition de chemin D_y de G_y de longueur au plus $k+1$ en temps $O(|V(G_y)|(|V(G_y)| + k^2))$. Ensuite, notons que $G'_x \setminus x$ et $G'_x \setminus e_1$ sont connexes et que $e_1 \neq x$ partagent une même face F_0.

— Si $x \in e_1$ ou $e_1 \in x$, l'algorithme calcule D'_x, une $\{x, e_1\}$-décomposition linéaire optimale de G'_x en temps $O(|E(G'_x)|)$ (voir Section 3.3.1.3).

— Sinon, les conditions du théorème 3.3.18 sont remplies et une $\{x, e_1\}$-décomposition linéaire D'_x de G'_x de longueur au plus $k+1$ peut être calculée en temps $O(|V(G'_x)| + k^2)$.

Ensuite, pour chaque $C \in \mathcal{C}_x$ avec $N(C) = \{v\}$, notre algorithme calcule une décomposition optimale D_c de $\bar{C}$ basée sur v. En utilisant plusieurs fois (une fois par composante de $\mathcal{C}_x$) certains des Lemmes 3.3.3-3.3.7 (selon si x et e_1 sont des arêtes ou non, qu'ils s'intersectent ou non, et que $N(C)$ intersecte e_1 ou non), une décomposition D_x de G_x de longueur au plus $k+1$ peut être calculée en temps $|\mathcal{C}_x|$ à partir de D'_x et des décompositions D_C, pour toutes composantes $C \in \mathcal{C}_x$. Enfin, d'après le lemme 3.3.8, la $\{x, y\}$-décomposition linéaire souhaitée de G de longueur au plus $k+1$ est obtenue à partir de D_x et D_y. Au total, la complexité en temps est de $O(|V(G_y)|(|V(G_y)| + k^2)) + O(|V(G'_x)| + k^2) = O(n(n + k^2))$.

Le dernier cas est celui où $e_1 = y$. Dans ce cas, on obtient G'_x à partir de G en supprimant les composantes de $G \setminus x$ ne contenant pas y et en supprimant également les composantes de $G \setminus y$ ne contenant pas x. On obtient une $\{x, y\}$-décomposition linéaire D'_x de G'_x de longueur au plus $k+1$ comme dans les deux points précédents. Ensuite, en utilisant les Lemmes 3.3.3-3.3.7, les composantes de $G \setminus G'_x$ peuvent être ajoutées comme ci-dessus pour obtenir la $\{x, y\}$-décomposition du chemin de G souhaitée, de longueur au plus $k+1$. □

3.4 Poursuite des travaux

L'étape suivante consisterait à concevoir un algorithme exact en temps polynomial (s'il existe) pour calculer la longueur linéaire des graphes planaires extérieurs. Notons que l'augmentation de la longueur $(+1)$ dans notre algorithme d'approximation provient de la propriété de contiguïté. L'exemple de la Figure 3.6 montre que nous ne pouvons pas éviter cette augmentation si nous conservons la propriété de contiguïté. De plus, la propriété GaD a été prouvée à partir d'une décomposition linéaire contiguë. Par conséquent, cette preuve doit également être adaptée au cas exact. Une autre question serait de savoir si notre algorithme pour les arbres peut être adapté

aux graphes cordaux. De plus, la complexité du calcul de la longueur linéaire (ou de la longueur arborescente) des graphes planaires reste ouverte.

Bien sûr, une fois l'étude de la longueur linéaire dans les graphes planaires extérieurs terminée, l'étude des graphes séries-parallèles est la suite logique en tant que super classe des graphes planaires extérieurs 2-connexes.

CHAPITRE 4

Les Chasseurs et Le Lapin

Dans ce chapitre, nous étudions une variante du jeu DES GENDARMES ET DU VOLEUR, le jeu DES CHASSEURS ET DU LAPIN. Ces travaux ont été réalisés en collaboration avec Nicolas Nisse, Foivos Fioravantes et Harmender Galhawat (Dissaux et al., 2023). Rappelons que nous sommes intéressés par le calcul du nombre minimum de chasseurs nécessaires pour capturer le lapin dans les arbres. Dans ce but, dans la section 4.2, nous commençons par nous intéresser à la définition (non triviale) d'une propriété de monotonie de ce jeu dans le but de nous aider à concevoir des stratégies de chasse. Nous montrons ensuite, dans la sous-section 4.2.1, quelques propriétés de telles stratégies de chasse monotones. Cela nous permet de prouver que $pw(G) \leq mh(G) \leq pw(G)+1$ dans la section 4.2.2. Cela implique notamment que le calcul de $mh(G)$ est un problème NP-complet. Nous nous intéressons ensuite, dans la section 4.3, à des classes de graphes simples, les graphes scindés, les graphes d'intervalles et les cographes pour lesquels nous caractérisons le nombre de chasseurs monotone et parfois le nombre de chasseurs non-monotone. Nous adaptons aussi l'algorithme du calcul de la largeur linéaire des arbres (Parsons, 1978) pour calculer le nombre de chasseurs monotone des arbres. Finalement, nous montrons qu'il existe des arbres pour lesquels la différence, entre le nombre de chasseurs monotone et non monotone, est arbitrairement grande, c'est-à-dire, qu'autoriser la recontamination permet de diminuer (considérablement) le nombre de chasseurs nécessaires. Finalement, nous montrons que le calcul du nombre de chasseurs monotone ou non monotone admet un noyau polynomial par rapport au nombre de sommets minimum couvrant un graphe G, $vc(G)$.

4.1 Préliminaires

Sauf si nous précisons le contraire, nous traiterons toujours dans ce chapitre des graphes $G = (V, E)$ non vides, finis, non dirigés, connexes et simples. Pour tout $v \in V$ et $X \subseteq V$, désignons $N_X(v) = \{u \in X \mid \{u, v\} \in E\}$ comme étant le *voisinage ouvert* de v dans X et notons le *voisinage fermé* de v dans X par $N_X[v] = (N_X(v) \cup \{v\}) \cap X$. Si $X = V$, on écrit simplement $N(v)$ et $N[v]$ respectivement.

CHASSEURS ET LAPIN. Le jeu DES CHASSEURS ET DU LAPIN est joué entre deux joueurs, l'un étant le "Chasseur" et l'autre étant le "Lapin", sur un graphe non vide, fini, non dirigé, connexe et simple $G = (V, E)$. Soit $k \in \mathbb{N}^*$ n'importe quel entier non nul. Le joueur "chasseur" contrôle k chasseurs et le joueur "lapin" contrôle un seul lapin. Le jeu va se dérouler sous forme de *tours*,

le premier étant particulier pour le joueur “Lapin”, il fera office d'initialisation. Plus précisément, pour le premier tour, le tour 0, le joueur “Lapin” place le lapin sur un sommet $r_0 \in V$. Le lapin est *invisible*, c'est-à-dire que sa position n'est pas connue des chasseurs et donc du joueur “Chasseur”. Notons que la position du lapin ne sera jamais connue par le joueur “Chasseur” qu'importe le tour de jeu. Concernant les tours suivant, c'est-à-dire, pour n'importe quel tour $i \geq 1$, le joueur “Chasseur” choisit d'abord un sous-ensemble non vide $S_i \subseteq V$ d'au plus k sommets de G (nous disons que les sommets de S_i sont *tirés* au tour i). Si la position actuelle r_{i-1} du lapin est tirée, c'est-à-dire si $r_{i-1} \in S_i$ (on dit que le lapin est *capturé*), alors le joueur “Chasseur” gagne et le jeu s'arrête. Sinon, le lapin est “effrayé” par les tirs et doit donc partir de sa position actuelle r_{i-1} vers un sommet adjacent $r_i \in N(r_{i-1})$, et le tour suivant commence. Le joueur lapin gagne s'il évite de se faire tirer dessus pour toujours.

Une *Stratégie de chasse* dans $G = (V, E)$ est une séquence finie $\mathcal{S} = (S_1, \ldots, S_\ell)$ de sous-ensembles non vides de sommets de G. Soit $h(\mathcal{S}) := \max_{1 \leq i \leq \ell} |S_i|$ et disons que $\mathcal{S}$ *utilise* $h(\mathcal{S})$ chasseurs. La *longueur* d'une stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans $G = (V, E)$ égale au nombre de tours de la stratégie, c'est-à-dire, la longueur de la stratégie $\mathcal{S} = (S_1, \ldots, S_\ell)$ égale ℓ. Une *trajectoire pour le lapin dans G à partir de $W \subseteq V$* (W sera toujours supposé non vide) est une marche $(r_0, \ldots, r_\ell)$ à partir de W, *i.e.*, $r_0 \in W$ et $r_i \in N(r_{i-1})$ pour chaque $1 \leq i \leq \ell$. Une stratégie de chasse est *gagnante par rapport à W* si, pour toute trajectoire de lapin $(r_0, \ldots, r_\ell)$ partant de W, il existe $0 \leq j < \ell$ tel que $r_j \in S_{j+1}$, c'est-à-dire que le lapin est finalement capturé quelle que soit sa trajectoire à partir de W. Étant donné une stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_\ell)$, une trajectoire de lapin $(r_0, \ldots, r_\ell)$ partant de W est *gagnante contre $\mathcal{S}$* si $r_i \notin S_{i+1}$ pour chaque $0 \leq i < \ell$. Une *stratégie de chasse gagnante* est une stratégie de chasse gagnante par rapport à V et une *trajectoire de lapin* est une trajectoire de lapin partant de V, c'est-à-dire, nous précisons où se déroulent les stratégies/trajectoires seulement si elles sont restreintes à un sous-ensemble de sommets de V.

Le *nombre de chasseurs de $G = (V, E)$ par rapport à $W \subseteq V$*, noté $h_W(G)$, est l'entier minimum k tel qu'il existe une stratégie de chasse gagnante par rapport à W et utilisant k chasseurs. Nous notons $h_V(G)$ par $h(G)$ quand il sera évident par contexte que nous parlons du nombre de chasseurs de G. Notons que, pour des raisons techniques, pour un graphe G ne contenant qu'un seul sommet, nous fixons $h(G) = 0$. Intuitivement, ceci est conforme à la "partie localisation" du jeu puisque le lapin est déjà localisé. Le joueur “Lapin” a une *stratégie de survie $\mathcal{R}$ à partir de $W \subseteq V$ contre $k \geq 1$ chasseurs* si, pour toute stratégie de chasseur $\mathcal{S}$ utilisant k chasseurs, il existe une trajectoire de lapin $\mathcal{R}(\mathcal{S})$ qui est gagnante contre $\mathcal{S}$. Notons que, si une telle stratégie $\mathcal{R}$ existe, alors $h_W(G) > k$. Une stratégie de survie correspond donc à une façon de créer des trajectoires (dépendantes de la stratégie de chasse utilisée par les chasseurs).

Les lemmes suivants seront utilisés tout au long de ce chapitre. Dans (Abramovskaya et al., 2016), il a été montré que le nombre de chasseurs est clos par sous-graphes. Nous montrons d'abord que ce résultat s'étend trivialement au cas où les positions de départ du lapin sont restreintes à n'importe quel ensemble $W \subseteq V$.

Lemme 4.1.1. *Soient $G = (V, E)$ un graphe quelconque et H un sous-graphe de G, et soit $W \subseteq V$ avec $W \cap V(H) \neq \emptyset$. Alors, $h_{W \cap V(H)}(H) \leq h_W(G) \leq h(G)$.*

Démonstration. Par définition, $h_W(G) \leq h(G)$. Montrons l'autre inégalité.

Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasseur gagnante dans G par rapport à W. Soit $\mathcal{S}' = (S'_1, S'_2, \ldots, S'_\ell)$ une stratégie telle que, pour chaque $1 \leq i \leq \ell$, $S'_i = S_i \cap V(H)$ si $S_i \cap V(H) \neq \emptyset$ et S'_i consiste en n'importe quel sommet de $V(H)$ dans le cas contraire. Alors, $\mathcal{S}'$ est une stratégie

de chasse gagnante dans H par rapport à $W \cap V(H)$. En effet, toute trajectoire de lapin $(r_0 \in W \cap V(H), r_1, \ldots, r_\ell)$ dans H partant de $W \cap V(H)$ est aussi une trajectoire partant de W dans G. Rappelons que $\mathcal{S}$ est une stratégie gagnante dans G par rapport à W. Par conséquent, il existe $i < \ell$ tel que $r_i \in S_{i+1}$, et puisque $r_i \in V(H)$, nous obtenons que $r_i \in S'_{i+1}$. Nous pouvons conclure, qu'il n'existe pas de trajectoire pour le lapin dans H partant de $W \cap V(H)$, gagnante contre $\mathcal{S}'$, et donc que $\mathcal{S}'$ est gagnante contre $W \cap V(H)$ dans H. De plus, $h(\mathcal{S}') \leq h(\mathcal{S})$, ce qui implique que $h_{W\cap V(H)}(H) \leq h_W(G)$. □

Pour toute stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_\ell)$, il est pratique d'identifier les positions potentielles d'un lapin (commençant dans $W \subseteq V$) après chaque tour. Définissons l'ensemble des positions possibles pour le lapin $\mathcal{Z}^W(\mathcal{S}) = (Z_0^W(\mathcal{S}), \ldots, Z_\ell^W(\mathcal{S}))$ contre $\mathcal{S}$ comme suit. Soit $Z_0^W(\mathcal{S}) = W$ et, pour chaque $0 < i \leq \ell$, soit $Z_i^W(\mathcal{S})$ l'ensemble des sommets v tels qu'il existe une trajectoire de lapin $(r_0, r_1, \ldots, r_i = v)$ telle que $r_0 \in W$ et, pour chaque $0 \leq j < i$, $r_j \notin S_{j+1}$. Formellement, pour tout indice $1 \leq i \leq \ell$, définissons $Z_i^W(\mathcal{S}) = \{x \in V(G) \mid \exists y \in (Z_{i-1}^W(\mathcal{S}) \setminus S_i) \wedge (\{x, y\} \in E(G))\}$. Intuitivement, $Z_i^W(\mathcal{S})$ est l'ensemble des sommets que le lapin (partant d'un sommet de W) peut atteindre à la fin du $i^{ème}$ tour sans avoir été capturé. Nous appellerons les sommets de $Z_i^W(\mathcal{S})$ les sommets *contaminés* après le tour i. Notons que si $\mathcal{S}$ est gagnante, alors $Z_\ell^W(\mathcal{S}) = \emptyset$. Dans ce qui suit, nous écrivons Z_i (resp., $Z_i(\mathcal{S})$) au lieu de $Z_i^W(\mathcal{S})$ lorsque $\mathcal{S}$ et W (resp., lorsque W) sont clairs dans le contexte.

Nous montrons maintenant que nous pouvons ne considérer que des stratégies de chasse composées uniquement de "tirs utiles". Une stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_\ell)$ est dite *parcimonieuse* si, pour chaque $1 \leq i \leq \ell$, $S_i \subseteq Z_{i-1}(\mathcal{S})$. Notons que, si $\mathcal{S}$ est parcimonieuse, alors $Z_i \neq \emptyset$ pour chaque $i < \ell$. Notons que si $\mathcal{S}$ est parcimonieuse, alors elle peut être retrouvée à partir de la séquence $\mathcal{Z}(\mathcal{S}) = (Z_0, \ldots, Z_\ell)$ des sommets contaminés pour chaque tour de $\mathcal{S}$. En effet, pour tout $1 \leq i \leq \ell$, $S_i = \{w \in Z_{i-1} \mid \exists x \in N(w) \backslash Z_i\}$.

Dans le lemme suivant, nous établissons que nous pouvons chasser le lapin de manière parcimonieuse sans augmenter le nombre de chasseurs requis.

Lemme 4.1.2. *Pour tout graphe $G = (V, E)$ et tout sous-ensemble non vide $W \subseteq V$, il existe une stratégie de chasse gagnante et parcimonieuse dans G par rapport à W utilisant $h_W(G)$ chasseurs.*

Démonstration. Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse gagnante par rapport à $W \subseteq V$ utilisant au plus $k \geq 1$ chasseurs. Soit $\mathcal{Z}(\mathcal{S}) = (Z_0(\mathcal{S}), \ldots, Z_\ell(\mathcal{S}))$ l'ensemble des sommets contaminés pour chaque tour de $\mathcal{S}$. S'il existe un entier $\ell' < \ell$ tel que $Z_{\ell'}(\mathcal{S}) = \emptyset$, alors $\mathcal{S} = (S_1, \ldots, S_{\ell'})$ est également une stratégie de chasse gagnante en ce qui concerne $W \subseteq V$ utilisant au plus k chasseurs. Par conséquent, nous pouvons supposer que $Z_i(\mathcal{S}) \neq \emptyset$ pour chaque $0 \leq i < \ell$.

De plus, s'il existe un entier $1 \leq i \leq \ell$ tel que $S_i \cap Z_{i-1}(\mathcal{S}) = \emptyset$, notons par h le plus petit de ces entiers et notons par $v \in Z_{h-1}(\mathcal{S})$ un des sommets contaminés à la fin du tour $h-1$. Alors, $\mathcal{S}' = (S_1, \ldots, S_{h-1}, \{v\}, S_{h+1}, \ldots, S_\ell)$ est aussi une stratégie gagnante (puisque $S_h \cap Z_{h-1}(\mathcal{S}) = \emptyset$) par rapport à $W \subseteq V$ en utilisant au plus $k \geq 1$ chasseurs. En répétant ce processus, nous pouvons supposer que, pour chaque $1 \leq i \leq \ell$, $S_i \cap Z_{i-1}(\mathcal{S}) \neq \emptyset$.

Finalement, soit $\mathcal{S} = (S'_1, S'_2, \ldots, S'_{\ell'})$ tel que, pour tout $1 \leq i \leq \ell'$, $S'_i = S_i \cap Z_{i-1}(\mathcal{S})$. Il est facile de voir que, pour tout $i \leq \ell'$, $Z_i(\mathcal{S}) = Z_i(\mathcal{S}')$, et donc $\mathcal{S}'$ est parcimonieuse. De plus, $\mathcal{S}'$ est une stratégie de chasse gagnante par rapport à W. En effet, puisque $\mathcal{S}$ est gagnante par rapport à W, pour toute trajectoire de lapin $(r_0, r_1, \ldots, r_\ell)$, il existe un entier $j < \ell$ tel que $r_j \in S_{j+1}$.

Soit i le plus petit de ces entiers. Par définition, $r_i \in Z_i \cap S_{i+1} = S'_{i+1}$ et donc $\mathcal{S}'$ est gagnant par rapport à W. De plus, $h(\mathcal{S}') \leq h(\mathcal{S})$. □

Notons que la Remarque 1.5.3 tient aussi pour les stratégies parcimonieuses dans les graphes bipartis $G = (V_r \cup V_w, E)$ par rapport à V_r (resp. V_w) (attention, la parité est inversée).

Remarque 4.1.3 – Soit $G = (V_r \cup V_w, E)$ un graphe biparti et $\mathcal{S}_r = (S_1, \ldots, S_\ell)$ une stratégie de chasse parcimonieuse dans G par rapport à V_r. Alors, pour chaque $1 \leq i \leq \lceil \ell/2 \rceil$, $S_{2i-1} \subseteq Z_{2i-2} \subseteq V_r$ et (si $2i \leq \ell$) $S_{2i} \subseteq Z_{2i-1} \subseteq V_w$.

Il faut remarquer qu'il existe des graphes $G = (V, E)$ et des stratégies de chasse $(S_1, \ldots, S_\ell)$ qui sont gagnantes dans G sans tirer sur tous les sommets, c'est-à-dire, $V \setminus \bigcup_{1 \leq i \leq \ell} S_i \neq \emptyset$. Par exemple, dans le graphe G qui consiste en une seule arête uv, la stratégie $(\{u\}, \{u\})$ est une stratégie de chasse gagnante utilisant un seul chasseur et ne tirant pas sur v. Notons que, dans cet exemple, il n'existe pas de stratégie de chasse parcimonieuse gagnante utilisant un chasseur et tirant à la fois sur u et v. Le lemme suivant, qui caractérise l'ensemble de ces sommets non tirés, sera utilisé tout au long du chapitre.

Lemme 4.1.4. *Soit H tout sous-graphe connexe non vide d'un graphe $G = (V, E)$. Soit $W \subseteq V$ tel que $W \cap V(H) \neq \emptyset$. Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ toute stratégie de chasse gagnante dans G par rapport à W. Si $S_i \cap V(H) = \emptyset$ pour tout $1 \leq i \leq \ell$, alors $|V(H)| = 1$.*

Démonstration. Soit $x \in V(H) \cap W$. Pour arriver à une contradiction, supposons que $|V(H)| \geq 2$. Soit $y \in N_H(x)$ (ce sommet existe puisque H est connexe). Remarquons que puisque $S_i \cap V(H) = \emptyset$ pour tout $1 \leq i \leq \ell$, $\{x, y\} \cap \bigcup_{1 \leq i \leq \ell} S_i = \emptyset$. Ainsi, le lapin peut osciller entre x et y indéfiniment sans se faire tirer dessus. Autrement dit, $\mathcal{R} = (r_0 = x, r_1 = y, r_2 = x, \ldots, r_\ell)$ est une trajectoire de lapin gagnante contre $\mathcal{S}$ à partir de $W \cap V(H)$. Ceci contredit le fait que $\mathcal{S}$ est une stratégie de chasse gagnante dans G par rapport à W. Nous pouvons donc en déduire que $|V(H)| = 1$. □

Le jeu DES CHASSEURS ET DU LAPIN a été particulièrement étudié dans les graphes bipartis (Abramovskaya et al., 2016 ; Bolkema & Groothuis, 2019 ; Gruslys & Méroueh, 2015) et nous poursuivons cette étude dans la section 4.4. Dans ce qui suit, les graphes bipartis sont désignés par $G = (V_r \cup V_w, E)$ où (V_r, V_w) est implicitement une bipartition de $V(G)$ telle que V_r et V_w sont respectivement des ensembles indépendants. Nous appelons les sommets de V_r (resp., de V_w) les sommets *rouges* (resp., *blancs*). Notons que, dans la majeure partie du document, nous considérerons seulement les stratégies de chasse par rapport à V, excepté dans la section 4.4 et au début de la section 4.2 pour la définition de stratégies de chasse monotones. Notons aussi que les Remarques 1.5.3 et 4.1.3 seront utilisé implicitement dans certaines preuve de la section 4.4.

Rappelons qu'il a été prouvé que, dans les graphes bipartis, la variante *rouge* du jeu, c'est-à-dire, la variante où le lapin est forcé de commencer sur un des deux ensembles de la bipartition, est équivalente au jeu original (Abramovskaya et al., 2016).

4.2 Monotonie

Dans les jeux classiques de poursuite-évasion de graphes, une notion importante est celle de la *monotonie*. Sans entrer dans les détails, dans ces jeux, une stratégie est *monotone* si la zone atteignable par le fugitif n'augmente jamais. Autrement dit, dans le cas particulier des jeux de

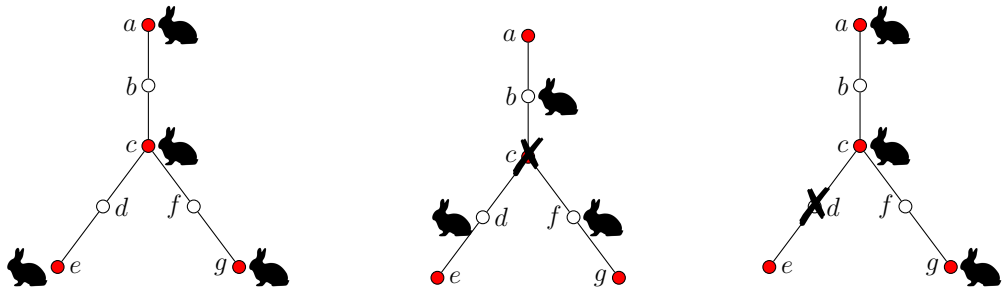
recherche dans les graphes, par exemple pour le jeu DES GENDARMES ET DU VOLEUR, une stratégie est monotone si, une fois qu'un chercheur (un gendarme) est retiré d'un sommet, il n'est jamais nécessaire d'occuper ce sommet lors d'un tour ultérieur (notez que, dans certains cas spécifiques, par exemple dans les graphes dirigés, ces deux définitions ne sont pas rigoureusement équivalentes (Adler, 2007)). Les stratégies monotones ont été largement étudiées (Bienstock & Seymour, 1991 ; Yang, Dyer, & Alspach, 2009 ; Mazoit & Nisse, 2008) car, d'une part, il est généralement plus facile de les concevoir et, d'autre part, les stratégies monotones ont une longueur polynomiale par rapport à la taille du graphe et, par conséquent, les problèmes de décision correspondants (existe-t-il une stratégie monotone utilisant k chercheurs ?) peuvent être prouvés comme étant dans NP.

Il est clair qu'une telle définition n'est pas adaptée au jeu DES CHASSEURS ET DU LAPIN. En effet, considérons le graphe qui consiste en une seule arête uv : le chasseur doit tirer sur un sommet, disons u, et, si le lapin était sur v, il se déplace vers u, c'est-à-dire que le sommet u est "recontaminé". Par conséquent, nous proposons de définir la monotonie dans le jeu DES CHASSEURS ET DU LAPIN comme suit (voir la définition formelle ci-dessous) : une fois qu'un sommet a été "nettoyé", si le lapin peut y accéder lors d'un tour ultérieur, il doit être capturé immédiatement.

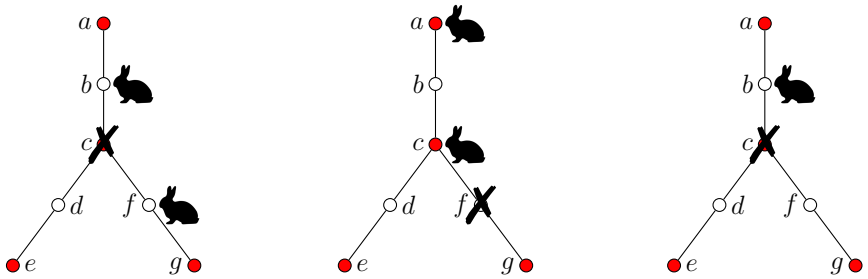
Dans les jeux classiques de recherche de graphes, le fait qu'un sommet soit nettoyé à un certain tour signifie que la stratégie du chercheur garantit que le fugitif ne peut pas occuper ce sommet à ce tour. Le fait d'être recontaminé peut alors être défini intuitivement par le fait qu'un sommet peut être atteint par le fugitif alors qu'il a été nettoyé lors d'un tour précédent. Cette définition intuitive n'a pas de sens dans le jeu DES CHASSEURS ET DU LAPIN et, en particulier, dans sa variante rouge dans les graphes bipartis. En effet, dans ce cas, chaque sommet rouge est nettoyé à chaque tour impair et donc, essayer de trouver une stratégie sans recontamination n'aurait donc aucun sens. Pour surmonter cette difficulté, nous proposons de définir le nettoyage d'un sommet à un tour donné par le fait que les actions des chasseurs garantissent que ce sommet ne peut pas être occupé par le lapin à ce tour.

Une autre difficulté vient du fait que, contrairement aux jeux de recherche de graphes classiques, un sommet peut être "nettoyé" sans avoir été tiré pendant le jeu. Rappelons, par exemple, notre discussion précédente pour le graphe constitué d'une seule arête. Comme exemple moins trivial, considérons une étoile à trois feuilles dont les arêtes ont été subdivisées une fois chacune. Ensuite, en supposant que les feuilles et le centre sont rouges, dans la variante rouge, il est possible pour un chasseur de gagner sans tirer sur les feuilles (alors que n'importe laquelle des feuilles peut être occupée par le lapin initialement). En effet, considérons la stratégie d'un chasseur qui, à chaque tour impair, tire sur le centre et, à chaque tour pair, tire sur un voisin arbitraire du centre qui n'a pas été tiré auparavant. La figure 4.1 illustre la stratégie ci-dessus.

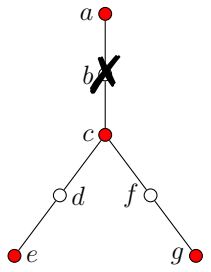
Par conséquent, deux actions des chasseurs peuvent nettoyer un sommet : soit un chasseur tire sur un sommet v au tour i (c'est-à-dire qu'il n'y a pas de trajectoire de lapin, telle que $r_{i-1} = v$, qui gagne contre une stratégie tirant sur v au tour i), soit les chasseurs tirent sur tous les sommets contaminés dans le voisinage de v. Dans ce cas, soit v était occupé et le lapin doit quitter v, soit il ne l'était pas et ne peut plus être occupé après le déplacement du lapin. Dans les deux cas, $v \notin Z_i$. Cette discussion motive la définition suivante de la monotonie des stratégies de chasse.



(a) $Z_0 = \{a, c, e, g\}$ (b) $S_1 = \{c\}$, $Z_1 = \{b, d, f\}$ (c) $S_2 = \{d\}$, $Z_2 = \{a, c, g\}$



(d) $S_3 = \{c\}$, $Z_3 = \{b, f\}$ (e) $S_4 = \{f\}$, $Z_4 = \{c, a\}$ (f) $S_5 = \{c\}$, $Z_5 = \{b\}$



(g) $S_6 = \{b\}$, $Z_6 = \emptyset$

Figure 4.1 – Exemple d'un graphe biparti (où $V_r = \{a, c, e, g\}$ correspond à la partie rouge de la bipartition, illustrée par les sommets rouges dans les figures) et d'une stratégie de chasse gagnante parcimonieuse par rapport à V_r, telle qu'aucun sommet dans $\{a, e, g\}$ n'est jamais tiré. Chaque sous-figure représente la situation à la fin du tour correspondant. Au tour 0, le lapin occupe n'importe quel sommet dans $\{a, c, e, g\}$. Ensuite, au tour 1, le chasseur tire sur le sommet c (représenté par la croix sur le sommet correspondant de la sous-figure (b)) et le lapin se déplace vers l'un des sommets de $\{b, d, f\}$. Le jeu se poursuit jusqu'à la fin du tour 6 (sous-figure (g)), moment où le chasseur est sûr de capturer le lapin au sommet b. Formellement, nous avons $\mathcal{S} = (\{c\}, \{d\}, \{c\}, \{f\}, \{c\}, \{b\})$ and $\mathcal{Z}(\mathcal{S}) = (\{a, c, e, g\}, \{b, d, f\}, \{a, c, g\}, \{b, f\}, \{a, c\}, \{b\}, \emptyset)$.

4.2.1 Définition de stratégies monotones et leurs propriétés

Étant donné un graphe G, une stratégie de chasse $\mathcal{S}$ par rapport à $W \subseteq V$ dans G est *monotone* si pour tout sommet $v \in V$, une fois que v a été nettoyé, il est alors tiré chaque fois que le lapin peut potentiellement l'atteindre. Formellement, nous dirons qu'un sommet v est *nettoyé* durant le tour

i si soit $v \in S_i$, ou soit $N(v) \cap Z_{i-1} \neq \emptyset$ et $N(v) \cap Z_{i-1} \subseteq S_i$. Rappelons que, dans la deuxième condition, il est nécessaire que $N(v) \cap Z_{i-1} \neq \emptyset$, sinon des sommets pourrait être nettoyés sans même tirer sur le voisinage fermé de ces sommets (comme explicité dans le paragraphe précédant cette section). Une stratégie $\mathcal{S} = (S_1, \ldots, S_\ell)$ est *monotone* si, pour tout sommet $v \in V$, s'il existe un indice i tel que v est nettoyé au tour i, alors pour tout tour $j > i$ ultérieur au tour i tel que v est contaminé pendant le tour j, v est tiré pendant le tour $j+1$, c'est-à-dire, $v \in S_{j+1}$ pour tout $j \geq i$ tel que $v \in Z_j$. Nous dirons aussi que le sommet v est *recontaminé* au tour j s'il existe $i \leq j$ tel que v est nettoyé au tour i et $v \in Z_j \setminus S_{j+1}$.

Définissons le NOMBRE DE CHASSEURS MONOTONE d'un graphe G par rapport à $W \subseteq V(G)$, noté par $mh_W(G)$, comme le nombre k minimum tel que k chasseurs ont au moins une stratégie de chasse gagnante monotone in G par rapport à W. Notons aussi par $mh(G)$, le nombre de chasseurs monotone $mh_V(G)$ d'un graphe G (par rapport à V). Plus précisément, quand nous ne précisons pas l'ensemble de position possible pour le lapin au début du jeu, alors le lapin peut commencer où il veut. Notons que, par les définitions précédentes, nous obtenons que :

Proposition 4.2.1. *Pour tout graphe $G = (V, E)$ et $W \subseteq V$, $h_W(G) \leq mh_W(G) \leq mh(G)$.*

La fin de cette section est dévouée à prouver des propriétés générales sur les stratégies de chasse (non-)monotones. Les deux premières propriétés seront d'ailleurs réutilisées pour obtenir d'autres propriétés.

Proposition 4.2.2. *Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$, une stratégie de chasse dans un graphe $G = (V, E)$. Pour tout sommet $v \in V$ et tout indice $1 \leq i \leq \ell$, s'il existe un sommet $u \in N(V)$ et un sommet $x \in N(u)$ (possiblement, $x = v$) tel que u et x n'ont jamais été tirés respectivement jusqu'au tour i et jusqu'au tour $i-1$, c'est-à-dire, $u \notin \bigcup_{j \leq i} S_j$ et $x \notin \bigcup_{j<i} S_j$, alors $v \in Z_p$ pour chaque $p \leq i$.*

Démonstration. La proposition est clairement valide si $p = 0$ puisque $Z_0 = V$. Si $p = 1$, alors, par hypothèse, il existe un sommet $u \in N(v) \setminus S_1$, et donc, il existe une trajectoire $(r_0 = u, r_1 = v)$, ce qui implique que $v \in Z_1$. Par conséquent, nous pouvons nous concentré sur le cas où $p > 1$.

Le lapin peut suivre la stratégie de survie suivante dépendant de la parité de p :

1. p est impair : Le lapin peut alors suivre la trajectoire suivante : $(r_0 = u, r_1 = x, r_2 = u, \ldots, r_{p-2} = x, r_{p-1} = u, r_p = v)$ où, pour tout $q < p$, $r_q = u$ si q est pair et $r_q = x$ si q est impair.
2. p est pair : Le lapin peut alors suivre la trajectoire suivante : $(r_0 = x, r_1 = u, r_2 = x, \ldots, r_{p-2} = x, r_{p-1} = u, r_p = v)$ où, pour tout $q < p$, $r_q = x$ si q est pair et $r_q = u$ si q est impair.

Dans les deux cas, pour tout indice $0 \leq j < p$, $r_j \notin S_{j+1}$ puisque $p \leq i$, $u \notin \bigcup_{j \leq i} S_j$ et $x \notin \bigcup_{j<i} S_j$. Par conséquent, $v \in Z_p$. □

Le prochain lemme nous montre que, comme prévu, si les chasseurs suivent une stratégie de chasse monotone, alors l'ensemble des sommets contaminés ne peut pas augmenter au fur et à mesure que le jeu avance.

Lemme 4.2.3. *Soit $G = (V, E)$ un graphe contenant au moins deux sommets. Pour toute stratégie de chasse monotone $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans G, et pour tous $0 \leq p \leq i \leq \ell$, $Z_i \subseteq Z_p$.*

Démonstration. Évidement, cette propriété est vraie si $p = 0$ ou si $i = 1$, puisque $Z_0 = V$. Par conséquent, supposons que $p \geq 1$ et $i > 1$. Soit v n'importe quel sommet dans Z_i. Puisque $v \in Z_i$, il existe une trajectoire $R = (r_0, \ldots, r_{i-2} = x, r_{i-1} = u, r_i = v)$ pour le lapin telle que, pour tout $0 \leq j < i$, $r_j \notin S_{j+1}$. Par définition d'une trajectoire, nous avons aussi que $u \in N(v)$ et $x \in N(u)$. De plus, par monotonie de $\mathcal{S}$, et puisque $u \in Z_{i-1} \setminus S_i$ (resp. $x \in Z_{i-2} \setminus S_{i-1}$), nous obtenons que $u \notin \bigcup_{q \leq i} S_q$ (resp. $x \notin \bigcup_{q \leq i-1} S_q$), c'est-à-dire, les sommets u et x n'ont jamais été tirés respectivement avant le tour i et avant le tour $i-1$, sinon ils seraient recontaminés respectivement au tour $i-1$ et au tour $i-2$. Par la Proposition 4.2.2, nous avons donc que $v \in Z_p$ pour tout $p \leq i$. □

Le prochain lemme démontre que, pour toute stratégie de chasse non-monotone, il doit exister un sommet, tiré à un certain tour et ensuite recontaminé. Grâce à ce lemme, il nous suffira de montrer qu'il n'existe pas de sommet tiré puis recontaminé pour certifier qu'une stratégie de chasse quelconque est monotone (Cela n'est pas évident puisqu'un sommet peut être recontaminé sans avoir été tiré).

Lemme 4.2.4. *Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$, une stratégie de chasse non-monotone gagnante dans un graphe quelconque $G = (V, E)$. Alors, il existe un sommet $v \in V$ et $1 \leq i \leq \ell$ tel que $v \in Z_{i-1} \setminus S_i$ and $v \in \bigcup_{p<i} S_p$.*

Démonstration. Pour arriver à une contradiction, supposons que l'énoncé est faux, c'est-à-dire,, pour tout sommet v, si $v \in Z_{i-1} \setminus S_i$ alors $v \notin \bigcup_{p<i} S_p$.

Puisque $\mathcal{S}$ est non-monotone et gagnante, il existe un sommet u tel que u est nettoyé au tour $1 \leq q \leq \ell - 2$, et est ensuite recontaminé au tour $j > q$ (c'est-à-dire,, $u \in Z_j \setminus S_{j+1}$). De plus, par notre supposition, u est nettoyé en tirant, durant le tour q, sur tous ces voisins contaminé pendant le tour $q-1$, c'est-à-dire,, $Z_{q-1} \cap N(u) \subseteq S_q$.

Montrons que $N(u) \subseteq \bigcup_{p \leq q} S_p$. Supposons qu'il existe un sommet $x \in N(u)$ tel que $x \notin \bigcup_{p \leq q} S_p$. Puisque $u, x \notin \bigcup_{p<q} S_q$ et $u \in N(x)$, par la Proposition 4.2.2, nous obtenons que $x \in Z_{q-1}$. Rappelons que $N(u) \cap Z_{q-1} \subseteq S_q$. Par conséquent, $x \in S_q$, et donc, nous obtenons que $N(u) \subseteq \bigcup_{p \leq q} S_p$.

Puisque $u \in Z_j$, il existe $w \in (N(u) \cap Z_{j-1}) \setminus S_j$ et $w \in \bigcup_{p<j} S_p$, ce qui contredit qu'il n'existe pas de sommet v recontaminé après avoir été nettoyé en tirant dessus. □

Maintenant, nous adaptons les lemmes 4.1.1 et 4.1.2 aux stratégies de chasse monotones.

Lemme 4.2.5. *Pour tout sous-graphe connexe non-vide H d'un graphe $G = (V, E)$, $mh(H) \leq mh(G)$. Plus précisément, s'il existe une stratégie de chasse monotone et gagnante $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans G, alors il existe aussi une stratégie de chasse monotone et gagnant $\mathcal{S}'$ dans H utilisant au plus $\max_{1 \leq i \leq \ell} |S_i \cap V(H)|$ chasseurs.*

Démonstration. Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$, une stratégie de chasse monotone et gagnante dans G.

If $|V(H)| = 1$, l'énoncé tient clairement puisque que $mh(H) = 0$. Par conséquent, nous supposerons que $|V(H)| > 1$. Soit m comme le plus petit entier tel que $S_m \cap V(H) \neq \emptyset$ et soit u n'importe quel sommet dans $S_m \cap V(H)$ (Notons que puisque $|V(H)| > 1$ et par le Lemme 4.1.4, un tel entier m existe). Définissons $\mathcal{S}' = (S'_1, \ldots, S'_\ell)$ une nouvelle stratégie de chasse telle que pour tout $1 \leq i \leq \ell$,

$$S'_i = \begin{cases} S_i \cap V(H), & \text{si } S_i \cap V(H) \neq \emptyset \\ \{u\}, & \text{sinon} \end{cases}$$

Pour prouver que $\mathcal{S}'$ est monotone, nous aurons besoin du résultat suivant :

Assertion 4.2.5.1 – Pour tout $0 \leq i \leq \ell$ et pour tout sommet v dans $V(H)$, si $v \in Z_i(\mathcal{S}')$, alors $v \in Z_i(\mathcal{S})$.

Démonstration. Posons $\mathcal{R} = (r_0, \ldots, r_i = v)$, la trajectoire dans H tel que pour tout $0 \leq j < i$, $\{r_j r_{j+1}\} \in E(H)$, $r_j \notin S'_{j+1}$ et $r_i = v$ (une telle trajectoire existe puisque $v \in Z_i(\mathcal{S}')$). Par construction de $\mathcal{S}'$, pour tout $1 \leq j \leq \ell$, $S_j \cap V(H) \subseteq S'_j$. Par conséquent, $\mathcal{R}$ est aussi une trajectoire dans G telle que $r_j \notin S_{j+1}$, pour tout $0 \leq j < i$. Donc, $v \in Z_i(\mathcal{S})$. ◇

Prouvons que $\mathcal{S}'$ est une stratégie de chasse monotone gagnante dans H. Premièrement, nous prouvons que $\mathcal{S}'$ est effectivement une stratégie de chasse gagnante dans H. Pour arriver à une contradiction, supposons que $\mathcal{S}'$ n'est pas une stratégie gagnante, c'est-à-dire, $Z_\ell(\mathcal{S}') \neq \emptyset$. Nous obtenons donc, par l'Assertion 4.2.5.1, que $Z_\ell(\mathcal{S}) \neq \emptyset$, ce qui contredit que $\mathcal{S}$ est une stratégie de chasse gagnante dans G.

Il nous reste à prouver que $\mathcal{S}'$ est monotone. Pour arriver à une contradiction, supposons que $\mathcal{S}'$ n'est pas monotone. Rappelons que si une stratégie n'est pas monotone, alors il existe un sommet $v \in V(H)$ et $1 \leq q < i \leq \ell$ tel que $v \in S'_q$ et $v \in Z_i(\mathcal{S}')$ par le Lemme 4.2.4. Par l'Assertion 4.2.5.1 et puisque $v \in Z_i(\mathcal{S}')$, $v \in Z_i(\mathcal{S})$. Puisque $S_p \cap V(H) \subseteq S'_p$ pour tout $1 \leq p \leq \ell$ et puisque $v \notin S'_{i+1}$, $v \notin S_{i+1}$.

Si $v = u$, alors $i + 1 > m$ (puisque $u \in S'_p$ pour tout $1 \leq p \leq m$) et donc $v \in S_m$ et dans $Z_i(\mathcal{S}) \setminus S_{i+1}$, contredisant la monotonie de $\mathcal{S}$.

Sinon, $v \neq u$. Par construction de $\mathcal{S}'$, $S'_p \setminus \{u\} \subseteq S_p$ pour tout $1 \leq p \leq \ell$. Par conséquent, $v \in S_q$ et $v \in Z_i(\mathcal{S}) \setminus S_{i+1}$, contredisant la monotonie de $\mathcal{S}$.

Finalement, le fait que $h(\mathcal{S}') \leq \max_{1 \leq i \leq \ell} |S_i \cap V(H)| \leq h(\mathcal{S})$ complète la preuve. □

Lemme 4.2.6. *Pour n'importe quel graphe $G = (V, E)$ et n'importe quel entier $k \geq mh(G)$, il existe une stratégie de chasse monotone, parcimonieuse et gagnante dans G utilisant au plus k chasseurs.*

Démonstration. Posons $\mathcal{S} = (S_1, \ldots, S_\ell)$, une stratégie de chasse monotone gagnante dans G et utilisant au plus $k \geq mh(G)$ chasseurs telle que ℓ est minimum. Si $\mathcal{S}$ est parcimonieuse, alors S est la stratégie décrite dans l'énoncé. Sinon, parmi ces stratégies (monotone gagnante dans G utilisant au plus k chasseurs), prenons celle qui maximise le premier tour $1 \leq j \leq \ell$ qui nous certifie que $\mathcal{S}$ n'est pas parcimonieuse. Il y a donc plusieurs cas à considérer :

— Soit $\mathcal{Z}(\mathcal{S}) = (Z_0(\mathcal{S}), \ldots, Z_\ell(\mathcal{S}))$ la séquence des ensembles de sommets contaminés pour chaque tour de $\mathcal{S}$. S'il existe un entier $\ell' < \ell$ tel que $Z_{\ell'}(\mathcal{S}) = \emptyset$, alors $\mathcal{S} = (S_1, \ldots, S_{\ell'})$ est aussi une stratégie de chasse monotone gagnante dans G utilisant au plus k chasseurs, contredisant la minimalité de ℓ.
Nous pouvons donc supposé à partir de maintenant que $Z_i(\mathcal{S}) \neq \emptyset$ pour tout $0 \leq i < \ell$.

— Soit $1 \leq j \leq \ell$, le plus petit indice tel que $S_j \setminus Z_{j-1}(\mathcal{S}) \neq \emptyset$ (si un tel entier n'existe pas, alors $\mathcal{S}$ est parcimonieuse et correspond donc à la stratégie décrite dans l'énoncé). Si $S_j \cap Z_{j-1}(\mathcal{S}) \neq \emptyset$, alors nous pouvons juste remplacer S_j par $S_j \cap Z_{j-1}(\mathcal{S})$ pour obtenir une nouvelle stratégie de chasse monotone et gagnante dans G, contredisant la minimalité de j.
Nous pouvons aussi supposé à partir de maintenant que $S_j \cap Z_{j-1}(\mathcal{S}) = \emptyset$. Notons que cela implique $j < \ell$ (puisque dans ce cas, $\mathcal{S}$ n'est pas gagnante dans G).

— S'il existe, notons par i le plut petit indice, tel que $S_{j+i} \cap Z_{j+i-1}(\mathcal{S}) \neq \emptyset$. Posons $v \in S_{j+i} \cap Z_{j+i-1}(\mathcal{S})$. Puisque $v \in Z_{j+i-1}(\mathcal{S})$, par le Lemme 4.2.3, $v \in Z_{j-1+i'}(\mathcal{S})$ pour tout $0 \leq i' < i$.

Par conséquent, pour tout $0 \leq i' < i$, remplaçons $S_{j+i'}$ par $\{v\}$. Il ne reste plus qu'à prouver que cette nouvelle stratégie $\mathcal{S}'$ monotone gagnante dans G, ce qui contredirait la maximalité de j. Premièrement, notons que, pour tout $0 \leq h < j$, $S_h = S'_h$ et donc $Z_h(\mathcal{S}) = Z_h(\mathcal{S}')$. Par définition, $Z_j(\mathcal{S}) = \{x \in V \mid \exists y \in Z_{j-1}(\mathcal{S}) \setminus S_j \wedge (\{x,y\} \in E)\}$ et puisque $S_j \cap Z_{j-1} = \emptyset$, nous avons aussi $Z_j(\mathcal{S}) = \{x \in V \mid \exists y \in Z_{j-1}(\mathcal{S}) \wedge (\{x,y\} \in E)\}$. D'un autre côté, $Z_j(\mathcal{S}') = \{x \in V \mid \exists y \in Z_{j-1}(\mathcal{S}') \setminus S'_j \wedge (\{x,y\} \in E)\} = \{x \in V \mid \exists y \in Z_{j-1}(\mathcal{S}) \setminus \{v\} \wedge (\{x,y\} \in E)\}$ puisque $Z_{j-1}(\mathcal{S}') = Z_{j-1}(\mathcal{S})$ et $S'_j = \{v\}$. Par conséquent, $Z_j(\mathcal{S}') \subseteq Z_j(\mathcal{S})$. Par induction sur $j \leq i' \leq \ell$ et en utilisant les mêmes arguments, nous obtenons que $Z_{i'}(\mathcal{S}') \subseteq Z_{i'}(\mathcal{S})$ pour tout $j \leq i' \leq \ell$. $\mathcal{S}'$ est donc une stratégie de chasse gagnante dans G utilisant au plus $k \leq mh(G)$ chasseurs (parce que $Z_\ell(\mathcal{S}') \subseteq Z_\ell(\mathcal{S}) = \emptyset$). Il ne nous reste plus qu'à montrer que $\mathcal{S}'$ est monotone.
Pour arriver à une contradiction, supposons que $\mathcal{S}'$ n'est pas monotone, et donc, par le Lemme 4.2.4, il existe un sommet x et un indice $1 < m \leq \ell$ tel que $x \in Z_{m-1}(\mathcal{S}') \setminus S'_m$ et $x \in \bigcup_{h<m} S'_h$.
Si $x \neq v$, alors, par définition de $\mathcal{S}'$ (pour tout $1 \leq r \leq \ell$, soit $S'_r = S_r$ ou $S'_r = \{v\}$) et parce que $Z_r(\mathcal{S}') \subseteq Z_r(\mathcal{S})$ pour tout $1 \leq r \leq \ell$, nous obtenons que $x \in \bigcup_{h<m} S_h$ et $x \in Z_{m-1}(\mathcal{S}) \setminus S_m$ contredisant que la monotonie de $\mathcal{S}$. Considérons maintenant l'autre possibilité, c'est-à-dire, $x = v$. Rappelons que nous avons déjà prouvé que $v \in Z_{j-1}(\mathcal{S}) \setminus S_j$. Par conséquent, par monotonie de $\mathcal{S}$, $v \notin \bigcup_{r<j} S_r$ ce qui implique que $v \notin \bigcup_{r<j} S'_r$. Puisque $v \in S'_{j+i'}$ pour tout $0 \leq i' \leq i$, nous avons $m > j + i$. Cela veut aussi dire que $v \in Z_{m-1}(\mathcal{S}) \setminus S_m$ (parce que $Z_r(\mathcal{S}') \subseteq Z_r(\mathcal{S})$ pour tout $1 \leq r \leq \ell$ et $S'_r = S_r$ pour tout $r \geq m > j + i$) et $v \in S_{j+i}$, contredisant la monotonie de $\mathcal{S}$.
Nous pouvons donc finalement supposer que $S_{j+i'} \cap Z_{j+i'-1}(\mathcal{S}) = \emptyset$ pour tout $0 \leq i'$ tel que $j + i' \leq \ell$.

— Rappelons que $j < \ell$ et que $Z_{j-1}(\mathcal{S}) \neq \emptyset$. Soit $v \in Z_{j-1}(\mathcal{S})$. Puisque $v \in Z_{j-1}(\mathcal{S})$, il existe $w \in N(v) \cap Z_{j-2}(\mathcal{S}) \setminus S_{j-1}$. De plus, puisque $w \in Z_{j-2}(\mathcal{S}) \setminus S_{j-1}$ (resp. $v \in Z_{j-1}(\mathcal{S}) \setminus S_j$), nous obtenons que $w \notin S_r$ (resp. $v \notin S_r$) pour tout $r \leq j$ (sinon, cela contredit la monotonie de $\mathcal{S}$). Rappelons que pour tout $0 \leq i'$ tel que $j + i' \leq \ell$, $S_{j+i'} \cap Z_{j+i'-1}(\mathcal{S}) = \emptyset$. Par conséquent, $w \notin S_r$ et $v \notin S_r$ pour tout $r \leq \ell$. Par la Proposition 4.2.2, $v \in Z_\ell(\mathcal{S})$, ce qui contredit que $\mathcal{S}$ est gagnante.

La preuve est donc complète. □

Pour conclure cette section, nous donnons une formulation alternative du Lemme 4.2.3. Plus précisément, nous montrons que, si le joueur "chasseur" suit une stratégie de chasse monotone, après avoir tiré sur un sommet, il doit être tiré continuellement, tant que le lapin peut atteindre ce sommet.

Lemme 4.2.7. *Soit $G = (V, E)$ un graphe et soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse monotone, parcimonieuse et gagnante dans G.*

— *S'il existe $1 \leq i < j \leq \ell$ tels que $v \in S_i \cap S_j$, alors $v \in S_{i+1}$.*
— *S'il existe un entier $1 \leq i < \ell$ tel que $v \notin Z_{i-1}$, alors $v \notin S_j$ pour tout $j \geq i$.*

Démonstration. Pour prouver le premier cas, supposons que $v \in S_i \cap S_j$. Puisque $\mathcal{S}$ est parcimonieuse, nous obtenons que $v \in Z_{i-1} \cap Z_{j-1}$. Par le Lemme 4.2.3 et puisque $v \in Z_j$, $v \in Z_{i'}$ pour tout $i' < j$. Puisque $v \in S_i$ et $v \in Z_{p-1}$ pour tout $i \leq p \leq j$ et par monotonie de $\mathcal{S}$, nous obtenons donc que $v \in S_p$ pour tout $i \leq p \leq j$.

Pour le second cas, le fait que $v \notin Z_{i-1}$ et le Lemme 4.2.3 impliquent que $v \notin Z_j$ pour tout $j \geq i$. Puisque $\mathcal{S}$ est parcimonieuse, $v \notin S_j$ pour tout $j \geq i$. □

De façon surprenante, le lemme précédent n'est pas une caractérisation des stratégies monotones. Effectivement, considérons le chemin (a, b, c, d) avec 4 sommets comme exemple. Nous pouvons vérifier que la stratégie de chasse $(\{a\}, \{b, c\}, \{b, c\})$ est parcimonieuse et gagnante (par rapport à V) et cette stratégie satisfait les conditions du lemme précédent, mais elle n'est pas pour autant monotone (car $a \in S_1$ et $a \in Z_1 \setminus S_2$).

4.2.2 Nombre de chasseurs monotone et largeur linéaire

Dans cette section, nous bornons le nombre de chasseurs monotone par la largeur linéaire, c'est-à-dire, $pw(G) \leq mh(G) \leq pw(G) + 1$. Rappelons qu'il est déjà connu que le nombre de chasseurs (monotone) est inférieur à la largeur linéaire (Abramovskaya et al., 2016), c'est-à-dire, $h(G) \leq pw(G) + 1$. Cette borne est plutôt intuitive puisque la largeur linéaire d'un graphe G est équivalente au nombre de gendarmes requis pour capture un voleur arbitrairement rapide et invisible, de façon monotone (Bienstock & Seymour, 1991), alors que dans notre cas, le lapin semble plus faible que le voleur, puisqu'il ne peut se déplacer que d'une case et est contraint de bouger à chaque tour. Intuitivement, le nombre de chasseurs (monotone) pourrait être beaucoup plus petit que la largeur linéaire, ce qui est le cas pour le nombre de chasseurs, mais pas dans le cas monotone. Au contraire, nous montrons que la monotonie rend ces deux jeux presque similaires, puisque ces deux paramètres ne peuvent différer que d'un.

Rappelons qu'une décomposition linéaire d'un graphe $G = (V, E)$ est une séquence $P = (X_1, \ldots, X_p)$ de sous-ensemble de sommets, appelé *sacs*, tel que (1) $\bigcup_{i \leq p} X_i = V$; (2) pour tout $\{u, v\} \in E$, il existe $i \leq p$ avec $\{u, v\} \subseteq X_i$; et (3) : pour tout $1 \leq i \leq j \leq q \leq p$, $X_i \cap X_q \subseteq X_j$. La *largeur* $w(P)$ de P est la taille d'un plus gros sac de P, i.e., $w(P) = \max_{i \leq p} |X_i| - 1$. La *largeur linéaire* $pw(G)$ de G est la largeur minimum des décompositions linéaires de G. Une décomposition linéaire de G de largeur $pw(G)$ est dite *optimale*. Une décomposition linéaire est *réduite* si aucun sac n'est contenu dans un autre. Il est bien connu que n'importe quel graphe admet une décomposition linéaire réduite et optimale.

Théorème 4.2.8. *Pour n'importe quel graphe $G = (V, E)$, $pw(G) \leq mh(G) \leq pw(G) + 1$.*

Démonstration. Premièrement, posons $P = (X_1, \ldots, X_\ell)$, une décomposition linéaire réduite de G de largeur k. Nous affirmons que P est une stratégie de chasse monotone dans G utilisant au plus $k + 1$ chasseurs. Cela vient directement du fait très connu, que pour tout $1 \leq i < \ell$, l'ensemble $X_i \cap X_{i+1}$ sépare $\bigcup_{1 \leq j \leq i} X_j \setminus X_{i+1}$ de $\bigcup_{i < j \leq \ell} X_j \setminus X_i$, et donc $Z_i \subseteq \bigcup_{i < j \leq \ell} X_j$ pour tout $1 \leq i \leq \ell$. En particulier, $Z_\ell = \emptyset$, et donc P est une stratégie de chasse gagnante. De plus, comme P est une stratégie pour les gendarmes gagnante et monotone dans G (par rapport aux règles du jeu DES GENDARMES ET DU VOLEUR), P est aussi une stratégie de chasse monotone. Pour conclure, $mh(G) \leq pw(G) + 1$.

Pour prouver l'autre inégalité, posons $\mathcal{S} = (S_1, \ldots, S_\ell)$, une stratégie de chasse parcimonieuse, monotone et gagnante dans G utilisant au plus $k \geq mh(G)$ chasseurs (cette stratégie existe par le Lemme 4.2.6). Nous aurons aussi besoin de l'assertion suivante certifiant que pour chaque sommet v non tiré pendant la stratégie, il existe un tour j tel que tout le voisinage de v est tiré pendant le tour j :

Assertion 4.2.8.1 – Pour tout $v \in V \setminus \bigcup_{1 \leq i \leq \ell} S_i$, il existe $1 \leq j \leq \ell$ tel que $N(v) \subseteq S_j$.

Démonstration. Pour arriver à une contradiction, supposons qu'il existe un sommet v non tiré, c'est-à-dire, $v \notin \bigcup_{1 \leq i \leq \ell} S_i$, tel que pour n'importe quel tour $1 \leq i \leq \ell$, il existe au moins un

sommet non tiré dans le voisinage de v, c'est-à-dire, $u_i \in N(v) \setminus S_i$. Dans ce cas, $\mathcal{R} = (r_0 = v, u_1, v, u_3, \dots, u_{2i-1}, v, u_{2i+1}, v \dots)$ est une trajectoire gagnante contre $\mathcal{S}$ pour le lapin, c'est-à-dire, $r_i \notin S_{i+1}$ pour n'importe quel $0 \leq i < \ell$, contredisant le fait que $\mathcal{S}$ soit une stratégie de chasse gagnante. ◇

Notre but est donc de construire une décomposition linéaire $\mathcal{P}$ de G de largeur k, prouvant que $pw(G) \leq mh(G)$. Nous la construisons par induction.

Commençons avec $\mathcal{P}_0 = \emptyset$ et posons $Y_0 = V \setminus \bigcup_{1 \leq i \leq \ell} S_i$, l'ensemble des sommets jamais tirés, pendant toute la stratégie $\mathcal{S}$. Pour tout $1 \leq i \leq \ell$, nous notons par $\mathcal{P}_i$ la séquence calculé à l'étape i de l'induction et nous notons par Y_i l'ensemble des sommets jamais tirés pendant toute la stratégie $\mathcal{S}$ et non présents dans $\mathcal{P}_i$. Définissons maintenant $\mathcal{P}_{i+1}$ et Y_{i+1} à partir de $\mathcal{P}_i$ et Y_i. Pour cela, nous devons gérer l'ensemble $H_{i+1} = \{u_1^{i+1}, \dots, u_{r_{i+1}}^{i+1}\} = \{v \in Y_i \mid N(v) \subseteq S_{i+1}\}$ des sommets qui ne sont jamais tirés, ne sont pas présents dans la séquence jusqu'à présent calculée, et dont leurs voisinages sont complètement tirés pendant le tour $i+1$. Nous allons donc rajouter plusieurs sacs à la suite de $\mathcal{P}_i$ grâce à l'opérateur $\odot$ dénotant la concaténation de deux séquences. Nous pouvons donc définir $\mathcal{P}_{i+1} = \mathcal{P}_i \odot (S_{i+1} \cup \{u_1^{i+1}\}, \dots, S_{i+1} \cup \{u_{r_{i+1}}^{i+1}\})$ et définir $Y_{i+1} = Y_i \setminus H_{i+1}$. Finalement, notons par $\mathcal{P} = (X_1, \dots, X_r)$ la décomposition linéaire $\mathcal{P}_\ell$ obtenue.

Premièrement, notons que par construction, pour tout $1 \leq i \leq r$, $|X_i| \leq k+1$, et donc $w(\mathcal{P}) \leq k$. Il ne nous reste donc qu'à prouver que $\mathcal{P}$ satisfait les trois propriétés des décompositions linéaires.

Par construction, $\bigcup_{1 \leq i \leq \ell} S_i \subseteq \bigcup_{1 \leq i \leq r} X_i$. De plus, par l'Assertion 4.2.8.1, $Y_0 \subseteq \bigcup_{1 \leq i \leq r} X_i$. Par conséquent, $\bigcup_{1 \leq i \leq r} X_i = V$ et donc, la propriété (1) des décompositions linéaires est satisfaite par $\mathcal{P}$.

Par construction, pour tout sommet v jamais tiré ($v \in Y_0$), il existe un unique $1 \leq i \leq r$ tel que $v \in X_i$ (v est contenu dans un seul sac, celui correspondant au premier tour tel que le voisinage de v est complètement tiré). La propriété (3) est donc satisfaite par ces sommets dans Y_0. Pour tous les autres sommets, c'est-à-dire, $v \in V \setminus Y_0$, notons par $1 \leq i \leq j \leq r$ les indices minimaux et maximaux tels que $v \in X_i \cap X_j$. Notons aussi par $1 \leq i' \leq j' \leq \ell$ les indices tels que X_i et X_j ont été construits respectivement de $S_{i'}$ et $S_{j'}$. Par le Lemme 4.2.7, $v \in S_{p'}$ pour tout $i' \leq p' \leq j'$. Par conséquent, par construction, $v \in X_p$ pour tout $i \leq p \leq j$. La propriété (3) des décompositions linéaires est donc respectée par tous les sommets de G.

Il ne reste plus qu'à prouver la deuxième propriété des décompositions linéaires. Pour ce faire, considérons n'importe quelle arête $\{u, v\} \in E$. Premièrement, supposons que u n'est jamais tiré, c'est-à-dire, $u \in Y_0$. Par l'Assertion 4.2.8.1, $v \notin Y_0$. De plus il existe un sac X_j contenant u, qui par construction, contient le voisinage de u, et donc v. Nous pouvons donc supposer que $u, v \in V \setminus Y_0$. Pour arriver à une contradiction, supposons qu'aucun sac ne contienne u et v, c'est-à-dire, que pour tout $1 \leq i \leq r$, $|\{u, v\} \cap X_i| \leq 1$. Sans perte de généralité, nous pouvons définir $M = \max\{1 \leq j \leq r \mid u \in X_j\} < m = \min\{1 \leq j \leq r \mid v \in X_j\}$ (ces deux entiers m et M sont bien définis puisque les propriétés (1) et (3) sont satisfaites). Notons aussi par $1 \leq M' \leq m' \leq \ell$ les indices tels que X_M a été créé à partir de $S_{M'}$ et X_m a été construit à partir de $S_{m'}$. Par définition de $\mathcal{P}$, $u \in S_{M'} \bigcup_{M' < i \leq \ell} S_i$ et $v \in S_{m'} \setminus \bigcup_{1 < i < m'} S_i$. Puisque $\mathcal{S}$ est parcimonieuse, $v \in Z_{m'-1}(\mathcal{S})$ et donc, il existe un sommet $w \in N(v) \cap Z_{m'-2} \setminus S_{m'-1}$. Puisque $\mathcal{S}$ est monotone, ce sommet w n'a jamais été tiré avant le tour m', c'est-à-dire, $w \notin \bigcup_{1 \leq i < m'} S_i$. Par la Proposition 4.2.2, $u \in Z_{m'-1}(\mathcal{S})$. Ainsi, $u \in (Z_{m'-1}(\mathcal{S}) \cap S_{M'}) \setminus S_{m'}$, c'est-à-dire, le sommet u est recontaminé au tour m', contredisant la monotonie de $\mathcal{S}$. Par conséquent, pour toute arête

$\{u, v\} \in E$, il existe $1 \leq j \leq r$ tel que $u, v \in X_j$, c'est-à-dire, la propriété (2) des décompositions linéaires est satisfaite par $\mathcal{P}$.

Nous pouvons donc conclure que $\mathcal{P}$ est une décomposition linéaire de largeur au plus k, prouvant que, $pw(G) \leq mh(G)$. □

Le Théorème 4.2.8 a d'importantes conséquences :

Corollaire 4.2.9. *Sauf si* P = NP, *il n'existe aucun algorithme polynomial décidant si* $mh(G) \leq k$, *et ce, qu'importe le graphe* G *et l'entier* k.

Démonstration. Cela vient du Théorème 4.2.8 et du fait que la largeur linéaire ne peut pas être approchée par un ratio de $(+1)$, c'est-à-dire, il n'y a pas d'algorithme polynomial calculant des décompositions linéaires d'un graphe G quelconque de largeur au plus $pw(G) + 1$ (Bodlaender et al., 1995) (sauf si P = NP). □

De plus, le Théorème 4.2.8 implique que la recontamination n'aide pas dans le jeu DES CHASSEURS ET DU LAPIN.

Corollaire 4.2.10. *Il existe* $\varepsilon > 0$ *tel que, pour tout* $k \in \mathbb{N}$, *il existe un arbre* T *avec* $h(T) \geq k$ *et* $mh(T) \geq (1 + \varepsilon)h(T)$.

Démonstration. Pour tout $n \in \mathbb{N}$, notons par T_n (resp. B_n) l'arbre enraciné défini récursivement comme suit : T_0 (resp. B_0) est un sommet, et, pour tout $n > 0$, T_n (resp. B_n) est obtenu à partir de trois (resp. deux) copies de T_{n-1} (resp. B_{n-1}) et un nouveau sommet r (la racine de T_n) tel que r est adjacent aux trois racines des trois copies de T_{n-1} (resp. B_{n-1}). Pour tout $n \geq 0$, T_n contient donc $\frac{3^{n+1}-1}{2}$ sommets et, par le Lemme de Parson (Parsons, 1978), $pw(T_n) = n = \Theta(\log_3(V(T)))$. Pour tout $n \in \mathbb{N}$, notons par T_n^* (resp. B_n^*) l'arbre obtenue à partir de T_n (resp. B_n) en sous-divisant chaque arête une fois. Pour tout $n \geq 0$, T_n^* contient donc $3^{n+1} - 2$ sommets et, puisque T_n est un mineur de T_n^*, $pw(T_n^*) \geq pw(T_n) = n = \Theta(\log_3(V(T)))$. De plus, B_n^* est un sous-grahe isométrique de T_n^*, et donc $h(T_n^*) \geq h(B_n^*) \geq (\frac{1}{4} - \mathcal{O}(1)) \log_2(V(T))$ (Gruslys & Méroueh, 2015). D'un autre côté, il a été montré dans (Gruslys & Méroueh, 2015) que, pour tout arbre T, $h(T) \leq \lceil \frac{\log_2 |V(T)|}{2} \rceil$. Le résultat suit pour $(1 + \varepsilon) = 2\frac{\ln(2)}{\ln(3)}$. □

4.3 Nombre de chasseurs (monotone) de quelques classes de graphes

Dans cette section, nous caractérisons le nombre de chasseurs monotone de plusieurs classes de graphes telles que les graphes scindés ("Split graphs" en anglais), les graphes d'intervalles, les cographes, et les arbres. En particulier, pour toutes ces classes, nos résultats impliquent l'existence d'un algorithme polynomial pour déterminer le nombre de chasseurs monotone.

4.3.1 Graphes scindés ou d'intervalles

Un graphe $G = (V, E)$ est un graphe *scindé* si $V = C \cup I$ peut être partitionné en deux ensembles, C et I, induisant respectivement, une clique maximale par inclusion et un ensemble indépendant. Notons que, étant donné un graphe scindé G, une partition (C, I) peut être calculée en temps linéaire (Hammer & Simeone, 1981). Dans ce qui suit, nous noterons les graphes scindés par $G = (C \cup I, E)$ où C induit une clique maximal par inclusion et I est un ensemble indépendant. Rappelons aussi les résultats sur la largeur linéaire des graphes scindés :

Lemme 4.3.1. *(Gustedt, 1993) Soit $G = (C \cup I, E)$ un graphe scindé. Alors, $|C| - 1 \leq pw(G) \leq |C|$.*

Premièrement, nous avons cette observation simple :

Proposition 4.3.2. *Soit $G = (C \cup I, E)$ un graphe scindé. Alors, $|C| - 1 \leq h(G) \leq mh(G) \leq |C|$.*

Démonstration. Par le Lemme 4.1.1, $h(G) \geq h(G[C])$, et par le Lemme 1.5.7, $h(G[C]) \geq \delta(G[C]) = |C| - 1$. Par conséquent, $h(G) \geq |C| - 1$. De plus, la stratégie de chasse consistant à tiré sur tous les sommets de C deux fois est clairement une stratégie gagnante et monotone dans G. Par conséquent, $h(G) \leq mh(G) \leq |C|$. □

Le prochain théorème caractérise complètement le nombre de chasseurs des graphes scindés.

Théorème 4.3.3. *Soit $G = (C \cup I, E)$ un graphe scindé. Alors, $h(G) = |C|$ si et seulement si pour toute paire de sommets $x, y \in C$, il existe un sommet $z \in N_I(x) \cap N_I(y)$. Sinon, $h(G) = |C| - 1$.*

Démonstration. Premièrement, nous montrons que si pour toute paire de sommets $x, y \in C$, il existe $z \in I$ tel que $\{x, z\} \in E$ et $\{y, z\} \in E$, alors $h(G) = |C|$. Nous prouvons que $h(G) > |C| - 1$ en montrant qu'il existe une stratégie de survie contre $|C| - 1$ chasseurs. Plus précisément, qu'importe la stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_\ell)$ (fixé) que ces $|C| - 1$ chasseurs utilisent, nous montrons comment construire une trajectoire $\mathcal{R} = (r_0, r_1, \ldots, r_{\ell-1})$ pour le lapin tel que pour tout $i \geq 0$, $r_i \notin S_{i+1}$. Puisque $|S_1| \leq |C| - 1$, il existe au moins un sommet v de C non tiré pendant le premier tour. Posons donc $r_0 = v$. Nous avons donc construit une trajectoire pour le lapin gagnante contre (S_1).

Maintenant, pour $i \geq 0$, nous allons supposer que nous avons construit une trajectoire $(r_0, r_1, \ldots, r_i)$ gagnante contre $(S_1, \ldots, S_i, S_{i+1})$ et finissant sur un sommet de C, c'est-à-dire, $r_i \in C$. Rappelons que $\mathcal{S}$ utilise au plus $|C| - 1$ chasseurs, et donc il existe au moins un sommet v dans C qui n'est pas tiré pendant le tour $i + 2$. Il existe deux cas dépendants de v et r_i.

Si $r_i \neq v$, alors $r_{i+1} = v$. Ainsi, par hypothèse d'induction, $(r_0, r_1, \ldots, r_{i+1})$ est une trajectoire gagnante contre $(S_1, \ldots, S_i, S_{i+2})$.

Sinon, $S_{i+2} = C \setminus \{r_i\}$. De plus, observons qu'il existe au moins un sommet $w \in C$ tel que $w \notin S_{i+3}$ (puisque $|S_{i+3}| < |C|$). Il est possible que w soit le même sommet que r_i, mais cela n'a pas d'importance. Rappelons que nous sommes dans le cas où il existe $z \in I$ tel que $\{w, z\}, \{r_i, z\} \in E$. De plus $z \notin S_{i+2}$ puisque $S_{i+2} = C \setminus \{r_i\}$ et $z \notin C$. Nous pouvons donc définir $r_{i+1} = z$ and $r_{i+2} = w$ pour obtenir une trajectoire $(r_0, r_1, \ldots, r_{i+2})$ gagnante contre $(S_1, \ldots, S_i, S_{i+3})$ finissant sur C.

Par conséquent, $h(G) \geq |C|$. Puisque $h(G) \leq |C|$ (par la Proposition 4.3.2), nous obtenons que $h(G) = |C|$.

Deuxièmement, nous montrons que s'il existe deux sommets distincts $x, y \in C$ tels que $N_I(x) \cap N_I(y) = \emptyset$ (c'est-à-dire, il n'existe pas de sommet z tel que $\{x, z\}, \{y, z\} \in E$), alors $h(G) \leq |C| - 1$ (et donc $h(G) = |C| - 1$ par la Proposition 4.3.2). Notons qu'il suffit de décrire une stratégie de chasse gagnante utilisant $|C| - 1$ chasseurs. Soit $\mathcal{S} = (S_1, S_2, S_3, S_4, S_5)$ où $S_1 = S_2 = S_5 = C \setminus \{y\}$ et $S_3 = S_4 = C \setminus \{x\}$. Cette stratégie de chasse utilise dont bien $|C| - 1$ chasseurs par définition. Il reste donc à prouver que $\mathcal{S}$ est gagnante. Posons n'importe quelle trajectoire $\mathcal{R} = (r_0, \ldots, r_4)$ pour le lapin. Si le lapin n'est pas capturé au premier tour, alors soit $r_0 = y$ ou soit $r_0 \in I$:

Case 1. $\mathbf{r_0 = y}$: Dans ce cas, si le lapin survie au-delà du premier tour, c'est-à-dire, $r_1 \notin S_2$ alors $r_1 \in N_I(y)$. Cependant, puisque $N_I(x) \cap N_I(y) = \emptyset$, le lapin ne peut pas se rendre ensuite sur x

ou rester sur I, c'est-à-dire, $r_2 \in C \setminus \{x\}$. Puisque $S_3 = C \setminus \{x\}$, il n'y a donc pas de trajectoire gagnante contre $\mathcal{S}$ commençant par y.

Case 2. $\mathbf{r_0 \in I}$: Dans ce cas, si le lapin survie au-delà du premier tour, c'est-à-dire, $r_1 \notin S_2$, alors $r_1 = y$. De même, si le lapin veut survivre au troisième tour, il peut soit se rendre sur x, soit se rendre sur I (en partant de y) :

2.a $\mathbf{r_2 \in N_I(y)}$: Ce cas est similaire au cas 1. Puisque $N_I(x) \cap N_I(y) = \emptyset$, le lapin pourra seulement se rendre sur $C \setminus \{x\}$ pendant le quatrième tour, c'est-à-dire, $r_3 \in C \setminus \{x\}$. Le lapin sera alors directement tiré puisque $S_4 = C \setminus \{x\}$, ce qui implique qu'il n'y a pas de trajectoire gagnante contre $\mathcal{S}$ commençant sur I sauf si le lapin va ensuite sur y puis x.

2.b $\mathbf{r_2 = x}$: Dans ce cas, si le lapin veut survivre au quatrième tour, c'est-à-dire, $r_3 \notin S_4$, alors il doit se rendre sur I, c'est-à-dire, $r_3 \in N_I(x)$. Par conséquent, puisque $N_I(x) \cap N_I(y) = \emptyset$, le lapin sera obligé de se rendre sur $r_4 \in C \setminus y$. Comme $S_5 = C \setminus \{y\}$, il n'y a donc pas de trajectoire gagnante contre $\mathcal{S}$ commençant par I.

Il n'y a donc aucune trajectoire gagnante contre $\mathcal{S}$, c'est-à-dire, $\mathcal{S}$ est gagnante dans G, prouvant que $h(G) = |C| - 1$ s'il existe $x, y \in C$ tel que $N(x) \cap N(y) = \emptyset$. □

La caractérisation précédente nous permet de montrer que le nombre de chasseurs et la largeur linéaire des graphes scindés sont équivalents.

Corollaire 4.3.4. *Pour tout graphe scindé $G = (C \cup I, E)$, $h(G) = pw(G)$.*

Démonstration. Si $|C| = 1$ et $I = \emptyset$, alors $pw(G) = h(G) = 0$. Si $|C| = 1$ et $I \neq \emptyset$, alors $pw(G) = h(G) = 1$. Nous pouvons donc supposé que $|C| > 1$.

Puisque $pw(G), h(G) \in \{|C| - 1, |C|\}$ par le Lemme 4.3.1 et la Proposition 4.3.2, supposons en premier que $h(G) = |C|$. Par le Théorème 4.3.3, pour toute paire de sommets distinct $x, y \in C$, il existe $z \in N_I(x) \cap N_I(y)$. Pour arriver à une contradiction, supposons qu'il existe une décomposition linéaire $P = (X_1, \ldots, X_\ell)$ réduite optimale de G de largeur $|C| - 1$. Il est très connu qu'il existe un entier $1 \leq i \leq \ell$ tel que $C \subseteq X_i$. De plus, puisque P a largeur $|C| - 1$, $X_i = C$. Montrons que $1 < i < \ell$. Supposons par contradiction que $i = 1$, c'est-à-dire, $X_1 = C$ (le cas $i = \ell$ est symétrique). Puisque P a largeur $|C|-1$ et est réduite (en particulier, $X_1 \setminus X_2 \neq \emptyset$), posons $v \in X_1 \setminus X_2$ et $z \in N_I(v)$ (z existe puisque $|C| > 1$ et pour toute paire de sommets distincts dans C ont un voisin commun dans I). Puisque $v \in X_1 = C$ et $v \notin \bigcup_{1<j\leq\ell} X_j$, il n'y a donc aucun sac contenant z et v, contredisant que la propriété (2) des décompositions linéaires est respectée par P. Nous avons donc que $1 < i < \ell$. À présent, posons $x \in C \setminus X_{i-1}$ et $y \in C \setminus X_{i+1}$ (x et y existent puisque P est réduite et $C = X_i$). Posons $z \in N_I(x) \cap N_I(y)$. Notons que $x \neq y$, sinon la propriété (3) des décompositions linéaires n'est pas respectée. Par conséquent, il doit exister un sac X_j avec $1 \leq j \leq \ell$ contenant y et z (propriété (2)). Puisque $y \notin \bigcup_{i<h\leq\ell} X_h$ (propriété (3)), $j < i$ et puisque $z \in X_j \setminus X_i$, $z \notin \bigcup_{i<h\leq\ell} X_h$ (propriété (3)). Finalement, puisque $x \notin \bigcup_{1\leq h<i} X_h$, il n'y a pas de sac contenant x et z, contredisant que la propriété (2) des décompositions linéaires est respectée par P.

Deuxièmement, supposons que $h(G) = |C| - 1$. Par le Théorème 4.3.3, il existe une paire de sommets $x, y \in C$ telle que $N_I(x) \cap N_I(y) = \emptyset$. Soient $N_I(x) = \{x_1, \ldots, x_m\}$ et $I \setminus N_I(x) = \{y_1, \ldots, y_t\}$. Soit $P = (x_1 \cup (C \setminus \{y\}), \ldots, x_m \cup (C \setminus \{y\}), C, y_1 \cup (C \setminus \{x\}), \ldots, y_t \cup (C \setminus \{x\}))$. Notons que cette séquence est une décomposition linéaire de largeur $|C| - 1$. Puisque $pw(G) \geq |C| - 1$ par le Lemme 4.3.1, nous obtenons que $pw(G) = |C| - 1$. □

Ensuite, nous caractérisons aussi le nombre de chasseurs monotone des graphes scindés. Pour cela, nous aurons besoin du Lemme suivant :

Lemme 4.3.5. *Soit G un graphe quelconque. Si G contient une clique C comme sous-graphe telle que $N(v) \setminus C \neq \emptyset$ pour tout $v \in C$, alors $mh(G) \geq |C|$.*

Démonstration. Par les Lemmes 4.1.1, 1.5.7 et la Proposition 4.2.1, $mh(G) \geq h(G) \geq h(C) \geq |C| - 1$. Soit $H = G[N[C]]$. Nous allons montrer que $mh(H) \geq |C|$ et donc, $mh(G) \geq |C|$ par le Lemme 4.2.5.

Supposons, pour arriver à une contradiction, que $mh(H) = |C| - 1$. Par le Lemme 4.2.6, il existe une stratégie de chasse parcimonieuse monotone et gagnante $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans H utilisant $|C| - 1$ chasseurs.

De plus, il doit exister un entier $1 \leq i \leq \ell$ tel que $|S_i \cap C| = |C| - 1$. En effet, dans le cas contraire, la trajectoire $(r_0, \ldots, r_\ell)$, où $r_0 \in C \setminus S_1$ et $r_j \in C \setminus (S_{j+1} \cup \{r_{j-1}\})$ pour tout $1 \leq j \leq \ell$, est gagnante contre $\mathcal{S}$. Notons par i le plus petit indice tel que $|S_i \cap C| = |C| - 1$, notons aussi par v le sommet de C non tiré pendant le tour i, et finalement notons par w un voisin de v n'étant pas dans C (ce sommet existe d'après les hypothèses de l'énoncé). Remarquons que v n'a jamais été tiré avant le tour i, sinon il serait recontaminé durant le tour i, contredisant la monotonie de $\mathcal{S}$.

Si $w \in \bigcup_{1 \leq j < i} S_j$, alors la trajectoire $(r_0, \ldots, r_{i-1} = v)$ telle que $r_{i-1} = v$ et $r_j \in C \setminus (S_{j+1} \cup \{r_{j+1}\})$ pour tout $1 \leq j < i-1$, ce qui est possible par minimalité de i. Si $w \notin S_{i+1}$, alors définissons $r_i = w$, c'est-à-dire, w est recontaminé au tour $i+1$ (puisque $w \in Z_i \cap \bigcup_{1 \leq j < i} S_j) \setminus S_{i+1}$), ce qui contredit donc la monotonie de $\mathcal{S}$. Nous pouvons donc supposer que $w \in S_{i+1}$ et donc il existe $z \in C \setminus (S_{i+1} \cup \{v\})$. Définissons alors $r_i = z$, c'est-à-dire, z est recontaminé au tour $i+1$ (puisque $z \in Z_i \cap S_i) \setminus S_{i+1}$), contredisant la monotonie de $\mathcal{S}$. Le fait que $w \in \bigcup_{1 \leq j < i} S_j$, implique donc une contradiction dans tous les cas. Nous obtenons donc que $w \notin \bigcup_{1 \leq j < i} S_j$.

Soit $j > i$ le plus petit indice tel que $w \in S_j$, ou, si w n'est jamais tiré, définissons alors $j > i$ le plus petit indice tel que $v \in S_j$ (au moins un des indices est bien défini, sinon, le lapin peut utiliser la trajectoire oscillant entre v et w pour survivre indéfiniment). Dans les deux cas, posons $z \in C \setminus (S_j \cup \{v\})$. Puisque v et w ne sont jamais tirés avant le tour j, par la Proposition 4.2.2, nous obtenons que $z \in (Z_{j-1} \cap S_i) \setminus S_j$, c'est-à-dire, z est recontaminé au tour j, contredisant la monotonie de $\mathcal{S}$. □

Rappelons qu'un sommet du graphe G est dit *simplicial* si ses voisins (et v) induisent une clique. En particulier, dans un graphe scindé $= (C \cup, E)$, un sommet $v \in C$ est simplicial si et seulement si $N(v) \setminus C = \emptyset$ (parce que C est censée être une clique maximale par inclusion).

Théorème 4.3.6. *Soit $G = (C \cup I, E)$ un graphe scindé. $mh(G) = |C| - 1$ si et seulement si il existe au moins un sommet simplicial dans C. Sinon, $mh(G) = |C|$.*

Démonstration. Remarquons en premier, que s'il n'existe aucun sommet simplicial dans C, alors par le Lemme 4.3.5, $mh(G) \geq |C|$ et donc, par la Proposition 4.3.2, $mh(G) = |C|$. Dans le cas contraire, notons par v un sommet simplicial de C quelconque. $S = (C \setminus v, C \setminus v)$ est une stratégie de chasse monotone et gagnante dans G, c'est-à-dire, $mh(G) \leq |C| - 1$, et donc, par la Proposition 4.3.2, $mh(G) = |C| - 1$. □

Notons que les précédents résultats impliquent qu'il existe des graphes scindés G pour lesquels $mh(G) > h(G)$, c'est-à-dire, la recontamination peut aider dans les graphes scindés. Par exemple, un tel graphe G peut être obtenu à partir d'une clique $C = \{x_1, \ldots, x_n\}$ contenant n sommets et d'un ensemble indépendant $I = \{i_1, \ldots, i_n\}$ contenant aussi n sommets de telle sorte que pour tout $1 \leq j \leq n$, $\{x_j, i_j\} \in E(G)$, et pour tout $h \neq j$, $\{x_h, i_j\} \notin E$. Par le Théorème 4.3.6 et le Lemme 4.3.3, nous obtenons bien que $mh(G) = n$ et $h(G) = n - 1$.

Pour conclure cette section, nous utilisons le Lemme 4.3.5 pour les graphes d'intervalles. Rappelons qu'un *graphe d'intervalles* est le graphe d'intersection d'un ensemble d'intervalles de la droite des réels. Il est très connu que, pour ces graphes d'intervalles G, $pw(G) = \omega(G) - 1$ où $\omega(G)$ est la taille maximum des cliques de G, et que G admet une décomposition linéaire optimale où chaque sac induit une clique.

Théorème 4.3.7. *Soit G un graphe d'intervalles. $h(G) = mh(G) = \omega(G) - 1$ si toute clique maximum contient un sommet simplicial (dans G). Dans le cas contraire, $mh(G) = \omega(G)$.*

Démonstration. Par le Théorème 4.2.8, $pw(G) \leq mh(G) \leq pw(G) + 1 = \omega(G)$. De plus, par le Lemme 1.5.7, $h(G) \geq \omega(G) - 1$. S'il existe une clique maximum ne contenant aucun sommet simplicial, alors, par le Lemme 4.3.5, $mh(G) = \omega(G)$. Dans le cas contraire, posons $(X_1, \ldots, X_\ell)$ une décomposition linéaire de G optimale telle que chaque sac induit une clique. Pour tout $1 \leq i \leq \ell$, si X_i contient un sommet simplicial v_i, définissons $Y_i = \{v_i\}$, sinon, définissons $Y_i = \emptyset$. $(X_1 \setminus Y_1, X_1 \setminus Y_1, X_2 \setminus Y_2, X_2 \setminus Y_2, X_3 \setminus Y_3, \ldots, X_\ell \setminus Y_\ell, X_\ell \setminus Y_\ell)$ est donc une stratégie de chasse monotone utilisant $\omega(G) - 1$ chasseurs. □

Nous avons donc $\omega(G) - 1 \leq h(G) \leq \omega(G)$. Malheureusement, calculer $h(G)$ pour un graphe d'intervalles G quand une clique maximum ne contient aucun sommet simplicial (dans G) parait beaucoup plus complexe.

4.3.2 Cographes

La classe des *cographes* peut être définie de façon récursive comme suit (Corneil, Perl, & Stewart, 1985) : Pour le cas de base, un sommet seul est un cographes. Ensuite, nous pouvons construire un cographe G à partir de deux autres cographes A et B, soit grâce à l'union disjointe $A \cup B$, c'est-à-dire, $G = (V(A) \cup V(B), E(A) \cup E(B))$, soit grâce à l'opération "join" $A \bowtie B$, consistant en l'union disjointe, puis en rajoutant aussi toutes les arêtes possibles entre les sommets de A et les sommets de B, c'est-à-dire, $G = (V(A) \cup V(B), E(A) \cup E(B) \cup \{\{a, b\} \mid a \in V(A), b \in V(B)\})$. Notons qu'il existe des *décompositions de cographes* G, correspondant aux différentes séquences d'unions et de joins permettant de construire G à partir d'un ensemble de sommets indépendants. De plus, une décomposition d'un cographe G peut être calculée en temps linéaire (Corneil et al., 1985).

Théorème 4.3.8. *$mh(G)$ peut être calculé en temps linéaire dans la classe des cographes.*

Démonstration. Soient A et B deux cographes. Nous prouvons que :
— $mh(A \cup B) = max(mh(A), mh(B))$, et ;
— $mh(A \bowtie B) = min(mh(A) + |V(B)|, |V(A)| + mh(B))$.
Ensuite, puisque nous pouvons calculer une décomposition d'un cographe G en temps linéaire (Corneil et al., 1985), nous pouvons aussi calculer $mh(G)$ en temps linéaire en utilisant cette décomposition.

Le fait que $mh(A \cup B) = max(mh(A), mh(B))$ est trivial. Par conséquent, fixons $G = A \bowtie B$ et notons respectivement par $\mathcal{S}^A = (S_1^A, \ldots, S_\ell^A)$ et $\mathcal{S}^B$, des stratégies de chasse monotones, gagnantes respectivement dans A et B utilisant respectivement $mh(A)$ et $mh(B)$ chasseurs. Remarquons que $\mathcal{S}^A \cup V(B) = (S_1^A \cup V(B), \ldots, S_\ell^A \cup V(B))$ et $\mathcal{S}^B \cup V(A)$ sont tous deux des stratégies de chasse monotones et gagnantes dans G. Par conséquent, $mh(G) \leq min(mh(A) + |V(B)|, |V(A)| + mh(B))$.

Notons par $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse monotone, parcimonieuse et gagnante dans G utilisant au plus $k \geq mh(G)$ chasseurs (cette stratégie existe par le Lemme 4.2.6) telle que ℓ est minimisé parmi ces stratégies. Si $\ell = 1$, alors $k = |S_1| = |V(G)| \geq min(mh(A) + |V(B)|, |V(A)| + mh(B))$. Par conséquent, supposons à présent que $\ell > 1$. Notons par $\mathcal{Z} = (Z_0, \ldots Z_\ell)$ l'ensemble des sommets contaminé pour chaque tour de la stratégie de chasse $\mathcal{S}$. Puisque ℓ est minimisé par $\mathcal{S}$, $(S_2, \ldots, S_\ell)$ ne peut pas être une stratégie de chasse monotone, parcimonieuse et gagnante de G, et donc $Z_1 \neq V$. Notons alors par v un des sommets de G non contaminé à la fin du tour 1, c'est-à-dire, $v \in V \setminus Z_1$.

Sans perte de généralité, nous pouvons supposer que $v \in V(A)$ (le cas où $v \in V(B)$ est symétrique). Puisque $v \notin Z_1$ et $Z_0 = V$, $N(v) \subseteq S_1$. Puisque $B \subseteq N(v)$, nous obtenons que $V(B) \subseteq S_1$. De plus, $V(B) \subseteq Z_1$. Effectivement, si nous supposons le contraire, nous obtenons que pour tout $w \in V(B)$, $N(w) \subseteq S_1$, et donc $V(A) \subseteq S_1$. En résumé, nous obtenons que $S_1 = V(G)$. Cela contredit que $\ell > 1$.

À présent, nous prouvons par induction que, pour tout $1 \leq i < \ell$, $V(B) \subseteq Z_i$ et $V(B) \subseteq S_i$. Puisque $V(B) \subseteq Z_1 \cap S_1$, supposons que $V(B) \subseteq Z_i \cap S_i$ pour $1 \leq i < \ell - 1$. Premièrement, par monotonie et parce que $V(B) \subseteq Z_i \cap S_i$, nous savons que $V(B) \subseteq S_{i+1}$. Pour prouver que $V(B) \subseteq Z_{i+1}$, supposons au contraire qu'il existe un sommet $b \in V(B)$ tel que $v \notin Z_{i+1}$. Cela implique que $V(A) \cap Z_i \subseteq S_{i+1}$. Par conséquent, $Z_i = (V(A) \cap Z_i) \cup (V(B) \cap Z_i) \subseteq S_{i+1}$, et donc, que $Z_{i+1} = \emptyset$, contredisant la minimalité de ℓ. Par conséquent, pour tout $1 \leq i < \ell$, $V(B) \subseteq Z_i$ et $V(B) \subseteq S_i$ et en particulier $V(B) \subseteq Z_{\ell-1} \cap S_{\ell-1}$. Par monotonie, cela implique que $V(B) \subseteq S_\ell$ et donc que tous les sommets de B sont continuellement tirés pendant $\mathcal{S}$.

Définissons $\mathcal{S} \cap V(A) = (S_1 \cap V(A), \ldots, S_\ell \cap V(A))$ la stratégie de chasse sur $G[A]$. Puisque $\mathcal{S}$ est monotone et gagnante dans G, n'utilise que k chasseur et que $V(B) \subseteq S_i$ pour tout $1 \leq i \leq \ell$, $\mathcal{S} \cap V(A)$ est monotone et gagnante dans $G[A]$ et n'utilise que $k - V(B)$ chasseurs. Par conséquent, nous obtenons que $k - |V(B)| \geq mh(A)$. □

Encore une fois, calculer le nombre de chasseurs (non-monotone) parait beaucoup plus complexe. En particulier, le lemme suivant montre que la recontamination aide dans les cographes.

Lemme 4.3.9. *Pour tout $k \geq 2$, il existe un cographe G tel que $h(G) \geq k$ et $mh(G) = \frac{3}{2}h(G) - 1$.*

Démonstration. Soit $a \geq 1$. Définissons par A et B deux cographes (isomorphique) consistant en l'union disjoint d'une clique contenant a sommets (dénotées par K_A et K_B) et d'un ensemble indépendant contenant a sommets. Notons que A et B contiennent tous deux $2a$ sommets. Soit $G = A \bowtie B$. Clairement, $h(A) = mh(A) = h(B) = mh(B) = a - 1$ et, par le Théorème 4.3.8, $mh(G) = 3a - 1$. Notons aussi $h(G) \geq 2a$ par le Lemme 1.5.7. De plus, $(A, K_A \cup K_B, K_B, A)$ est une stratégie de chasse (non-monotone) gagnante dans G utilisant $2a$ chasseurs et donc, $h(G) = 2a$. G est donc bien un cographe tel que $mh(G) = \frac{3}{2}h(G) - 1$. □

4.3.3 Arbres

Cette section est dévouée au calcul du nombre de chasseurs monotone dans les arbres. En résumé, nous allons d'abord montrer qu'il existe (presque) l'équivalent du Lemme de Parson (Parsons, 1978) pour le nombre de chasseurs monotone. Grâce à ce lemme, nous serons capables de décrire un algorithme polynomial calculant le nombre de chasseurs monotone, dont l'idée générale correspond à celle de l'algorithme permettant de calculer la largeur linéaire des arbres (Ellis et al., 1994).

Mais tout d'abord, commençons par le cas simple des chemins.

Proposition 4.3.10. *Soit P un chemin. Si P contient au moins 4 sommets, alors, $1 = h(P) < mh(P) = 2$.*

Démonstration. Le fait que $h(P) = 1$ a déjà été prouvé dans (Abramovskaya et al., 2016) et ce, qu'importe le nombre de sommets de P (sauf si $V(P) = 1$ et donc par définition $h(P) = 0$).

Notons que pour tout chemin $P = (p_1, \ldots, p_n)$ contenant n sommets (possiblement $n < 4$), la stratégie $\mathcal{S} = (\{p_1, p_2\}, \{p_2, p_3\}, \ldots, \{p_i, p_{i+1}\}, \{p_{i+1}, p_{i+2}\}, \ldots, \{p_{n-1}, p_n\})$ est monotone et gagnante dans P. Par conséquent, $mh(P) \leq 2$, et donc, il ne nous reste plus qu'à prouver que $mh(P) > 1$ si P contient au moins 4 sommets. Par le Lemme 4.2.5, il est donc suffisant de montrer que $mh(P) > 1$ quand P contient exactement 4 sommets, c'est-à-dire, $P = (p_1, p_2, p_3, p_4)$. Notons qu'il est possible de partitionner P en une clique $\{p_2, p_3\}$ et un ensemble indépendant $\{p_1, p_4\}$ de taille 2 tel que $\{p_1, p_2\}, \{p_3, p_4\} \in E(P)$ and $\{p_1, p_4\}, \{p_2, p_4\} \notin E(P)$. Par conséquent, et par le Lemme 4.3.5, $mh(P) > 2$ si P contient au moins 4 sommets. □

Avant d'adapter le Lemme de Parsons au nombre de chasseurs monotone, nous avons besoin des deux lemmes techniques suivants :

Proposition 4.3.11. *Soient $G = (V, E)$ un graphe connexe quelconque et H un sous-graphe connexe de G. Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse parcimonieuse monotone et gagnante dans G. De plus, posons un entier $1 \leq i \leq \ell$ et deux sommets $x, y \in V(H)$ tels que $x \in \bigcup_{j<i} S_j$, $y \in Z_{i-1}$ et leur distance (entre x et y) dans H est minimisée. Si $x, y \notin S_i$, alors $\{x, y\} \in E(H)$.*

Démonstration. Remarquons en premier que $x \neq y$. Effectivement, dans le cas contraire, $x \in (\bigcup_{j<i} S_j \cap Z_{i-1}) \setminus S_i$, c'est-à-dire, x est recontaminé au tour i, ce qui contredit la monotonie de $\mathcal{S}$. Donc, pour le reste de la preuve, $x \neq y$. De plus, nous supposerons, pour arriver à une contradiction, que $\{x, y\} \notin E$. Avec le fait que H soit connexe, il existe forcément un chemin entre x et y dans H. Notons par P un plus court chemin entre x et y dans H. Notons aussi par a le voisin de x dans P. Puisque nous avons supposé que $xy \notin E$, nous obtenons que $y \neq a$. Par minimalité de la distance entre x et y, nous avons donc que $a \notin Z_{i-1}$ et $a \notin \bigcup_{j<i} S_j$. Notons finalement $b \neq x$ le voisin de a dans P différent de b. Remarquons que $b \notin \bigcup_{j<i} S_j$ même si $b = y$. Dans le cas où $b \neq y$, $b \notin \bigcup_{j<i} S_j$ par minimalité de la distance entre x et y. Dans le cas contraire, c'est-à-dire, $b = y$, par les hypothèses de l'énoncé, $y \in Z_{i-1} \setminus S_i$, et donc $y \notin \bigcup_{j<i} S_j$, sinon $\mathcal{S}$ n'est pas monotone.

Par conséquent ($a, b \notin \bigcup_{j<i} S_j$), par la Proposition 4.2.2, $a \in Z_q$ pour tout $q < i$, contredisant une des hypothèses précédemment faites, l'hypothèse $a \notin Z_{i-1}$. □

Lemme 4.3.12. *Soient $G = (V, E)$ un graphe quelconque et $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse parcimonieuse monotone et gagnante dans G utilisant au plus k chasseurs. Soit H, un sous-graphe connexe quelconque contenant au moins 2 sommets. Si $S_i \cap V(H) \neq \emptyset$ et $S_j \cap V(H) \neq \emptyset$ pour n'importe quels indices $1 \leq i < j \leq \ell$, alors $S_z \cap V(H) \neq \emptyset$ pour tout $i \leq z \leq j$.*

Démonstration. Soit $2 \leq i + 1 < j \leq \ell$ tel que $V(H) \cap S_i \neq \emptyset$ et $V(H) \cap S_j \neq \emptyset$. Pour arriver à une contradiction, supposons qu'il existe $i < z < j$ tel que $S_z \cap V(H) = \emptyset$. Notons par X l'ensemble des sommets de H tirés avant le tour z, c'est-à-dire, $X = V(H) \cap \bigcup_{q<z} S_q$. Puisque $V(H) \cap S_i \neq \emptyset$ et $i < z$, $X \neq \emptyset$. Notons aussi par Y, l'ensemble des sommets de H contaminé à la fin du tour z, c'est-à-dire, $Y = V(H) \cap Z_{z-1}$. Puisque $S_j \cap V(H) \neq \emptyset$ et $\mathcal{S}$ est parcimonieuse, nous avons aussi $Z_{j-1} \cap V(H) \neq \emptyset$. Soit $u \in Z_{j-1} \cap S_j \cap V(H)$. Par le Lemme 4.2.3, $u \in Z_q$ pour tout $q < j$. En particulier, $u \in Z_{z-1}$ et donc $Y \neq \emptyset$. Posons $x \in X$ et $y \in Y$ tel que $dist_H(x, y)$

est minimisé. Par la Proposition 4.3.11, $\{x, y\} \in E$. Par conséquent, puisque $y \in Z_{z-1} \setminus S_z$, nous avons $x \in Z_z$. Par conséquent, puisque $\mathcal{S}$ est monotone et $x \in \bigcup_{q<z} S_q$, nous avons $x \in S_{z+1}$. Par le Lemme 4.2.7, puisque $x \in \bigcup_{q<z} S_q$ et $x \in S_{z+1}$, nous avons $x \in S_z$, ce qui contredit les hypothèses de l'énoncé. □

Soit T un arbre quelconque et notons par v n'importe lequel de ses sommets. Une *branche* en v est n'importe quelle composante connexe de $T - v$. Une *étoile* est n'importe quel arbre contenant au moins 2 sommets, et contenant au plus un sommet de degré au moins 2. En résumé, le Lemme de Parson (Parsons, 1978) énonce que pour tout arbre T et n'importe quel entier $k \in \mathbb{N}$, $pw(T) \geq k + 1$ si et seulement si il existe un somme v dans T tel qu'il existe au moins trois branches en v nécessitant une largeur linéaire au moins k. Le lemme suivant est son adaptation au nombre de chasseurs monotone et la différence notable entre ces deux lemmes sont les cas de bases.

Lemme 4.3.13. *[Adaptation du Lemme de Parson] Soit $T = (V, E)$ un arbre connexe quelconque.*
- *$mh(T) = 0$ si et seulement si $|V| = 1$;*
- *$mh(T) = 1$ si et seulement si T est une étoile ;*
- *$mh(T) = 2$ si et seulement si T n'est pas une étoile et contient un chemin P tel que $T \setminus P$ est une forêt d'étoile et de sommets isolés ;*
- *Pour tout $k \geq 3$, $mh(T) \geq k$ si et seulement si il existe un sommet $v \in V$ tel qu'il existe au moins 3 branches en v nécessitant au moins $k - 1$ chasseur pour avoir une stratégie de chasse monotone gagnante.*

Démonstration. Le premier point est trivial par définition. Ensuite, si T est une étoile, alors tirer deux fois sur le sommet de degré au moins 2 est une stratégie de chasse monotone et gagnante utilisant un seul chasseur, c'est-à-dire, $mh(T) \leq 2$. Notons aussi qu'une étoile contient au moins deux sommets, et donc $mh(T) > 1$. Si T n'est pas une étoile (and $|V(T)| > 1$), il contient un chemin avec au moins 4 sommets comme sous-graphe. Par la Proposition 4.3.10 et le Lemme 4.2.5, il s'ensuit que $mh(T) \geq 2$, ce qui conclut la preuve du second point. Si T n'est pas réduit à une étoile et contient un chemin P tel que $T \setminus P$ est une forêt d'étoiles et de sommets isolés, il est facile de montrer que $mh(T) \leq 2$. Sinon, T contient un sommet v tel qu'au moins trois composantes de $T - v$ contiennent un chemin avec 4 sommets. Le "si" du quatrième point montre alors que $mh(T) > 2$ et conclut la preuve du troisième point.

Prouvons le quatrième point. Soit $k \geq 3$.

Preuve de $\Leftarrow$: Supposons d'abord qu'il existe un sommet v et trois branches B_1, B_2 et B_3 en v, telles que $mh(B_1), mh(B_2), mh(B_3) \geq k-1$. Nous allons montrer que $mh(T) \geq k$. Pour arriver à une contradiction, supposons aussi que $mh(T) < k$. Par le Lemme 4.2.6, il existe une stratégie de chasse parcimonieuse, monotone et gagnante $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans T qui utilise au plus $k-1$ chasseurs. Soit $Z = (Z_1, \ldots, Z_\ell)$ l'ensemble des sommets contaminés par rapport à $\mathcal{S}$.

Pour $j \in \{1, 2, 3\}$, $1 \leq i_j \leq \ell$ est l'entier minimum tel que $V(B_j) \cap Z_{i_j} = \emptyset$. Notons que, par le Lemme 4.2.3, $V(B_j) \cap Z_q = \emptyset$ pour tout $i_j \leq q \leq \ell$, et puisque $\mathcal{S}$ est parcimonieux, $V(B_j) \cap S_q = \emptyset$ pour tout $i_j < q \leq \ell$. Notons aussi que, puisque $mh(B_j) \geq k - 1 \geq 2$, B_j a au moins deux sommets. Puisque $Z_{i_j-1} \cap V(B_j) \neq \emptyset$ et $Z_{i_j} \cap V(B_j) = \emptyset$, cela implique que $S_{i_j} \cap V(B_j) \neq \emptyset$ (sinon, $u \in Z_{i_j} \cap V(B_j)$ où $w \in (Z_{i_j-1} \cap V(B_j))$ et $u \in N(w) \cap V(B_j)$, ce qui est contradictoire avec la définition de i_j). Supposons que $i_1 \leq i_2 \leq i_3$.

Nous allons montrer qu'il existe un tour j_2 au cours duquel tous les $k - 1$ chasseurs devront tirer sur des sommets de B_2, et que $v \in Z_{j_2-1}$, ce qui conduira à une contradiction.

Pour tout $1 \leq i \leq 3$, soit j_i un indice tel que $|S_{j_i} \cap V(B_i)| = k-1$. Ces indices existent, car sinon, par le Lemme 4.2.5, $mh(B_i) < k-1$. Notons que pour tout $i, j \in \{1,2,3\}$ tel que $i \neq j$, puisque $\mathcal{S}$ utilise au plus $k-1$ chasseurs, $S_{j_i} \cap V(B_i) \neq \emptyset$, $S_{j_j} \cap V(B_j) \neq \emptyset$ et $S_{i_j} \cap V(B_j) \neq \emptyset$, $j_i \neq j_j$ et $j_i \neq i_j$. Nous allons d'abord montrer que $j_2 < i_3$, ce qui nous permettra de prouver que $v \in Z_{j_2-1}$. Observons que $j_2 \leq i_2 \leq i_3$. Ainsi, puisque $j_2 \neq i_3$, $j_2 < i_3$. Par conséquent, par la Proposition 4.2.2 et le Lemme 4.3.12, $V(B_3) \cup \{v\} \subseteq Z_q$ pour tout $q \leq j_2$. En particulier, $v \in Z_{j_2-1}$. De plus, puisque $S_{j_2} \subseteq V(B_2)$, $v \notin S_{j_2}$. Par conséquent, $x \in Z_{j_2}$ où x est le voisin de v dans B_1, c'est-à-dire $Z_{j_2} \cap V(B_1) \neq \emptyset$.

Puisque $Z_{i_1} \cap V(B_1) = \emptyset$, si $i_1 < j_2$, alors il y a une contradiction avec le Lemme 4.2.3. Sinon, $j_2 < i_1 \leq i_2$ (car $j_2 \neq i_1$) et donc soit $j_1 < j_2 < i_1$, soit $j_2 < j_1 < i_2$ (car $j_1 \neq j_2$, $j_1 \leq i_1 \leq i_2$ et $j_1 \neq i_2$), ce qui contredit le Lemme 4.3.12.

Preuve de $\Rightarrow$: Supposons maintenant que, pour tout $v \in V$, au plus deux branches en v ont un nombre de chasseurs monotone au moins égal à $k-1$. Prouvons qu'il existe une stratégie de chasse parcimonieuse, monotone et gagnante $\mathcal{S}$ dans T utilisant au plus $k-1$ chasseurs.

Tout d'abord, supposons qu'il existe un chemin $P = (v_1, \ldots, v_p)$ tel que pour toute composante connexe C de $T \setminus P$, $mh(C) < k-1$. La stratégie de chasse suivante est parcimonieuse, monotone, gagnante et utilise au plus $k-1$ chasseurs. La stratégie consiste en p phases exécutées séquentiellement de $i = 1$ à p. La phase i consiste à tirer v_i à chaque tour, et à utiliser les $k-2$ tirs restants pour éliminer séquentiellement chaque composante connexe de $T \setminus P$ adjacente à v_i (ceci est possible puisque chacune de ces branches en v a un nombre de chasseurs monotone au plus égal à $k-2$). Enfin, le dernier tour de la phase i (sauf pour $i = p$) consiste à tirer à la fois sur v_i et v_{i+1} (rappelons que $k-1 \geq 2$).

Montrons maintenant qu'un tel chemin P existe. Soit X l'ensemble des sommets v tels qu'il existe exactement deux branches en v ayant un nombre de chasseurs monotone au moins égal à $k-1$. Supposons d'abord que $X \neq \emptyset$ et montrons qu'il induit un chemin. Posons x et y n'importe quelle paire de sommets dans X et z tout sommet interne du chemin entre x et y. Soit B (resp., B') la branche en z contenant x (resp., contenant y). Rappelons qu'il existe une branche B_x en x avec $mh(B_x) \geq k-1$ et notons que cette branche B_x est un sous-graphe de B et donc, par le Lemme 4.2.5, $mh(B) \geq k-1$. De même, $mh(B') \geq k-1$ et donc, $z \in X$ et donc X induit un sous-arbre de T. S'il existe un sommet w de degré au moins trois dans $T[X]$, par des arguments similaires, il existe au moins trois branches en w avec un nombre de chasseurs monotone au moins égal à $k-1$, contredisant l'hypothèse initiale. Par conséquent, X induit un chemin $(v_2, \ldots, v_{p-1})$. Notons par B_1 la branche en v_2 ne contenant pas v_3 telle que $mh(B_1) = k-1$ (si v_3 n'existe pas, B_1 est une branche quelconque en v_2 telle que $mh(B_1) \geq k-1$) et définissons v_1 comme étant le voisin de v_2 dans B_1. De façon symétrique, posons B_p la branche en v_{p-1} ne contenant pas v_{p-2} telle que $mh(B_p) = k-1$ (si v_{p-2} n'existe pas, B_p doit être une branche quelconque en v_2 différente de B_1 et telle que $mh(B_p) \geq k-1$) et définissons v_p comme le voisin de v_{p-1} dans B_p. Alors, $P = (v_1, v_2, \ldots, v_{p-1}, v_p)$ satisfait les conditions souhaitées.

Enfin, si $X = \emptyset$, posons v_1 comme n'importe quel sommet de T. Nous construisons le chemin P à partir de v_1 comme suit. Supposons qu'un chemin $(v_1, \ldots, v_i)$ ait déjà été construit pour un certain $i \geq 1$. S'il existe une branche B en v_i, ne contenant pas v_{i-1} (si $i > 1$), et avec un nombre de chasseurs monotone au moins $k-1$ (si elle existe, une telle branche doit être unique puisque $X = \emptyset$), alors, définissons v_{i+1} comme le voisin de v_i dans B. Le processus se termine lorsqu'une telle branche B n'existe pas et le chemin obtenu satisfait les conditions souhaitées.

Ceci conclut la preuve. □

Nous concevons un algorithme de programmation dynamique pour calculer le nombre de chasseurs monotone d'un arbre T. Commençons par enraciner T à n'importe quel sommet $r \in V(T)$. Pour n'importe quel sommet $u \in V(T)$, nous noterons par $T[u]$ le sous-arbre de T induit par u et tous les descendants de u. Soit $T[u, x, \ldots, y]$ le sous-arbre obtenu à partir de $T[u]$ après avoir supprimé tous les sommets de $V(T[x]) \cup \cdots \cup V(T[y])$ (où $x, \ldots, y \in V(T[u])$). Enfin, pour $k \geq 2$, un sommet $x \in V(T)$ est dit *k-critique* dans l'arbre T enraciné en r si et seulement si $mh(T[x]) = k$ et s'il existe deux enfants (et donc voisin de x) v_1 et v_2 de x dans T tels que $mh(T[v_1]) = mh(T[v_2]) = k$. Dans le cas $k = 1$, nous dirons qu'un sommet x est 1-critique si et seulement si $mh(T[x]) = 1$ et qu'il existe un unique fils v tel que $mh(T[v]) = 1$.

Remarque 4.3.14 – Rappelons que $mh(T[w]) \geq 1$ si $T[w]$ contient au moins une arête, c'est-à-dire que w a au moins un fils. Par conséquent, si un sommet w est 1-critique dans $T[w]$, alors $T[w]$ est une étoile centrée sur le fils w' de w tel que $mh(T[w']) = 1$, c'est-à-dire que w' est le seul sommet de $T[w]$ ayant un degré au moins égal à 2. De plus, si un sommet w n'est pas 1-critique et que $mh(T[w]) = 1$, alors $T[w]$ est une étoile centrée en w, c'est-à-dire que w est le seul sommet de degré au moins 2, à moins que $|V(T[w])| = 2$, auquel cas w peut aussi être de degré 1.

Le lemme suivant est utilisé tout au long de la preuve du corollaire 4.3.16 et du lemme 4.3.17.

Lemme 4.3.15. *Soit T un arbre quelconque enraciné en un sommet $v \in V(T)$. Soit $k = \max_{1 \leq i \leq d}\{mh(T[v_i]\}$, où $v_1, \ldots, v_d$ sont les enfants de v dans T. Pour tout sommet w dans T, il existe au plus 1 branche en w dans T ayant un nombre de chasseurs monotone au moins $k + 1$. De plus, $mh(T) \leq k + 1$.*

Démonstration. Remarquons d'abord que, par définition de k, il n'y a pas de branche en v qui ait un nombre de chasseurs monotone $k + 1$. Par conséquent, posons w comme n'importe quel sommet dans $V(T) \setminus \{v\}$ et désignons par x l'enfant de v tel que $w \in V(T[x])$. Par définition de k, $mh(T[x]) \leq k$. Pour tout fils w de z, $T[z]$ est un sous-arbre de $T[x]$. Ainsi, par le Lemme 4.2.5, $mh(T[z]) \leq k$ et donc il y a au plus 1 branche en w qui a un nombre de chasseurs monotone $k + 1$. Pour conclure, il est évident qu'appliquer séquentiellement l'ensemble des stratégies de chasse pour toutes les branches en v, tout en tirant continuellement sur v, est une stratégie de chasse monotone gagnante utilisant $k + 1$ chasseurs, c'est-à-dire, $mh(T) \leq k + 1$. □

Le corollaire suivant, obtenu à partir du lemme 4.3.13 et du lemme 4.3.15, décrit comment calculer $mh(T)$ d'un arbre enraciné T, de bas en haut, depuis ses feuilles jusqu'à la racine, de telle sorte que, pour tout $u \in V(T)$, $mh(T[u])$ est calculé à partir des valeurs $(mh(T[u_i]))_{u_i \; enfant \; de \; u}$ et des sommets critiques que les sous-arbres $T[u_i]$ contiennent.

Corollaire 4.3.16. *Soit T un arbre quelconque enraciné. Pour tout sommet $u \in T$ et pour $u_1, \ldots, u_d$ ses d enfants dans T tels que $k = mh(T[u_1]) \geq mh(T[u_2]) \geq \ldots mh(T[u_d])$:*

1. *Si $d = 0$, alors $mh(T[u]) = 0$;*
2. *Si $k = 0$ et $d > 0$, alors $mh(T[u]) = 1$;*
3. *Si $k = 1$, $d = 1$ et le seul fils de u n'est pas 1-critique, alors $mh(T[u]) = 1$;*
4. *Si $k = 1$, $d = 1$ et le seul fils de u est 1-critique, alors $mh(T[u]) = 2$;*
5. *Si $k = 1$ et $d \geq 2$, alors $mh(T[u]) = 2$;*
6. *Si $k > 1$ et $mh(T[u_3]) = k$, alors $mh(T[u]) = k + 1$;*

7. *Si $k > 1$, $mh(T[u_2]) = k$ ($mh(T[u_3]) < k$ ou $d = 2$), et $T[u_1]$ ou $T[u_2]$ contient un sommet k-critique, alors $mh(T[u]) = k + 1$;*
8. *Si $k > 1$, $mh(T[u_2]) = k$ ($mh(T[u_3]) < k$ ou $d = 2$), et ni $T[u_1]$ ni $T[u_2]$ ne contient de sommet k-critique, alors $mh(T[u]) = k$;*
9. *Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[u_2]) < k$ ou $d = 1$), $T[u_1]$ contient un sommet k-critique et $mh(T[u, x]) = k$, alors $mh(T[u]) = k + 1$;*
10. *Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[u_2]) < k$ ou $d = 1$) et $T[u_1]$ contient un sommet k-critique et $mh(T[u, x]) < k$, alors $mh(T[u]) = k$;*
11. *Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[u_2]) < k$ ou $d = 1$) et $T[u_1]$ ne contient aucun sommet k-critique, alors $mh(T[u]) = k$.*

Démonstration. Chacun de ces différents cas peut être prouvé grâce au Lemme 4.3.13

1. Si $d = 0$, alors $T[u]$ ne contient qu'un sommet et donc par le Lemme 4.3.13, $mh(T[u]) = 0$.
2. Si $k = 0$ et $d > 0$, alors il existe un fils w de u tel que $mh(T[w]) = 0$ et il n'y a aucun enfant w' de u tel que $mh(T[w']) > 0$. De coup, $T[u]$ contient au moins une arête et consiste donc en une étoile centrée en u. Par le Lemme 4.3.13, nous obtenons que $mh(T[u]) = 1$.
3. Si $k = 1$, $d = 1$ et le seul fils de u n'est pas 1-critique, c'est-à-dire, u_1 est le seul enfant de u et $mh(T[u_1]) = 1$. Donc, $T[u_1]$ contient au moins une arête et consiste donc en une étoile centrée en u_1 (car u_1 n'est pas 1-critique). Par conséquent, $T[u]$ est aussi une étoile centrée en u_1 et donc, par le Lemme 4.3.13, $mh(T[u]) = 1$.
4. Si $k = 1$, $d = 1$ et le seul enfant de u est 1-critique, c'est-à-dire, u_1 est le seul enfant de u, $mh(T[u_1]) = 1$ et il existe un enfant w de u_1 tel que $mh(T[w]) = 1$. Par le Lemme 4.3.13 et par définition d'une étoile, $T[w]$ contient au moins 1 arête. Notons donc par v un enfant de w dans $T[w]$. Par conséquent, $T[u]$ contient le chemin (v, w, u_1, u) et donc n'est pas une étoile. De plus, $T[u] \setminus u$ est une forêt d'étoile, et donc, par le Lemme 4.3.13, $mh(T[u]) = 2$.
5. Si $k = 1$ et $d \geq 2$, alors u a au moins deux enfants et $mh(T[u_1] \geq 1$. Par conséquent, par le Lemme 4.3.13 et par définition d'une étoile, $T[u_1]$ contient au moins une arête. Notons donc par w un enfant de u_1 in $T[u_1]$. Notons que $T[u]$ contient le chemin $P = (u_2, u, u_1, w)$ comme sous-graphe et que $T[u] \setminus u$ est une forêt d'étoile. Par conséquent, par le Lemme 4.3.13, $mh(T[u]) = 2$.
6. Si $k > 1$ et $mh(T[v_3]) = k$, alors $mh(T[u]) \geq k + 1$ par le Lemme 4.3.13. De plus, par le Lemme 4.3.15, nous obtenons que $mh(T[u]) = k + 1$.
7. Si $k > 1$, $mh(T[u_1]) = mh(T[u_2]) = k$ ($mh(T[v_3]) < k$ ou $d = 2$) et $T[u_1]$ ou $T[u_2]$ contient un sommet k-critique, notons par x le sommet k-critique dans $T[u] \setminus \{u\}$. Sans perte de généralité, nous pouvons supposer que $x \in T[u_1]$. Notons par y le parent de x dans $T[u]$ (il est possible que $y = u$). Notons aussi par B_1 et B_2 les deux branches en x dans $T[x]$ telle que $mh(B_1) = mh(B_2) = k$. La dernière branche requise est B_y la branche en x dans $T[u]$ contenant y. Remarquons que B_y contient $T[u_2]$ comme sous-graphe. Nous avons donc, par le Lemme 4.2.5, que $mh(B_y) \geq mh(T[u_2]) = k$. Par conséquent, puisqu'il existe trois branches de x nécessitant un nombre de chasseurs monotone au moins k et par le Lemme 4.3.13, nous obtenons que $mh(T[u]) \geq k + 1$. Finalement, par le Lemme 4.3.15 et Lemme 4.3.13, $mh(T[u]) \leq k + 1$, et donc $mh(T[u]) = k + 1$.

8. Si $k > 1$, $mh(T[u_1]) = mh(T[u_2]) = k$ ($mh(T[v_3]) < k$ ou $d = 2$) et ni $T[u_1]$ ni $T[u_2]$ ne contiennent de sommet k-critique, alors il n'y a pas trois branches en u nécessitant un nombre de chasseurs monotone au moins k. Notons aussi que pour tout $w \in T[u_1]$ (resp. $w \in T[u_2]$), w n'est pas k-critique et donc, il existe au plus une branche en w dans $T[w]$ nécessitant un nombre de chasseurs monotone au moins k. Par conséquent, puisque toute branche en w dans $T[u]$ est aussi une branche en w dans $T[u_1]$, sauf si c'est la branche contenant le parent de w, nous obtenons donc qu'il existe au plus 2 branches en w dans $T[u]$ nécessitant un nombre de chasseurs monotone au moins k. Finalement, si $d > 2$, pour tout $2 < j \leq d$ et pour tout sommet $w \in T[u_j]$, notons par y le parent de w dans $T[u_j] \cup \{u\}$. Notons que pour tout sommet $z \in N(w) \setminus \{y\}$, $T[z]$ est un sous-graphe de $T[u_j]$ et donc, par le Lemme 4.2.5 $mh(T[z]) \leq mh(T[u_j]) < k$. Par conséquent, il n'y a pas de sommet $w \in T[u]$ tel qu'il existe 3 branches en v nécessitant un nombre de chasseurs monotone au moins k. Par le Lemme 4.3.13, cela implique que $mh(T[u]) \leq k$. Par le Lemme 4.2.5 et parce que $mh(T[u_1]) = k$, nous obtenons que $mh(T[u]) = k$.
9. Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[u_2]) < k$ ou $d = 1$), $T[u_1]$ contient un sommet k-critique et $mh(T[u, x]) = k$, alors, il y a 3 branches en x nécessitant un nombre de chasseurs monotone au moins k dans $T[u]$. Par conséquent, par le Lemme 4.3.13, $mh(T[u]) \geq k + 1$. Finalement, par le Lemme 4.3.15, $mh(T[u]) \leq k + 1$ and donc $mh(T[u]) = k + 1$.
10. Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[u_2]) < k$ ou $d = 1$), $T[u_1]$ contient un sommet x k-critique et $mh(T[u, x]) < k$, alors, il n'y a pas trois branches en u nécessitant un nombre de chasseurs monotone au moins k. Posons w n'importe quel sommet de $T[u] \setminus \{u\}$. Supposons en premier, que w n'est pas k-critique. Il y a donc, au plus une branche en w dans $T[w]$ qui nécessite un nombre de chasseurs monotone au moins k. Par conséquent, et puisqu'il n'existe qu'une autre branche en w dans $T[u]$ (celle contenant son parent), il y a au plus deux branches en w dans $T[u]$ nécessitant un nombre de chasseurs monotone au moins k.

    Supposons à présent que w est k-critique. Par conséquent, $w \in T[u_1]$ puisque $mh(T[u_j]) < k$ pour tout $2 \leq j \leq d$. En fait, $w = x$. Effectivement, si nous supposons que $w \neq x$, c'est-à-dire, il existe deux sommets k-critique dans $T[u_1]$, alors la branche en w dans $T[u_1]$ contenant son parent contient $T[x]$ comme sous-graphe et/ou la branche en x dans $T[u_1]$ contenant son parent contient $T[w]$. Dans les tous les cas, par le Lemme 4.2.5 il existe au moins un sommet ayant trois branches nécessitant un nombre de chasseurs monotone au moins k. Par conséquent, par le Lemme 4.3.13, $mh(T[u_1]) = k + 1$, contredisant la définition de k. En résumé, pour tout sommet $w \in T[u] \setminus x$, il existe au plus deux branches en w nécessitant un nombre de chasseurs monotone au moins k. Rappelons que par hypothèse, $mh(T[u, x]) < k$. Cela implique qu'il existe au plus 2 branches en x nécessitant un nombre de chasseurs monotone au moins k (sinon, par le Lemme 4.3.13 et le Lemme 4.2.5, $k < mh(T[x]) \leq mh(T[u_1])$, une contradiction). Par conséquent, il n'y a aucun sommet dans $T[u]$ ayant 3 branches nécessitant un nombre de chasseurs monotone au moins k, et donc, par le Lemme 4.3.13, $mh(T[u]) \leq k$ et par le Lemme 4.2.5, $mh(T[u]) \geq mh(T[u_1]) \geq k$.
11. Si $k > 1$, $mh(T[u_1]) = k$ ($mh(T[v_2]) < k$ ou $d = 1$) et $T[u_1]$ ne contient pas de sommet k-critique, alors il n'existe pas de sommet w dans $T[u]$ ayant 3 branches nécessitant un nombre de chasseurs monotone au moins k (comme dans le cas précédent). Par conséquent,

par le Lemme 4.3.13, $mh(T[u]) \leq k$ et par le Lemme 4.2.5, $mh(T[u]) \geq mh(T[u_1]) \geq k$, et donc, $mh(T[u]) = k$.

□

Nous avons besoin d'une nouvelle définition technique pour pouvoir finalement décrire notre algorithme qui va globalement construire une étiquette pour chaque sommet à partir des étiquettes de ses enfants. Rappelons que, pour tout $u, x_1, \ldots, x_p \in V(T[u])$, l'arbre $T[u, x_1, \ldots, x_p]$ correspond au sous-arbre obtenu à partir de $T[u]$ en supprimant les sommets de $\bigcup_{i \leq p} V(T[x_i])$.

Définition 4.3.1 (Étiquette $\lambda(u, T)$ d'un arbre T enraciné en u.). Pour tout arbre $T[u]$, l'étiquette $\lambda(u, T[u])$ de u est une liste d'entiers $(a_1, \ldots, a_p)$, où $a_1 > a_2 > \cdots > a_p \geq 0$, telle que a_p peut être marqué par une étoile $(*)$ et telle qu'il existe un ensemble de sommets $\{\ell_1, \ldots, \ell_p\}$ tels que :

- $mh(T[u]) = a_1$;
- pour tout $1 \leq i < p$, $mh(T[u, \ell_1, \ldots, \ell_i]) = a_{i+1}$ et ℓ_i est un sommet a_i-critique dans $T[u, \ell_1, \ldots, \ell_{i-1}]$. Nous dirons que ℓ_i *est associé* à a_i ;
- $\ell_p = u$. Si a_p n'est pas marqué par une étoile $(*)$, alors u n'est pas un sommet a_p-critique dans $T[u, \ell_1, \ldots, \ell_{p-1}]$. Si a_p est marqué (par une étoile), alors u est un sommet a_p-critique. Dans les deux cas, $T[u, \ell_1, \ldots, \ell_{p-1}, \ell_p] = T[u, u]$ est l'arbre vide.

Exemples. Dans les exemples suivants, nous commençons avec deux arbres T_1 et T_2 dont leurs racines ont "presque" les mêmes étiquettes et montrons qu'ajouter un sommet adjacent à leurs racines nous permet d'obtenir deux nouveaux arbres dont les racines ont des étiquettes "très différentes". En particulier, cela montre l'important d'une étoile marquant le dernier élément d'une étiquette.

Premièrement, soit T_1 l'arbre enraciné en un sommet u_1 ayant comme étiquette $\lambda(u_1, T_1) = (a_1, a_2, a_3) = (3, 2, 1)$. Un tel arbre T_1 est représenté dans la Figure 4.2. Puisque $a_1 = 3$, nous obtenons par définition que $mh(T_1[u_1]) = 3$ et qu'il existe un sommet $\ell_1^1 \in T_1[u_1]$ qui est 3-critique. De plus, $mh(T_1[u_1, \ell_1^1]) = a_2 = 2$ et il existe un sommet $\ell_2^1 \in T_1[u_1, \ell_1^1]$ qui est 2-critique. Finalement, $mh(T_1[u_1, \ell_1^1, \ell_2^1]) = a_3 = 1$ et puisque a_3 n'est par marqué d'une étoile, il n'existe pas de sommet 1-critique dans $T_1[u_1, \ell_1^1, \ell_2^1]$. Par les remarques précédentes, $T_1[u_1, \ell_1^1, \ell_2^1]$ est une étoile centrée sur u_1 et $\ell_3^1 = u_1$ (de plus, l'étoile contient au moins 2 sommets puisque $mh(T_1[u_1, \ell_1^1, \ell_2^1]) > 0$).

Soit T_1' l'arbre obtenu à partir de T_1 en ajoutant un sommet u (la racine de T_1') adjacent à u_1. Notons que, ℓ_1^1 (resp., ℓ_2^1) est toujours 3-critique (resp., 2-critique) dans $T_1'[u]$ (resp., dans $T_1'[u, \ell_1^1]$). Notons aussi que $T_1'[u, \ell_1^1, \ell_2^1]$ correspond en fait à l'arbre obtenu à partir de $T_1[u_1, \ell_1^1, \ell_2^1]$ en ajoutant le sommet u adjacent à u_1. Ainsi, $T_1'[u, \ell_1^1, \ell_2^1]$ est une étoile (contenant au moins 3 sommets) centrée sur u_1, et donc u est 1-critique dans $T_1'[u, \ell_1^1, \ell_2^1]$. Par conséquent, l'étiquette de u dans T_1' est $(3, 2, 1^*)$.

Deuxièmement, soit T_2 l'arbre enraciné en u_2 ayant une étiquette $\lambda(u_2, T_2) = (a_1, a_2, a_3) = (3, 2, 1^*)$. De façon similaire à l'exemple précédent, il existe ℓ_1^2 et ℓ_2^2 étant respectivement un sommet 3-critique de $T_2[u_1]$ et un sommet 2-critique de $T_2[u_1, \ell_1^2]$. De plus, puisque a_3 est marqué d'une étoile, il existe un sommet 1-critique ℓ_3^2 dans $T_2[u_2, \ell_1^2, \ell_2^2]$. Rappelons que, par définition d'une étiquette, $\ell_3^2 = u_2$, et donc $T_2[u_2, \ell_1^2, \ell_2^2]$ est une étoile contenant 3 sommets centrée sur le seul enfant de u_2.

Soit T_2' l'arbre obtenu à partir de T_2 en ajoutant un sommet u' (la racine de T_2') adjacent à u_2. Notons que $T_2'[u', \ell_1^2, \ell_2^2]$ correspond en fait à l'arbre obtenu à partir de $T_2[u_2, \ell_1^2, \ell_2^2]$ en ajoutant le

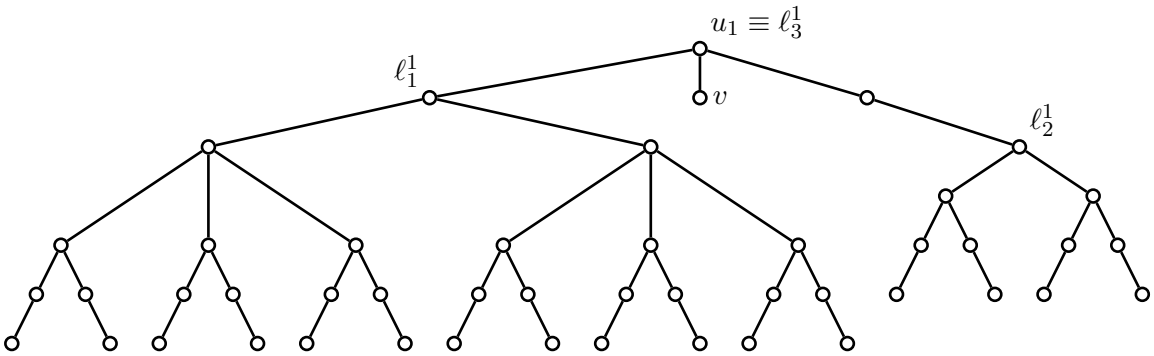


Figure 4.2 – Un exemple d'arbre T_1, décrit dans les exemples de la Définition 4.3.1, tel que $\lambda(u_1, T_1[u_1]) = (3, 2, 1)$. Observons que $mh(T_1[\ell_1^1]) = 3$ et que ℓ_1^1 est un sommet 3-critique puisqu'il existe deux branches de ℓ_1^1 nécessitant un nombre de chasseurs monotone au moins 3. De façon similaire, $mh(T_1[\ell_2^1]) = 2$ et ℓ_3^1 est un sommet 2-critique. Finalement, le graphe $T_1[u_1, \ell_1^1, \ell_1^2]$ est une étoile avec 3 sommets, centrée sur u_1. Ajouter au moins une feuille adjacente à u_1 nous donne un arbre T_1' tel que $\lambda(v, T_1'[v]) = (3, 2, 1)$. Ajouter une feuille u_2 adjacente à v nous donne un arbre T_2 tel que $\lambda(u_2, T_2[u_2]) = (3, 2, 1^*)$. Ajouter une feuille u' adjacente à u_2 nous donne un arbre T_2' tel que $\lambda(u', T_2'[u']) = (4)$.

sommet u' adjacent à u_2. Par conséquent, $T_2'[u', \ell_1^2, \ell_2^2]$ contient un chemin de 4 sommets et donc $mh(T_2'[u', \ell_1^2, \ell_2^2]) > 1$. Cela implique qu'il existe trois branches en ℓ_2^2 dans $T_2'[u', \ell_1^2]$ nécessitant un nombre de chasseurs monotone au moins 2, et donc, $mh(T_2'[u', \ell_1^2]) \geq 3$. À son tour, cela implique qu'il existe trois branches en ℓ_1^2 dans $T_2'[u']$ nécessitant un nombre de chasseurs monotone au moins 3. Ainsi, nous obtenons que $mh(T_2'[u]) \geq 4$ et $\lambda(u', T_2') = (4)$.

Assertion 4.3.16.1 – Il n'existe pas d'arbre T enraciné en $v \in V(T)$tel que v a $(1, 0)$ comme étiquette dans $T[v]$.

Démonstration. Pour arriver à une contradiction, supposons qu'il existe un arbre T enraciné en $v \in V(T)$ tel que v a $lambda = (a_1, a_2) = (1, 0)$ comme étiquette in $T[v]$. Alors, par définition des étiquettes, il existe un sommet ℓ_1 in $T[v]$ tel que ℓ_1 est 1-critique (parce que $a_1 = 1$) et $\ell_1 \neq v$ puisque $\lambda \neq (a_1)$. Par définition d'un sommet 1-critique, nous obtenons que ℓ_1 a un enfant y tel que $mh(T[y]) = 1$. Puisque $mh(T[y]) = 1$, nous obtenons que $T[y]$ contient au moins une arête et donc il existe $x \in N(y)$. Puisque $T[v]$ est connexe, il existe (ℓ_1, v)-chemin $P = (p_1 = \ell_1, \dots p_q = v)$. Ainsi, $P' = (x, y, p_1 = \ell_1, \dots, p_q = v)$ est un chemin contenant au moins 4 sommets. Par conséquent, $mh(T[v]) > 1$ ce qui contredit que $\lambda(v) = (1, 0)$ puisque $a_1 = mh(T[v])$. □

La notation suivante est utilisée dans l'Algorithme 4.1 et dans sa preuve (Lemme 4.3.17). Pour toute séquence $\lambda = (a_1, \dots, a_p)$ et tout entier $a > a_1$, $a \odot \lambda = (a_1' = a, a_2' = a_1, \dots, a_{p'-1}' = a_{p-1}, a_{p'}' = a_p)$, c'est-à-dire, $\odot$ dénote la concaténation de 2 séquences. Nous notons le préfixe de λ par $pref(\lambda) = (a_1, \dots, a_{p-1})$ et le dernier élément de λ par $t(\lambda) = a_p$.

Lemme 4.3.17. *Soit T un arbre quelconque, enraciné en $u \in V(T)$. L'Algorithme 4.1, prenant en entrée T et $v \in V(T)$, et les étiquettes des enfants de v dans T, retourne l'étiquette du sommet v dans $T[v]$.*

Algorithme 4.1 Soit T un arbre enraciné en v. Soient $v_1, \ldots, v_d$ les d enfants de v dans $T[v]$ et soient $\lambda_1, \ldots, \lambda_d$ leurs étiquettes correspondantes.

si $d = 0$ **alors**
  retourner (0) ;
fin si
Soit $\lambda = ()$ et soit $k = \max_{i \in \{1,\ldots,d\}} \max_{q \in \{1,\ldots,|\lambda_i|\}} \lambda_i$;
pour m de 0 à k **faire**
  Soit $n(m)$ le nombre d'enfants contenant m ou m^* dans leurs étiquettes ;
  si $m = 0$ **alors**
    si $n(m) \geq 1$ **alors**
      $\lambda \leftarrow (1)$;
    sinon
      $\lambda \leftarrow (0)$;
    fin si
  sinon si $m = 1$ **alors**
//Case 3 : un enfant de v a 1 dans son étiquette, mais ne contient pas de sommet 1-critique et il n'existe pas d'enfant v contenant 0 dans son étiquette.
    si $n(m) = 1, \forall 1 \leq i \leq d, m \notin pref(\lambda_i), t(\lambda_i) \neq m^*$ et $\lambda = (0)$ **alors**
      $\lambda \leftarrow (1^*)$;
//Case 4 : v a un enfant unique, et cet enfant est un sommet 1-critique, ou
//Case 5 : v a au moins 2 enfants dont au moins un contenant 1 dans son étiquette
    sinon si $n(m) \geq 1$ **alors**
      $\lambda \leftarrow (2)$;
    fin si
  sinon si $m > 1$ **alors**
// Invariant/intuition : à chaque étape, $\lambda = \lambda(v, T_{m-1})$ où T_{m-1} est le sous-arbre obtenu à partir de T en retirant les sous-arbres $T[W]$ tel que $w \neq v$ et $mh(T[w]) \geq m$.
//Case 6 : au moins trois enfants de v ont m ou m^* dans leurs étiquettes
    si $n(m) \geq 3$ **alors**
      $\lambda \leftarrow (m+1)$;
//Case 7 : deux enfants de v ont m ou m^* dans leurs étiquettes et au moins un de ces sous-arbres contient un sommet m-critique
    sinon si $n(m) = 2$ et $\exists 1 \leq i \leq d, m \in pref(\lambda_i)$ ou $t(\lambda_i) = m^*$ **alors**
      $\lambda \leftarrow (m+1)$;
//Case 8 : deux enfants de v ont m ou m^* dans leurs étiquettes. De plus, leurs sous-arbres ne contiennent pas de sommets m-critique
    sinon si $n(m) = 2$ et, $\forall 1 \leq i \leq d, m \notin pref(\lambda_i)$ et $t(\lambda_i) \neq m^*$ **alors**
      $\lambda \leftarrow (m^*)$;
//Case 9 : un enfant de v a m ou m^* dans son étiquette. De plus, son sous-arbre contient un sommet m-critique et $mh(T_{m-1}) = m$
    sinon si $n(m) = 1, \exists 1 \leq i \leq d, m \in pref(\lambda_i)$ ou $t(\lambda_i) = m^*$, et λ contient m ou m^* **alors**
      $\lambda \leftarrow (m+1)$;
//Case 10 : un enfant de v a m ou m^* dans son étiquette. De plus, son sous-arbre contient un sommet m-critique et $mh(T_{m-1}) < m$
    sinon si $n(m) = 1, \exists 1 \leq i \leq d, m \in pref(\lambda_i)$ ou $t(\lambda_i) = m^*$, et λ ne contient ni m ni m^* **alors**
      $\lambda \leftarrow (m) \odot \lambda$;
//Case 11 : un enfant de v a m ou m^* dans son étiquette. De plus, son sous-arbre ne contient aucun sommet m-critique
    sinon si $n(m) = 1$ et, $\forall 1 \leq i \leq d, m \notin pref(\lambda_i)$ et $t(\lambda_i) \neq m^*$ **alors**
      $\lambda \leftarrow (m)$;
    fin si
  fin si
fin pour
retourner λ ;

Démonstration. Notre but est de prouver que l'algorithme retourne l'étiquette de l'arbre $T[v]$ enraciné en v, noté par $\lambda(v, T[v]) = (a_1^v, \ldots, a_{p^v}^v)$. Notons aussi par $d_{T[v]}(v) = d$ le nombre d'enfants de v. Notons par λ^{alg} l'étiquette calculée par l'Algorithme 4.1.

Observons en premier que, si $d = 0$, alors $mh(T[v]) = 0$ par le Corollaire 4.3.16, $mh(T[v]) = 0$, et donc, par définition d'une étiquette, $\lambda(v, T[v]) = (0)$. Notons que dans ce cas, l'Algorithme 4.1 retourne (0) à la ligne 2. Par conséquent, $\lambda^{alg} = \lambda(v, T[v]) = (0)$.

Nous pouvons donc supposer que $d > 0$, à partir de maintenant. Notons par $v_1, \ldots v_d$ les d enfants de v dans $T[v]$. Pour tout $1 \leq i \leq d$, notons par $\lambda_i = (a_1^i, \ldots, a_{p_i}^i)$ l'étiquette de v_i dans $T[v_i]$, c'est-à-dire, $\lambda_i = \lambda(v_i, T[v_i]) = (a_1^i, \ldots, a_{p_i}^i)$. Sans perte de généralité, supposons que $a_1^1 \geq a_1^2 \geq \cdots \geq a_1^d$, c'est-à-dire, les enfants de v sont triés par rapport à la plus grande valeur dans leur étiquette. Par définition, notons que $k = a_1^1 = \max_{i \in \{1,\ldots,d\}} \max_{q \in \{1,\ldots,|\lambda_i|\}} a_q^i$.

Pour chaque $1 \leq i \leq d$, notons $T_k^i = T[v_i]$. De plus, pour $0 \leq m \leq k$, si $m = a_j^i$ pour n'importe quel $1 \leq j \leq p_i$, notons par T_{m-1}^i l'arbre obtenu à partir de T_m^i en retirant $T[\ell_j^i]$ où ℓ_j^i est le sommet associé à a_j^i. Sinon, notons T_m^i aussi par $T^i m - 1$, c'est-à-dire, $T_m^i = T_{m-1}^i$. Finalement, pour tout $0 \leq m \leq k$, nous noterons par T_m le sous-arbre de $T[u]$ induit par $\bigcup_{1 \leq i \leq d} V(T_m^i)$. Intuitivement, T_m est le sous-arbre obtenu à partir de $T[v]$ en supprimant les sous-arbres $T[w]$ pour tout sommet $w \neq v$ tel que $mh(T[w]) \geq m + 1$. Notons que $T_k = T[v] = T$ et que $mh(T_m) \leq m + 1$ pour tout $m \leq k$. Notons aussi que T_0 consiste seulement en v et, peut être, quelques-uns de ses enfants.

Nous allons prouver par induction sur $0 \leq m \leq k$ qu'après la $(m+1)^e$ itération de la boucle "for" de l'algorithme, la valeur courante de λ, la variable de l'Algorithme 4.1, dénotée par λ^m, est égale à l'étiquette $\lambda(v, T_m)$, c'est-à-dire, l'étiquette de v dans T_m. Si l'induction tient, alors, quand $m = k$, l'Algorithme retourne $\lambda^{alg} = \lambda^k = \lambda(v, T_k) = \lambda(v, T[v])$, ce qui conclut la preuve.

Soit $n(m)$ le nombre d'enfants w de v, tel que m ou m^* est contenu dans leurs étiquettes.

Le cas de base est pour $m = 0$.

— Supposons premièrement que $n(0) \geq 1$. Rappelons que, par définition de T_0, pour tout enfant w de v dans T_0, $mh(T_0[w]) \leq 0$, c'est-à-dire, $T_0[w]$ contient seulement w. Par conséquent, v n'est pas 1-critique dans T_0. Ainsi, par le cas 2 du Corollaire 4.3.16 et par définition d'une étiquette, $mh(T_0) = 1$ et $\lambda(v, T_0) = (1)$, ce qui correspond à λ^0 (ligne 9 de l'Algorithme 4.1).

— Supposons à présente que $n(0) = 0$, c'est-à-dire, T_0 contient seulement v. Par le Cas 1 du Corollaire 4.3.16, $mh(T_0) = 0$. Ainsi, $\lambda(v, T_0) = (0)$. De plus, puisque, $n(m) = 0$, $\lambda^0 = (0)$ (ligne 11 de l'Algorithme 4.1).

Supposons à présent que $m = 1$. Il suit du cas $m = 0$ que $\lambda^0 = \lambda(v, T_0)$. Avant d'analyser les différents sous-cas pour $m = 1$, rappelons que $\lambda(v, T_m)$ ne peut pas avoir $(1, 0)$ comme étiquette par l'Assertion 4.3.16.1.

— Supposons en premier que $n(1) = 0$. Cela implique que $T_1 = T_0$, et donc, nous obtenons que $\lambda^1 = \lambda^0 = \lambda(v, T_0) = \lambda(v, T_1)$ (dans ce cas, l'Algorithme 4.1 ne fait rien pendant la 2^e itération de la boucle).

— Supposons ensuite que $n(1) = 1$, et notons par v_i l'enfant de v tel que (1) ou (1^*) est dans l'étiquette de v_i.

  — Supposons en premier que nous sommes dans le Cas 3 (ligne 16 de l'Algorithme 4.1), c'est-à-dire, 1 (pas 1^*) est le dernier élément de l'étiquette de v_i, et $\lambda^0 = (0)$. Rappelons que puisque $\lambda^0 = (0)$, et par le cas $m = 0$, T_0 contient seulement un sommet. Ainsi, v n'a qu'un enfant w dans T_1, et $mh(T_1[w]) = 1$ (parce que $n(1) = 1$),

c'est-à-dire, T_1 est une étoile centrée sur w (mais enraciné en v). Donc, par cas 3 du Corollaire 4.3.16, $mh(T_1) = 1$. De plus, v est 1-critique dans T_1. Par conséquent, $\lambda(v, T_1) = (1^*)$, ce qui correspond à λ^1 (ligne 16 de l'Algorithme 4.1).

— Supposons ensuite que nous sommes dans le Cas 5 (ligne 20 de l'Algorithme 4.1), c'est-à-dire, 1 (pas 1^*) est le dernier élément de l'étiquette de v_i, et $\lambda^0 = (1)$. Observons que si $\lambda^0 = (1)$, et par le cas $m = 0$, alors T_0 est une étoile centrée sur (et enraciné en) v. Ainsi, v a au moins 2 enfants dans T_1 (puisque $v_i \in V(T_1) \setminus V(T_0)$) mais il existe un enfant v_i de $v$qui n'est pas une feuille (c'est-à-dire, v_i est la racine du sous-arbre dont le nombre de chasseurs monotone est au moins 1). Donc, par le Cas 5 du Corollaire 4.3.16, $mh(T_1) = 2$. Par le Lemme 4.3.15, pour tout sommet $w \in T_1$, il existe au plus 1 branche en w qui a un nombre de chasseurs monotone au moins 2 (donc il n'y a pas de sommets 2-critiques). Par conséquent, $\lambda(v, T_1) = (2)$, ce qui correspond à λ^1 (ligne 20 de l'Algorithme 4.1).

— Considérons à présent le Cas 4 (ligne 20 de l'Algorithme 4.1), c'est-à-dire, le dernier élément de l'étiquette de v_i est 1^* et v_i l'unique enfant de v. Par le Cas 4 du Corollaire 4.3.16, $mh(T_1) = 2$. Par le Lemme 4.3.15, pour tout sommet $w \in T_1$, il existe au plus 1 branche en w qui a un nombre de chasseurs monotone au moins 2 (donc il n'y a pas de sommets 2-critiques). Ainsi, $\lambda(v, T_1) = (2)$, ce qui correspond à λ^1 (ligne 20 de l'Algorithme 4.1).

— Finalement, supposons que $n(m) = n(1) \geq 2$. Ainsi, dans T_1, v a au moins 2 deux enfants qui sont les racines de sous-arbres qui ont un nombre de chasseurs monotone au moins 1. Par le Cas 5 du Corollaire 4.3.16, $mh(T_1) = 2$ (rappelons que par définition de T_1, $mh(T_1) \leq 2$). De plus, par le Lemme 4.3.15, pur tout sommet $w \in T_1$, il y a au plus 1 branches en w qui a un nombre de chasseurs monotone au moins 2 (donc il n'y a pas de sommet 2-critique). Par conséquent, $\lambda(v, T_1) = (2)$, ce qui correspond à λ^1 (ligne 20 de l'Algorithme 4.1).

Nous sommes maintenant prêts à prouver l'étape d'induction. Soit $m \geq 2$ et supposons que $\lambda(v, T_{m-1}) = \lambda^{m-1}$. Nous allons prouver que $\lambda(v, T_m) = \lambda^m$.

— **Cas 6.** Nous sommes dans le cas où $n(m) \geq 3$ (ligne 26 de l'Algorithme 4.1). Puisque $n(m) \geq 3$, dans T_m, v a au moins 3 enfants v_j, $v_{j'}$ et $v_{j''}$ tel que $mh(T_m^j) = mh(T_m^{j'}) = mh(T_m^{j''}) = m$ (et $mh(T_m^i) \leq m$ pour tout $1 \leq i \leq d$). Ainsi, nous sommes dans le cas 6 du Corollaire 4.3.16, et donc, $mh(T_m) = m + 1$. Notons aussi que, par le Lemme 4.3.15 et pour tout sommet $w \in T_m$, il existe au plus 1 branches B en w avec $mh(B) \geq m + 1$. Par conséquent, T_m n'a pas de sommet $m + 1$-critique, et donc $\lambda(v, T_m) = (m + 1)$. Pour conclure, ligne 26 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m + 1)$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

— **Case 7.** Nous sommes dans le cas où $n(m) = 2$ et où il existe $1 \leq i \leq d$ tel que $m \in pref(\lambda_i)$ ou $t(\lambda_i) = m^*$, c'est-à-dire, T_m^i contient un sommet m-critique (ligne 29 de l'Algorithme 4.1). Notons par y_i le sommet m-critique de T_m^i. Puisque $n(m) = 2$, dans T_m, v a exactement 2 enfants v_j et $v_{j'}$ (et $i \in \{j, j'\}$) tel que $mh(T_m^j) = mh(T_m^{j'}) = m$ et $mh(T_m^q) < m$ pour tout autre enfant v_q de v dans T_m. Puisque y_i est m-critique, nous sommes dans le cas 7 du Corollaire 4.3.16. Ainsi, $mh(T_m) = m + 1$. Notons aussi que, par le Lemme 4.3.15, pour tout sommet $w \in T_m$, il existe au plus 1 branche B en w avec $mh(B) \geq m + 1$. Par conséquent, T_m n'a pas de sommet $m + 1$-critique, et donc $\lambda(v, T_m) = (m + 1)$. Pour conclure, ligne 29 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m + 1)$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

— **Case 8.** Nous sommes dans les cas où $n(m) = 2$, $m \notin pref(\lambda_i)$ et $t(\lambda_i) \neq m^*$ pour tout $1 \leq i \leq d$, c'est-à-dire, T_m^i ne contient aucun sommet m-critique (ligne 32 de l'Algorithme 4.1). Puisque $n(m) = 2$, dans T_m, v a exactement 2 enfants v_j et $v_{j'}$ tel que $mh(T_m^j) = mh(T_m^{j'}) = m$ et $mh(T_m^q) < m$ pour tout autre enfant v_q de v dans T_m. Puisque v_j et $v_{j'}$ ne sont pas m-critique, nous sommes dans le Cas 8 du Corollaire 4.3.16. Ainsi, $mh(T_m) = m$. Puisque $n(m) = 2$, v est clairement un sommet m-critique, et donc $\lambda(v, T_m) = (m^*)$. Pour conclure, ligne 32 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m^*)$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

— **Case 9.** Nous sommes dans le cas où $n(m) = 1$, il existe $1 \leq i \leq d$ tel que $m \in pref(\lambda_i)$, ou $m^* = t(\lambda_i)$ (c'est-à-dire, il existe un sommet m-critique dans T_m^i) et λ_{m-1} contient m ou m^*, c'est-à-dire, $mh(T_{m-1}) = m$ (ligne 35 de l'Algorithme 4.1). Soit y_i le sommet m-critique dans T_m^i. Puisque $n(m) = 1$, dans T_m, v a exactement 1 enfant v_j (et donc, $i = j$) tel que $mh(T_m^j) = m$ et $mh(T_m^q) < m$ pour tout autre enfant v_q de v dans T_m. Par conséquent, $T_{m-1} = T_m[v, y_i]$. Ainsi, par définition des étiquettes et puisque $m \in \lambda^{m-1}$, $mh(T_{m-1}) = mh(T_m[v, y_i]) = m$. Nous sommes donc dans le Cas 9 du Corollaire 4.3.16. Ainsi, $mh(T_m) = m + 1$. Notons aussi que, par le Lemme 4.3.15, pour tout sommet $w \in T_m$, il existe au moins 1 branche B en w avec $mh(B) \geq m + 1$. Par conséquent, T_m n'a pas de sommet $(m+1)$-critique, et donc $\lambda(v, T_m) = (m+1)$. Pour conclure, ligne 35 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m + 1)$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

— **Case 10.** Nous sommes dans le cas où $n(m) = 1$, où il existe $1 \leq i \leq d$ tel que $m \in pref(\lambda_i)$, ou $m^* = t(\lambda_i)$ (c'est-à-dire, il y a un sommet m-critique dans T_m^i), et $m \notin \lambda^{m-1}$ (ligne 38 de l'Algorithme 4.1). Soit y_i le sommet m-critique dans T_m^i. Puisque $n(m) = 1$, dans T_m, v_i est le seul enfant de v tel que $mh(T_m^i) = m$ et $mh(T_m^q) < m$ pour tout autre enfant v_q de v in T_m. Par conséquent, $T_{m-1}[v] = T_m[v, y_i]$. Par définition des étiquettes et puisque $m \notin \lambda^{m-1}$, $mh(T_{m-1}[v]) = mh(T_m[v, y_i]) < m$. Par conséquent, nous sommes dans les Cas 10 du Corollaire 4.3.16. Ainsi, $mh(T_m) = m$. Soit $\lambda^{m-1} = \lambda(v, T_{m-1}) = (a_1, \ldots, a_p)$. Rappelons qu'il existe des sommets $\ell_1, \ell_2, \ldots, \ell_{p-1}$ tels que ℓ_h est a_h-critique dans $T_{m-1}[v, \ell_1, \ldots, \ell_{h-1}]$ pour tout $1 \leq h < p$. Puisque $T_{m-1}[v] = T_m[v, y_i]$, ℓ_h est a_h-critique dans $T_m[v, y_i, \ell_1, \ldots, \ell_{h-1}]$ pour tout $1 \leq h < p$. De plus, pour tout $1 \leq h \leq p$, $T_{m-1}[v, \ell_1, \ldots, \ell_{h-1}] = T_m[v, y_i, \ell_1, \ldots, \ell_{h-1}]$, et donc $mh(T_m[v, y_i, \ell_1, \ldots, \ell_{h-1}]) = mh(T_{m-1}[v, \ell_1, \ldots, \ell_{h-1}]) = a_h$. Par conséquent, $\lambda(v, T_m) = (m, a_1, \ldots, a_p)$. Pour conclure, ligne 38 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m) \odot \lambda^{m-1}$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

— **Case 11.** Finalement, nous sommes dans le cas où $n(m) = 1$, $m \notin pref(\lambda_i)$ et $t(\lambda_i) \neq m^*$ pour tout $1 \leq i \leq d$, c'est-à-dire, T_m^i ne contient aucun sommet m-critique (ligne 41 de l'Algorithme 4.1). Puisque $n(m) = 1$, dans T_m, v a exactement 1 enfant v_i tel que $mh(T_m^i) = m$ et $mh(T_m^q) < m$ pour tout autre enfant v_q de v dans T_m. Puisque qu'il n'y a pas de sommets m-critique dans T_m, nous sommes dans le cas 11 du Corollaire 4.3.16. Ainsi, $mh(T_m) = m$. Notons aussi que v n'est pas m-critique puisque $n(m) = 1$. Par conséquent, T_m n'a pas de sommet m-critique, et donc $\lambda(v, T_m) = (m)$. Pour conclure, ligne 41 de l'Algorithme 4.1 retourne précisément $\lambda^m = (m)$. Ainsi, $\lambda^m = \lambda(v, T_m)$.

□

Par conséquent, le théorème suivant suit :

Théorème 4.3.18. *Le nombre de chasseurs monotone de n'importe quel arbre peut être calculé en temps polynomial.*

4.4 La recontamination aide beaucoup dans les arbres

Jusqu'à présent, nous avons investigué le jeu DES CHASSEURS ET DU LAPIN avec la propriété de monotonie puisque les stratégies monotones sont souvent plus faciles à comprendre. Les précédents travaux sur ce jeu dans les graphes bipartis $G = (V_r \cup V_w, E)$ ont montré qu'il était suffisant d'étudier sa *variante rouge*, c'est-à-dire, quand le lapin est obligé de commencer sur un sommet de V_r. Plus précisément, le Lemme 1.5.4 nous rappelle ce résultat, $h(G) = h_{V_r}(G)$ pour tout graphe biparti $G = (V_r \cup V_w, E)$ qui a permis d'obtenir beaucoup de résultats sur le jeu DES CHASSEURS ET DU LAPIN (Abramovskaya et al., 2016; Bolkema & Groothuis, 2019; Gruslys & Méroueh, 2015). Par conséquent, il est intéressant de considérer la propriété de monotonie dans le cas de la variante rouge. Cette section est dévouée à cette étude dans le cas des arbres. Rappelons que $mh_{V_r}(G)$ dénote le nombre minimum de chasseurs requis pour tuer le lapin commençant sur V_r dans un graphe biparti $G = (V_r \cup V_w, E)$ de façon monotone. Il peut être vérifié que, dans (Gruslys & Méroueh, 2015), les auteurs ont montré que, pour tout arbre T, $mh_{V_r}(T) \leq \lceil \frac{\log_2 |V(T)|}{2} \rceil$ (ils montrent $h_{V_r}(T) \leq \lceil \frac{\log_2 |V(T)|}{2} \rceil$ grâce à une stratégie monotone sur V_r (Gruslys & Méroueh, 2015)). En particulier, la preuve du Corollaire 4.2.10 montre donc que :

Corollaire 4.4.1. *Il existe $\varepsilon > 0$ tel que, pour tout $k \in \mathbb{N}$, il existe un arbre T avec $mh_{V_r}(T) \geq k$ et $mh(T) \geq (1+\varepsilon) mh_{V_r}(T)$.*

Par conséquent, la Proposition 4.3.10 et le Corollaire 4.4.1 nous montre déjà qu'il existe des graphes G pour lesquels $mh_{V_r}(G) < mh(G)$. Le résultat principal de cette section est qu'il existe une famille infinie d'arbres T tels que la différence entre $mh_{V_r}(T)$ et $h_{V_r}(T)$ est arbitrairement grande. En particulier, cela améliore les résultats du Corollaire 4.2.10 puisque $mh_{V_r}(G) \leq mh(G)$ et $h_{V_r}(G) = h(G)$ pour tout graphe G.

Plus précisément, cette section est dévouée à prouver que :

Théorème 4.4.2. *Pour tout $i \geq 3$, il existe un arbre T tel que $mh_{V_r}(T) \geq i$ et $h_{V_r}(T) = h(T) = 2$.*

Démonstration. Dans la Section 4.4.1, nous définissons une famille $(T_{i,2i})_{i \geq 3}$ d'arbres tels que $h_{V_r}(T_{i,2i}) = 2$ pour tout $i \geq 3$ (Lemme 4.4.10). Puis, dans la Section 4.4.2, le Lemme 4.4.15 prouve que $mh_{V_r}(T_{i,2i}) \geq i$ pour tout $i \geq 3$. □

Pour prouver les Lemmes 4.4.10 et 4.4.15 plus bas, nous devons d'abord adapter les Lemmes et les Propositions techniques de la section 4.2.1 dans le cas de la variante rouge des graphes bipartis. Dans tous les cas, ces Lemmes et ces Propositions sont des adaptations directes des Lemmes et des Propositions de la section 4.2.1. De ce fait, leurs preuves ne sont pas données, mais peuvent être trouvé dans l'annexe de notre rapport (Dissaux et al., 2023).

La Proposition suivante est une adaptation de la Proposition 4.2.2 pour la variante rouge du jeu.

Proposition 4.4.3. *Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse dans un graphe biparti quelconque $G = (V_r \cup V_w, E)$ par rapport à V_r. Soient $v \in V_r$ (resp. $v \in V_w$) et $1 \leq i \leq \ell$. S'il existe un sommet $u \in N(v)$ et un sommet $x \in N(u)$ (possiblement $x = v$) tel que $u \notin \bigcup_{j \leq i} S_j$ et $x \notin \bigcup_{j<i} S_j$, alors $v \in Z_{2p}$ pour tout $2p \leq i$ (resp. $v \in Z_{2p+1}$ pour tout $2p+1 \leq i$).*

Le prochain lemme adapte le Lemme 4.2.3 dans la variante rouge du jeu.

Lemme 4.4.4. *Soit $G = (V_r \cup V_w, E)$ un graphe biparti contenant au moins 2 sommets. Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse monotone dans G par rapport à V_r. Pour n'importe quel indice $0 \leq p < i \leq \lceil \ell/2 \rceil$, $Z_{2i} \subseteq Z_{2p}$ et $Z_{2i+1} \subseteq Z_{2p+1}$.*

Le Lemme 4.4.5 est une adaptation directe, dans la variante rouge du jeu, du Lemme 4.2.4. La seule différence dans leurs preuves est l'utilisation de la Proposition 4.4.3 au lieu de la Proposition 4.2.2.

Lemme 4.4.5. *Soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse non-monotone dans un graphe biparti $G = (V_r \cup V_w, E)$ par rapport à V_r. Alors, il existe un sommet $v \in V$ et $1 \leq i \leq \ell$ tels que $v \in Z_{i-1} \setminus S_i$ et $v \in \bigcup_{p<i} S_p$.*

Le Lemme 4.4.6 est une adaptation directe, dans la variante rouge, du Lemme 4.2.5. La seule différence dans leurs preuves est l'utilisation de la Proposition 4.4.3 et du Lemme 4.4.5 au lieu de la Proposition 4.2.2 et du Lemme 4.2.4.

Lemme 4.4.6. *Pour tout sous-graphe connexe non-vide H d'un graphe biparti $G = (V_r \cup V_w, E)$, $mh_{V_r \cap V(H)}(H) \leq mh_{V_r}(G)$. De plus, si $|V(H)| > 1$, nous obtenons que, s'il existe une stratégie de chasse monotone $\mathcal{S} = (S_1, \ldots, S_\ell)$ dans G par rapport à V_r, alors il existe une stratégie de chasse monotone $\mathcal{S}'$ dans H par rapport à $V_r \cap V(H)$ utilisant au plus $\max_{1 \leq i \leq \ell} |S_i \cap V(H)|$ chasseurs.*

Le prochain lemme adapte le Lemme 4.2.6 dans la variante rouge du jeu.

Lemme 4.4.7. *Pour tout graphe biparti $G = (V_r \cup V_w, E)$ et pour tout $k \geq mh_{V_r}(G)$, il existe une stratégie de chasse monotone parcimonieuse dans G par rapport à V_r utilisant k chasseurs.*

Le prochain lemme adapte le Lemme 4.2.7 dans la variante rouge du jeu.

Lemme 4.4.8. *Soit $G = (V_r \cup V_w, E)$ un graphe biparti et soit $\mathcal{S} = (S_1, \ldots, S_\ell)$ une stratégie de chasse monotone parcimonieuse dans G par rapport à V_r.*

- *S'il existe $1 \leq i < j \leq \ell$ tel que $v \in S_i \cap S_j$, alors $v \in S_{i+2}$.*
- *Si $v \in V_r$ (resp., $v \in V_w$) et qu'il existe un tour impair (resp., pair) $1 \leq i < \ell$ tel que $v \notin Z_{i-1}$, alors $v \notin S_j$ pour tout $j \geq i$.*

4.4.1 La famille des arbres $(T_{i,q})_{i \geq 3, q \geq 6}$: définition et nombre de chasseurs

Dans cette section, nous prouvons qu'il existe une différence arbitrairement grande entre le nombre de chasseurs et le nombre de chasseurs monotone dans la variante rouge du jeu. Plus précisément, nous concevons une famille infinie d'arbres $(T_i)_{i \geq 3}$ ayant une différence arbitraire entre ces deux paramètres.

Définissons $S_{k,q}$ comme l'arbre enraciné obtenu à partir de $q \geq 6$ chemins de taille $k \geq 3$ (avec k arêtes) en identifiant une extrémité de chaque chemin en un seul et même sommet appelé la *racine*, notée par c, de $S_{k,q}$. De façon équivalente, $S_{k,q}$ peut être obtenu à partir d'une étoile enracinée en c de degré q en sous-divisant chaque arête $k-1$ fois. À partir de maintenant, notons par (V_r, V_w) la bipartition de $V(S_{k,q})$ et supposons que $c \in V_r$.

Lemme 4.4.9. *Pour tout $k, q \in \mathbb{N}$ tel que $k \geq 3$ et $q \geq 6$, $h(S_{k,q}) = mh_{V_r}(S_{k,q}) = 2$.*

Démonstration. Le fait que $h(S_{k,q}) > 1$ vient de la caractérisation des arbres ayant un nombre de chasseurs 1 dans (Britnell & Wildon, 2013). Rappelons que nous supposons, sans perte de généralité, que le centre c de $S_{k,q}$ est dans V_r. Nous allons prouver que $mh_{V_r}(S_{k,q}) \leq 2$ et donc $h(S_{k,q}) = mh_{V_r}(S_{k,q}) = 2$ par le Lemme 1.5.4 et la Proposition 4.2.1.

Définissons la stratégie $\mathcal{S}$ par rapport à V_r utilisant 2 chasseurs comme suit. À chaque tour impair, le premier chasseur tire sur c. Le second chasseur tire séquentiellement sur chaque chemin $P = (v_1, \ldots, v_k)$ de $S_{k,q} \setminus c$ en tirant itérativement sur $v_1, v_2, \ldots, v_k$ (en commençant par v_1 pendant un tour pair).

Formellement, notons par $P^1, \ldots, P^q$ les q branches en c dans $S_{k,q}$, et notons $P^i = (v_1^i, \ldots, v_k^i)$ les sommets du chemin P^i pour tout $1 \leq i \leq q$ (où v_1^i est le voisin de c se trouvant dans P^i). Dans le cas où k est pair, la stratégie $\mathcal{S}$ correspond à $(\{c\}, \{v_1^1\}, \{c, v_2^1\}, \{v_3^1\}, \{c, v_4^1\}, \ldots, \{v_{k-1}^1\}, \{c, v_k^1\}, \{v_1^2\}, \{c, v_2^2\}, \ldots, \{v_{j-1}^i\}, \{c, v_j^i\}, \ldots, \{c, v_k^q\})$. Dans le cas où k est impair, la stratégie $\mathcal{S}$ correspond à $(\{c\}, \{v_1^1\}, \{c, v_2^1\}, \{v_3^1\}, \{c, v_4^1\}, \ldots, \{c, v_{k-1}^1\}, \{v_k^1\}, \{c\}, \{v_1^2\}, \{c, v_2^2\}, \ldots, \{v_k^{i-1}\}, \{c\}, \{v_1^i\}, \{c, v_2^i\}, \ldots, \{v_{j-1}^i\}, \{c, v_j^i\}, \ldots, \{v_k^q\})$.

Clairement, cette stratégie est monotone et gagnante dans $S_{k,q}$ par rapport à V_r dans les deux cas (Quelle que soit la parité de k). □

Notons par $\mathcal{S}_1$ la stratégie décrite dans la précédente preuve et notons par ℓ_1 le plus petit entier pair plus grand ou égale à la longueur de $\mathcal{S}_1$ (la longueur égale $1 + qk$ si k est pair et $q(k+1)$ si k est impair).

La construction des arbres $T_{i,q}$. Pour tout $i \geq 2$ et tout $q \geq 6$, définissons l'arbre $T_{i,q}$ comme suit. Premièrement, $T_{1,q} = S_{3,q}$. Ensuite, pour tout $i > 1$, supposons que $T_{i-1,q}$ a été construit récursivement et qu'il existe une stratégie de chasse gagnante de longueur ℓ_{i-1} utilisant 2 chasseurs dans $T_{i-1,q}$ par rapport à V_r (c'est le cas pour $i_1 = 1$ par le Lemme 4.4.9 et ce sera prouver pour $i-1 > 1$ dans le lemme suivant). Définissons $T_{i,q}$ comme l'union disjointe de q copies $T_i^1, \ldots, T_i^q$ de $T_{i-1,q}$ et d'un sommet c_i étant la racine de $T_{i,q}$. Finalement, pour tout $1 \leq j \leq q$, nous ajoutons un chemin P_i^j de longueur p_i^j (défini ci-dessous) entre la racine c_i^j de T_i^j et c_i (et donc, c_i et c_i^j sont à distance p_j^i dans $T_{i,q}$).

Les longueurs p_i^j sont définies récursivement comme suit. Définissons en premier $p_i^1 = 2$ et ensuite, pour tout $1 < j \leq q$, définissons p_i^j comme le plus petit entier pair plus grand ou égal à la longueur $\ell_{i-1} + \sum_{1\leq k<j} p_i^k$ (nous montrerons dans le prochain lemme que ℓ_i est égal au plus petit entier pair plus grand ou égal à $q\ell_{i-1} + \sum_{1\leq j\leq q} jp_i^j$).

Finalement, supposons que $c_i \in V_r$ et puisque p_i^j est pair, remarquons que $c_i, c_i^1, \ldots, c_i^q$ sont tous de V_r.

Lemme 4.4.10. *Pour tout $i \in \mathbb{N}^*$ et $q \geq 6$, $h_{V_r}(T_{i,q}) = 2$.*

Démonstration. Puisque $T_{i,q}$ contient $S_{3,q}$ commet sous-graphe et par les Lemmes 4.4.9 et 4.1.1, nous obtenons que $h_{V_r}(T_{i,q}) \geq 2$.

Il ne reste qu'à prouver que $h_{V_r}(T_{i,q}) \leq 2$ par induction sur i. Plus précisément, nous prouvons qu'il existe une stratégie de chasse $\mathcal{S}_i = (S_1, \ldots, S_{\ell_i})$ gagnante dans $T_{i,q}$ par rapport à V_r utilisant 2 chasseurs et telle que, pour tout $j \geq 1$, si la racine c_i de $T_{i,q}$ est dans Z_j, alors $c_i \in S_{j+1}$ (que nous appellerons la propriété $(*)$). Remarquons que la stratégie $\mathcal{S}_1$ définit dans le Lemme 4.4.9 satisfait la propriété $(*)$. Supposons donc par induction que pour $i > 1$ une telle stratégie $\mathcal{S}_{i-1}$ a bien été définie pour $T_{i-1,q}$.

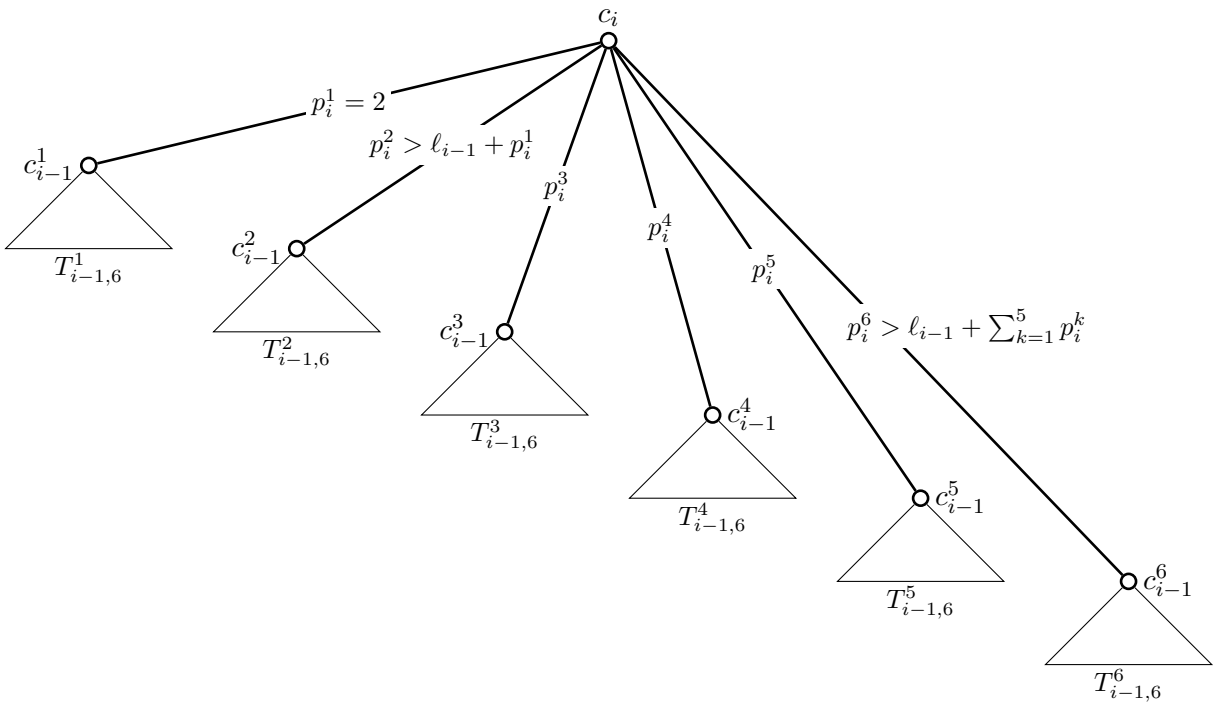


Figure 4.3 – Le graphe $T_{i,6}$. Les étiquettes sur les arêtes sont utilisées pour représenter leurs longueurs respectives. En particulier, pour tout $2 \leq j \leq 6$, nous avons $p_i^j > \sum_{1 \leq k \leq j-1} p_i^k + \ell_{i-1}$, où $p_i^1 = 2$ et ℓ_{i-1} est égale au nombre de tours pour nettoyer n'importe quel copie du graphe $T_{i-1,6}$.

Rappelons que, pour tout $1 \leq j \leq q$, c_i^j dénote la racine de la copie T_i^j de $T_{i-1,q}$ reliée à c_i dans T_i par le chemin P_i^j de longueur au moins $\ell_{i-1} + \sum_{1 \leq k < j} p_i^k$. De plus, rappelons que $c_i \in V_r$.

Définissons maintenant la stratégie $\mathcal{S}_i$ comme suit. Elle procède en q phases et certifie que, pour n'importe quel tour h, si $c_i \in Z_h$, alors $c_i \in S_{h+1}$ (propriété $(*)$) et que, pour n'importe quel tour h arrivant après la j^{th} phase, $(Z_h \setminus c_i) \cap \bigcup_{k \leq j} (V(T_i^k) \cup V(P_i^k)) = \emptyset$. Cela implique, qu'au tour ℓ_i, si lapin est vivant, il doit se trouver sur c_i. Mais durant la dernière phase, nous montrerons aussi que le lapin ne peut pas être sur c_i, prouvant que $\mathcal{S}_i$ est une stratégie de chasse gagnante (avec la propriété $(*)$).

Pour tout $1 \leq j \leq q$, notons par i_0^j le dernier tour de la phase j ($i_0^0 = 0$) et supposons par induction sur j que i_0^j est pair. De plus supposons par induction sur j que $(Z_{i_0^{j-1}} \setminus c_i) \cap \bigcup_{k \leq j-1} (V(T_i^k) \cup V(P_i^k)) = \emptyset$. Décrivons à présent la phase j^{th} qui procédera en 2 sous-phases.

De façon très informelle, dans la première sous-phase, les chasseurs vont "pousse" le lapin vers les sous-arbres T_i^q, puis T_i^{q-1} jusqu'au sous-arbre T_i^j. Ensuite, pendant la seconde sous-phase, les deux chasseurs vont nettoyer le sous-arbre T_i^j (sans que le lapin puisse s'échapper de T_i^j s'il y était). Les longueurs des chemins reliant les racines des sous-arbres à c_i (illustré dans la Figure 4.3) garantissent que le lapin ne peut atteindre c_i avant que T_i^j soit complètement nettoyé.

Formellement, durant la première sous-phase, le premier chasseur va tirer sur c_i pendant tous les tours impairs. Par conséquent, le lapin ne peut s'échapper de la branche en c_i dans $T_{i,q} \setminus c_i$ qu'il occupait à la fin de la $(j-1)^{th}$ phase. De plus, pendant la même première sous-phase, le second

chasseur va séquentiellement tirer sur les sommets de $P_i^q, P_i^{q-1}, \ldots, P_i^j$ dans cet ordre, des voisins de c_i aux sommets $c_i^q, c_i^{q-1}, \ldots, c_i^j$. Plus précisément, pour tout $j \leq k \leq q$, notons le $i^{\text{ème}}$ chemin par $P_i^k = (v_0^k = c_i, v_1^k, \ldots, v_{p_i^k}^k = c_i^k)$. Le second chasseur va donc commencer à tirer pendant le tour $i_0^j + 2$ sur v_1^q et va séquentiellement tirer sur $v_2^q, v_3^q, \ldots, v_{p_i^q}^q = c_i^q, v_1^{q-1}, v_2^{q-1}, \ldots, v_{p_i^{q-1}}^{q-1} = c_i^{q-1}, v_1^{q-2}, \ldots, c_i^j$. Remarquons qu'après le tour où le deuxième chasseur tire sur c_i^q, si le lapin occupait un sommet de $T_i^q \cup P_i^q$ au début de cette j^{eme} phase, alors il doit occuper un sommet à distance au moins p_i^q de c_i (c'est-à-dire qu'il se trouve dans T_i^q). De façon similaire, après le tour où le deuxième chasseur tire sur c_i^{q-1}, si le lapin occupait un sommet de $T_i^{q-1} \cup P_i^{q-1}$ au début de cette j^{eme} phase, alors il doit occuper un sommet à distance au moins p_i^{q-1} de c_i. De plus, si le lapin occupait un sommet de $T_i^q \cup P_i^q$ au début de cette j^{eme} phase, alors il doit occuper un sommet à distance au moins $p_i^q - p_i^{q-1}$ de c_i (puisqu'il a déjà eu p_i^{q-1} tours entre les tires sur c_i^q et les tire sur c_i^{q-1}). Avec des arguments similaires, et par définition de p_i^k pour tout $j < k \leq q$, après le tour où le chasseur tire sur c_i^j, le lapin doit être à distance ℓ_{i-1} de c_i s'il occupait un sommet de $\bigcup_{j<k\leq q} T_i^k \cup P_i^k$ a la fin de la $(j-1)^{\text{ème}}$ phase. De plus, le lapin ne peut être sur un sommet de $\bigcup_{k\leq j-1}(V(T_i^k) \cup V(P_i^k))$ puisque le premier chasseur a tiré continuellement sur c_i pendant les tours impairs. Finalement, Le lapin ne peut être dans P_i^j puisque le second chasseur vient juste de tirer séquentiellement sur $v_1^j, v_2^j, \ldots, v_{p_i^j}^j = c_i^j$ (il a "poussé" le lapin dans T_i^j).

La seconde sous-phase de la phase j, durant laquelle les deux chasseurs vont exécuter la stratégie de chasse $\mathcal{S}_{i-1}$ dans T_i^j (le tir du second chasseur sur c_i^j pendant la première sous-phase, c'est-à-dire, le dernier tour de la première sous-phase, peut être utilisé comme premier tour de la stratégie $\mathcal{S}_{i-1}$). Par les hypothèses d'induction, la stratégie $\mathcal{S}_{i-1}$ certifie que le lapin ne peut s'échapper par la racine c_i^j de T_i^j sans être capturé par le tir du premier chasseur (si c_i^j est dans Z_j pendant un tour q, alors $c_i^j \in S_{q+1}$). Par conséquent, si le lapin occupe un sommet de T_i^j au début de la seconde sous-phase de la phase j, alors le lapin ne peut pas s'échapper de ce sous-arbre et est éventuellement capturé par les deux chasseurs exécutant $\mathcal{S}_{i-1}$. Sinon, puisque $\mathcal{S}_{i-1}$ a une longueur au plus ℓ_{i-1}, le lapin ne peut pas atteindre c_i avant les derniers tirs des chasseurs sur T_i^j. Notons par i_0^j le dernier tour de la phase j. Remarquons que i_0^j est pair puisque p_i^h et ℓ_h sont tous pairs pour tout $1 \leq h \leq q$, et i_0^{j-1} est aussi pair par induction. Par conséquent, la $j^{\text{ème}}$ phase se termine après le tour (pair) i_0^j avec la propriété souhaitée : Le lapin ne peut occuper qu'un sommet de $c_i \cup \bigcup_{j<k\leq q}(V(T_i^k) \cup V(P_i^k))$ (pendant le tour i_0^j), c'est-à-dire, $(Z_{i_0^j} \setminus c_i) \cap \bigcup_{k\leq j}(V(T_i^k) \cup V(P_i^k)) = \emptyset$.

Pour conclure, remarquons que $\mathcal{S}_i$ est une stratégie gagnante de longueur au plus $q\ell_{i-1} + \sum_{1\leq j\leq q} jp_i^j$ puisque chaque phase j procède en $\ell_{i-1} + \sum_{j\leq k\leq q} p_i^k$ tours. □

Théorème 4.4.11. *Pour n'importe quel arbre T, il existe une sous-division T' de T telle que $h(T') \leq 2$.*

Démonstration. Notons par q le degré maximum de T. Posons r n'importe quel sommet de T, et notons par i l'excentricité de r (c'est-à-dire, la plus grande distance entre r et un autre sommet de T). Alors, il existe une sous-division T' de T étant un sous-graphe de $T_{i,\max\{6,q\}}$ (chaque sommet de T correspond à un sommet de degré au moins 3 de $T_{i,\max\{6,q\}}$ et r correspond à la racine de $T_{i,\max\{6,q\}}$). Par les Lemmes 4.1.1, 1.5.4 et 4.4.10, $h(T') \leq 2$. □

Corollaire 4.4.12. *Pour tout $\ell \geq 0$, il existe un arbre T et une sous-division T' de T tels que $h(T) - h(T') \geq \ell$.*

4.4.2 Stratégies monotones dans les arbres $(T_{i,q})_{i\geq 3, q\geq 6}$

Avant de prouver le Lemme 4.4.15, nous avons besoin de quelques résultats supplémentaires. Remarquons que le lemme suivant est une adaptation de la Proposition 4.3.11 pour la variante rouge du jeu.

Lemme 4.4.13. *Soit $G = (V_r \cup V_w, E)$ un graphe biparti et soit H un sous-graphe connexe de G. Soit $\mathcal{S} = (S_1, \dots, S_\ell)$ une stratégie de chasse monotone parcimonieuse et gagnante dans G par rapport à V_r. Posons un indice $1 \leq i \leq \ell$ et des sommets $x, y \in V(H)$ tel que $x \in \bigcup_{j<i} S_j$, $y \in Z_{i-1}$ et tel que la distance entre x et y dans H est minimisé. Si $x, y \notin S_i$, alors $xy \in E(H)$.*

Démonstration. Remarquons en premier que si $x = y$, alors $\mathcal{S}$ n'est pas monotone puisque $y = x \in (\bigcup_{j<i} S_j \cap Z_{i-1}) \setminus S_i$. Par conséquent, nous pouvons supposer que $x \neq y$. Soit P un plus court chemin entre x et y dans H (il existe puisque H est connexe). Supposons que $S_i \subseteq V_r$ et donc que i est impair (le cas où $S_i \subseteq V_w$ et i est impair est similaire). Puisque $y \in Z_{i-1}$ et $S_i \subseteq Z_{i-1}$ (car $\mathcal{S}$ est parcimonieuse), $y \in V_r$. Soit a le voisin de x dans P.

Pour arriver à une contradiction, supposons que $a \neq y$. Par minimalité de la distance entre x et y, $a \notin Z_{i-1}$ et $a \notin \bigcup_{j<i} S_j$. Soit b le voisin de a dans P différent de x. Si b est différent de y, alors, par minimalité de la distance entre x et y, $b \notin Z_{i-1}$ et $b \notin \bigcup_{j<i} S_j$. Par conséquent, et par la Proposition 4.4.3, si $a \in V_r$, alors $a \in Z_i$. Au contraire, si $b \in V_r$, alors $b \in Z_i$. Puisque que a et b sont adjacents, un des deux est dans V_r et donc dans Z_i contredisant la minimalité de la distance entre x et y.

Nous pouvons donc supposer que $b = y$. Cela implique $x \in V_r$. Remarquons aussi que $y \notin S_j$ pour tout $j \leq i$. En effet, supposer que $b \neq y$ contredirait la monotonie de $\mathcal{S}$, puisque $y \notin S_i$ et $y \in Z_{i-1}$.

En conclusion, par la Proposition 4.4.3, et puisque $a, y \notin \bigcup_{j\leq i-1} S_j$, nous obtenons que $x \in Z_{i-1}$. Cela contredit la monotonie de $\mathcal{S}$ puisque $x \notin S_i$ et $x \in \bigcup_{j<i} S_j$. □

Avant de prouver le prochain Lemme, nous aurons besoin d'une définition supplémentaire. Pour un graphe G et une stratégie de chasse $\mathcal{S} = (S_1, \dots, S_\ell)$ dans G par rapport à $X \subseteq V$, un sous-ensemble de sommets $W \subseteq V$ est *définitivement nettoyé au tour i* si $W \cap Z_j(\mathcal{S}) = \emptyset$ et $W \cap S_{j+1} = \emptyset$ pour tout $i \leq j \leq \ell$.

Intuitivement, le prochain lemme certifie que si le degré de la racine r de l'arbre T est assez grand comparé au nombre de chasseurs, alors, quand la première branche en r est définitivement nettoyée par une stratégie de chasse monotone, il existe un certain nombre de branches (nous n'avons besoin que de deux branches non tirées pour la suite) qui n'ont jusqu'alors jamais été tirées (sur aucun de leurs sommets).

Lemme 4.4.14. *Soit $T = (V_r \cup V_w, E)$ un arbre enraciné en un sommet $c \in V_r$ dont ces voisins sont $N(c) = \{v_1, \dots, v_d\}$ avec $d \geq 2k$. Pour tout $1 \leq i \leq d$, notons par B_i la branche en c contenant v_i et supposons que $|V(B_i)| \geq 2$. Soit $\mathcal{S} = (S_1 \dots, S_p)$ une stratégie de chasse parcimonieuse et gagnante dans T par rapport à V_r utilisant au plus $k-1$ chasseurs. Soit $1 \leq j \leq p$ le plus petit indice tel qu'il existe $1 \leq \alpha \leq d$ tel que $V(B_\alpha)$ est définitivement nettoyé au tour j. Alors, il existe $1 \leq \beta < \gamma \leq d$, $\alpha \notin \{\beta, \gamma\}$, tel que $(\bigcup_{1\leq i\leq j} S_i) \cap V(B_\beta) = \emptyset$ et $(\bigcup_{1\leq i\leq j} S_i) \cap V(B_\gamma) = \emptyset$.*

Démonstration. Soient j et α définis comme dans l'énoncé et, sans perte de généralité, supposons que $\alpha = 1$. B_1 est donc la première branche en c nettoyée au tour j.

Assertion 4.4.14.1 – Pour tout sommet $v \in V(T)$ tel qu'il existe au moins 3 branches en v contenant chacune au moins 2 sommets, définissons q le plus petit entier tel qu'une branche B en v est définitivement nettoyée au tour q. Alors, $S_q \cap V(B) \neq \emptyset$.

Démonstration. Par minimalité de q, soit $Z_{q-1} \cap V(B) \neq \emptyset$ ou $S_q \cap V(B) \neq \emptyset$. Si $S_q \cap V(B) \neq \emptyset$, alors l'énoncé de l'assertion tient. Par conséquent, supposons que $S_q \cap V(B) = \emptyset$. Alors $Z_{q-1} \cap V(B) \neq \emptyset$, car sinon B était déjà définitivement nettoyée au tour $q-1$. Soit $x \in Z_{q-1} \cap V(B)$. Puisque $x \notin S_q$, $N_B(x) \subseteq Z_q$. Puisque B est connexe et contient au moins 2 sommets, $N_B(x) \neq \emptyset$. Par conséquent, $Z_q \cap V(B) \neq \emptyset$, ce qui contredit le fait que B soit définitivement nettoyée au tour q. ◇

Grâce à l'Assertion 4.4.14.1, nous savons que $V(B_1) \cap S_j \neq \emptyset$.

Par conséquent, et puisque $|S_j| \leq k-1$, il existe au plus $k-2$ autres branches en c (différentes de B_1), pouvant être tirées pendant le tour j. Sans perte de généralité, supposons que $B_2, \ldots, B_{k-1}$ sont les branches en c qui peuvent aussi être tirées pendant le tour j (elles ne le sont pas forcément). Plus précisément, $S_j \subseteq \{c\} \cup \bigcup_{1\leq h<k} V(B_h)$.

Pour arriver à une contradiction, supposons qu'il existe au plus une branche, sans perte de généralité B_k, tel que $\bigcup_{1\leq i\leq j} S_i \cap V(B_k) = \emptyset$. Autrement dit, nous supposons que pour tout $k < h \leq d$, il existe $j_h < j$ et $x_h \in V(B_h) \cap S_{j_h}$.

Pour tout $k < h \leq d$, dénotons par j_h^* le plus petit entier tel que B_h est définitivement nettoyée au tour j_h^*. Par l'Assertion 4.4.14.1, pour tout $k < h \leq d$, nous obtenons que $S_{j_h^*} \cap V(B_h) \neq \emptyset$. Cela implique entre autre que, pour tout $k < h \leq d$, $j_h^* \neq j$ puisque $V(B_h) \cap S_j = \emptyset$. De plus, avec la minimalité de j, nous obtenons que $j_h^* > j$ pour tout $k < h \leq d$.

Nous allons prouver à présent que, pour tout $k < h \leq d$, il existe un sommet $y_h \in Z_{j-1} \cap V(B_h)$. Pour arriver à une contradiction, supposons que pour un certain $k < h \leq d$, nous avons $Z_{j-1} \cap V(B_h) = \emptyset$. Rappelons que $S_{j_h^*} \cap V(B_h) \neq \emptyset$, et notons par z n'importe quel sommet dans $S_{j_h^*} \cap V(B_h)$. Puisque $\mathcal{S}$ est parcimonieuse, $z \in Z_{j_h^*-1}$. Il doit donc exister une trajectoire $R = (r_0, \ldots, r_{j_h^*-1} = z)$ pour le lapin telle que $r_q \notin S_{q+1}$ pour tout $0 \leq q < j_h^* - 1$.

Remarquons que $r_{j-1} \notin V(B_h)$ puisque $Z_{j-1} \cap V(B_h) = \emptyset$. Cela implique notamment qu'il existe $j-1 \leq m < j_h^* - 1$ tel que $r_m = c$ (puisque R est une trajectoire et que $r_{j-1} \notin V(B_h)$ et $r_{j_h^*-1} \in V(B_h)$). Le fait que $r_m = c$ and $r_m \notin S_{m+1}$, implique que $v_1 \in Z_{m+1}$ où v_1 est le voisin de c dans B_1. Cela contredit que B_1 soit définitivement nettoyée au tour j ($j-1 \leq m$). Pour conclure cette deuxième contradiction, nous savons à présent que pour tout $k < h \leq d$, il existe un sommet $y_h \in Z_{j-1} \cap V(B_h)$.

Finalement, pour tout $k < h \leq d$, posons deux sommets x_h et y_h tels que $x_h \in V(B_h) \cap \bigcup_{1\leq i<j} S_i$ et $y_h \in Z_{j-1} \cap V(B_h)$ et tels que la distance entre ces deux sommets soit minimisée. Notons que $y_h \notin \bigcup_{1\leq i<j} S_i$, car sinon, $\mathcal{S}$ ne serait pas monotone puisque $y_h \in Z_{j-1} \setminus S_j$. Puisque $S_j \cap V(B_h) = \emptyset$ et par le Lemme 4.4.13, nous obtenons que $x_h y_h \in E(B_h)$. Par conséquent, $x_h \in Z_j$ pour tout $k < h \leq d$. Puisque $\mathcal{S}$ est une stratégie monotone, que x_h a déjà été tiré avant le tour j et que $x_h \in Z_j$, nous obtenons que $x_h \in S_{j+1}$ pour tout $k < h \leq d$. Vu que $d \geq 2k$, nous obtenons donc que $|S_{j+1}| \geq k$, ce qui contredit que la stratégie $\mathcal{S}$ utilise au plus $k-1$ chasseurs. □

Finalement, nous allons avoir besoin d'une définition supplémentaire : Pour tout arbre $T = (V_r \cup V_w, E)$, pour tout sommet $v \in V(T)$, et pour n'importe quelle branche B en v tel que $|V(B)| > 1$. Pour toute stratégie de chasse $\mathcal{S} = (S_1, \ldots, S_p)$ dans T par rapport à V_r, posons m le plus petit entier tel que $S_m \cap V(B) \neq \emptyset$ et posons $u \in S_m \cap V(B)$ (par le Lemme 4.1.4, un tel

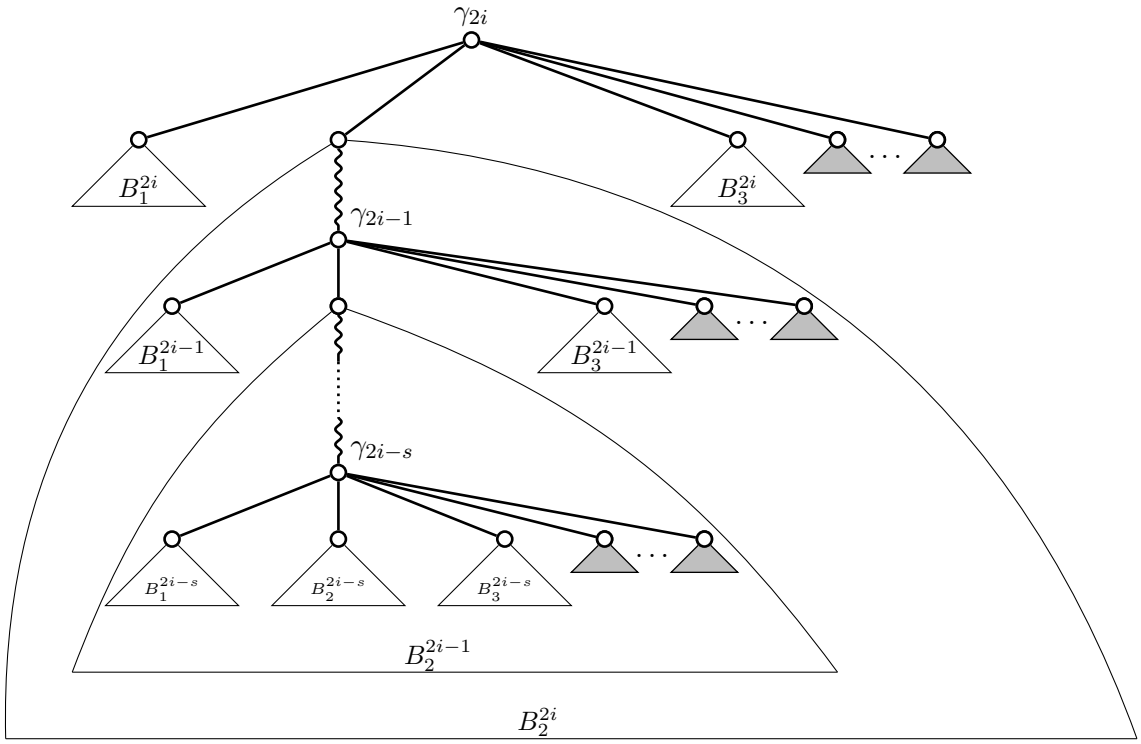


Figure 4.4 – Une représentation de l'arbre $T_{2i,d}$ illustrant les notations utilisées dans la preuve du Lemme 4.4.15. Les arêtes ondulées sont utilisées pour représenter des chemins de sommets internes de degré 2. La branches B_1^{2i} est la première branche dans $T_{2i,d}$ en γ_{2i} qui est définitivement nettoyée, et cela arrive durant le tour j_{2i}. Les sommets de B_2^{2i} et de B_3^{2i} n'ont jamais été tirés avant le tour $j_{2i}+1$. Parmi ces deux branches B_2^{2i} et B_3^{2i}, la première à être définitivement nettoyée est la branche B_2^{2i}, et cela arrive durant le tour $j'_{2i} > j_{2i}$. La couleur grise est utilisée sur les triangles pour montrer que nous ne savons pas qu'elle est l'état de ses branches quand ces branches seront définitivement nettoyées ou même si elles l'ont déjà été. Observons que B_2^{2i} contient une copie de $T_{2i-1,d}$, enraciné en γ_{2i-1}. Puisque B_2^{2i} est définitivement nettoyée au tour j'_{2i}, nous pouvons itérer les mêmes arguments et définir B_1^{2i-1} comme la première branche dans $T_{2i-1,d}$ en γ_{2i} étant définitivement nettoyée au tour $j_{2i} < j_{2i-1} < j'_{2i}$, jusqu'à atteindre les feuilles de B_2^{2i}.

entier m existe puisque $|V(B)| > 1$ et B est connexe implique que $V_r \cap V(B) \neq \emptyset$). Définissons $\mathcal{S}_B = (S'_1, \dots S'_p)$ comme la *restriction* de $\mathcal{S}$ sur B tel que pour tout $1 \leq i \leq \ell$,

$$S'_i = \begin{cases} S_i \cap V(B), & \text{si } S_i \cap V(B) \neq \emptyset \\ \{u\}, & \text{sinon} \end{cases}$$

Rappelons que, $h(\mathcal{S}_B) \leq \max_{1 \leq i \leq \ell} S_i \cap V(B)$ par le Lemme 4.4.6.

Lemme 4.4.15. *Pour n'importe quels entiers $i \in \mathbb{N}^*$, $i \geq 3$, et $d \geq 2i$, $mh_{V_r}(T_{2i,d}) \geq i$.*

Démonstration. Notons par γ_{2i} la racine de $T_{2i,d}$.

Pour arriver à une contradiction, supposons que $mh_{V_r}(T_{2i,d}) < i$. Par le Lemme 4.4.7, il existe une stratégie de chasse $\mathcal{S}_{2i} = (S_1^{2i}, \dots, S_\ell^{2i})$ parcimonieuse monotone et gagnante par rapport à V_r utilisant au plus $i-1$ chasseurs.

Posons $1 \leq j_{2i} \leq \ell$ le plus petit indice tel qu'une branche en γ_{2i}, sans perte de généralité B_1^{2i}, est définitivement nettoyée au tour j_{2i}. Par le Lemme 4.4.14, il existe deux branches en γ_{2i}, sans perte de généralité B_2^{2i} et B_3^{2i}, tels que $(\bigcup_{1\leq q\leq j_{2i}} S_q) \cap V(B_2^{2i}) = \emptyset$ et $(\bigcup_{1\leq q\leq j_{2i}} S_q) \cap V(B_3^{2i}) = \emptyset$.

Posons $1 \leq j'_{2i} \leq \ell$ le plus petit indice tel qu'au moins une branche, B_2^{2i} ou B_3^{2i}, est définitivement nettoyée au tour j'_{2i}.

Remarquons que B_2^{2i} (resp., B_3^{2i}) est connexe et contient au moins deux sommets (et donc au moins un sommet dans V_r). Par conséquent, par le Lemme 4.1.4, au moins un sommet de B_2^{2i} (resp., B_3^{2i}) doit être tiré avant que la branche soit définitivement nettoyée. Ainsi, $j_{2i} < j'_{2i}$. Sans perte de généralité, supposons que B_2^{2i} est définitivement nettoyée au tour j'_{2i} (possiblement, B_3^{2i} peut aussi être définitivement nettoyée au tour j'_{2i}).

Nous prouvons à présent par induction sur $0 \leq h < 2i$, qu'il existe $1 \leq j_{2i} < j_{2i-1} < \cdots < j_{2i-h} < j'_{2i-h} \leq j'_{2i-h+1} \leq \cdots \leq j'_{2i-1} \leq j'_{2i} \leq \ell$ et $\{B_1^{2i-s}, B_2^{2i-s}, B_3^{2i-s}\}_{0\leq s\leq h}$ tels que, pour tout $0 \leq s \leq h$:

1. $(\bigcup_{1\leq q\leq j_{2i-s}} S_q) \cap V(B_2^{2i-s}) = \emptyset$ et $(\bigcup_{1\leq q\leq j_{2i-s}} S_q) \cap V(B_3^{2i-s}) = \emptyset$;
2. B_1^{2i-s}, B_2^{2i-s} et B_3^{2i-s} sont des branches sommet-disjoints en la racine γ_{2i-s} de la copie de $T_{2i-s,d}$ contenue dans $B_2^{2i-(s-1)}$ (avec $B_2^{2i+1} = T_{2i,d}$ pour $s = 0$), chacun contenant une copie de $T_{2i-(s+1),d}$ (avec $T_{0,d} = \emptyset$ pour $s = h = 2i-1$) ;
3. B_1^{2i-s} est définitivement nettoyée au tour j_{2i-s} (pas avant, c'est-à-dire, pour tout $x < j_{2i-s}$, B_1^{2i-s} n'est pas définitivement nettoyée au tour x), B_2^{2i-s} est définitivement nettoyée au tour j'_{2i-s} (pas avant) et B_3^{2i-s} est définitivement nettoyée au tour $x \geq j'_{2i-s}$ (pas avant).

Voir la Figure 4.4 pour une illustration de ces notations.

Pour $h = 0$, nous avons déjà prouvé l'existence de ces branches et de ces indices. Supposons que l'induction tienne pour $0 \leq h < 2i-1$ et prouvons que l'induction tient pour $h+1$. Notons par F la copie de $T_{2i-(h+1),d}$ (enracinée en $\gamma_{2i-(h+1)}$) contenue dans B_2^{2i-h} et posons $1 \leq j_{2i-(h+1)} \leq j'_{2i-h}$ le plus petit indice tel qu'une branche B en $\gamma_{2i-(h+1)}$ dans F est définitivement nettoyée au tour $j_{2i-(h+1)}$. Notons que chacune des branches en $\gamma_{2i-(h+1)}$ dans F est connexe et contient au moins deux sommets (et donc au moins un dans V_r). Par conséquent, par le Lemme 4.1.4, au moins un sommet de B doit être tiré avant que B soit définitivement nettoyée. Ainsi, $j_{2i-h} < j_{2i-(h+1)}$. Sans perte de généralité, supposons que $B = B_1^{2i-(h+1)}$. Notons par $\mathcal{S}_{2i-(h+1)}$ la restriction de $\mathcal{S}_{2i-h}$ sur F. Par le Lemme 4.4.14, pour la stratégie $\mathcal{S}_{2i-(h+1)}$ sur F, il existe au moins deux branches en $\gamma_{2i-(h+1)}$, sans perte de généralité $B_2^{2i-(h+1)}$ et $B_3^{2i-(h+1)}$, telles que $(\bigcup_{1\leq q\leq j_{2i-(h+1)}} S_q) \cap V(B_2^{2i-(h+1)}) = \emptyset$ et $(\bigcup_{1\leq q\leq j_{2i-(h+1)}} S_q) \cap V(B_3^{2i-(h+1)}) = \emptyset$. Aussi, nous savons par le Lemme 4.1.4 que $j_{2i-(h+1)} < j'_{2i-(h+1)}$. Finalement, $B_1^{2i-(h+1)}$, $B_2^{2i-(h+1)}$ et $B_3^{2i-(h+1)}$ sont toutes contenues dans B_{2i-h} et, donc, $max(j_{2i-(h+1)}, j'_{2i-(h+1)}) = j'_{2i-(h+1)} \leq j'_{2i-h}$. Cela conclut la fin de l'étape de l'induction.

Pour tout $1 \leq s \leq 2i$, notons par H_s le sous-graphe induit par B_1^s, B_3^s et γ_s (donc H_s est connexe et les sous-graphes H_s et $H_{s'}$ sont sommet-disjoints pour tout $s \neq s'$). Puisque B_1^s est définitivement nettoyée au tour j_s, contient au moins deux sommets et par le Lemme 4.1.4, il existe un sommet $x'_s \in V(B_1^s) \cap \bigcup_{1\leq q\leq j_{2i}} S_q$.

Notons que B_3^s est connexe et contient au moins deux sommets (et donc au moins un dans V_r). Par conséquent, par le Lemme 4.1.4, au moins un sommet de B_3^s doit être tiré avant que la branche ne soit définitivement nettoyée. Ainsi, puisque B_3^s est définitivement nettoyée au tour

$z_s \geq j'_s$, mais n'est pas définitivement nettoyée à un tour précédent, $S_{z_s} \cap V(B_3^s) \neq \emptyset$. De plus, puisque $\mathcal{S}_{2i}$ est parcimonieuse, tout sommet $v \in S_{z_s} \cap V(B_3^s)$ est dans Z_{z_s-1}. Cela implique que $(N(v) \cap Z_{z_s-2}) \setminus S_{z_s-1} \neq \emptyset$. Posons w_s dans $(N(v) \cap Z_{z_s-2}) \setminus S_{z_s-1}$ et remarquons que $w_s \in N[V(B_3^s)] = V(B_3^s) \cup \{\gamma_s\}$. Puisque $\mathcal{S}_{2i}$ est monotone, nous obtenons que $w_s \notin S_j$ pour tout $j < z_s$ et que $N(w_s) \not\subseteq S_j$ pour tout $j < z_s$, c'est-à-dire, w_s n'a pas été tiré avant le tour z_s. Notons aussi que si $w_s = \gamma_s$, alors le voisin de γ_s dans B_1^s est dans Z_{z_s-1}, une contradiction puisque $z_s > j_1$ et B_1^s est définitivement nettoyée au tour $j_s \leq j_1$. Ainsi, $w_s \neq \gamma_s$ et donc $w_s \in V(B_3^s)$.

De façon similaire, il existe un sommet $w'_s \in N(w_s) \cap Z_{z_s-3} \setminus S_{z_s-2}$ tel que w'_s n'a jamais été tiré avant le tour $z_s - 1$. Ainsi, nous avons deux sommets adjacents w_s et w'_s dans $N[B_3^s]$ qui n'ont jamais été tirés avant le tour $z_s - 1$. Par conséquent, puisque $\{w_s, w'_s\} \cap V_r \neq \emptyset$, il existe une trajectoire pour le lapin $(\dots, w_s, w'_s, w_s, w'_s, \dots)$ consistant à osciller entre w_s et w'_s qui implique que $\{w_s, w'_s\} \cap Z_j \neq \emptyset$ pour tout $j < z_s$. En particulier, puisque $j_1 - 1 < z_s$, il existe un sommet $y'_s \in N[B_3^s] \cap Z_{j_1-1}$.

Pour tout $1 \leq s \leq 2i$, posons x_s et y_s, deux sommets dans $V(H_s)$ tels que $x_s \in V(H_s) \cap \bigcup_{1 \leq q \leq j_1} S_q$, $y_s \in V(H_s) \cap Z_{j_1-1}$ et la distance entre x_s et y_s est minimisé. Définissons $\mathcal{P} = \{s \mid 1 \leq s \leq 2i, y_s \in S_{j_1} \cup S_{j_1+1}$ où $x_s \in S_{j_1} \cup S_{j_1+1}\}$, c'est-à-dire, $\mathcal{P}$ est l'ensemble d'indices s tels qu'au moins un sommet, y_s ou x_s, est tiré pendant le tour j_1 ou $j_1 + 1$. Puisque $\mathcal{S}_{2i}$ utilise au plus $i - 1$ chasseurs, nous obtenons que $|\mathcal{P}| \leq 2(i - 1)$. Ainsi, nous pouvons prendre un indice $1 \leq s^* \leq 2i$ tel que $s^* \notin \mathcal{P}$, c'est-à-dire, $x_{s^*}, y_{s^*} \notin S_{j_1} \cup S_{j_1+1}$. Il suit du Lemme 4.4.13 que $x_{s^*}y_{s^*} \in E(T_{2i,d})$. Puisque $y_{s^*} \in Z_{j_{2i}-1} \setminus S_{j_1}$, nous obtenons que $x_{s^*} \in Z_{j_1}$. Dans tous les cas, $x_{s^*} \in \bigcup_{1 \leq q \leq j_1} S_q \setminus S_{j_1+1}$ et donc, x_{s^*} est recontaminé, contredisant la monotonie de $\mathcal{S}_{2i}$. □

4.5 Kernelisation par la couverture par sommets minimum

Bien que le calcul du nombre de chasseurs monotone soit NP-difficile, il est possible que ce problème (ou la version non-monotone) soit FPTpar certains paramètres. Dans cette section, nous nous intéressons à l'étude de ces problèmes paramétrés par *le nombre de sommets couvrants*.

Rappelons quelques définitions basiques de la complexité paramétrée. Une instance d'une version paramétrée Π_p d'un problème de décision Π est une paire (I, t), où I est une instance de Π et t est un entier positif, appelé le *paramètre*, associé avec I. L'instance paramétrée Π_p est *résoluble à paramètre fixé* (Fixed-Parameter Tractable) s'il existe un algorithme (appelé *algorithme FPT*) qui, étant donné une instance (I, t) de Π_p, le résout en temps $f(t) \cdot |I|^{\mathcal{O}(1)}$, où f est une fonction de t.

Définition 4.5.1 (Instances équivalentes). Soit Π_1 et Π_2, deux problèmes paramétrés. Deux instances, $(I, t) \in \Pi_1$ et $(I', t') \in \Pi_2$, sont *équivalentes* si (I, t) est une instance vraie si et seulement si (I', t') est une instance vraie.

Un problème (de décision) paramétré Π_p admets un *noyau* de taille $f(t)$, pour une fonction f qui dépend seulement de t, s'il existe un algorithme (appelé un *algorithme de kernelisation*) qui, étant donné une instance (I, t) de Π_p, retourne, en temps $(|I| + t)^{\mathcal{O}(1)}$, une instance équivalente (I', t') de Π_p tel que $|I'| \leq f(t)$ et $t' \leq t$. Si la fonction f est polynomiale, alors le problème admets un *noyau polynomial*. Il est bien connu qu'un problème de décision paramétré est FPT si et seulement si il admet un noyau (Cygan et al., 2015b).

Rappelons qu'une *couverture par sommets* d'un graphe G est un ensemble de sommets $U \subseteq V(G)$ tel que pour toute arête $\{u, v\} \in E(G), U \cap \{u, v\} \neq \emptyset$. La taille minimum d'une couverture par sommets est appelée *le nombre de sommets couvrants* (*vertex cover number*) de G et est notée par $vc(G)$. Dans ce qui suit, nous considérons le problème des CHASSEURS ET DU LAPIN paramétré par le nombre de sommets couvrants. Plus précisément, nous considérons une instance $((G, k), t)$ où le problème revient à décider si $h(G) \leq k$ et où le paramètre t est n'importe quelle borne supérieure pour le nombre de sommets couvrant $vc(G)$.

Premièrement, nous avons l'observation suivante.

Proposition 4.5.1. *Pour tout graphe connexe G, $h(G) \leq mh(G) \leq vc(G)$.*

Démonstration. Posons U, une couverture par sommets de G et notons par I l'ensemble de sommets indépendants $V(G) \setminus U$. Les chasseurs peuvent gagner simplement en tirant sur U deux fois, c'est-à-dire, $\mathcal{S} = (S_1 = U, S_2 = U)$. Si le lapin comment sur un sommet $u \in U$, alors il est tiré immédiatement. Sinon, il se trouvait sur $v \in I$ et doit bouger sur un sommet $u \in U$ durant le premier tour (puisque I est un ensemble indépendant), c'est-à-dire, $Z_1(\mathcal{S} = U$. Puisque $S_2 = U$, nous obtenons que $Z_2(\mathcal{S} = \emptyset$, et donc $\mathcal{S}$ est une stratégie gagnante. Notons aussi que $\mathcal{S}$ est monotone. Par conséquent, $h(G) \leq mh(G) \leq vc(G)$. □

Soit U une couverture par sommets de taille $t \geq vc(G)$ de G et soit I l'ensemble de sommets indépendants $V(G) \setminus U$. Pour chaque sous-ensemble de sommets $S \subseteq U$, nous définissons la classe d'équivalence : $\mathcal{C}_S = \{v \mid v \in I \text{ et } N(v) = S\}$.

Ensuite, nous avons le lemme crucial suivant.

Lemme 4.5.2. *Pour tout graphe connexe $G = (V, E)$, pour toute couverture par sommets $U \subseteq V$ de G, et pour un certain $k \geq 1$, posons $S \subseteq U$ tel que $|\mathcal{C}_S| > k + 1$. Soit $\mathcal{C}_S = \{v_1, \ldots, v_q\}$. Alors, $h(G) \leq k$ (resp., $mh(G) \leq k$) si et seulement si $h(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) \leq k$ (resp., $mh(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) \leq k$).*

Démonstration. Par le Lemme 4.1.1, $h(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) \leq h(G)$. De façon similaire, par le Lemme 4.2.5, $mh(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) \leq mh(G)$. Par conséquent, il ne reste plus qu'à prouver que, si $h(G) > k$ (resp., $mh(G) > k$), alors $h(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) > k$ (resp., $mh(G[V \setminus \{v_{k+2}, \ldots, v_q\}]) > k$). Définissons $H = G[V \setminus \{v_{k+2}, \ldots, v_q\}]$ et notons par $X = \{v_1, \ldots, v_{k+1}\}$ les sommets de $\mathcal{C}_S \cap V(H)$.

Dans ce qui suit, nous montrons que si $h(G) > k$ (resp., $mh(G) > k$), alors $h(H) > k$ (resp., $mh(H) > k$). Pour y arriver, nous devons montrer que s'il existe une stratégie de survie pour le lapin dans G contre k chasseurs, alors il en existe une pour le lapin dans H contre k chasseurs.

(1) $\boldsymbol{h(G) > k \implies h(H) > k}$ **:** Soit $\mathcal{S} = (S_1, S_2, \ldots, S_\ell)$ une stratégie de chasse (pas forcément gagnante) dans H utilisant k chasseurs. Remarquons que $\mathcal{S}$ est une stratégie de chasse dans G utilisant au plus k chasseurs. Puisque $h(G) > k$, il existe une trajectoire $\mathcal{R}' = (r'_0, r'_1, \ldots, r'_{\ell-1})$ pour le lapin dans G gagnante contre $\mathcal{S}$, c'est-à-dire, tel que $r'_i \notin S_{i+1}$ pour tout $0 \leq i < \ell$. Définissons $\mathcal{R} = (r_0, \ldots, r_{\ell-1})$ une trajectoire pour le lapin tel que, pour tout $0 \leq i < \ell$, $r_i = r'_i$ si $r'_i \in V(H)$ et, dans le cas contraire, posons r_i n'importe quel sommet de $X \setminus S_{i+1}$ (un tel sommet existe puisque $|S_{i+1}| \leq k$ et $|X| > k$). Notons que $r'_i \neq r_i$ seulement si $r'_i \notin V(H)$ et donc $r'_i \in \mathcal{C}_S \setminus X$. Cela implique que, si $r'_i \notin V(H)$, alors $r'_{i-1}, r'_{i+1} \in S \subset V(H)$ (puisque $N(r'_i) = S$). Par conséquent, $r_{i-1} = r'_{i-1}$, $r_{i+1} = r'_{i+1}$ et $r_{i-1}, r_{i+1} \in N_H(r_i)$ (puisque $r_i \in X$ et donc $N_G(r_i) = N_H(r_i) = S$). Nous pouvons donc conclure que, $\mathcal{R}$ est une trajectoire pour le

lapin dans H, et donc, par construction, $r_i \notin S_{i+1}$ pour tout $0 \leq i < \ell$, c'est-à-dire, $\mathcal{S}$ n'est pas une stratégie de chasse gagnante dans h et donc $h(H) > k$.

(2) $\boldsymbol{mh(G) > k \implies mh(H) > k}$ **:** Soit $\mathcal{S} = (S_1, S_2, \ldots, S_\ell)$ une stratégie de chasse monotone (pas forcément gagnante) dans H utilisant k chasseurs. Remarquons que, $\mathcal{S}$ est aussi une stratégie de chasse dans G utilisant k chasseurs. Puisque $mh(G) > k$, pour toute stratégie de chasse $\mathcal{S}$ dans G utilisant k chasseurs, il existe une trajectoire pour le lapin R' tel que soit R' est gagnante dans G contre $\mathcal{S}$ (c'est-à-dire, $h(G) > k$) ou soit un sommet est recontaminé (c'est-à-dire, $mh(G) > k$). À présent, construisons R à partir de R' de la même façon que le paragraphe précédent. Si le lapin n'est n'est jamais tiré pendant la trajectoire $\mathcal{R}'$ dans G contre $\mathcal{S}$, alors, avec les mêmes arguments utilisés dans (1), $\mathcal{R}$ est gagnante dans H. Par conséquent, nous supposons que même s'il existe un tour q tel que $r_q \in S_{q+1}$, il doit exister un tour $q' < q$ tel que $r_{q'}$ est recontaminé au tour q', c'est-à-dire, il existe $q'' < q'$ tel que $r'_q \in S_{q''+1} \setminus S_{q'+1}$ (par le Lemme 4.2.4). Puisque seulement les sommets de H sont tirés dans $\mathcal{S}$, $r_{q'} \in V(H)$. Par conséquent, $r_{q'}$ est aussi recontaminé au tour q' par $\mathcal{R}$ dans H contre la stratégie $\mathcal{S}$. Nous obtenons donc que $mh(H) > k$. □

Finalement, nous présentons notre résultat de kernelisation.

Théorème 4.5.3. *Le problème prenant en entrée un graphe connexe G contenant n sommets et un entier $k \geq 1$, décidant si $h(G) \leq k$ (resp., $mh(G) \leq k$), paramétré par une borne supérieure t de $vc(G)$, admet un noyau de taille au plus $4^t(t+1) + 2t$. De plus, le problème peut être résolu en temps $(4^t(t+1) + 2t)^{t+1} \cdot n^{\mathcal{O}(1)}$.*

Démonstration. L'algorithme de kernelisation procède comme suit. Premièrement, si $k > t$, alors l'algorithme répond que $h(G) \leq mh(G) \leq k$ (c'est correct par la Proposition 4.5.1). Sinon, posons U une couverture par sommets de taille au plus $2t$ dans G (elle peut être calculée en temps $\mathcal{O}(tn)$ par des algorithmes d'approximation classique de ratio 2 pour la couverture par sommets utilisant des couplages maximaux (Williamson & Shmoys, 2011)). Soit H le sous-graphe de G obtenu comme suit. Pour tout $S \subseteq U$, si $|\mathcal{C}_S| > k+1$, alors nous retirons $|\mathcal{C}_S| - (k+1)$ sommets de $\mathcal{C}_S$. Par le Lemme 4.5.2 (utilisé itérativement sur chaque sous-ensemble $S \subseteq U$), $h(G) \leq k$ (resp., $mh(G) \leq k$) si et seulement si $h(H) \leq k$ (resp., $mh(H) \leq k$). De plus, $|V(H)| = |U| + \sum_{S \subseteq U} |\mathcal{C}_S \cap V(H)| \leq 2t + 2^{2t}(k+1) \leq 4^t(t+1) + 2t$ (La dernière égalité tient par la Proposition 4.5.1). Par conséquent, l'algorithme décrit si dessus est l'algorithme de kernelisation désiré.

Finalement, en utilisant l'algorithme XP (Abramovskaya et al., 2016), il peut être décidé en temps $|V(H)|^{k+1}$ si $h(H) \leq k$ ou non. Puisque, par la Proposition 4.5.1, $k \leq t$, cela nous donne l'algorithme FPT décidant si $h(G) \leq k$ (resp., $mh(G) \leq k$) en temps $(4^t(t+1) + 2t)^{t+1} \cdot n^{\mathcal{O}(1)}$. □

4.6 Poursuite des travaux

Dans ce chapitre, nous avons étudié le jeu DES CHASSEURS ET DU LAPIN en définissant la monotonie de ce jeu. En utilisant cette nouvelle notion de monotonie, nous avons caractérisé le nombre de chasseurs monotone de plusieurs classes de graphes. De plus, nous avons établi que, contrairement à plusieurs jeux de recherche, la recontamination aide dans ce jeu, c'est-à-dire, $h(G)$ peut être arbitrairement plus petit que $mh(G)$. De plus, nos résultats nous ont permis de montrer

que le nombre de chasseurs monotone ne se comporte pas “bien” par rapport aux mineurs et aux sous-divisions, contrairement aux paramètres de la largeur arborescente et de la largeur linéaire.

Il reste encore plusieurs questions ouvertes qui semblent difficiles. La plus importante est celle sur la complexité du calcul du jeu DES CHASSEURS ET DU LAPIN. Même si nous avons prouvé que calculer $mh(G)$ est NP-difficile, la complexité du calcul de $h(G)$ reste ouverte, même si G est restreint à un arbre.

Nous avons aussi prouvé que le jeu DES CHASSEURS ET DU LAPIN, ainsi que sa variante monotone, sont FPTpar $vc(G)$ en concevant des noyaux exponentiels. Il n'est pas difficile de voir que chacun de ces jeux admets une composition “AND” paramétré par la taille de la solution (en prenant l'union disjointe d'instances). Par conséquent, puisque calculer $mh(G)$ est NP-difficile et que $pw(G) \leq mh(G) \leq pw(G) + 1$, il est peu probable que la variante monotone du jeu DES CHASSEURS ET DU LAPIN paramétrée par $k + pw(G)$ admette une compression polynomiale. Notons que nous ne pouvons pas dire la même chose de la version classique du jeu DES CHASSEURS ET DU LAPIN puisque nous ne savons pas encore si la complexité de calcul est NP-difficile. De plus, puisque $mh(G)$ est relié à $pw(G)$ et admet un noyau polynomial par rapport à $vc(G)$ (Chapelle, Liedloff, Todinca, & Villanger, 2017), il pourrait être intéressant de regarder si décider si $mh(G) \leq k$ (resp., $h(G) \leq k$) admet aussi un noyau de taille polynomiale quand le problème est paramétré par $vc(G)$. De plus, une autre direction de recherche intéressante serait d'étudier la complexité paramétrée de ces deux jeux en considérant des paramètres comme la taille de la solution, la largeur arborescente ou encore la largeur linéaire.

Finalement, nous proposons quelques questions ouvertes concernant le calcul de $h(G)$ pour plusieurs classes de graphes comme les arbres, les cographes et les graphes d'intervalles. Plus précisément, il serait fort intéressant de concevoir un algorithme polynomial similaire à l'algorithme 4.1, pour calculer $h(T)$ pour un arbre T (une question déjà posée dans (Abramovskaya et al., 2016)). La direction la plus naturelle pour proposer un tel algorithme est de définir la notion de monotonie, ce que nous avons fait. Malheureusement, le Théorème 4.4.2 implique qu'une telle approche ne marcha pas. Cela implique notamment le besoin de nouveaux outils et techniques pour la conception de cet algorithme.

CHAPITRE 5

Conclusion et Perspectives

Nous avons donc étudié les longueurs arborescente et linéaire des sous-classes de graphes simples de graphes planaires. Cette étude a pour but d'améliorer notre compréhension de ces paramètres et de leurs liens avec les largeurs arborescente et linéaire.

Rappel des contributions

Comme la longueur arborescente des cycles, des graphes planaires extérieurs et des grilles étaient déjà connues, nous nous sommes intéressés à la longueur arborescente des graphes série-parallèles. Nous avons aussi conçu un algorithme d'approximation de ratio $\frac{3}{2}$ pour le calcul de la longueur arborescente des graphes série-parallèles. Nous avons aussi caractérisé les graphes série-parallèles de longueur arborescente 2 en termes de sous-graphes isométriques et avons identifié deux familles interdites, les cycles et les graphes Dumbo. Malheureusement, nous n'avons pas réussi à généraliser cette caractérisation pour les graphes série-parallèles de longueur arborescente $k \geq 3$. En effet, le nombre de familles de graphes interdits en tant que sous-graphes isométriques croit trop vite par rapport à k. Néanmoins, cela est peut-être faisable si nous arrivons à les décrire d'une façon plus synthétique. Il est peut-être aussi possible d'abandonner cette caractérisation en termes de sous-graphes isométriques pour développer un algorithme polynomial, paramétré par la taille de la solution. Bien sûr, nous ne sommes même pas sûrs que cela soit possible, c'est-à-dire, la complexité du calcul de la longueur arborescente des graphes série-parallèles est peut-être NP-difficile.

En ce qui concerne la longueur linéaire, rien (ou presque) n'était connu avant cette thèse à part la complexité de calcul de ce paramètre. Nous avons donc étudié les classes de graphes les plus simples pour ce paramètre, les arbres, les cycles et les graphes planaires extérieurs. Nous sommes parvenus à concevoir un algorithme d'approximation de facteur additif $+1$ pour les graphes planaires extérieurs. La suite logique serait donc de l'adapter pour obtenir un algorithme exact si cela est possible.

Nous avons aussi étudié le jeu DES CHASSEURS ET DU LAPIN. Dans l'objectif de résoudre le calcul du nombre de chasseurs des arbres, nous avons défini une notion de monotonie pour ce jeu. Cela nous a permis notamment de prouver que le nombre de chasseurs monotone des arbres est calculable en temps polynomial et que le nombre de chasseurs monotone est équivalent à la largeur linéaire ($pw(G) \leq mh(G) \leq pw(G) + 1$). Cela nous a permis de prouver que calculer $mh(G)$ est un problème NP-difficile. Nous avons aussi montré que ce nombre de chasseurs monotone peut être arbitrairement différent du nombre de chasseurs, et ce, même dans les arbres.

Pour espérer pouvoir calculer le nombre de chasseurs d'un arbre, il faudrait donc développer de nouvelles techniques (la définition de la monotonie est la technique classique). Le lemme de Parsons est peut-être adaptable pour le nombre de chasseurs des arbres, mais cette adaptation parait très compliquée. Nous avons aussi étudié d'autres classes de graphes simples, autre que les arbres, comme les graphes scindés, les graphes d'intervalles et les cographes. Nous avons caractérisé leur nombre de chasseurs monotone, et pour certains, nous avons caractérisé leur nombre de chasseurs. Finalement, nous avons prouvé que le nombre de chasseurs et le nombre de chasseurs monotone sont FPTpar $vc(G)$ en concevant des noyaux exponentiels.

Perspectives

En dehors de la suite directe de nos travaux, il reste beaucoup de questions intéressantes à étudier.

Longueur arborescente. Concernant la longueur arborescente, il serait intéressant d'étudier les algorithmes d'approximation. Par exemple, il serait intéressant de répondre à la question ouverte dans (Dourisboure & Gavoille, 2007) concernant l'algorithme $Disk - Tree$, "est-ce que cet algorithme termine pour $k \geq t\ell(G)$?". Notons aussi qu'il est théoriquement envisageable de concevoir des algorithmes d'approximation de ratio $\frac{3}{2}$ pour ce problème, mais aucun n'est connu avec un ratio inférieur à 3 dans le cas général ou dans les graphes planaires pour le moment.

De plus, la complexité du calcul de la longueur arborescente des graphes planaires reste ouverte. Rappelons aussi que nous sommes intéressés par la relation entre la largeur arborescente et la longueur arborescente. Il serait donc intéressant de concevoir des algorithmes permettant de transformer une décomposition arborescente de longueur bornée en décomposition arborescente de largeur bornée et vice versa.

Il pourrait aussi être intéressant de considérer la variante des Gendarmes et du Voleur équivalent à la largeur arborescente où nous évaluons les stratégies monotones par rapport à la distance entre les gendarmes. En effet, cette vision en termes de jeu de recherche semble équivalente à la longueur arborescente et pourrait donc nous permettre d'obtenir de nouveaux résultats, comme ce fut le cas pour la largeur arborescente des graphes avec la définition des buissons correspondant intuitivement à un ensemble de positions permettant aux fugitifs de s'échapper indéfiniment contre k gendarmes (Seymour & Thomas, 1993).

Une variante de la longueur arborescente pourrait aussi être très intéressante, sa variante connexe. Une décomposition arborescente $D = (T, \mathcal{X})$ de $G = (V, E)$ est connexe si pour toute arête e de l'arbre T, les sommets (de G) contenus respectivement dans les sacs de T_e^1 et dans les sacs de T_e^2, les deux arbres connexes de $T \setminus e$, induisent respectivement un sous-graphe connexe de G, c'est-à-dire, $G[\bigcup_{X \in V(T_e^1)} X]$ et $G[\bigcup_{X \in V(T_e^2)} X]$ sont connexes. En effet, il existe des exemples de graphes (les cycles) où la longueur arborescente et la longueur arborescente connexe sont différentes, mais la différence entre ces paramètres pourrait être assez faibles (possiblement un rapport $\frac{3}{2}$). De plus, cette variante connexe devrait être plus simple à calculer, puisque cela retire la complexité de trouver un bon séparateur dans un cycle. Par exemple, l'algorithme d'approximation de la longueur arborescente des graphes série-parallèles serait en fait un algorithme exact pour calculer la longueur arborescente connexe des graphes série-parallèles.

Longueur linéaire. Concernant la longueur linéaire, la suite directe, après les graphes planaires extérieurs, serait d'étudier les graphes série-parallèles. Malheureusement, au vu de nos résultats sur la longueur arborescente des graphes série-parallèles et de nos résultats sur la longueur linéaire des graphes planaires extérieurs, cela semble compliqué.

Concernant les autres perspectives pour la longueur linéaire, elles sont similaires à celles pour la longueur arborescente. Plus précisément, il serait intéressant de concevoir des algorithmes d'approximation pour la longueur linéaire de classes de graphes simples. De plus, la complexité du calcul de la longueur linéaire (ou de la longueur arborescente) des graphes planaires reste ouverte alors que la complexité du calcul de la largeur linéaire est NP-difficile dans les graphes planaires. Finalement, il pourrait être intéressant de définir une variante des Gendarmes et du Voleur où nous évaluons les stratégies par rapport à la distance entre les gendarmes, et une variante connexe de la longueur linéaire. Notons que cette fois-ci, nous n'avons pas d'exemple de graphes pour lesquels la longueur linéaire et la longueur linéaire connexe sont différentes.

Les chasseurs et le lapin. Concernant le jeu DES CHASSEURS ET DU LAPIN, la complexité du calcul du nombre de chasseurs reste un problème ouvert dans les graphes généraux, même si nous avons prouvé que calculer $mh(G)$ est NP-difficile. De plus, il serait intéressant de connaître le nombre de chasseurs pour d'autres classes de graphes que les arbres, par exemple, le nombre de chasseurs des graphes d'intervalles. De plus, il serait intéressant d'étudier la complexité paramétrée de ces deux jeux en considérant des paramètres comme la taille de la solution, la largeur arborescente ou encore la largeur linéaire.

Références

Abramovskaya, T. V., Fomin, F. V., Golovach, P. A., & Pilipczuk, M. (2016). How to hunt an invisible rabbit on a graph. *European Journal of Combinatorics*, *52*, 12–26. Consulté sur https://doi.org/10.1016/j.ejc.2015.08.002 doi: 10.1016/j.ejc.2015.08.002

Adler, I. (2004). Marshals, monotone marshals, and hypertree-width. *Journal of Graph Theory*, *47*(4), 275–296.

Adler, I. (2007). Directed tree-width examples. *J. Comb. Theory, Ser. B*, *97*(5), 718–725. Consulté sur https://doi.org/10.1016/j.jctb.2006.12.006 doi: 10.1016/j.jctb.2006.12.006

Amir, E. (2001). Efficient approximation for triangulation of minimum treewidth. In J. S. Breese & D. Koller (Eds.), *UAI '01 : Proceedings of the 17th conference in uncertainty in artificial intelligence, university of washington, seattle, washington, usa, august 2-5, 2001* (pp. 7–15). Morgan Kaufmann. Consulté sur https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=77&proceeding_id=17

Archdeacon, D. (1981). A kuratowski theorem for the projective plane. *Journal of Graph Theory*, *5*(3), 243-246. Consulté sur https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.3190050305 doi: https://doi.org/10.1002/jgt.3190050305

Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. of Discrete Mathematics*, *8*(2), 277–284. Consulté sur https://doi.org/10.1137/0608024 doi: 10.1137/0608024

Arnborg, S., & Proskurowski, A. (1989). Linear time algorithms for np-hard problems restricted to partial k-trees. *Discret. Appl. Math.*, *23*(1), 11–24. Consulté sur https://doi.org/10.1016/0166-218X(89)90031-0 doi: 10.1016/0166-218X(89)90031-0

Belmonte, R., Fomin, F. V., Golovach, P. A., & Ramanujan, M. S. (2017). Metric dimension of bounded tree-length graphs. *SIAM J. Discret. Math.*, *31*(2), 1217–1243. Consulté sur https://doi.org/10.1137/16M1057383 doi: 10.1137/16M1057383

Bienstock, D., & Seymour, P. D. (1991). Monotonicity in graph searching. *Journal of Algorithms*, *12*(2), 239–245. Consulté sur https://doi.org/10.1016/0196-6774(91)90003-H doi: 10.1016/0196-6774(91)90003-H

Blin, L., Burman, J., & Nisse, N. (2017). Exclusive graph searching. *Algorithmica*, *77*(3), 942–969. Consulté sur https://doi.org/10.1007/s00453-016-0124-0 doi: 10.1007/s00453-016-0124-0

Bodlaender, H. L. (1988). Some classes of graphs with bounded treewidth. *Bull. EATCS*, *36*, 116–125.

Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, *209*(1-2), 1–45. Consulté sur https://doi.org/10.1016/S0304-3975(97)00228-4 doi: 10.1016/S0304-3975(97)00228-4

Bodlaender, H. L., Cygan, M., Kratsch, S., & Nederlof, J. (2015). Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*,

243, 86–111. Consulté sur https://doi.org/10.1016/j.ic.2014.12.008 doi: 10.1016/j.ic.2014.12.008

Bodlaender, H. L., & Fomin, F. V. (2002). Approximation of pathwidth of outerplanar graphs. *J. Alg.*, *43*(2), 190–200.

Bodlaender, H. L., Fomin, F. V., Lokshtanov, D., Penninkx, E., Saurabh, S., & Thilikos, D. M. (2016). (Meta) Kernelization. *J. ACM*, *63*(5), 44 :1–44 :69. Consulté sur https://doi.org/10.1145/2973749 doi: 10.1145/2973749

Bodlaender, H. L., Gilbert, J. R., Hafsteinsson, H., & Kloks, T. (1995). Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, *18*(2), 238–255.

Bodlaender, H. L., & Kloks, T. (1996). Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, *21*(2), 358–402. Consulté sur https://doi.org/10.1006/jagm.1996.0049 doi: 10.1006/jagm.1996.0049

Bolkema, J., & Groothuis, C. (2019). Hunting rabbits on the hypercube. *Discrete Mathematics*, *342*(2), 360-372. Consulté sur https://www.sciencedirect.com/science/article/pii/S0012365X18303455 doi: https://doi.org/10.1016/j.disc.2018.10.011

Bonato, A., Finbow, S., Gordinowicz, P., Haidar, A., Kinnersley, W. B., Mitsche, D., ... Stacho, L. (2013). The robber strikes back. *CoRR*, *abs/1308.2843*. Consulté sur http://arxiv.org/abs/1308.2843

Bonato, A., & Nowakowski, R. (2011). The game of cops and robbers on graphs. 2011. *American Mathematical Society*.

Bouchitté, V., Kratsch, D., Müller, H., & Todinca, I. (2004). On treewidth approximations. *Discret. Appl. Math.*, *136*(2-3), 183–196. Consulté sur https://doi.org/10.1016/S0166-218X(03)00440-2 doi: 10.1016/S0166-218X(03)00440-2

Breisch, R. (1967). An intuitive approach to speleotopology. *Southwestern cavers*, *6*(5), 72–78.

Britnell, J. R., & Wildon, M. (2013). Finding a princess in a palace : a pursuit-evasion problem. *The Electronic Journal of Combinatorics*, *20*(1), 25. Consulté sur https://doi.org/10.37236/2296 doi: 10.37236/2296

Chapelle, M., Liedloff, M., Todinca, I., & Villanger, Y. (2017). Treewidth and pathwidth parameterized by the vertex cover number. *Discrete Applied Mathematics*, *216*, 114–129.

Chartrand, G., & Harary, F. (1967). Planar permutation graphs. *Annales De L Institut Henri Poincare-probabilites Et Statistiques*, *3*, 433-438.

Chekuri, C., & Chuzhoy, J. (2016). Polynomial bounds for the grid-minor theorem. *J. ACM*, *63*(5), 40 :1–40 :65. Consulté sur https://doi.org/10.1145/2820609 doi: 10.1145/2820609

Chimani, M., Mutzel, P., & Zey, B. (2012). Improved steiner tree algorithms for bounded treewidth. *Journal of Discrete Algorithms*, *16*, 67-78. Consulté sur https://www.sciencedirect.com/science/article/pii/S1570866712000858 (Selected papers from the 22nd International Workshop on Combinatorial Algorithms (IWOCA 2011)) doi: https://doi.org/10.1016/j.jda.2012.04.016

Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics : A survey. *Autonomous Robots*, *31*(299), 299–316.

Chuzhoy, J. (2016). Improved bounds for the excluded grid theorem. *CoRR*, *abs/1602.02629*. Consulté sur http://arxiv.org/abs/1602.02629

Chuzhoy, J., & Tan, Z. (2021). Towards tight(er) bounds for the excluded grid theorem. *J. Comb. Theory, Ser. B*, *146*, 219–265. Consulté sur https://doi.org/10.1016/j.jctb.2020.09.010 doi: 10.1016/j.jctb.2020.09.010

Corneil, D. G., Perl, Y., & Stewart, L. K. (1985). A linear recognition algorithm for cographs. *SIAM Journal on Computing*, *14*(4), 926-934.

Coudert, D., & Ducoffe, G. (2018). Revisiting decomposition by clique separators. *SIAM J. Discret. Math.*, *32*(1), 682–694. Consulté sur https://doi.org/10.1137/16M1059837 doi: 10.1137/16M1059837

Coudert, D., Ducoffe, G., & Nisse, N. (2016). To approximate treewidth, use treelength! *SIAM J. Discret. Math.*, *30*(3), 1424–1436. Consulté sur https://doi.org/10.1137/15M1034039 doi: 10.1137/15M1034039

Coudert, D., Huc, F., & Sereni, J.-S. (2007). Pathwidth of outerplanar graphs. *J. Graph Theory*, *55*(1), 27–41.

Courcelle, B., & Mosbah, M. (1993). Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, *109*(1&2), 49–82. Consulté sur https://doi.org/10.1016/0304-3975(93)90064-Z doi: 10.1016/0304-3975(93)90064-Z

Crytser, D., Komarov, N., & Mackey, J. (2020). Containment : A variation of cops and robber. *Graphs Comb.*, *36*(3), 591–605. Consulté sur https://doi.org/10.1007/s00373-020-02140-5 doi: 10.1007/s00373-020-02140-5

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., ... Saurabh, S. (2015a). *Parameterized algorithms*. Springer. Consulté sur https://doi.org/10.1007/978-3-319-21275-3 doi: 10.1007/978-3-319-21275-3

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., ... Saurabh, S. (2015b). *Parameterized algorithms* (1st éd.). Springer Publishing Company, Incorporated.

Demaine, E. D., Fomin, F. V., Hajiaghayi, M. T., & Thilikos, D. M. (2004). Bidimensional parameters and local treewidth. *SIAM J. Discret. Math.*, *18*(3), 501–511. Consulté sur https://doi.org/10.1137/S0895480103433410 doi: 10.1137/S0895480103433410

Demaine, E. D., & Hajiaghayi, M. (2008). The bidimensionality theory and its algorithmic applications. *Comput. J.*, *51*(3), 292–302. Consulté sur https://doi.org/10.1093/comjnl/bxm033 doi: 10.1093/comjnl/bxm033

Demaine, E. D., Hajiaghayi, M., & Thilikos, D. M. (2006). The bidimensional theory of bounded-genus graphs. *SIAM J. Discret. Math.*, *20*(2), 357–371. Consulté sur https://doi.org/10.1137/040616929 doi: 10.1137/040616929

Demaine, E. D., Hajiaghayi, M. T., Nishimura, N., Ragde, P., & Thilikos, D. M. (2004). Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. *J. Comput. Syst. Sci.*, *69*(2), 166–195. Consulté sur https://doi.org/10.1016/j.jcss.2003.12.001 doi: 10.1016/j.jcss.2003.12.001

Dieng, Y., & Gavoille, C. (2009). On the tree-width of planar graphs. *Electron. Notes Discret. Math.*, *34*, 593–596. Consulté sur https://doi.org/10.1016/j.endm.2009.07.099 doi: 10.1016/j.endm.2009.07.099

Diestel, R. (2012). *Graph theory, 4th edition* (Vol. 173). Springer.

Dinneen, M. J. (1995). *Vlsi layouts and dna physical mappings.*

Dirac, G. A. (1952). A property of 4-chromatic graphs and some remarks on critical graphs. *Journal of The London Mathematical Society-second Series*, 85-92.

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021a, septembre). Longueur Arborescente des Graphes Série-Parallèles. In *ALGOTEL 2021 - 23èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications.* La Rochelle, France. Consulté sur https://hal.science/hal-03217731

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021b). Treelength of series-parallel graphs. In C. E. Ferreira, O. Lee, & F. K. Miyazawa (Eds.), *Proceedings of the XI latin and american algorithms, graphs and optimization symposium, LAGOS 2021, online event / são paulo, brazil, may 2021* (Vol. 195, pp. 30–38). Elsevier. Consulté sur https://doi.org/10.1016/j.procs.2021.11.008 doi: 10.1016/j.procs.2021.11.008

Dissaux, T., Ducoffe, G., Nisse, N., & Nivelle, S. (2021c). Treelength of series-parallel graphs. *Discrete Applied Mathematics (accepté).*

Dissaux, T., Fioravantes, F., Galhawat, H., & Nisse, N. (2023, février). *Further results on the Hunters and Rabbit game through monotonicity* (Rapport technique). Inria - Sophia Antipolis. (soumis à une conférence internationale). Consulté sur https://hal.science/hal-03995642

Dissaux, T., & Nisse, N. (2022a). Longueur linéaire des graphes planaires extérieurs. In *ICGT 2022 - The 11th International Colloquium on Graph Theory and combinatorics.* Montpellier, France.

Dissaux, T., & Nisse, N. (2022b, mai). Longueur linéaire des graphes planaires extérieurs. In *AlgoTel 2022 - 24èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications.* Saint-Rémy-Lès-Chevreuse, France. Consulté sur https://hal.science/hal-03655647

Dissaux, T., & Nisse, N. (2022c). Pathlength of outerplanar graphs. In A. Castañeda & F. Rodríguez-Henríquez (Eds.), *LATIN 2022 : Theoretical informatics - 15th latin american symposium, guanajuato, mexico, november 7-11, 2022, proceedings* (Vol. 13568, pp. 172–187). Springer. (soumis à une revue internationale). Consulté sur https://doi.org/10.1007/978-3-031-20624-5_11 doi: 10.1007/978-3-031-20624-5_11

Dourisboure, Y., & Gavoille, C. (2007). Tree-decompositions with bags of small diameter. *Discret. Math.*, *307*(16), 2008–2029. Consulté sur https://doi.org/10.1016/j.disc.2005.12.060 doi: 10.1016/j.disc.2005.12.060

Dragan, F. F., & Köhler, E. (2014). An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs. *Algorithmica*, *69*(4), 884–905. Consulté sur https://doi.org/10.1007/s00453-013-9765-4 doi: 10.1007/s00453-013-9765-4

Dragan, F. F., Köhler, E., & Leitert, A. (2017). Line-distortion, bandwidth and path-length of a graph. *Algorithmica*, *77*(3), 686–713.

Dragan, F. F., & Leitert, A. (2016). Minimum eccentricity shortest paths in some structured graph classes. *J. Graph Algorithms Appl.*, *20*(2), 299–322.

Ducoffe, G., Legay, S., & Nisse, N. (2020). On the complexity of computing treebreadth. *Algorithmica*, *82*(6), 1574–1600. Consulté sur https://doi.org/10.1007/s00453-019-00657-7 doi: 10.1007/s00453-019-00657-7

Duffin, R. (1965). Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, *10*(2), 303-318. Consulté sur https://www.sciencedirect.com/science/article/pii/0022247X65901253 doi: https://doi.org/10.1016/0022-247X(65)90125-3

Ellis, J. A., Sudborough, I. H., & Turner, J. S. (1994). The vertex separation and search number of a graph. *Information and Computation*, *113*(1), 50–79.

Eppstein, D. (1992). Parallel recognition of series-parallel graphs. *Inf. Comput.*, *98*(1), 41–55. Consulté sur https://doi.org/10.1016/0890-5401(92)90041-D doi: 10.1016/0890-5401(92)90041-D

Fedorov, R. V., Levy, S., Kovaldzhi, A., & Yashchenko, I. (2011). *Moscow mathematical olympiads, 2000-2005* (Vol. 7). American Mathematical Soc.

Feige, U., Hajiaghayi, M., & Lee, J. R. (2008). Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, *38*(2), 629–657. Consulté sur https://doi.org/10.1137/05064299X doi: 10.1137/05064299X

Fellows, M. R., & Langston, M. A. (1994). On search, decision, and the efficiency of polynomial-time algorithms. *Journal of Computer and System Sciences*, *49*(3), 769-779. Consulté sur https://www.sciencedirect.com/science/article/pii/S0022000005800790 (30th IEEE Conference on Foundations of Computer Science) doi: https://doi.org/10.1016/S0022-0000(05)80079-0

Giannopoulou, A. C., Hunter, P., & Thilikos, D. M. (2012). Lifo-search : A min-max theorem and a searching game for cycle-rank and treedepth. *Discrete Applied Mathematics*, *160*(15), 2089–2097.

Gottlob, G., Leone, N., & Scarcello, F. (2003). Robbers, marshals, and guards : game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, *66*(4), 775–808.

Groenland, C., Joret, G., Nadara, W., & Walczak, B. (2023). Approximating pathwidth for graphs of small treewidth. *ACM Trans. Algorithms*, *19*(2), 16 :1–16 :19. Consulté sur https://doi.org/10.1145/3576044 doi: 10.1145/3576044

Gruslys, V., & Méroueh, A. (2015). Catching a mouse on a tree. *arXiv preprint arXiv :1502.06591*.

Gustedt, J. (1993). On the pathwidth of chordal graphs. *Discret. Appl. Math.*, *45*(3), 233–248. Consulté sur https://doi.org/10.1016/0166-218X(93)90012-D doi: 10.1016/0166-218X(93)90012-D

Gustedt, J., Mæhle, O., & Telle, J. (2002, 01). The treewidth of java programs. In (Vol. 2409, p. 86-97). doi: 10.1007/3-540-45643-0_7

Halin, R. (1976). S-functions for graphs. *Journal of Geometry*, *8*, 171-186. Consulté sur https://doi.org/10.1007/BF01917434 doi: 10.1007/BF01917434

Hammer, P. L., & Simeone, B. (1981). The splittance of a graph. *Combinatorica*, *1*, 275–284.

Haslegrave, J. (2014). An evasion game on a graph. *Discrete Mathematics*, *314*, 1–5.

Herlihy, M., Kuhn, F., Tirthapura, S., & Wattenhofer, R. (2006). Dynamic analysis of the arrow distributed protocol. *Theory Comput. Syst.*, *39*(6), 875–901.

Hopcroft, J. E., & Tarjan, R. E. (1973). Dividing a graph into triconnected components. *SIAM J. Comput.*, *2*(3), 135–158.

Ilcinkas, D., Nisse, N., & Soguet, D. (2009). The cost of monotonicity in distributed graph searching. *Distributed Computing*, *22*(2), 117–127. Consulté sur https://doi.org/10.1007/s00446-009-0089-1 doi: 10.1007/s00446-009-0089-1

Indyk, P. (2001). Algorithmic applications of low-distortion geometric embeddings. In *42nd annual symposium on foundations of computer science, FOCS* (pp. 10–33). IEEE.

Isaza, A., Lu, J., Bulitko, V., & Greiner, R. (2008). A cover-based approach to multi-agent moving target pursuit. In *proceedings of the fourth artificial intelligence and interactive digital entertainment conference* (pp. 54–59). AAAI Press.

Johnson, T., Robertson, N., Seymour, P. D., & Thomas, R. (2001). Directed tree-width. *Journal of Combinatorial Theory Series B*, *82*(1), 138–154.

Kammer, F. (2012). *Treelike and chordal graphs : Algorithms and generalizations* (Thèse de doctorat, University of Augsburg). Consulté sur http://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/1606

Kammer, F., & Tholey, T. (2016). Approximate tree decompositions of planar graphs in linear time. *Theor. Comput. Sci.*, *645*, 60–90. Consulté sur https://doi.org/10.1016/j.tcs.2016.06.040 doi: 10.1016/j.tcs.2016.06.040

Kawarabayashi, K., & Kobayashi, Y. (2012). Linear min-max relation between the treewidth of h-minor-free graphs and its largest grid. In C. Dürr & T. Wilke (Eds.), *29th international symposium on theoretical aspects of computer science, STACS 2012, february 29th - march 3rd, 2012, paris, france* (Vol. 14, pp. 278–289). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Consulté sur https://doi.org/10.4230/LIPIcs.STACS.2012.278 doi: 10.4230/LIPIcs.STACS.2012.278

Kinnersley, N. G. (1992). The vertex separation number of a graph equals its path-width. *Inf. Process. Lett.*, *42*(6), 345–350. Consulté sur https://doi.org/10.1016/0020-0190(92)90234-M doi: 10.1016/0020-0190(92)90234-M

Kirousis, L. M., & Papadimitriou, C. H. (1986). Searching and pebbling. *Theor. Comput. Sci.*, *47*(3), 205–218. Consulté sur https://doi.org/10.1016/0304-3975(86)90146-5 doi: 10.1016/0304-3975(86)90146-5

Korhonen, T. (2021). Single-exponential time 2-approximation algorithm for treewidth. *CoRR*, *abs/2104.07463*. Consulté sur https://arxiv.org/abs/2104.07463

Korhonen, T., & Lokshtanov, D. (2022). An improved parameterized algorithm for treewidth. *CoRR*, *abs/2211.07154*. Consulté sur https://doi.org/10.48550/arXiv.2211.07154 doi: 10.48550/arXiv.2211.07154

Kornai, A., & Tuza, Z. (1992). Narrowness, pathwidth, and their application in natural language processing. *Discrete Applied Mathematics*, *36*(1), 87-92. Consulté sur https://www.sciencedirect.com/science/article/pii/0166218X9290208R doi: https://doi.org/10.1016/0166-218X(92)90208-R

Kosowski, A., Li, B., Nisse, N., & Suchan, K. (2015). k-Chordal graphs : From cops and robber to compact routing via treewidth. *Algorithmica*, *72*(3), 758–777. Consulté sur https://doi.org/10.1007/s00453-014-9871-y doi: 10.1007/s00453-014-9871-y

Krauthgamer, R., & Lee, J. R. (2006). Algorithms on negatively curved spaces. In *47th annual IEEE symposium on foundations of computer science (FOCS 2006), 21-24 october 2006, berkeley, california, usa, proceedings* (pp. 119–132). IEEE Computer Society. Consulté sur https://doi.org/10.1109/FOCS.2006.9 doi: 10.1109/FOCS.2006.9

Kuratowski, C. (1930). Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, *15*(1), 271-283. Consulté sur http://eudml.org/doc/212352

LaPaugh, A. S. (1993). Recontamination does not help to search a graph. *Journal of the ACM*, *40*(2), 224–245. Consulté sur https://doi.org/10.1145/151261.151263 doi: 10.1145/151261.151263

Leaf, A., & Seymour, P. D. (2015). Tree-width and planar minors. *J. Comb. Theory, Ser. B*, *111*, 38–53. Consulté sur https://doi.org/10.1016/j.jctb.2014.09.003 doi: 10.1016/j.jctb.2014.09.003

Lokshtanov, D. (2010). On the complexity of computing treelength. *Discret. Appl. Math.*, *158*(7), 820–827. Consulté sur https://doi.org/10.1016/j.dam.2009.10.007 doi: 10.1016/j.dam.2009.10.007

Markou, E., Nisse, N., & Pérennes, S. (2017). Exclusive graph searching vs. pathwidth. *Inf. Comput.*, *252*, 243–260. Consulté sur https://doi.org/10.1016/j.ic.2016.11.007 doi: 10.1016/j.ic.2016.11.007

Matoušek, J., & Thomas, R. (1992). On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, *108*(1), 343-364. Consulté sur https://www.sciencedirect.com/science/article/pii/0012365X9290687B doi: https://doi.org/10.1016/0012-365X(92)90687-B

Mazoit, F., & Nisse, N. (2008). Monotonicity of non-deterministic graph searching. *Theoretical Computer Science*, *399*(3), 169–178. Consulté sur https://doi.org/10.1016/j.tcs.2008.02.036 doi: 10.1016/j.tcs.2008.02.036

Monien, B., & Sudborough, I. H. (1988). Min cut is NP-complete for edge weighted trees. *Theor. Comput. Sci.*, *58*, 209–229.

Nisse, N. (2019). Network decontamination. In P. Flocchini, G. Prencipe, & N. Santoro (Eds.), *Distributed computing by mobile entities, current research in moving and computing* (Vol. 11340, pp. 516–548). Springer. Consulté sur https://doi.org/10.1007/978-3-030-11072-7_19 doi: 10.1007/978-3-030-11072-7_19

Parra, A., & Scheffler, P. (1997). Characterizations and algorithmic applications of chordal graph embeddings. *Discret. Appl. Math.*, *79*(1-3), 171–188. Consulté sur https://doi.org/10.1016/S0166-218X(97)00041-3 doi: 10.1016/S0166-218X(97)00041-3

Parsons, T. D. (1978). Pursuit-evasion in a graph. In *Theory and applications of graphs, lncs* (Vol. 642, pp. 426–441). Springer.

Robertson, N., & Seymour, P. D. (1986a). Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, *7*(3), 309–322. Consulté sur https://doi.org/10.1016/0196-6774(86)90023-4 doi: 10.1016/0196-6774(86)90023-4

Robertson, N., & Seymour, P. D. (1986b). Graph minors. v. excluding a planar graph. *J. Comb. Theory, Ser. B*, *41*(1), 92–114. Consulté sur https://doi.org/10.1016/0095-8956(86)90030-4 doi: 10.1016/0095-8956(86)90030-4

Robertson, N., & Seymour, P. D. (1991). Graph minors. x. obstructions to tree-decomposition. *J. Comb. Theory, Ser. B*, *52*(2), 153–190. Consulté sur https://doi.org/10.1016/0095-8956(91)90061-N doi: 10.1016/0095-8956(91)90061-N

Robertson, N., & Seymour, P. D. (1995). Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, *63*(1), 65–110. Consulté sur https://doi.org/10.1006/jctb.1995.1006 doi: 10.1006/jctb.1995.1006

Robertson, N., & Seymour, P. D. (2004). Graph minors. XX. wagner's conjecture. *J. Comb. Theory, Ser. B*, *92*(2), 325–357. Consulté sur https://doi.org/10.1016/j.jctb.2004.08.001 doi: 10.1016/j.jctb.2004.08.001

Robertson, N., Seymour, P. D., & Thomas, R. (1994). Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, *62*(2), 323–348. Consulté sur https://doi.org/10.1006/jctb.1994.1073 doi: 10.1006/jctb.1994.1073

Seymour, P. D., & Thomas, R. (1993). Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, *58*(1), 22–33.

Seymour, P. D., & Thomas, R. (1994). Call routing and the ratcatcher. *Comb.*, *14*(2), 217–241. Consulté sur https://doi.org/10.1007/BF01215352 doi: 10.1007/BF01215352

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319-2323.

Thilikos, D. M. (2000). Algorithms and obstructions for linear-width and related search parameters. *Discrete Applied Mathematics*, *105*(1), 239-271. Consulté sur https://www.sciencedirect.com/science/article/pii/S0166218X0000175X doi: https://doi.org/10.1016/S0166-218X(00)00175-X

Thorup, M. (1998). All structured programs have small tree width and good register allocation. *Information and Computation*, *142*(2), 159-181. Consulté sur https://www.sciencedirect.com/science/article/pii/S0890540197926973 doi: https://doi.org/10.1006/inco.1997.2697

Valdes, J., Tarjan, R. E., & Lawler, E. L. (1982). The recognition of series parallel digraphs. *SIAM J. Comput.*, *11*(2), 298–313. Consulté sur https://doi.org/10.1137/0211023 doi: 10.1137/0211023

Wagner, K. V. (1937). Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, *114*, 570-590.

Williamson, D. P., & Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge University Press.

Yamaguchi, A., & Aoki-Kinoshita, K. (2014, 11). Chemical compound complexity in biological pathways. In (p. 471-493). doi: 10.1201/b17645-17

Yang, B., Dyer, D., & Alspach, B. (2009). Sweeping graphs with large clique number. *Discrete Mathematics*, *309*(18), 5770–5780. Consulté sur https://doi.org/10.1016/j.disc.2008.05.033 doi: 10.1016/j.disc.2008.05.033

Décomposition de graphes: longueur arborescente et jeux de poursuite

Thomas DISSAUX

Résumé

Les décompositions de graphes sont un outil permettant de représenter un graphe en plusieurs parties, appelées sacs, et structurées comme un arbre ou un chemin suivant si ce sont des décompositions arborescentes ou linéaires. Ces décompositions permettent de résoudre certains problèmes NP-difficiles en temps linéaire, si la taille maximum des sacs (i.e. la « largeur/width ») est bornée. Cela a motivé les travaux de ces 30 dernières années sur la largeur arborescente d'un graphe G (la plus petite largeur des décompositions arborescente de G). Cependant, il reste encore beaucoup de questions ouvertes comme la complexité du calcul de la largeur arborescente des graphes planaires. Pour pouvoir répondre à cette question, il peut être intéressant d'étudier une autre mesure des décompositions, la longueur. Cette mesure correspond au diamètre maximum des sacs d'une décomposition, et il a été prouvé qu'il existe une relation entre la longueur arborescente et la largeur arborescente dans les graphes planaires.

Nous nous intéressons donc dans le chapitre 2 à la longueur arborescente de classes de graphes planaires simples, comme les graphes série-parallèles. Nous explicitons une liste infinie de sous-graphes isométriques interdits pour les graphes série-parallèles de longueur arborescente 2. Grâce à cette liste, il est alors possible en temps polynomial de tester si un graphe série-parallèle a une longueur arborescente au plus 2 et, dans le cas d'une réponse positive, de calculer une décomposition arborescente optimale.

Nous nous intéressons aussi au cas de la longueur linéaire dans le chapitre 3. Nous nous focalisons sur les classes des arbres et des cycles pour lesquelles nous caractérisons la longueur linéaire. Nous nous intéressons aussi à la longueur linéaire des graphes planaires extérieurs et concevons un algorithme d'approximation de facteur additif 1.

Finalement, dans le chapitre 4, nous nous intéressons à une variante algorithmique des décompositions linéaires des graphes, par le biais d'un jeu de poursuite-évasion, le jeu *des Chasseurs et du Lapin*. Dans ce jeu, un groupe de chasseurs traque un lapin invisible qui est forcé de bouger à chaque étape sur une position voisine. Nous nous intéressons au nombre minimum $h(G)$ de chasseurs qui peuvent attraper le lapin quoi qu'il fasse sur un graphe G. Ce jeu a notamment été étudié dans le cas des graphes bipartis (grilles, arbres, hypercubes...) mais reste ouvert dans beaucoup de classes de graphes et notamment dans les arbres. Une notion très utile pour le calcul de stratégie dans les jeux de poursuite-évasion est la notion de monotonie. Nous définissons une variante monotone du jeu des chasseurs et du lapin, nous permettant entre autres, de prouver que, dans cette variante, le nombre minimum $mh(G)$ de chasseurs diffère d'au plus un de la largeur linéaire du graphe G. Ce résultat a d'importantes conséquences, comme le fait que le calcul de $mh(G)$ est NP-difficile. Nous caractérisons aussi $mh(G)$ pour plusieurs classes de graphes, comme les graphes scindés, les graphes d'intervalles, les arbres et les cographes. Nous étudions la différence entre mh et h dans ces classes de graphes et, en particulier, nous montrons qu'il peut exister une différence arbitrairement grande entre mh et h dans les arbres.

Mots-clés : Décomposition arborescente, Décomposition linéaire, Longueur arborescente, Longueur linéaire, Graphe planaire, Jeu des Chasseurs et du Lapin.