Coloration d'arêtes ℓ-distance et clustering : études et algorithmes auto-stabilisants

Kaouther Drira

Numéro d'ordre : 335 - 2010 Année 2010

THÈSE

Coloration d'arêtes ℓ-distance et Clustering : Etudes et Algorithmes auto-stabilisants

présentée devant
L'Université de Lyon

pour obtenir
Le grade de docteur

École doctorale Informatique et Mathématiques (InfoMaths)
Spécialité : Informatique

par
Kaouther DRIRA

Soutenue le 14 Décembre 2010 devant la commission d'examen

COMPOSITION DU JURY

Rapporteurs	M. Raphaël Couturier	Professeur–Université de Franche Comté
	M. Jean Pallo	Professeur–Université de Bourgogne
Examinateurs	Mme. Salima Hassas	Professeur–U.C.B. Lyon 1
	M. Lhouari Nourine	Professeur–Université Blaise Pascal
Directeurs	M. Hamamache Kheddouci	Professeur–U.C.B. Lyon 1
	Mme. Hamida Seba	Maître de Conférence–U.C.B. Lyon 1

Laboratoire d'Informatique pour l'Entreprise et les Systèmes de Production
LIESP, EA 4125

« *Celui qui déplace la montagne, c'est*
celui qui commence à enlever
les petites pierres.»
Confucius

Résumé

La coloration de graphes est un problème central de l'optimisation combinatoire. C'est un domaine très attractif par ses nombreuses applications. Différentes variantes et généralisations du problème de la coloration de graphes ont été proposées et étudiées. La coloration d'arêtes d'un graphe consiste à attribuer une couleur à chaque arête du graphe de sorte que deux arêtes ayant un sommet commun n'ont jamais la même couleur, le tout en utilisant le moins de couleurs possibles. Dans la première partie de cette thèse, nous étudions le problème de la coloration d'arêtes ℓ-distance, qui est une généralisation de la coloration d'arêtes classique. Nous menons une étude combinatoire et algorithmique du paramètre. L'étude porte sur les classes de graphes suivantes : les chaînes, les grilles, les hypercubes, les arbres et des graphes puissances. Le paramètre de la coloration d'arêtes ℓ-distance permet de modéliser des problèmes dans des réseaux assez grands. Cependant, avec la multiplication du nombre de nœuds, les réseaux sont de plus en plus vulnérables aux défaillances (ou pannes). Dans la deuxième partie, nous nous intéressons aux algorithmes tolérants aux pannes et en particulier les algorithmes auto-stabilisants. Nous proposons un algorithme auto-stabilisant pour la coloration propre d'arêtes. Notre solution se base sur le résultat de Vizing pour utiliser un minimum de couleurs possibles. Par la suite, nous proposons un algorithme auto-stabilisant de clustering destiné à des applications dans le domaine de la sécurité dans les réseaux mobiles ad hoc. La solution que nous proposons est un partitionnement en clusters basé sur les relations de confiance qui existent entre nœuds. Nous proposons aussi un algorithme de gestion de clés de groupe dans les réseaux mobiles ad hoc qui s'appuie sur la topologie de clusters préalablement construite. La sécurité de notre protocole est renforcée par son critère de clustering qui surveille en permanence les relations de confiance et expulse les nœuds malveillants de la session de diffusion.

Mots–Clés

Coloration d'arêtes – Algorithmes – Auto-stabilisation – Clustering – Réseaux mobiles ad hoc – Gestion de clés.

Abstract

Graph coloring is a famous combinatorial optimization problem and is very attractive for its numerous applications. Many variants and generalizations of the graph-coloring problem have been introduced and studied. An edge-coloring assigns a color to each edge so that no two adjacent edges share the same color. In the first part of this thesis, we study the problem of the ℓ-distance-edge-coloring, which is a generalization of the classical edge-coloring. The study focuses on the following classes of graphs : paths, grids, hypercubes, trees and some power graphs. We are conducting a combinatorial and algorithmic study of the parameter. We give a sequential coloring algorithm for each class of graph. The ℓ-distance-edge-coloring is especially considered in large-scale networks. However, with the increasing number of nodes, networks are increasingly vulnerable to faults. In the second part, we focus on fault-tolerant algorithms and in particular self-stabilizing algorithms. We propose a self-stabilizing algorithm for proper edge-coloring. Our solution is based on Vizing's result to minimize number of colors. Subsequently, we propose a self-stabilizing clustering algorithm for applications in the field of security in mobile ad hoc networks. Our solution is a partitioning into clusters based on trust relationships between nodes. We also propose a group key-management algorithm in mobile ad hoc networks based on the topology of clusters previously built. The security of our protocol is strengthened by its clustering criterion which constantly monitors trust relationships and expels malicious nodes out of the multicast session.

Key–Words

Edge-coloring – Algorithms – Self-stabilization – Clustering – Mobiles ad hoc networks – Key-management.

Table des matières

Liste des Algorithmes

Table des figures

Chapitre 1

Introduction

La théorie des graphes trouve différentes applications dans de nombreux domaines tels que la chimie, la biologie, les réseaux de télécommunications ou encore les réseaux sociaux. Les recherches en théorie des graphes sont essentiellement menées par des informaticiens du fait de l'importance des aspects algorithmiques (recherche de solutions). Il s'agit essentiellement de modéliser des problèmes. Les graphes constituent ainsi des outils de modélisation importants et très utilisés en informatique. On peut ainsi ramener un problème concret et complexe à un modèle mathématique plus clairement posé qui consiste en l'étude de sommets et arêtes (ou arcs). Un graphe $G = (V, E)$ est essentiellement défini par une relation binaire $E \in V \times V$ sur un ensemble V le plus souvent fini.

Malgré la simplicité apparente de leur définition, les graphes capturent une large part de la complexité algorithmique. Il est ainsi très important de bien comprendre la structure des graphes afin d'utiliser des modélisations pertinentes à base de graphes (i.e. des modélisations sur lesquelles les algorithmes de résolution sont efficaces).

La théorie des graphes constitue une branche à part entière des mathématiques, grâce aux travaux de Cayley, Menger, König, Berge, Erdös, etc. Son histoire débute avec les travaux d'Euler au $XVIII^{ième}$ siècle et trouve son origine dans l'étude de certains problèmes, tels que celui des ponts de Königsberg (est ce qu'il est possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir à son point de départ). Euler avait formulé qu'un graphe n'est *Eulérien* (*on peut distinguer une extrémité initiale et une extrémité finale de chaque arête, et ordonner l'ensemble des arêtes du graphe de telle façon que l'extrémité finale d'une arête corresponde à l'extrémité initiale de l'arête qui lui succède dans l'ordre*) si et seulement si chaque sommet a un nombre pair d'arêtes. Au milieu du $XIX^{ième}$ siècle, le mathématicien britannique Arthur Cayley s'est intéressé aux arbres, qui représentent un type particulier de graphe

n'ayant pas de cycle. Cayley a démontré que le nombre d'arbres que l'on peut construire sur n ($n > 1$) sommets numérotés est égal à n^{n-2}. En 1927, Menger a montré que pour toute paire de sommets u et v non adjacents d'un graphe connexe G, la taille minimum d'un sous-ensemble de sommets qui sépare u et v est égale au nombre maximum de chemins deux à deux indépendants joignant u et v. Un des premiers ouvrages, traitant de la théorie des graphes "*Theorie der endlichen und unendlichen Graphen*" écrit par König remonte à 1936. König est sans doute à l'origine de l'utilisation du terme *graphe* pour désigner ce qui était préalablement considéré comme un ensemble de points et de flèches. Parallèlement, un effort de synthèse important a été opéré en particulier par Claude Berge dans son ouvrage "*Théorie des graphes et ses applications*" publié en 1957, dans lequel il introduit une théorie des graphes unifiée et abstraite rassemblant de nombreux résultats de la littérature.

Le problème de coloration de graphes est une des origines de la théorie des graphes. Il remonte au $XIX^{ième}$ siècle lorsque Guthrie a remarqué que quatre couleurs étaient suffisantes pour colorer la carte d'Angleterre, sans donner la même couleur à deux régions ayant une frontière commune. Il a posé alors la question de savoir si quatre couleurs suffisaient toujours pour colorer n'importe quelle carte géographique de sorte que deux régions voisines n'aient pas la même couleur.

La coloration de graphes est un problème central de l'optimisation combinatoire, c'est un domaine très attractif par ses nombreuses applications. La coloration de graphes permet de modéliser de nombreux problèmes réels tels que l'allocation de fréquences, la diffusion dans les réseaux, le partage des ressources, le transfert de fichiers, etc. Un graphe peut être coloré de différentes manières. Nous pouvons colorer différents éléments d'un graphe : les sommets, les arêtes, une combinaison de ces éléments, des sous-structures, etc. A ceci peuvent s'ajouter différentes contraintes pour obtenir ainsi de nouveaux paramètres de coloration. Beaucoup de bornes ou de valeurs exactes des paramètres de coloration de graphes restent à trouver et des algorithmes satisfaisant ces paramètres restent à élaborer. Le fait que les problèmes de coloration soient NP-complets rend la tâche encore plus dure. Cependant, ces derniers peuvent être approchés en temps polynomial dans certaines classes de graphes. Dans ce cadre, nous nous sommes intéressés au paramètre de la coloration d'arêtes. Une coloration des arêtes d'un graphe G est une affectation de couleurs aux arêtes telles que les arêtes ayant une extrémité en commun sont de couleurs différentes. On cherche généralement à déterminer une coloration utilisant aussi peu de couleurs que possible. Le problème de la coloration d'arêtes a été posé en 1880 en relation avec la conjecture des quatre couleurs. Le premier article qui traite le problème de la coloration d'arêtes a été écrit par Tait en 1889. Dans cet article Tait a prouvé que le problème des quatre couleurs est équivalent au problème de la

coloration d'arêtes des graphes planaires cubiques avec trois couleurs. De nombreux résultats de coloration d'arêtes sont aujourd'hui connus. En particulier, Vizing a montré en 1964 que tout graphe non orienté de degré au plus Δ, admet une $(\Delta+1)$-coloration propre d'arêtes. Ce résultat montre que l'ensemble des graphes de degré au plus Δ peut être partitionné en deux classes : ceux dont l'indice chromatique est Δ (graphes de classe 1) et ceux dont l'indice chromatique est $\Delta + 1$ (graphes de classe 2). Certaines familles de graphes ont pu être classées, mais le problème général de classification des graphes simples quelconques est NP-complet.

Une généralisation de la coloration d'arêtes, la coloration d'arêtes ℓ-distance, $\ell \geq 0$, consiste à assigner une couleur de 1 à k à chaque arête, de sorte que chaque paire d'arêtes distantes d'au plus ℓ ne partagent pas la même couleur. Ce paramètre de coloration fait l'objet de la première partie de ce manuscrit.

Colorer un graphe avec le minimum de couleurs est un problème NP-complet. Par conséquent, le problème de la coloration d'arêtes ℓ-distance est lui aussi NP-complet et la recherche d'un algorithme polynomial résolvant le problème dans le cas le plus général semble difficile. Nous avons ainsi restreint notre étude à certaines classes de graphes. Pour chaque classe de graphe, nous avons défini un sous-graphe tel que les arêtes qui appartiennent à ce sous-graphe sont à distance au plus ℓ. Ce sous-graphe nous permet de déterminer la valeur de la borne inférieure qui est égale à son nombre d'arêtes. Les classes de graphes qui nous ont intéressées pour ce travail sont les chaînes, les grilles, les hypercubes, les arbres et les graphes puissances de certaines classes de graphes. Nous avons également mené une étude algorithmique permettant de donner un algorithme de coloration séquentiel pour chacune des classes de graphes étudiées. Le nombre de couleurs utilisées dépend du nombre d'arêtes du sous-graphe précédemment défini.

Le paramètre de la coloration d'arêtes ℓ-distance permet de modéliser des problèmes dans des réseaux assez grands. Cependant, avec la multiplication du nombre de sommets, il devient de plus en plus difficile de concevoir des algorithmes séquentiels. En effet, les sommets peuvent être géographiquement très éloignés les uns des autres. Par exemple, ils peuvent être situés dans des pays différents et être inter-connectés par un réseau comme internet. De plus, chaque sommet possède sa propre mémoire locale et échange de l'information avec les autres sommets à l'aide de messages envoyés via le réseau les connectant. Des pannes peuvent survenir et perturber ainsi le fonctionnement du réseau. L'auto-stabilisation permet de fournir un service qui ne pourrait pas être réalisé par un seul sommet en terme de fonctionnalité, de disponibilité, de temps de réponse, de fiabilité et de garantir qu'après l'occurrence de toute panne, le réseau retrouve un fonctionnement (ou comportement) correct en un temps fini. Les algorithmes auto-stabilisants se sont révélés être des solutions plus réalistes face aux problèmes de pannes transitoires. Le concept de

l'auto-stabilisation a été défini par Edsger Dijkstra en 1974. Pendant neuf ans, ce résultat resta marginal jusqu'à une intervention de Leslie Lamport qui fit remarquer l'importance et la puissance potentielle que présentait l'auto-stabilisation. Depuis, l'intérêt pour le domaine ne fait que croître.

La plupart des résultats sur les graphes ont été revus pour obtenir leurs versions auto-stabilisantes. En effet, l'auto-stabilisation constitue un moyen simple et efficace de tolérer des fautes ou des pannes. En effet, si l'on suppose qu'à un moment donné, suite à une panne transitoire, le système se trouve dans une configuration illégitime, on sait que, en l'absence de nouvelles pannes, il va retrouver un comportement correct en un temps fini, puis le conserver. On parle d'auto-stabilisation car ce retour à un comportement normal se fait sans aucune aide extérieure. Dans un système auto-stabilisant aucune hypothèse n'est imposée quant à son initialisation. Cela signifie que tous les canaux de communication et variables présents dans ce système ont une initialisation arbitraire. Un système auto-stabilisant démarre ainsi son exécution dans un état initial quelconque, et cette exécution se déroule en deux phases. La première phase, nommée *phase de convergence*, dure un temps fini pendant lequel le système ne se comporte pas correctement. La deuxième phase, dans laquelle le système se comporte correctement est appelée la *phase de correction*. Les algorithmes de coloration d'arêtes tolérants aux pannes sont des directions de recherche très intéressantes vue la diversité des applications de la coloration propre d'arêtes en particulier dans le domaine des réseaux de télécommunications. Nous nous sommes ainsi intéressés à l'auto-stabilisation dans les algorithmes de coloration d'arêtes, où le système entre dans un état légitime si pour chaque arête, il n'existe aucune arête voisine qui partage la même couleur. Nous avons développé un algorithme auto-stabilisant pour la coloration propre d'arêtes. Notre algorithme se base sur le résultat de Vizing pour utiliser le minimum de couleurs possibles.

L'auto-stabilisation prend naturellement une importance fondamentale dans les réseaux mobiles ad hoc. Un réseau mobile ad hoc ou MANET (Mobile Ad hoc Network) est composé de nœuds mobiles libres de se déplacer dans n'importe quelle direction. Un MANET peut se modéliser par un graphe. L'ensemble des sommets correspond aux nœuds du réseau. L'ensemble des arêtes représente les liens de communication entre les nœuds. Ces nœuds sont limités en consommation d'énergie, bande passante et puissance de calcul. Dans un MANET, il n'y a ni administration centralisée ni infrastructure préexistante. Fournir des solutions efficaces pour les protocoles de communication dans les réseaux mobiles ad hoc est donc une tâche extrêmement difficile en raison de la conjonction de ces différentes contraintes. La solution la plus appropriée doit passer par une meilleure organisation des nœuds afin d'introduire de la stabilité dans le réseau, masquer une partie de la dynamique des nœuds, réduire les interférences et la consommation d'énergie, optimiser la diffusion

d'information, augmenter la capacité du réseau en terme de débit par exemple, etc. La solution de partitionnement (appelée aussi clustering) du réseau a été souvent retenue. Le clustering consiste d'une façon générale à regrouper les nœuds dans des sous-groupes ou clusters. Les clusters peuvent être indépendants, recouvrants, interconnectés ou non. Des mécanismes de clustering doivent être adéquats à la nature du problème. Par exemple, le critère de clustering doit être adapté au problème afin de rendre les protocoles de communication plus efficaces. Les clusters doivent se former de manière dynamique où des chefs de clusters sont choisis parmi les nœuds les plus performants du réseaux. Certaines solutions ne répondent pas à la totalité des exigences des protocoles en terme de persistance, de robustesse, de passage à l'échelle, etc. Nous proposons un algorithme auto-stabilisant de clustering dans les MANETs. Notre solution de clustering se base sur le critère de la confiance. La confiance est une notion fondamentale pour maintenir un certain niveau de sécurité. C'est un aspect important dans la conception et l'analyse des réseaux de part son implication dans le développement des relations de confiance qui existent entre les nœuds. Afin de tester les performances de notre algorithme de clustering nous l'avons appliqué à la gestion de clés de groupe. Utiliser la confiance comme un critère de clustering est intéressant pour la gestion des clés. La confiance est un premier niveau de sécurité et les relations de confiance aident à expulser les nœuds malveillants qui risquent de perturber la session de groupe ou le processus de clustering.

Organisation du mémoire

Ce mémoire est structuré en deux parties. La première partie s'intéresse à l'étude du paramètre de coloration d'arêtes ℓ-distance. La deuxième partie décrit deux algorithmes de graphes auto-stabilisants utiles par exemple pour des applications en réseaux de télécommunications.

Partie I : Coloration d'arêtes ℓ-distance de graphes

Dans la première partie, nous présentons nos principaux résultats relatifs à la coloration d'arêtes ℓ-distance.

Dans le **Chapitre 2**, nous introduisons les notations usuelles de la théorie des graphes qui seront utilisées par la suite. Nous présentons également la plupart des classes de graphes que nous considérerons. Enfin, nous définissons quelques notions de colorations déjà étudiées dans la littérature avant de présenter le paramètre de coloration d'arêtes ℓ-distance sur lequel porte les deux chapitres suivants.

Le **Chapitre 3**, présente certaines classes de graphes pour lesquelles le pro-

blème peut être résolu de manière exacte en temps polynomial. Nous étudions le paramètre de coloration d'arêtes ℓ-distance des chaînes, des hypercubes, des grilles, des arbres k-aires complets et des arbres quelconques. Pour chacune de ces classes, nous donnons des valeurs exactes et des bornes de ce paramètre de coloration.

Après cela, dans le **Chapitre 4**, nous nous intéressons aux propriétés de cette coloration pour les graphes puissances. Une puissance p d'un graphe est obtenue en ajoutant des arêtes entre toute paire de sommets séparés d'une distance inférieure ou égale à p. Nous étendons l'étude précédemment menée sur le paramètre dans le chapitre précédent pour mener cette étude sur ces graphes plus complexes.

Partie II : Algorithmes auto-stabilisants dans les graphes

La deuxième partie s'intéresse aux aspects auto-stabilisants des algorithmes de graphes et de leur application dans les réseaux.

Le **Chapitre 5** introduit le paradigme de l'auto-stabilisation. Nous introduisons, par la suite, notre algorithme auto-stabilisant de coloration d'arêtes. Cette coloration est assurée avec au plus $(\Delta + 1)$ couleurs et une complexité polynomiale en $O(m(\Delta + n))$ mouvements, où Δ est le degré maximum du graphe.

Le **Chapitre 6** décrit notre algorithme auto-stabilisant de clustering destiné à des applications dans le domaine de la sécurité dans les réseaux mobiles ad hoc. L'utilisation d'une telle topologie permet de simplifier les problèmes liés à la mobilité, au dynamisme de l'environnement et à la sécurité. La topologie que nous proposons est un partitionnement basé sur les relations de confiance qui existent entre les nœuds. Elle forme une structure capable de s'adapter dynamiquement aux changements de l'environnement mobile. La confiance est liée à la sécurité, c'est un critère adapté au clustering dans les MANETs, qui génère des clusters stables et contribue à éviter les membres malveillants de chaque cluster. Nous présentons par la suite une application possible de notre algorithme de clustering. Nous présentons un algorithme de gestion de clés de groupe dans les réseaux mobiles ad hoc qui s'appuie sur la topologie de clusters préalablement construite. La sécurité de notre protocole est renforcée par son critère de clustering qui surveille en permanence les relations de confiance et expulse les nœuds malveillants de la session de diffusion. Nous avons simulé notre algorithme de clustering pour évaluer ses performances et les comparer avec les performances d'autres algorithmes de clustering existants.

Ce manuscrit se termine par une récapitulation de nos principales contributions, les améliorations à apporter et les perspectives ouvertes par cette thèse.

Liste des publications de l'auteur

Les travaux de recherche effectués durant cette thèse ont donné lieu à la publication d'articles dans une revue internationale et cinq conférences internationales. D'autres articles sont actuellement soumis à des revues internationales.

Article paru dans des journaux internationaux avec comité de lecture

- K. Drira, H. Seba and H. Kheddouci, *ECGK : an efficient clustering scheme for group key management in MANETs*. Computer Communications 33 (9), pages 1094-1107 (2010).

Articles parus dans des conférences internationales

- K. Drira, H. Seba, B. Effantin, and H. Kheddouci, *Distance edge coloring of power graphs*, 8th French Combinatorial Conference (8FCC'10), Paris-France, July 2010.
- K. Drira and H. Kheddouci, *A New Clustering Algorithm for MANETs*, 14th International Telecommunications Network Strategy and Planning Symposium (Networks'10), Warsaw-Poland, September 27-30, 2010.
- K. Drira, L. Dekar et H. Kheddouci. *A self-stabilizing (Delta+1)-edge-coloring algorithm of arbitrary graphs*, International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE Computer Society, Hiroshima, Japan, pp. 312-317 (2009)
- K. Drira , H. Seba and H. Kheddouci, *Distance edge coloring of trees*, International Optimization Conference INOC 2009, Pise, Italie.
- K. Drira, H. Kheddouci and N. Tabbane. *Virtual Dynamic Topology for Routing in Mobile Ad Hoc Networks*. Proceedings of the International Conference on Late advances in Networks (ICLAN'2006), pp 129-134, Paris, France

Articles soumis dans des journaux internationaux avec comité de lecture

- K. Drira, H. Seba and H. Kheddouci, *Distance-edge-coloring and collision-free communication in sensor networks* (En révision à Networks, 2009)
- K. Drira, H. Seba, B. Effantin and H. Kheddouci, *Distance-edge-coloring of some power graphs* (Soumis à Discrete Mathematics, 2010)

Première partie

Coloration d'arêtes ℓ-distance de graphes

Chapitre 2

Généralités sur les graphes et coloration d'arêtes ℓ-distance

Résumé

*Le problème de coloration de graphes est un problème fondamental en optimisation combinatoire. C'est un problème difficile pour lequel nous ne pouvons garantir de trouver la solution optimale en un temps polynomial. Dans ce chapitre, nous allons introduire les définitions et notations de la théorie des graphes nécessaires à la compréhension de la suite du chapitre. Nous n'abordons que les concepts utilisés. Pour une présentation détaillée, on peut se référer aux livres suivants [*Berge 57*] [*Berge 70*] [*Chartrand 08*] [*Gondran 86*] [*Harary 69*] [*Kocay 05*] [*Sache 74*]. Par la suite, nous définissons le problème de coloration de graphes ainsi que le paramètre de coloration que nous avons étudié.*

2.1 Préliminaires

2.1.1 Terminologie des graphes

Un **graphe non orienté** G est un couple (V, E) où V est l'ensemble des **sommets** ou **nœuds** et E l'ensemble des **arêtes**. Nous désignerons par $V(G)$ et $E(G)$, respectivement, l'ensemble des sommets et l'ensemble des arêtes d'un graphe G. Soient $n(G) = |V(G)|$ le nombre de sommets et $m(G) = |E(G)|$ le nombre d'arêtes. Le nombre de sommets est appelé **ordre** du graphe et le nombre d'arêtes est appelé **taille**. S'il n'y a pas d'ambiguïté nous noterons simplement V, E, n et m. Nous disons que les sommets u et v d'un graphe G sont **adjacents** si $\{u, v\} \in E(G)$. Le **voisinage** du sommet v, noté $N_G(v)$, est l'ensemble des sommets de G adjacents (ou voisins) au sommet v et le **voisinage fermé** de v, noté $N_G[v]$, désigne $N_G(v) \cup \{v\}$. Le **degré** d'un sommet v dans G, noté $d_G(v)$, est $|N_G(v)|$. Le degré minimum de G est $\delta_G = min\{d_G(v) : u \in V(G)\}$ et le degré maximum de G est $\Delta_G = max\{d_G(v) : u \in V(G)\}$. Si tous les sommets ont le même degré d, alors G est **régulier** de degré d ou **d-régulier**. De même que précédemment, s'il n'y a pas d'ambiguïté, nous utilisons les notations simplifiées $N(v)$, $N[v]$ et $d(v)$. Soient $u, v \in V$ deux sommets adjacents, une arête $e \in E$ est notée $e = (u, v) = (v, u)$. L'arête e est ainsi incidente aux sommets u et v. Deux arêtes ayant une extrémité $v \in V$ en commun sont dites incidentes en v. Nous disons qu'un graphe non orienté est **connexe** si pour chacune des paires de sommets distincts, u et v, il existe un chemin de u à v.

Proposition 2.1.1. *La somme des degrés des sommets d'un graphe non orienté est égale à deux fois le nombre d'arêtes du graphe.*

Un **graphe orienté** $G = (V, E)$ est défini comme un ensemble $V(G)$ de n sommets ou nœuds, et un ensemble $A(G)$ de m **arcs** reliant ces sommets. Un arc reliant le sommet u au sommet v est noté (u, v). Dans le cas d'un graphe orienté, le **degré sortant** d'un sommet x est le nombre d'arcs qui partent de x, noté $d_+(x)$, et son **degré entrant**, noté $d_-(x)$, est le nombre d'arcs arrivant au sommet x. Nous avons la relation $\sum d_+ = \sum d_- = |A(G)|$ (nombre d'arcs).

La Figure 2.1(a) représente un graphe non orienté et la Figure 2.1(b) représente un graphe orienté, tous d'eux d'ordre 5.

Un graphe **étiqueté** est un graphe où chaque arête est affectée soit d'une chaîne de caractères, soit d'un nombre. Un graphe **pondéré** est un graphe étiqueté où chaque arête est affectée d'un nombre réel positif, appelé poids de cette arête. Le **poids** d'un graphe est la somme des poids des arêtes qui le composent.

Notons qu'un couple de sommets $x, y \in V(G) \times V(G)$ peut être représenté plusieurs fois dans $E(G)$, nous parlons alors d'arête multiple (x, y) et de multi-

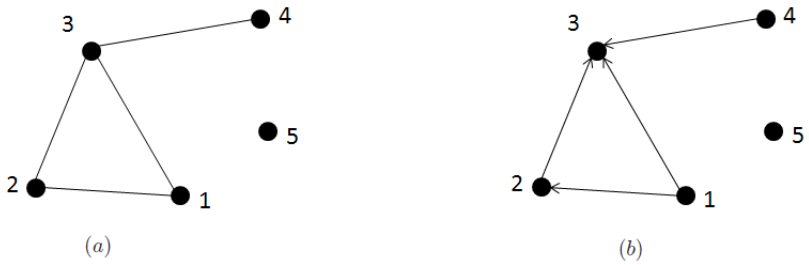

Figure 2.1 : *Exemples de graphe non orienté (a) et de graphe orienté (b).*

graphe G. Un graphe est **simple** s'il n'a pas de boucles ni d'arcs ou arêtes multiples.

Un **stable**, appelé aussi ensemble indépendant, est un ensemble de sommets deux à deux non adjacents.

Soit $G = (V, E)$ un graphe. Le **sous-graphe** $G' = (V', E')$ de G engendré par $V' \subset V$ est le graphe dont l'ensemble des sommets est V' et dont les arêtes (arcs) ont leur deux extrémités dans V'. Si $V' = V$ et $E' \subset E$ alors G' est un graphe **partiel** de G (nous pouvons l'obtenir à partir de G auquel nous avons retiré des arêtes (arcs)). Le sous-graphe de G **induit** par un sous-ensemble de sommets $S \subseteq V$, noté $G[S]$ est le graphe de sommets S et d'arêtes $E \cap S \times S$, c'est-à-dire le graphe obtenu en gardant de G ses sommets dans S et toutes les arêtes les reliant.

Deux graphes H et G sont **isomorphes** s'il existe une bijection f entre $V(H)$ et $V(G)$ de sorte que $\{u, v\} \in E(H)$ si et seulement si $\{f(u), f(v)\} \in E(G)$ [West 00].

Soit G un graphe. Un graphe G' est appelé **subdivision** du graphe G s'il se déduit de G par insertions successives d'un certain nombre de sommets "sur" des arêtes de G. Plus formellement, G' est une subdivision de G s'il s'obtient à partir de G par des opérations successives de remplacement d'une arête (u, v) de G (u et v étant des sommets de G) par un nouveau sommet w plus deux arêtes (u, w) et (v, w).

Soit G un graphe, nous appelons **distance** entre v et v', que nous notons $dist(v, v')$, la longueur du plus court chemin entre v et v'. On a les propriétés suivantes :

- la **distance entre deux arêtes** $e = (u, v)$ et $e' = (u', v')$ est définie de la manière suivante :
  $dist(e, e') = min\{dist(u, u'), dist(u, v'), dist(v, u'), dist(v, v')\}$
- l'**excentricité** d'un sommet u de G est $exc(G) = \max\{dist(u, v) | v \in V\}$
- le **diamètre** d'un graphe est $diam(G) = \max\{exc(v) | v \in V\}$
- le **rayon** d'un graphe $rayon(G) = \min\{exc(u, v) | v \in V\}$

Le **dual d'un graphe** G, noté $L(G)$, est le graphe d'incidence des arêtes de G : les arêtes de G sont les sommets de $L(G)$ et deux sommets de $L(G)$ sont adjacents

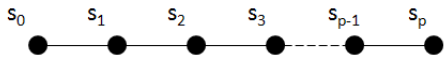

Figure 2.2 : *Une chaîne de longueur p est une suite finie $(s_0, s_1, ..., s_p)$ de sommets du graphe G.*

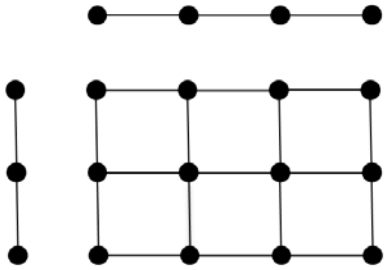

Figure 2.3 : *Grille $M_{3,4} = P_3 \square P_4$.*

si leurs arêtes correspondantes dans G sont incidentes à un même sommet. Les graphes duaux sont plus connus sous leur appellation anglaise de *line-graphs*. Toute coloration des arêtes d'un graphe G correspond à une coloration des sommets de son dual $L(G)$.

2.1.2 Quelques classes de graphes

a. Chemin, chaîne

Un **chemin** ou **chaîne** est une succession d'arêtes ou d'arcs parcourus dans le même sens. Nous parlons de chaîne si nous ne tenons pas compte de la direction des arcs (voir Figure 2.2). Nous parlons ainsi de chaîne dans les graphes non orientés. Le nombre d'arêtes parcourues s'appelle la longueur de la chaîne. L'adjectif **élémentaire** s'applique quand la chaîne ou le chemin parcourt des sommets distincts.

b. Circuit, cycle

Si une chaîne revient à son point de départ, nous parlons de **cycle** dans un graphe non orienté et de **circuit** dans un graphe orienté.

c. Grille

Le produit cartésien de graphes est défini par le produit cartésien des ensembles de sommets de chaque facteur et l'ensemble des arêtes est défini à partir des ensembles d'arêtes de chaque facteur. Le produit cartésien de chaînes P_n et P_m est une grille à deux dimensions $M_{n,m}$, $M_{n,m} = P_n \square P_m$ (voir Figure 2.3)

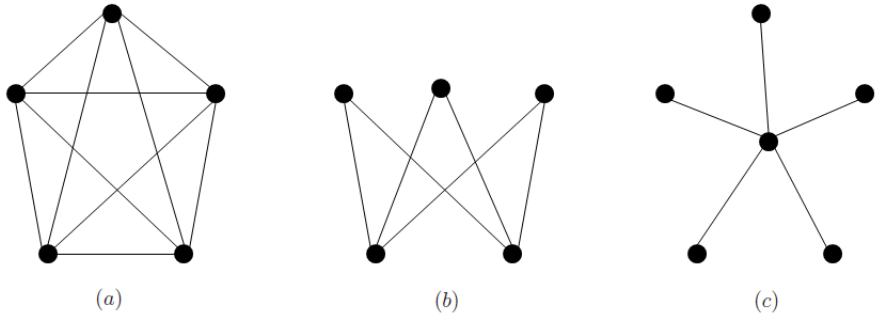

Figure 2.4 : *Graphe complet K_5 (a), graphe biparti complet $K_{3,2}$ (b) et étoile $K_{1,5}$ (c).*

d. Graphe complet (clique) et biparti

Un **graphe complet** à n sommets, noté K_n, est un graphe tel que toutes les paires de sommets sont jointes par une arête. Un graphe complet à n sommets a $\frac{n(n-1)}{2}$ arêtes. Chaque sommet v a un degré $d(v) = n-1$. K_n est un graphe $(n-1)$-régulier. Une **clique** est un sous-graphe complet d'un graphe. La recherche d'une clique dans un graphe est un problème NP-Complet [Karp 72] [Cook 71].

Un graphe G est **biparti** si l'ensemble de ses sommets $V(G)$ peut être partitionné en deux classes X et Y de sorte que deux sommets de la même classe ne soient jamais adjacents. Nous notons habituellement un tel graphe biparti $G = (X, Y, E(G))$. Si pour tout $x \in X$ et pour tout $y \in Y$ nous avons $(x, y) \in E(G)$, le graphe G est **biparti complet**. Le graphe biparti complet avec $|X| = m$ et $|Y| = n$ est noté $K_{m,n}$. Les graphes de la forme $K_{1,n}$ sont appelés **étoiles**.

La Figure 2.4 représente le graphe complet K_5 (a), le graphe biparti complet $K_{3,2}$ (b) et l'étoile $K_{1,5}$ (c).

e. Graphe planaire

Un graphe **planaire** est un graphe qui a la particularité de pouvoir se représenter sur un plan sans qu'aucune arête n'en croise une autre (voir Figure 2.6).

Théorème 2.1.2. *[Kuratowski 30] Un graphe G est planaire si et seulement s'il ne contient pas de subdivision de K_5 ou de $K_{3,3}$.*

Dans la Figure 2.5, le graphe est une subdivision de K_5.

f. Hypercube

Un hypercube de dimension d (également appelé d-cube), noté H_d est le produit cartésien de K_2 par lui même d fois : $H_d = H_{d-1} \square K_2$, $H_1 = K_2$. La construction d'un cube se fait par la translation du cube de dimension inférieure selon un axe

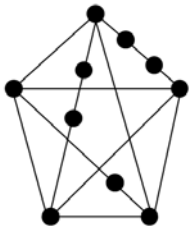

Figure 2.5 : *Exemple de subdivision de K_5.*

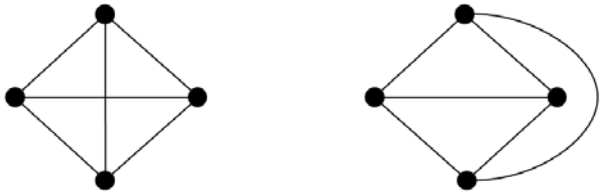

Figure 2.6 : *Exemple de graphe planaire.*

perpendiculaire aux dimensions de ce cube. Autrement dit, nous traçons un hypercube de dimension $d-1$, H^1_{d-1}, nous reproduisons son image H^2_{d-1} et nous relions les points deux à deux avec des arêtes connecteurs (voir Figure 2.7).

Soit V l'ensemble des sommets d'un graphe simple et E l'ensemble de ses arêtes. Un isomorphisme d'un graphe G vers un graphe H est la bijection $f : V(G) \to V(H)$ telle que (u, v) dans $E(G)$ si et seulement si $(f(u), f(v))$ dans $E(H)$ [West 00]. Les deux hypercubes H^1_{d-1} et H^2_{d-1} sont isomorphes. Ainsi, un isomorphisme de H^1_{d-1} vers H^2_{d-1} est une bijection, si pour tout $e^1 = (u, v)$ dans $E(H^1_{d-1})$ il existe une image e^2 de e^1 telle que $e^2 = (f(u), f(v))$ dans $E(H^2_{d-1})$.

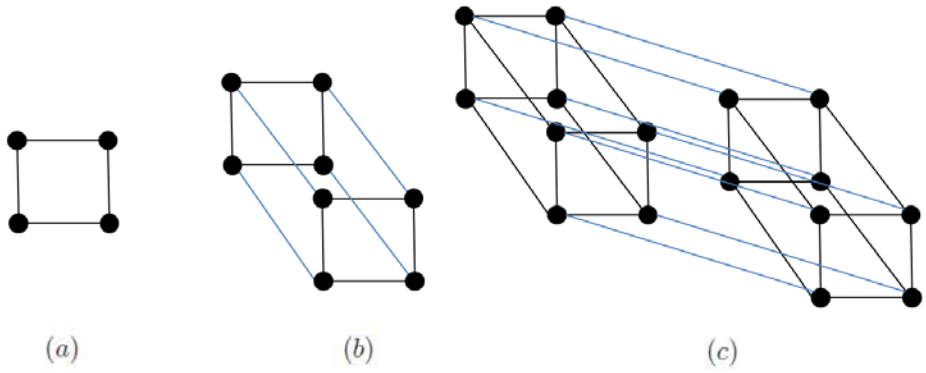

Figure 2.7 : *Hypercubes H_2 (a), H_3 (b) et H_4 (c).*

g. Arbre et forêt

- Un **arbre** $T = (V, E)$ est un graphe connexe et sans cycle. Pour un arbre T de racine r :
  - Le **père** d'un sommet u est l'unique voisin de u sur le chemin de la racine à u. La **racine** r est le seul sommet sans père.
  - Les **fils** d'un sommet u sont les voisins de u autres que son père.
  - Une **feuille** est un sommet sans fils. Les feuilles correspondent aux sommets de degré 1.
  - Les sommets de degré supérieur ou égal à deux sont appelés **sommets internes**.
  - Les **ancêtres** d'un sommet u sont les sommets trouvés sur le chemin unique entre ce sommet et la racine. De la même manière, les ancêtres d'une arête e sont les arêtes trouvées sur le chemin unique entre un sommet de e et la racine.
  - La **hauteur** h de l'arbre T est la longueur du plus long chemin de la racine à une feuille.
  - Nous représentons habituellement un graphe de racine r par **niveaux**. Au niveau 0 apparaît la racine, au niveau 1 ses fils, au niveau 2 les fils de ses fils, etc... Le niveau d'un sommet correspond à sa profondeur dans l'arbre, c'est à dire la longueur de son chemin à la racine.
  - Un arbre admet un ou deux **centres** (suivant la parité du diamètre). Un centre est un sommet d'excentricité minimum par lequel passe toutes les chaînes élémentaires de longueur maximum.
  - Soient v ou/et v' les centres de T. Alors, les **branches** de T sont les sous-graphes obtenus en supprimant les centres de T ainsi que leurs arêtes incidentes.
- Une **forêt** est un graphe sans cycle, composée de l'union d'arbres deux-à-deux disjoints (réunion d'arbres).
- Un **arbre k-aire** (avec $k \geq 2$) est un arbre dont la racine est de degré k, et les autres sommets sont soit de degré $k+1$ soit de degré 1 (les feuilles). Si $k = 2$, il est appelé arbre **binaire**. Un **arbre k-aire complet** est un arbre k-aire où toutes les feuilles sont au même niveau (*i.e.* à la même distance de la racine) (voir Figure 2.8).

h. Graphe puissance

Le graphe de puissance $p \geq 0$, noté G^p, d'un graphe G est obtenu à partir de G en ajoutant une arête entre chaque paire de sommets à distance au plus p dans G. De cette définition, nous remarquons facilement que $G^1 = G$ et G^0 est un stable

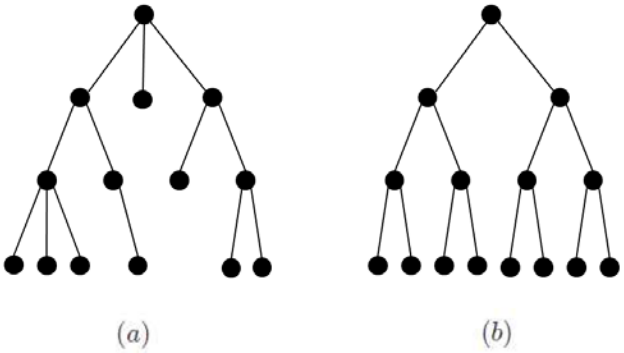

Figure 2.8 : *Exemples d'un arbre quelconque (a) et d'un arbre 2-aire complet(b).*

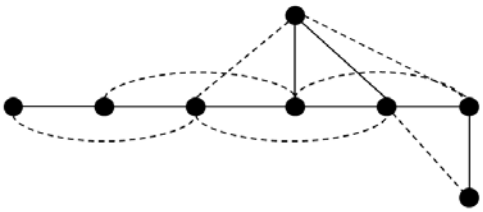

Figure 2.9 : *Exemple de graphe G^2. Les arêtes pleines représentent les arêtes de G et celles en pointillées représentent les arêtes de la puissance 2.*

(voir Figure 2.9).

2.2 Quelques paramètres de graphes

Différents paramètres associés aux graphes ont été introduits, dont nous pouvons donner une représentation simple et visuelle. La plupart de ces paramètres de graphes servent à modéliser le monde réel et offrent un support puissant de résolution pour ces problèmes. Les applications sont très nombreuses. Elles justifient une recherche importante en algorithmique. L'importance des réseaux de transport et de communication, qui sont d'ailleurs de plus en plus des réseaux évolutifs (pour gérer les connexions des utilisateurs à des serveurs de téléphonie mobile, ou à des réseaux pair-à-pair), explique le foisonnement des problématiques. Dans ce qui suit, nous présentons quelques paramètres de la théorie des graphes qui nous seront utiles pour la suite.

2.2.1 Ensembles dominants

Un sous-ensemble D de V est dit ***dominant*** si et seulement si tout nœud de V est soit dans D soit voisin d'un nœud de D [West 00].

Il peut exister différentes classes d'ensembles dominants, nous pouvons citer par

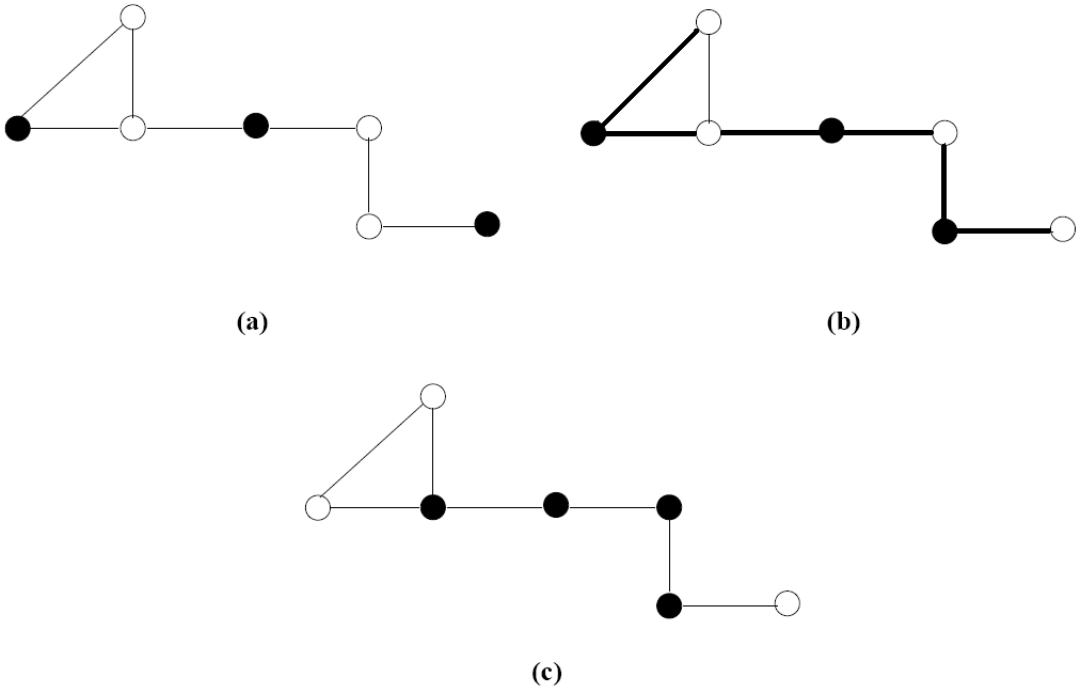

Figure 2.10 : *Ensembles dominants [*Erciyes 07*] : (a)* IDS *(b)* WCDS *(c)* CDS.

exemple les ensembles dominants connectés (*Connected Dominating Sets : CDS*), les ensembles dominants faiblement connectés (*Weakly Connected Dominating Sets : WCDS*) et les ensembles dominants indépendants (*Independent Dominating Sets : IDS*) [Haynes 98] [Erciyes 07].

- Un *IDS* est un ensemble dominant du graphe G tel que les sommets ne sont pas adjacents entre eux (les sommets du IDS forment un stable). La Figure 2.10.a montre un exemple de cet ensemble (sommets colorés en noir).
- Un *WDS* (*weakly dominating set*) est un ensemble dominant faiblement. C'est un sous-ensemble $D \subseteq V$ ayant pour sous-graphe induit D_w. D_w étant le sous graphe incluant tous les sommets de D, leurs voisins et toutes les arêtes du graphe initial ayant au moins un des deux sommets extrémités appartenant à D. D est dit un ensemble dominant faiblement connecté (*weakly-connected dominating set*) si D est dominant et D_w est connexe. La Figure 2.10.b montre un exemple de cet ensemble (D contient les sommets colorés en noir et D_w est le sous-graphe composé des arêtes en gras ainsi que leurs sommets incidents).
- Un *CDS* est un ensemble dominant connecté si le sous-graphe formé par cet ensemble est connexe. La Figure 2.10.c montre un exemple de cet ensemble (sommets colorés en noir).

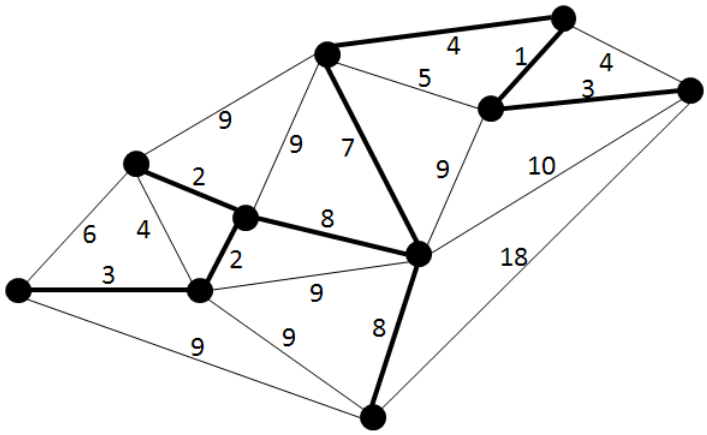

Figure 2.11 : *Un* minimum spanning tree *d'un graphe donné.*

2.2.2 Arbre couvrant

Soit $G = (V, E)$ un graphe connexe. Un arbre couvrant de G est un sous-graphe $T = (V, E')$ qui soit un arbre. Le *MST* (*minimum spanning tree*) est un arbre couvrant d'un graphe pondéré sur les arêtes. Le poids total des arêtes de cet arbre est minimal. L'algorithme de Prim [Prim 57] est un algorithme glouton qui permet de trouver un arbre couvrant minimal dans un graphe connexe pondéré et non-orienté. En d'autres termes, cet algorithme trouve un sous-ensemble d'arêtes formant un arbre sur l'ensemble des sommets du graphe initial, et tel que la somme des poids de ces arêtes soit minimale. L'algorithme consiste à choisir arbitrairement un sommet et à faire croître un arbre à partir de ce sommet. Chaque augmentation se fait de la manière la plus économique possible.

La Figure 2.11 montre un *minimum spanning tree* d'un graphe G.

2.2.3 Les clusters

Un cluster est un graphe réduit dont les sommets satisfont ou partagent une certaine propriété.

2.2.4 Colorations de graphes

Depuis plusieurs décennies, la coloration de graphes est un domaine très attractif de la théorie des graphes par ses nombreuses applications. La coloration de graphes permet de modéliser de nombreux problèmes réels, depuis le placement de personnes autour d'une table ou de pièces sur un échiquier jusqu'aux différents problèmes d'ordonnancement et de planning de la vie de tous les jours et dans le domaine des réseaux/télécom [Barrett 06] [Gandham 05] [Kaklamanis 97] [Lewis 10] [Molloy 02] [Skiena 08].

Définition 2.2.1. Une *k-coloration* d'un graphe $G = (V, E)$ est une application $c : V \to C$, où C est un ensemble de k couleurs (généralement un ensemble d'entiers, $C = \{1, 2, \cdots, k\}$). Si pour toute arête (u, v) de $E : C(u) \neq C(v)$ alors la coloration est dite propre. Un graphe admettant une k-coloration propre est dit *k-colorable*.

La coloration de graphes est un outil permettant de caractériser les graphes. Il n'y a pas une unique façon de colorer les graphes mais plusieurs. Nous pouvons déjà vouloir colorer différents éléments d'un graphe (les sommets, les arêtes, une combinaison de ces éléments, des sous-structures, etc) et à ceci peuvent s'ajouter différentes contraintes. La contrainte la plus courante est celle de la propreté.

2.3 Problèmes de coloration de graphes

2.3.1 Un peu d'histoire...

Le problème de coloration de graphes est une des origines de la théorie des graphes. L'un des problèmes les plus célèbres et les plus productifs est le problème des quatre couleurs. Ce problème est resté pendant plus d'un siècle sans solution [Appel 77] [Gonthier 08]. Le problème des quatre couleurs consistait à trancher la question suivante : peut-on colorer n'importe quelle carte géographique de sorte que deux régions voisines n'aient pas la même couleur en n'utilisant que quatre couleurs au maximum ?

Les premiers résultats de coloration de graphes concernent presque exclusivement les graphes planaires : il s'agissait alors de colorer des cartes. En 1852, le jeune mathématicien F. Guthrie s'est demandé s'il est toujours possible de colorer une carte de géographie à l'aide de quatre couleurs de sorte à ce que des pays voisins aient des couleurs différentes. La première trace écrite de la conjecture des quatre couleurs apparaît dans une lettre que A. D. Morgan envoya à W. R. Hamilton en 1852. De nombreuses tentatives de démonstration de la conjecture (incorrectes) furent alors proposées par Cayley, Kempe et bien d'autres. La carte à colorer a été remplacée par un graphe, chaque pays étant représenté par un sommet et deux pays voisins étant reliés par une arête.

Les Figures 2.12 et 2.13 illustrent l'exemple de coloration de la carte des régions de France et son graphe associé.

L'étude de cette conjecture entraîna de nombreux développements en théorie des graphes, par P. G. Tait, P. J. Heawood, F. Ramsey et H. Hadwiger. En 1890, P. J. Heawood fit remarquer que la démonstration de A. Kempe était fausse. Il montra quant à lui le théorème des cinq couleurs en reprenant des idées de A. Kempe. De nombreux travaux ont été publiés lors du siècle suivant pour réduire le nombre de couleurs à quatre, jusqu'à la démonstration finale de deux chercheurs américains, K.

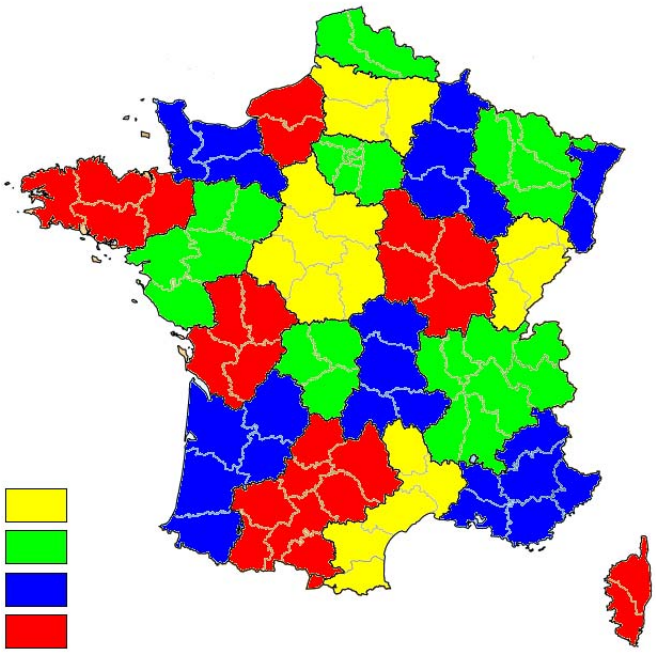

Figure 2.12 : *Exemple de coloration de la carte des régions de France.*

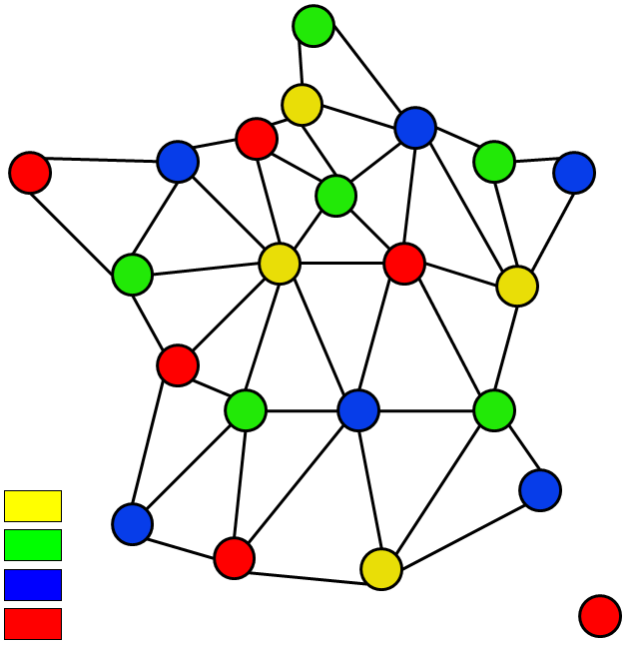

Figure 2.13 : *Le graphe associé à la carte des régions de France.*

Appel et W. Haken en 1976. Ils ont pu répondre affirmativement à cette conjecture des quatre couleurs. Plus d'un siècle s'est donc écoulé entre l'énoncé et la résolution de ce problème en apparence très simple. Pour une note historique détaillée vous pouvez vous référer à [Kubale 04] [Wilson 04].

2.3.2 Coloration sommet

Le paramètre le plus étudié et connu est le *nombre chromatique* de G, noté $\chi(G)$ [Malaguti 09][Pardalos 98]. Il est défini comme le nombre minimum de couleurs nécessaires pour colorer les sommets d'un graphe G de sorte que deux sommets adjacents n'aient pas la même couleur (coloration propre de sommets).

Brooks [Brooks 41] a montré que $\chi(G) \leq \Delta + 1$, pour un graphe G de degré maximum Δ. Nous avons égalité si et seulement si, soit $\Delta \neq 2$ et G admet un sous-graphe $K_{\Delta+1}$ comme composante connexe, soit $\Delta = 2$ et G admet un cycle C_{2K+1} comme composante connexe. La détermination du nombre chromatique dans le cas général est un problème NP-difficile.

Le problème d'allocation de fréquences peut être modélisé par un problème de coloration propre de sommets. Le problème d'allocation de fréquences a pour but d'affecter une fréquence à chacune des antennes d'un réseau mobile en minimisant les risques d'interférences dans les communications. Un réseau permet aux mobiles de communiquer entre eux sur de larges zones. Chaque mobile a besoin d'une fréquence pour permettre le transport des informations par ondes radio. Dans le réseau mobile *GSM*, par exemple, le nombre de mobiles ne cesse d'augmenter, alors que le nombre de fréquences allouées est très limité. Il est donc nécessaire d'affecter une même fréquence à plusieurs mobiles du réseau. Deux mobiles utilisant des fréquences trop proches peuvent subir des dégradations, nommées interférences. Lorsque le niveau d'interférence est trop élevé, la qualité du signal devient si mauvaise que la communication est impossible. La performance du réseau mobile dépend en grande partie de la façon dont les fréquences sont attribuées aux mobiles, constituant le problème d'allocation de fréquences. Ce problème se modélise par un graphe de la manière suivante :

- Chaque mobile est représenté par un sommet.
- Deux sommets sont reliés par une arête dès que les deux mobiles communiquent.

Le problème d'affectation de fréquences radio revient à associer à chaque sommet du graphe une couleur (correspondant à une fréquence) de sorte que deux sommets reliés par une arête ne reçoivent pas la même couleur, le but étant de minimiser le nombre de couleurs utilisées.

D'autres paramètres de coloration sont dérivés du nombre chromatique. Le *nombre chromatique borné* $\chi_k(G)$ est le nombre minimum de couleurs nécessaires pour une coloration propre d'un graphe G telle que chaque couleur apparaisse au plus k fois. Jarvis et Zhou [Jarvis 01] utilisent ce paramètre pour modéliser des ensembles de tâches ne pouvant être exécutées en même temps. Ils ont évalué la complexité de recherche de ce paramètre pour des arbres avec k quelconque.

Le *nombre chromatique harmonieux* est le nombre minimum de couleurs néces-

saires à la coloration d'un graphe G telle que la coloration soit propre et que les extrémités de deux arêtes distinctes ne soient pas colorées par la même paire de couleurs. La détermination de ce paramètre pour un graphe quelconque est un problème NP-difficile [Hopcroft 83]. Ce paramètre a été très étudié pour les arbres. En 1997, Lu [Lu 97] détermine une valeur exacte de ce paramètre pour les arbres binaires complets et en 1999, Edwards [Edwards 99] étudie ce paramètre pour les arbres k-aires complets.

Le *nombre chromatique fort* $\chi_s(G)$ est le nombre minimum de couleurs nécessaires pour une *coloration forte* du graphe G telle que la coloration soit propre $(C_1, ..., C_k)$ et que pour chaque sommet u de G, il existe une couleur i pour laquelle u soit adjacent à tous les sommets de la classe de couleur C_i. Le concept de coloration forte a été introduit en 2006 par Zverovich [Zverovich 06]. Les auteurs donnent une caractérisation de tous les graphes 3-fortement colorables et prouvent que si un graphe admet une 3-coloration forte alors elle est unique. La *coloration forte stricte* de graphes est une coloration forte qui n'admet pas la présence de la classe de couleur vide [Haddad 09]. Le *nombre chromatique fort strict* $\chi_{ss}(G)$ est le nombre minimum de couleurs nécessaires pour une coloration forte stricte du graphe G. La détermination de ce paramètre pour un graphe quelconque est un problème NP-complet. Ce paramètre a été étudié pour les arbres [Haddad 09]. Le paramètre de la coloration forte stricte peut être appliqué à la diffusion dans les réseaux [Haddad 08]. En effet, la notion de dominance (i.e. pour chaque sommet v, il existe une classe de couleur C_i telle que v soit adjacent à tous les sommets de C_i) induite par la coloration forte stricte permet d'optimiser la diffusion dans les réseaux. Soit $G = (V, E)$ un graphe et soit $C_{ss} = \{C_1, ..., C_k\}$ une coloration forte stricte de G. Un ensemble fortement dominant (ou *strong dominating set*, dénoté par SDS) dans le graphe G induit par la coloration C_{ss} est un sous-ensemble de sommets $S \subseteq V$ tel que $v \in S$ si et seulement si v est coloré avec une couleur dominée (i.e. $S = V(DCS)$). Après la construction de l'ensemble fortement dominant (SDS) sur le graphe modélisant le réseau, chaque sommet v connaît tous les sommets qu'il domine et connaît tous les sommets qui le dominent (informations offertes par des fonctions de dominance et dominance inverse [Haddad 08]). La diffusion est alors effectuée en deux phases : une diffusion intra-cluster et un acheminement inter-cluster. Pour la première phase, un sommet voulant diffuser une information l'envoie à un élément de l'ensemble fortement dominant de son cluster, en se basant sur sa fonction de dominance. Ces derniers assurent la diffusion dans tout le cluster en se basant sur les fonctions de dominance et dominance inverse. Le fait que chaque sommet domine une seule classe de couleur réduit énormément le phénomène de redondance. En ce qui concerne la deuxième phase, les messages unicasts permettent d'acheminer l'information entre les clusters.

Tous ces paramètres minimisent le nombre de couleurs nécessaires à la coloration de sommets d'un graphe. Il existe d'autres paramètres qui cherchent à maximiser ce nombre de couleurs. Nous pouvons citer par exemple le *nombre achromatique* $\psi(G)$, qui représente le nombre maximum de couleurs nécessaires à la coloration d'un graphe G telle que la coloration soit propre et que chaque paire de couleurs apparaisse au moins sur une des arêtes de G. Ce paramètre a été introduit en 1970 par Harary et Hedetniemi [Harary 70] et il fournit une borne supérieure au nombre chromatique : pour tout graphe G, $\chi(G) \leq \psi(G)$. Yannakakis et Gavril [Yannakakis 80] ont prouvé que le calcul de ce paramètre pour un graphe quelconque était NP-difficile. Dans [Farber 86], Farber et al. montrent que le problème est NP-difficile pour les graphes bipartis, et Cairnie et Edwards [Cairnie 97] montrent que ce problème est NP-difficile pour les arbres.

Le *nombre Grundy* $\Gamma(G)$ [Christen 79] est le nombre maximum de couleurs nécessaires à la coloration d'un graphe G pour que la coloration soit propre et pour que chaque sommet u dans G ayant la couleur $c(u)$ est adjacent à toutes les couleurs de 1 jusqu'à $c(u) - 1$. Ces sommets sont dit *sommets Grundy*. Ce paramètre a été étudié pour différentes classes de graphes telles que les arbres [Hedetniemi 82], les graphes puissances [Germain], le produit cartésien de graphes [Effantin 07], etc.

Le *nombre b-chromatique* $\varphi(G)$ représente le nombre maximum k de couleurs nécessaires à la coloration d'un graphe G telle que la coloration soit propre et que pour chaque couleur $i \in \{1, .., k\}$, il existe un sommet u de couleur i, adjacent à des sommets colorés avec chaque couleur $i \in \{1, .., k\}$ et $i \neq j$. Ces sommets sont appelés *sommets dominants*. Nous parlons dans ce cas de la *b-coloration*. Le but du problème de la b-coloration étant de maximiser le nombre de couleurs, en fonction des contraintes données précédemment. Irving et Manlove [Irving 99] ont prouvé que trouver le nombre b-chromatique d'un graphe quelconque est un problème NP-difficile. Ils donnent cependant un algorithme permettant de calculer en temps polynomial le nombre b-chromatique d'un arbre. D'autres études ont été menées pour borner le nombre b-chromatique sur des graphes particuliers. Kouider et Mahéo [Kouider 02] ont donné dans un premier temps des valeurs exactes du nombre b-chromatique pour des graphes simples. Dans [Kouider 06], ils évaluent également des bornes et valeurs exactes pour la somme cartésienne de deux graphes. Le paramètre a été étudié pour la somme de deux étoiles, la somme d'une étoile et d'une chaîne et la somme de deux graphes complets. Dans [Effantin 03], Effantin et al. ont donné une valeur exacte du nombre b-chromatique pour les graphes puissances d'une chaîne, d'un cycle et d'un arbre binaire complet. Ils ont aussi étendu leurs résultats aux graphes puissances d'un arbre k-aire complet [Effantin 05a] et aux graphes puissances d'un *caterpillar* complet [Effantin 05b]. La b-coloration

peut avoir des applications très intéressantes dans des domaines variés. En effet, la b-coloration permet de décomposer un système en un ensemble de communautés, où chaque communauté contient un sommet adjacent à toutes les autres communautés. Ce genre d'organisation peut être très utile dans le clustering des données [Elghazel 08]. Soit la représentation topologique d'un ensemble d'individus à grouper $X = \{X_1, ..., X_n\}$ par un graphe complet, non orienté et pondéré $G = (V, E)$ pour lequel les sommets $\{v_1, ..., v_n\}$ sont les individus à classer (le sommet v_i correspond à l'individu X_i) et les arêtes sont les liens pondérés par les dissimilarités entre les paires de données. Le graphe G est traditionnellement représenté par un tableau de dissimilarités symétrique de taille $n \times n$ $D = \{d(X_i, X_j | X_i, X_j \in X)\}$. Le clustering des données consiste à construire des clusters. Un cluster représente un ensemble d'éléments semblables, et les éléments de différents clusters sont différents. En conséquence, un cluster doit avoir une homogénéité interne élevée et une hétérogénéité forte entre les éléments de différents clusters. Ces deux conditions s'élèvent à affirmer que les arêtes entre deux sommets d'un même cluster doivent avoir des faibles pondérations (indiquant une similarité élevée), et ceux entre les sommets de deux clusters doivent être à forte pondération (indiquant une similarité faible). Le problème de clustering est par consequent ramené à un problème de b-coloration dans le graphe de dissimilarités G [Elghazel 08].

2.3.3 Coloration arête

Nous détaillons dans la suite un autre paramètre de coloration de graphes qui est la coloration d'arêtes.

Le problème de coloration d'arêtes a été posé en 1880 en relation avec la conjecture des quatre couleurs. Le premier article qui traite le problème de la coloration d'arêtes a été écrit par Tait en 1889 [Fiorini 77]. Dans cet article Tait a prouvé que le problème des quatre couleurs est équivalent au problème de la coloration d'arêtes des graphes planaires cubiques avec trois couleurs.

La coloration des arêtes d'un graphe consiste à attribuer une couleur à chaque arête du graphe telle que deux arêtes ayant un sommet commun n'aient jamais la même couleur, le tout en utilisant le moins de couleurs possibles. La Figure 2.14 donne un exemple de coloration d'arêtes. La Figure 2.14(a) donne un exemple de coloration d'arêtes non propre et la Figure 2.14(b) donne un exemple de coloration d'arêtes propre. Nous vérifions en effet que chaque sommet ne possède jamais deux arêtes adjacentes de la même couleur.

L'*indice chromatique* $\chi'(G)$ est le nombre minimum de couleurs nécessaires à la coloration d'arêtes d'un graphe telle que deux arêtes adjacentes n'aient pas la même couleur. Le théorème de Vizing est un théorème de la théorie des graphes qui stipule que la coloration des arêtes d'un graphe G peut s'effectuer à l'aide de $\Delta_G + 1$

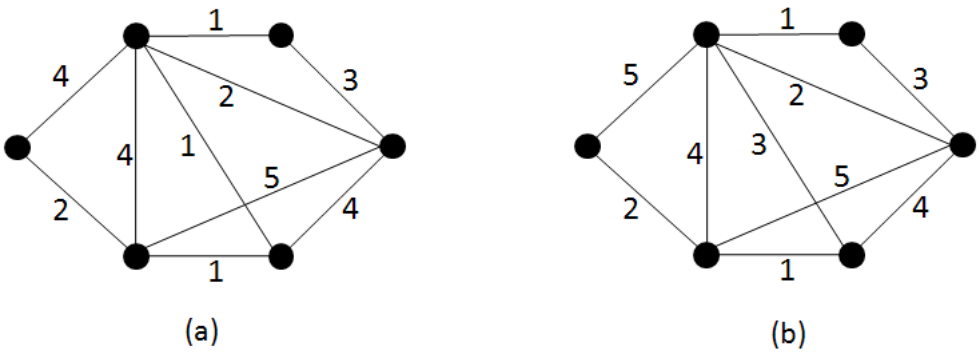

Figure 2.14 : *Exemples de coloration d'arêtes.*

couleurs au maximum, où Δ_G est le degré maximum du graphe G. Autrement dit $\chi'(G) = \Delta_G$ ou $\chi'(G) = \triangle_G$+1. Quand $\chi'(G) = \Delta_G$, nous dirons que G est de "classe 1", dans le cas contraire, il sera de "classe 2". Le problème de décider qu'un graphe appartient à la classe 1 ou la classe 2 est un problème de classification [Beineke 73] [Erdos 77] [Jakobsen 73] [Sanders 01] [Vizing 65]. En 1981, Holyer a prouvé que la détermination de l'appartenance d'un graphe simple à l'une ou l'autre de ces deux classes est un problème NP-complet [Holyer 81]. Cependant, pour certains cas particuliers, il existe des règles permettant de conclure rapidement. Par exemple, Vizing, en 1965, a établi qu'un graphe simple et planaire est de classe 1 si son degré maximum est supérieur ou égal à 8 (théorème de Vizing [Vizing 64] [Vizing 65])). Les graphes bipartis (théorème de König [Konig 16]) font partie de la classe 1. Vizing a même donné une procédure qui permet de déterminer une coloration des arêtes de G en $\Delta(G) + 1$ couleurs. Son graphe est coloré avec $\Delta(G)$ ou $\Delta(G) + 1$. Il est par contre très difficile de savoir si $\chi'(G)$ vaut $\Delta(G)$ ou $\Delta(G) + 1$ lorsque G n'a aucune propriété particulière.

La preuve de Vizing utilise deux structures de données : un *fan* et un (c, d)-*path*. Nous détaillons ci-dessous ces deux structures de données que nous allons utiliser dans le Chapitre 5. Soit $e_0 = (u, v)$ une arête non colorée. Alors, un fan F est une séquence d'arêtes distinctes $e_0, e_1, ..., e_k$ incidentes au sommet u tel qu'il existe une séquence de couleurs distinctes $\alpha_0, \alpha_1, ..., \alpha_{k-1}$ qui satisfont les conditions suivantes :

1. α_i est une couleur manquante [1] au niveau du sommet v_i, $0 \leq i \leq k - 1$;
2. $e_i = (u, v_i)$, $1 \leq i \leq k$, est colorée avec α_{i-1} ; et
3. $\alpha_0, \alpha_1, ..., \alpha_{k-1}$ sont distinctes.

Puisque G est un graphe simple, les arêtes $e_0, e_1, ..., e_k$ sont distinctes. Une rotation d'un fan F signifie re-colorer e_i avec α_i pour chaque i, $0 \leq i \leq k - 1$ et effacer la

1. Nous disons que pour un sommet v il lui manque la couleur α, si v n'admet pas d'arêtes incidentes de couleur α.

couleur α_{k-1} de e_k. Une rotation de F entraîne une coloration d'une autre arête de G telle que e_k au lieu de e_0 soit non-colorée. Si une couleur manquante α_k au niveau v_k est aussi une couleur manquante au niveau u, alors une rotation du fan rend α_k une couleur manquante au niveau des deux extrémités de l'arête non-colorée $e_k = (u, v_k)$ et par conséquent colorer e_k avec α_k complète la coloration propre d'arêtes de G.

Si α_k n'est pas une couleur manquante au niveau u, alors il existe un sommet v_j voisin à u tel que $C(u, v_j) = \alpha_k$. Soit P un (β, α_k)-path P ((c, d)-path tel que $c = \beta$ et $d = \alpha_k$). β est une couleur manquante au niveau du sommet u telle que $\beta \neq \alpha_k$ et β n'est pas une couleur manquante au niveau du sommet v_j. Cela signifie qu'il existe une arête incidente à v_j tel que la couleur de cette arête est β. Le (β, α_k)-path P définit la chaîne maximale qui commence par le sommet v_j et colorée avec β et α_k alternées tout au long des arêtes successives de P. Il est indiqué dans [Misra 92] que (β, α_k)-path est simple et unique. L'opération d'échanger les couleurs des arêtes dans P (β par α_k et α_k par β) est nommée *inversion*. Si l'état initial de la coloration du graphe est valide alors la coloration du graphe après inversion de la chaîne maximale (β, α_k)-path reste valide [Misra 92]. Après l'opération d'inversion, une rotation de $F = e_0, e_1, ..., e_j$ est possible dans ce cas. En effet, l'inversion de P rend β une couleur manquante au niveau de v_j. Puisque β devient une couleur manquante au niveau des deux extrémités de e_j, colorer e_j avec β complète la coloration propre d'arêtes de G. Le problème de partage des ressources peut être considéré comme un problème de coloration propre d'arêtes. En effet, dans le problème de partage des ressources, il y a un ensemble de nœuds et un ensemble de ressources. Si un nœud peut accéder à une certaine ressource, alors nous connectons les deux par une arête pour construire un réseau. La prochaine étape consiste à appliquer une coloration propre d'arêtes à ce réseau. Par la suite, un nœud est autorisé à accéder à la ressource lorsque l'indice de l'intervalle de temps correspond à la couleur de l'arête. De cette façon, la planification des ressources se fait à l'avance et le conflit des ressources est ainsi éliminé.

Différentes variantes et généralisations de la coloration d'arêtes ont été introduites et étudiées. L'indice chromatique de la *coloration propre d'arêtes à sommets adjacents distincts*(SAD) $\chi'_a(G)$ est le nombre minimum de couleurs nécessaires à la coloration d'arêtes d'un graphe telle que la coloration soit propre et que pour tout $x, y \in V$, $x \neq y$, $S(x) \neq S(y)$ avec $S(x)$ contient l'ensemble de couleurs de toutes les arêtes incidentes à x. Ce paramètre de coloration a été introduit et étudié par Burris dans [Burris 93]. Dans [Burris 93] [Balister 02] [Balister 03], le nombre de la coloration SAD est calculé pour quelques familles de graphes comme les chaînes, les cycles, les graphes bipartis, etc. La conjecture suivante est posée dans [Zhang 02].

Conjecture : Soit un graphe G de degré maximum Δ et d'ordre au moins 6

alors $\Delta \leq \chi'_a(G) \leq \Delta + 2$.

Le problème de la *f-coloration d'arêtes* d'un multi-graphe $G = (X, E)$ est une généralisation de la coloration d'arêtes. Ce problème revient à trouver une affectation d'un nombre minimum $\chi'_f(G)$ de couleurs aux arêtes de G de manière que, pour chaque sommet v et chaque couleur i, nous avons $|C_i(v)| \leq f(v)$, où $C_i(v)$ est l'ensemble des arêtes de couleur i incidentes à v [Zhou 95]. Soit $\Delta_f(G) = max_{v \in V}[d(v)/f(v)]$ avec $d(v)$ le degré du sommet v. Il est connu que $\chi'_f(G)$ est égal à $\Delta_f(G)$ ou $\Delta_f(G)+1$ pour tout graphe simple G [Hakimi 86]. Le problème de la $[g, f]$-coloration d'arêtes suppose la condition suivante : $g(v) \leq |C_i(v)| \leq f(v)$ pour tout sommet $v \in X$ et toute couleur i. Une couleur i apparaît ainsi au moins $g(v)$ fois et au plus $f(v)$ fois au niveau du sommet v. Le problème de la $[g, f]$-*coloration d'arêtes* est aussi connu sous le nom du problème de $[g, f]$-*factorisation* [Liu 94] [Liu 05]. Le problème de transfert de fichiers dans un réseau d'ordinateurs peut être considéré comme un problème de f-coloration des arêtes. Soient un ensemble X d'ordinateurs, chacun capable de communiquer directement avec chaque autre ordinateur, et un ensemble E de grands fichiers devant être transférés entre divers ordinateurs. Chaque ordinateur $v \in X$ dispose de $f(v)$ ports identiques lui permettant de gérer jusqu'à $f(v)$ transferts à tout instant. Chaque fichier est transféré directement de son origine à sa destination. Une fois qu'un transfert a commencé, il s'effectue sans interruption, et tous les transferts nécessitent le même temps. Le but est de trouver un ordonnancement des transferts de manière à minimiser le temps total requis pour effectuer tous les transferts. Ce problème peut être modélisé à l'aide d'un multi-graphe $G^f = (X, E, f)$ dont les sommets représentent les ordinateurs et où chaque fichier est représenté par une arête liant les deux ordinateurs concernés par le fichier [Paschos 07].

Dans la section suivante, nous nous intéressons à une autre généralisation de la coloration d'arêtes qui est le paramètre de la coloration d'arêtes ℓ-distance.

2.4 La coloration d'arêtes ℓ-distance

La coloration d'arêtes ℓ-distance, $\ell \geq 0$, est une généralisation de la coloration d'arêtes classique qui consiste à assigner une couleur de 1 à k à chaque arête, telle que chaque deux arêtes qui sont à distance au plus ℓ ne partagent pas la même couleur. Autrement dit, soit un entier $\ell \geq 0$, une coloration d'arêtes ℓ-distance de G colore toutes les arêtes du graphe telles que toutes arêtes e et e', avec $dist(e, e') \leq \ell$, ont des couleurs différentes. Il est évident que la coloration d'arêtes 0-distance est la coloration propre d'arêtes classique. Le nombre minimum de couleurs utilisées pour colorer un graphe avec une coloration d'arêtes ℓ-distance est appelé l'*indice ℓ-chromatique* et noté $\chi'_\ell(G)$.

La *coloration de sommets ℓ-distance* a été définie en 1969 par Kramer et al. [Kramer 69]. La coloration de sommets ℓ-distance a été largement étudiée dans la littérature [Kramer 72], [Speranza 75], [Fertin 03].

Kramer donne le résultat suivant sur le nombre minimum de couleurs pour la coloration de sommets ℓ-distance, *nombre ℓ-chromatique*, noté $\chi_\ell(G)$.

Théorème 2.4.1. *[Kramer 72] Soit $G = (V, E)$ un graphe non orienté, fini, connexe et sans boucle et ℓ un entier positif, $\ell \geq 2$ alors*

$$\chi_\ell(G) = \ell + 1,$$

si et seulement si l'une des conditions suivantes est remplie :

1. $|V| = \ell + 1$,
2. *E se réduit à un seul chemin de longueur $L \geq \ell$,*
3. *E se réduit à un seul cycle, dont la longueur est un multiple de $\ell + 1$.*

Le même résultat a été trouvé par Speranza en 1975 [Speranza 75]. Fertin et al. [Fertin 03] ont obtenu une valeur exacte du paramètre pour une grille à deux dimensions :

$$\chi_\ell(G) = \lceil \frac{1}{2}(\ell + 1)^2 \rceil.$$

Une étude détaillée de cette coloration de sommets ℓ-distance est présentée dans [Kramer 08].

Seuls quelques travaux sur la coloration d'arêtes ℓ-distance ont été proposés. Ito et al. [Ito 07] ont développé deux algorithmes pour la coloration d'arêtes ℓ-distance. Dans [Ito 07], les auteurs présentent un algorithme polynomial qui résout le problème de la coloration d'arêtes ℓ-distance dans un k-arbre partiel, ensuite ils donnent un algorithme polynomial 2-approximatif pour les graphes planaires. Un résultat plus récent [Kang 09] de cette coloration donne une borne inférieure du paramètre de la coloration d'arêtes ℓ-distance.

Proposition 2.4.2. *[Kang 09] Soit un entier $\ell \geq 2$. Pour certaines valeurs arbitraires de Δ, il existe un graphe régulier de degré Δ tel que*

$$\chi'_\ell(G) > \frac{\Delta \ell}{2\,(\ell - 1)^{\ell - 1}}.$$

Le paramètre de la coloration d'arêtes ℓ-distance est relié à la coloration sommet du graphe puissance du graphe dual.

$$\chi'_\ell(G) = \chi_{\ell+1}(L(G)) = \chi((L(G))^{\ell+1}).$$

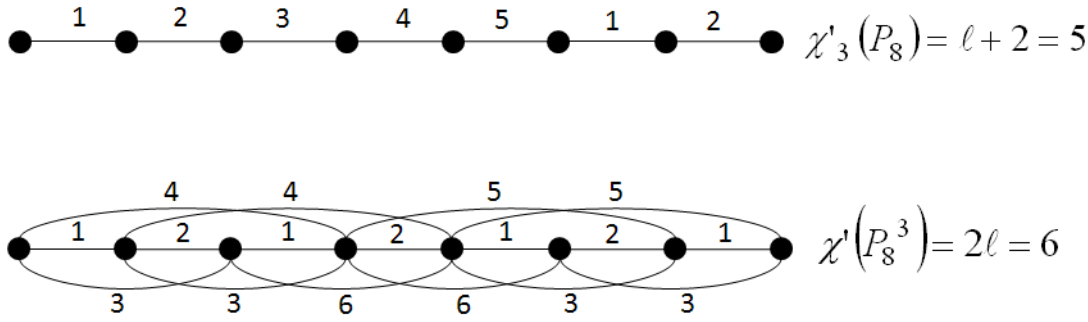

Figure 2.15 : *Vizing vs Coloration d'arêtes ℓ-distance.*

Le théorème de Brooks peut fournir une borne supérieure de l'indice ℓ-chromatique [Kang 09].

$$\chi'_\ell(G) \leq \Delta((L(G))^{\ell+1}) + 1 \leq 1 + 2\sum_{i=1}^{\ell+1}(\Delta - 1)^i + 1.$$

Le problème de la coloration d'arêtes ℓ-distance est un problème original qui est NP-difficile en général et ne peut pas être résolu d'une manière efficace dans les graphes généraux [Ito 07].

Il est intéressant de noter que le problème de la coloration d'arêtes ℓ-distance est différent de la détermination de l'indice chromatique, en utilisant l'algorithme de Vizing, de son graphe puissance. En effet, si nous considérons la chaîne P_8 d'ordre 8, pour $\ell = 3$, nous avons $\chi'_3(P_8) = \ell + 2 = 5$ (voir chapitre 3). Cependant, l'indice chromatique de son graphe puissance est égal à $\chi'(P_8^3) = 2\ell = 6$ (voir Figure 2.15).

2.5 Conclusion

Ce chapitre nous a permis de rappeler les bases du vocabulaire utilisé dans la théorie des graphes afin de pouvoir appréhender la suite au niveau de l'utilisation des graphes pour la résolution de problèmes combinatoires. Dans la suite de cette partie, nous étudions le problème de la coloration d'arêtes ℓ-distance de différentes classes de graphes comme les chaînes, les hypercubes, les grilles, les arbres et des graphes puissances. Dans l'ensemble de ce mémoire, sauf mention contraire, les graphes considérés seront simples, connexes et non orientés.

Chapitre 3

Coloration d'arêtes ℓ-distance de quelques classes de graphes

Résumé

Dans ce chapitre, nous étudions la coloration d'arêtes ℓ-distance pour certaines classes de graphes (chaînes, hypercubes, grilles, arbres). Nous nous intéresserons à l'aspect algorithmique de ce paramètre, en donnant pour chaque classe de graphes des méthodes de colorations.

3.1 Coloration d'arêtes ℓ-distance d'une chaîne

Théorème 3.1.1. *Soit P_n une chaîne d'ordre $n > 0$. L'indice ℓ-chromatique de P_n est donné par :*

$$\chi'_\ell(P_n) = \begin{cases} \ell + 2 & \text{si } n \geq \ell + 3, \quad (a) \\ n - 1 & \text{si } n \leq \ell + 3. \quad (b) \end{cases}$$

Démonstration. ***Cas(a)*** Dans un premier temps, nous démontrons par construction la borne supérieure de l'indice ℓ-chromatique. Soit Q l'ensemble de couleurs défini par $Q = \{1, 2, 3, ..., \ell + 2\}$. Nous colorons d'une manière cyclique les arêtes de P_n en utilisant les couleurs de Q. Pour une coloration d'arêtes ℓ-distance d'un graphe G, la distance entre deux arêtes distinctes ayant la même couleur est égale au moins à $(\ell + 1)$. La coloration obtenue est une coloration d'arêtes ℓ-distance. Ainsi, $\chi'_\ell(P_n) \leq \ell + 2$. La Figure *3.1* montre une coloration d'arêtes 1-distance de P_5. Dans un second temps, pour prouver la borne inférieure, soient e et e' deux arêtes de P_n telles que $dist(e, e') = \ell + 1$. Si nous supposons que ces deux arêtes ont la même couleur, toutes les arêtes intermédiaires qui relient e et e' auront des couleurs différentes. Il y a $\ell + 1$ arêtes entre e et e'. Ainsi, le nombre de couleurs est au moins égal à $(\ell + 1) + 1$ couleurs. Nous avons, $\chi'_\ell(P_n) \geq \ell + 2$. Par conséquent, $\chi'_\ell(P_n) = \ell + 2$.

Cas(b) Si $n \leq \ell + 3$, toutes les arêtes de P_n sont à une distance au plus ℓ les unes des autres. Par conséquent, toutes ces arêtes auront des couleurs différentes. D'où, $\chi'_\ell(P_n) = |E(P_n)| = n - 1$. □

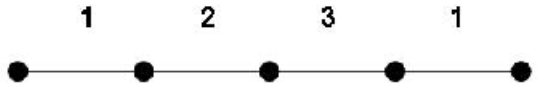

Figure 3.1 : *Coloration d'arêtes* 1*-distance de* P_5.

3.2 Coloration d'arêtes ℓ-distance d'un hypercube

3.2.1 Construction du sous-graphe $S_{\ell,d}$

Définition 3.2.1. Soient $\ell \geq 0$, $d \geq \ell + 1$ et H_d un hypercube à d-dimensions. Le sous-graphe $S_{\ell,d} \subset H_d$ est défini comme suit :

- $S_{\ell,d} = G \cup E'$, avec $G = H_{\ell+1}$ et $E' \subset E(H_d)$ sont des arêtes adjacentes à G.
- pour tout $v \in V(H_\ell)$ (avec $H_\ell \subset G$ et v est aussi une extrémité de E'), le degré de v dans $S_{\ell,d}$ est égal à d et $|E'| = (d - \ell - 1)\, 2^\ell$.

La Figure 3.2 montre un exemple de construction de quelques sous-graphes $S_{\ell,d}$.

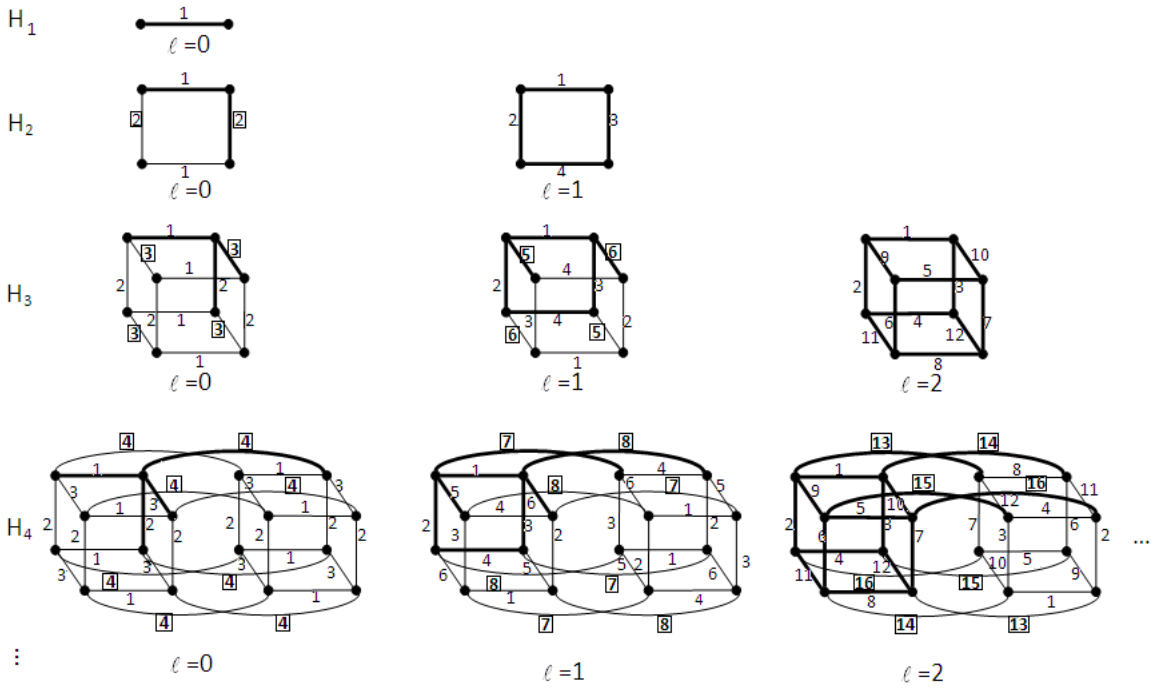

Figure 3.2 : *Exemples de la coloration d'arêtes ℓ-distance de $H_{i,1\leq i\leq 4}$. Le sous-graphe $S_{\ell,d}$ est donné par les arêtes en gras. Les couleurs encadrées sont les couleurs des arêtes connecteurs.*

Le lemme suivant présente quelques propriétés du sous-graphe $S_{\ell,d}$.

Lemme 3.2.2. *Soient $\ell \geq 0$, $d \geq \ell + 1$, H_d un hypercube et $S_{\ell,d} \subset H_d$ un sous graphe de H_d. Alors, $|E(S_{\ell,d})| = d2^\ell$ et pour tout $e, e' \in E(S_{\ell,d})$, $dist(e, e') \leq \ell$.*

Démonstration. Selon la Définition 3.2.1, le nombre d'arêtes de $S_{\ell,d}$ est égal au nombre d'arêtes de $H_{\ell+1}$ plus le nombre d'arêtes E', soit $(\ell+1)\,2^\ell + (d-\ell-1)\,2^\ell = d2^\ell$.

Pour tout $e, e' \in E(H_{\ell+1})$, nous avons $dist(e, e') \leq \ell$ et chaque arête de E' a une extrémité qui appartient à $V(H_{\ell+1})$. Par conséquent, pour tout $e, e' \in E(S_{\ell,d}), dist(e, e') \leq \ell$. □

3.2.2 Résultat général

Dans la suite, nous présentons le résultat principal de cette section.

Théorème 3.2.3. *Soit H_d un hypercube à d-dimension. Alors, l'indice ℓ-chromatique de H_d est donné par :*

$$\chi'_\ell(H_d) = \begin{cases} d2^{d-1} & \text{si } \ell \geq d-1, \quad (a) \\ d2^\ell & \text{si } \ell \leq d-1. \quad (b) \end{cases}$$

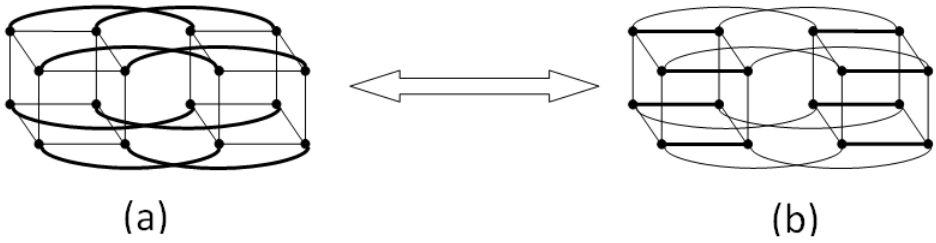

Figure 3.3 : *Changer la position des sommets ne modifie pas la forme de H_4. En effet, les arêtes connecteurs (les arêtes en gras) du premier hypercube (a) deviennent des arêtes de H_3^1 et H_3^2 dans le second hypercube (b).*

Démonstration. ***Cas (a)***. Soient $e, e' \in E(H_d)$ deux arêtes du d-cube H_d. Alors, la distance entre ces deux arêtes est au plus égale à d. Cependant, si $\ell \geq d-1$, alors $dist(e,e') \leq \ell$. Toutes les arêtes de H_d auront des couleurs différentes et $\chi'_\ell(H_d) = |E(H_d)| = d2^{d-1}$.

Cas (b). Pour satisfaire la coloration d'arêtes ℓ-distance de l'hypercube H_d, nous appliquons un algorithme récursif de $d-\ell$ itérations. Durant la première itération, nous colorons le sous-graphe $H_{\ell+1}$ avec $|E(H_{\ell+1})|$ couleurs. Puis, à chaque itération i, nous colorons l'hypercube $H_{\ell+i+1},\ i \in [1, d-\ell-1]$ comme suit :

1. colorer $H^2_{\ell+i}$ avec les couleurs de $H^1_{\ell+i}$ comme suit : soit $e^1 \in (H^1_{\ell+i})$ et $e^2 \in (H^2_{\ell+i})$ (e^2 est l'image de e^1) : si $c(e^1) \in \left[1, (\ell+1)2^\ell\right]$ alors $c(e^2) = (\ell+1)2^\ell + 1 - c(e^1)$ sinon pour $k \in \mathbb{N}^*$:$c(e^1) \in \left[(\ell+k)2^\ell + 1, (\ell+k+1)2^\ell\right]$ alors $c(e^2) = (\ell+k+\frac{1}{2})2^{\ell+1} + 1 - c(e^1)$.
2. colorer les arêtes qui connectent $H^1_{\ell+i}$ et $H^2_{\ell+i}$. D'après la Figure 3.3, pour colorer les arêtes connecteurs, nous utilisons la même technique de coloration qui colore les deux hypercubes $H^1_{\ell+i}$ et $H^2_{\ell+i}$. En effet, nous pouvons modifier la position des sommets de l'hypercube de manière à ce que les arêtes connecteurs appartiennent aux deux hypercubes de dimensions inférieures. Nous utilisons 2^ℓ nouvelles couleurs. La coloration est une coloration d'arêtes ℓ-distance, puisque les couleurs utilisées pour colorer les deux $(d-1)$-cubes et les arêtes connecteurs sont différentes.

Pour $\ell \leq d-1$, nous avons pour tout $e^1 \in E\left(H^1_{d-1}\right)$, il existe $e^2 \in E\left(H^2_{d-1}\right)$ telle que $dist(e,e') > \ell$. Par conséquent, ces deux arêtes peuvent avoir la même couleur, i.e. $c(e^1) = c(e^2)$.

Selon notre algorithme de coloration, les arêtes de $H_{\ell+1}$ ont des couleurs différentes. Ainsi, le nombre total de couleurs utilisées dans H_d est égal à la somme des couleurs des arêtes connecteurs (i.e. $(d-(\ell+1))\,2^\ell$) et des couleurs de $H_{\ell+1}$ (i.e. $\left((\ell+1)2^\ell\right)$). Ainsi, nous avons la borne inférieure de l'indice ℓ-chromatique, $\chi'_\ell(H_d) \leq d2^\ell$.

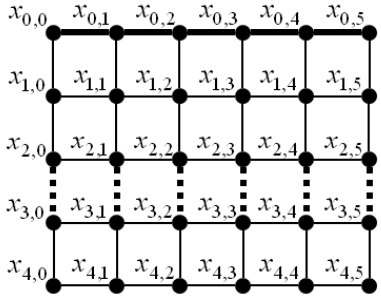

Figure 3.4 : *$M_{5,6}$ est une grille à deux dimensions. Les arêtes continues en gras sont un exemple de la séquence des arêtes horizontales $\mathcal{H}_0 = ((x_{0,0}, x_{0,1}), (x_{0,1}, x_{0,2}), (x_{0,2}, x_{0,3}), (x_{0,3}, x_{0,4}), (x_{0,4}, x_{0,5}))$ et les arêtes pointillées en gras sont un exemple de la séquence des arêtes verticales $\mathcal{V}_2 = ((x_{2,0}, x_{3,0}), (x_{2,1}, x_{3,1}), (x_{2,2}, x_{3,2}), (x_{2,3}, x_{3,3}), (x_{2,4}, x_{3,4}), (x_{2,5}, x_{3,5}))$.*

D'après le Lemme 3.2.2, l'indice ℓ-chromatique est donné par $\chi'_\ell(H_d) \geq d2^\ell$. Par conséquent, nous pouvons déduire la valeur exacte de l'indice ℓ-chromatique et $\chi'_\ell(H_d) = d2^\ell$. □

3.3 Grilles à deux dimensions

3.3.1 Préliminaires

Nous utilisons les notations suivantes le long de cette partie. La grille à deux dimensions $M_{n,m} = G(V(G), E(G))$ admet un ensemble de sommets $V(G) = \{x_{i,j}$ with $0 \leq i \leq n-1,\ 0 \leq j \leq m-1\}$ et un ensemble d'arêtes $E(G) = \{(x_{i,j}, x_{i,j+1}),\ 0 \leq i \leq n-1,\ 0 \leq j \leq m-2\} \cup \{(x_{i,j}, x_{i+1,j}), 0 \leq i \leq n-2, 0 \leq j \leq m-1\}$.

Nous notons par $\mathcal{V}_k$, avec $0 \leq k \leq n-2$, la séquence des arêtes verticales de $M_{n,m}$, $\mathcal{V}_k = ((x_{k,0}, x_{k+1,0}), ..., (x_{k,m-1}, x_{k+1,m-1}))$. Pour une coloration propre c de $M_{n,m}$, nous écrivons $c(\mathcal{V}_k) = (1, 2, \ldots m)$ pour dire que la première arête de $\mathcal{V}_k$ est colorée par la couleur 1, la seconde arête de $\mathcal{V}_k$ par la couleur 2 et ainsi de suite. Similairement, nous notons par $\mathcal{H}_k$, avec $0 \leq k \leq n-1$, la séquence des arêtes horizontales de $M_{n,m}$, $\mathcal{H}_k = ((x_{k,0}, x_{k,1}), ..., (x_{k,m-2}, x_{k,m-1}))$. Nous notons $c(\mathcal{H}_k) = (1, 2, \ldots m-1)$ l'affectation des couleurs de $\mathcal{H}_k$, pour dire que la première arête de $\mathcal{H}_k$ est colorée par la couleur 1, la seconde arête de $\mathcal{H}_k$ par la couleur 2 et ainsi de suite (voir Figure 3.4).

3.3.2 Construction du sous-graphe S_ℓ

Nous donnons tout d'abord quelques définitions que nous utilisons par la suite pour prouver la borne inférieure de l'indice ℓ-chromatique de la grille à deux di-

mensions. Comme pour les hypercubes, nous définissons dans un premier temps un sous-graphe tel que la distance entre n'importe quelle paire d'arêtes est au plus égale à ℓ. La borne inférieure de l'indice ℓ-chromatique est ainsi égale au nombre d'arêtes de ce sous-graphe. Nous distinguons deux cas selon les valeurs de n et m.

Définition 3.3.1. Soient $\ell \geq 0$, $n, m \geq \ell+3$ et $M_{n,m}$ une grille à deux dimensions. Soit $v = x_{i,j} \in V(M_{n,m})$ avec $\left\lfloor \frac{\ell}{2} \right\rfloor + 1 \leq i \leq n - \left\lfloor \frac{\ell}{2} \right\rfloor - 1$ et $\left\lfloor \frac{\ell}{2} \right\rfloor + 1 \leq j \leq m - \left\lfloor \frac{\ell}{2} \right\rfloor - 2$. Alors, $Q_v = \{u \in V(M_{n,m}) : dist(u,v) \leq \left\lfloor \frac{\ell}{2} \right\rfloor + 1\}$. Nous définissons le sous-graphe $S_\ell \subset M_{n,m}$ comme suit :

$$S_\ell = \begin{cases} M_{n,m}[Q_v] & \text{si } \ell \text{ est pair,} \\ M_{n,m}[Q_v] \cup M_{n,m}[Q_{v'}] \cup \{e_\ell\} & \text{si } \ell \text{ est impair,} \end{cases}$$

tel que $v = x_{i,j}$, $v' = x_{i,j+1}$ et $e_\ell = (x_{i+\lfloor \frac{\ell}{2} \rfloor + 1, j}, x_{i+\lfloor \frac{\ell}{2} \rfloor + 1, j+1})$.

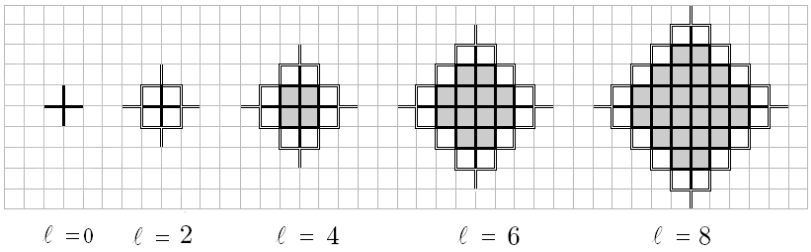

Figure 3.5 : *Exemples de sous-graphes S_ℓ dans le cas ℓ pair. Pour ℓ pair fixé, le sous-graphe S_ℓ contient le sous-graphe $S_{\ell-2}$ (traits pleins) + les arêtes identifiées par les doubles traits.*

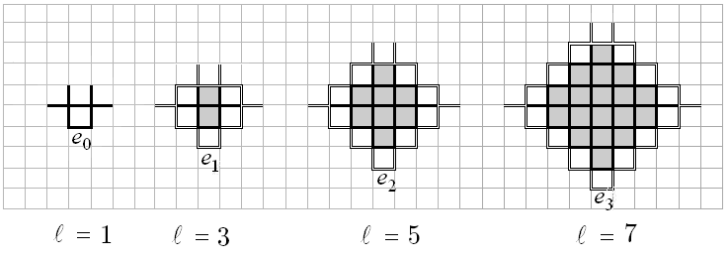

Figure 3.6 : *Exemples de sous-graphes S_ℓ dans le cas ℓ impair. Pour ℓ impair fixé, le sous-graphe S_ℓ contient le sous-graphe $S_{\ell-2}$ (traits pleins) + les arêtes identifiées par les doubles traits.*

La Figure 3.5 (respectivement la Figure 3.6) donne quelques exemples de construction de S_ℓ lorsque ℓ est pair (respectivement ℓ est impair).

Si $n < \ell + 3$ ou $m < \ell + 3$, nous ne pouvons pas appliquer la Définition 3.3.1. Uniquement une partie de S_ℓ que nous notons $S"_\ell$ peut être construite. De plus, si $n+m < \ell+4$ ou ($n+m = \ell+4$ et n ou $m \neq 2$), alors $S"_\ell$ contient toutes les arêtes

de la grille (voir Théorème 3.3.6). La Définition 3.3.2 illustre comment construire le sous-graphe $S"_\ell$.

Définition 3.3.2. Soient $\ell \geq 0$, $M_{n,m'}$ et $M_{n,m}$ deux grilles à deux dimensions. Soient $n < \ell + 3$ et $M_{n,m'} \subset M_{n,m}$ avec $m' = m$ si $m < \ell + 3$ et $m' = \ell + 3$ si $m \geq \ell + 3$. Le sous-graphe $S"_\ell$ est défini par :

$$S"_\ell = \begin{cases} M_{n,m'} - (\Re_a \cup \Re_b \cup \Re_c \cup \Re_d \cup (x_{\ell+1,c}, x_{\ell+1,c+1})) & \text{si } n = \ell + 2 \text{ et } \ell \text{ est impair,} \\ M_{n,m'} - (\Re_a \cup \Re_b \cup \Re_c \cup \Re_d) & \text{sinon,} \end{cases}$$

où $a, b, c, d \in \mathbb{N}$ et $\Re_a, \Re_b, \Re_c, \Re_d \subset M_{n,m'}$ sont les quatre *corners* de $M_{n,m'}$ qui ne sont pas dans $S"_\ell$. Un *corner* $\Re_i, i \in \{a, b, c, d\}$ est un sous-graphe de $M_{n,m'}$ défini comme suit : soit $v \in M_{n,m'}$ un sommet de degré 2 et $Q'_v = \{u \in V(M_{n,m'}) : dist(u, v) \leq i\}$. Alors, $\Re_i = M_{n,m'}[Q'_v]$. Un *corner* est un sous-graphe qui appartient à $M_{n,m'}$ et n'appartient pas à $S"_\ell$. Chaque *corner* $\Re_i, i \in \{a, b, c, d\}$ est formé de deux côtés ayant le même nombre d'arêtes égal à $i = \frac{n+m'-\ell-4+\theta_i}{2}$, avec $\theta_i \in \{-1, 0, 1\}$, $i \in \{a, b, c, d\}$ et $\theta_i = \pm(n + m' + \ell) \bmod 2$, $\theta_a = -\theta_d$ et $\theta_b = -\theta_c$ (voir Figure 3.7).

La Figure 3.8 montre quelques exemples de $S"_\ell$ pour différentes valeurs de ℓ, n et m'.

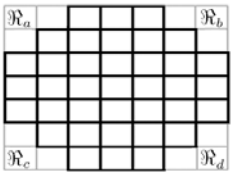

Figure 3.7 : $\ell = 8$, $n = 8$ et $m = m' = 8$. Le sous-graphe $S"_\ell$ est donné par les arêtes en gras. Les autres arêtes appartiennent aux corners $\Re_a, \Re_b, \Re_c$ ou $\Re_d$.

3.3.3 Résultats préliminaires

Le lemme suivant présente quelques propriétés du sous-graphe S_ℓ.

Lemme 3.3.3. *Pour $n, m \geq \ell + 3$, soit $S_\ell \subset M_{n,m}$ un sous-graphe de $M_{n,m}$. Alors, $|E(S_\ell)| = (\ell + 2)^2 - (\ell \bmod 2)$ et pour tout $e, e' \in E(S_\ell)$, $dist(e, e') \leq \ell$.*

Démonstration. ***Cas 1*** Supposons que ℓ est pair. Soit $\ell = 2\ell'$. Nous pouvons vérifier que $|E(S_{\ell'+1})| = |E(S_{\ell'})| + 4(2(\ell' + 1) + 1)$ pour tout $i \geq 0$ (voir Figure 3.5). Nous avons ainsi une suite arithmétique et $|E(S_\ell)| = \sum_{i=0}^{\ell'} 4(2i+1) = (2\ell'+2)^2 = (\ell+2)^2$.

Cas 2 Supposons que ℓ est impair. Soit $\ell = 2\ell'+1$. Comme dans le premier cas, nous avons $|E(S_{\ell'+1})| = |E(S_{\ell'})| + 8\,(\ell' + 2)$. Par conséquent, $|E(S_\ell)| = \sum_{i=0}^{\ell'} 8(i + 1) = (2\ell' + 2)(2\ell' + 4) = (\ell + 2)^2 - 1$.

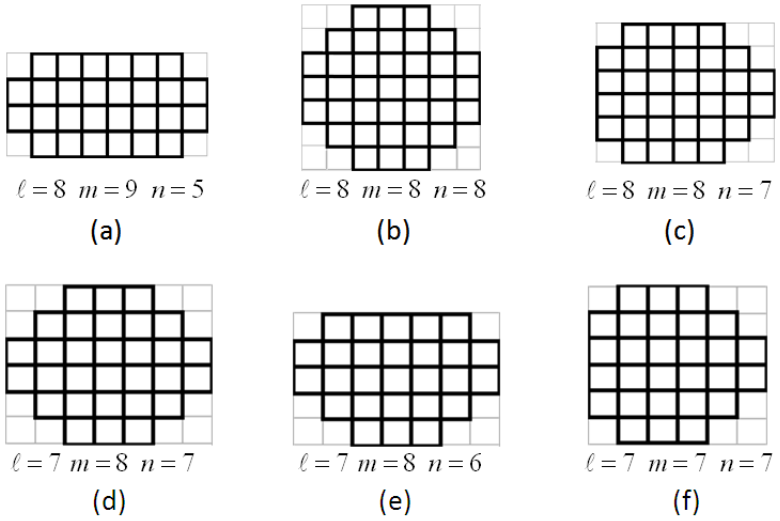

Figure 3.8 : Exemples de $\Re$ en fonction de n, $m = m'$ et ℓ. Le sous-graphe $S"_\ell$ est donné par les arêtes en gras.

D'après la Définition 3.3.1, la distance entre chaque paire d'arêtes qui appartiennent à $E(S_\ell)$ est égale au plus à ℓ. Ainsi, pour tout $e, e' \in E(S_\ell)$, $dist(e, e') \leq \ell$. □

Le lemme suivant présente quelques propriétés du sous-graphe $S"_\ell$.

Lemme 3.3.4. *Pour $n + m > \ell + 4$, soit $S"_\ell \subset M_{n,m'}$ un sous-graphe de $M_{n,m'} \subset M_{n,m}$ tel que $m' = m$ si $m \leq \ell + 2$ et $m' = \ell + 3$ si $m > \ell + 2$. Alors, pour tout $e, e' \in E(S"_\ell)$, $dist(e, e') \leq \ell$ et*

$$|E(S"_\ell)| = \begin{cases} |E(M_{n,m'})| - \frac{1}{4}\sum_{i\in\{a,b,c,d\}}((n+m'-\ell-3-\theta_i)^2-1) & \\ -(\ell \bmod 2) & \text{si } n = \ell+2, \quad (a) \\ |E(M_{n,m'})| - \frac{1}{4}\sum_{i\in\{a,b,c,d\}}((n+m'-\ell-3-\theta_i)^2-1) & \text{si } n < \ell+2, \quad (b) \end{cases}$$

avec $\theta_i = \pm(n + m' + \ell) \bmod 2$, $\theta_a = -\theta_d$ et $\theta_b = -\theta_c$, $i = \frac{n+m'-\ell-4+\theta_i}{2}$ et $i \in \mathbb{N}$.

Démonstration. Selon la Définition 3.3.2, chaque *corner* définit deux suites arithmétiques égales. La première est composée des arêtes qui appartiennent à la séquence des arêtes verticales et la deuxième à la séquence des arêtes horizontales. Le nombre d'éléments pour chacune est égal à $i = \frac{n+m'-\ell-4+\theta_i}{2}$ tel que $i \in \{a, b, c, d\}, i \in \mathbb{N}$. La somme des arêtes de chaque *corner* est égale à $|E(\Re_i)| = \frac{1}{4}((n+m'-\ell-3-\theta_i)^2-1)$, $\theta_i \in \{-1, 0, 1\}$.

Si $n = \ell + 2$ et ℓ est impair, alors les arêtes qui n'appartiennent pas à $S"_\ell$ sont $E(\Re_a), E(\Re_b), E(\Re_c), E(\Re_d)$ et e'. L'arête e' n'appartient pas $S"_\ell$ car la distance entre les arêtes $e = (x_{0,a}, x_{0,a+1})$ et $e' = (x_{n-1,m'-d-2}, x_{n-1,m'-d-1})$ est égale

à $dist(e, e') = \ell - \frac{\theta_a + \theta_d}{2} = \ell$ (car d'après la Définition 3.3.2 $\theta_a = -\theta_d$) (voir Figure 3.9). □

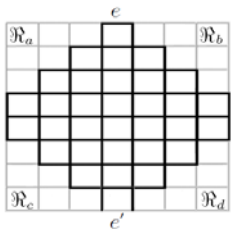

Figure 3.9 : $\ell = 7$, $m = 8$ et $n = 9$.$e = (x_{0,3}, x_(0, 4))$ et $e' = (x_{8,3}, x_{8,4})$. Le sous-graphe $S"_\ell$ est donné par les arêtes en gras. Les autres arêtes appartiennent aux corners $\Re_a, \Re_b, \Re_c, \Re_d$ ou à e'.

3.3.4 Grilles de dimensions $\geq \ell + 3$

Dans cette partie, nous donnons la valeur exacte de l'indice ℓ-chromatique d'une grille à deux dimensions $M_{n,m}$ pour différentes valeurs de n et m. Nous traitons le cas où les dimensions sont au moins de $\ell + 3$. Il est donné par le théorème suivant.

Théorème 3.3.5. *Soit $M_{n,m}$ une grille à deux dimensions telle que $n, m \geq \ell + 3$. Alors, l'indice ℓ-chromatique de $M_{n,m}$ est donné par :*

$$\chi'_\ell(M_{n,m}) = (\ell + 2)^2 - (\ell \mod 2).$$

Démonstration. Pour obtenir la borne inférieure de $\chi'_\ell(M_{n,m})$, nous considérons le sous-graphe $S_\ell \subset M_{n,m}$ défini dans la Définition 3.3.1. Selon le Lemme 3.3.3, pour tout $e, e' \in E(S_\ell)$, $dist(e, e') \leq \ell$ et $|E(S_\ell)| = (\ell + 2)^2 - (\ell \mod 2)$. Ainsi, toutes les arêtes du sous-graphe S_ℓ auront des couleurs différentes. Par conséquent, $\chi'_\ell(M_{n,m}) \geq (\ell + 2)^2 - (\ell \mod 2)$.

Pour la borne supérieure, si ℓ est pair, la coloration c de $\mathcal{H}_k, 0 \leq k \leq n - 1$ est définie comme suit :

- Pour $0 \leq k \leq \frac{\ell}{2}$, soit Q l'ensemble des couleurs défini de la manière suivante : $Q = \{k(\ell+2)+1, k(\ell+2)+2, \cdots, (k+1)(\ell+2)\}$. Nous colorons d'une manière cyclique les arêtes de $\mathcal{H}_k$ avec les couleurs de Q.
- Pour $\frac{\ell}{2} + 1 \leq k \leq \ell + 1$, soit Q_1 l'ensemble des couleurs défini de la manière suivante :
  $Q_1 = \left\{\left(k - \frac{\ell}{2} - 1\right)(\ell + 2) + 1, \left(k - \frac{\ell}{2} - 1\right)(\ell + 2) + 2, \cdots, \left(k - \frac{\ell}{2}\right)(\ell + 2)\right\}$.
  Dans ce cas, nous colorons les $\frac{\ell}{2} + 1$ premières arêtes de $\mathcal{H}_k$ avec les $\frac{\ell}{2} + 1$ dernières couleurs de Q_1. Puis, nous colorons d'une manière cyclique les autres arêtes de $\mathcal{H}_k$ avec les couleurs de Q_1.

– Pour $k > \ell + 1$: $c(\mathcal{H}_k) = c(\mathcal{H}_{k \bmod (\ell+2)})$.

La coloration des arêtes verticales $\mathcal{V}_k$ est similaire à celle des arêtes horizontales $\mathcal{H}_k$ décrite précédemment. Il suffit seulement de considérer la grille $M_{m,n}$ au lieu de $M_{n,m}$ et utiliser des couleurs différentes. La Figure 3.10 montre un exemple de cette coloration. Comme $c(\mathcal{H}_k) = c(\mathcal{H}_{k \bmod (\ell+2)})$ pour $k > \ell + 1$, nous traitons le cas $k \leq \ell+1$. D'abord, il suffit de vérifier qu'il n'existe aucun conflit de couleurs entre les arêtes de $\mathcal{H}_k$. Pour cela, nous devons prouver que pour tout $e_i = (x_{k,i}, x_{k,i+1}) \in \mathcal{H}_k$, pour tout $e_j = (x_{k',j}, x_{k',j+1}) \in \mathcal{H}_{k'}$ telles que $c(e_i) = c(e_j)$, nous avons $dist(e_i, e_j) = |j - i - 1| + |k' - k| > \ell$. D'après l'algorithme de coloration, il existe deux cas en fonction de k et k'.

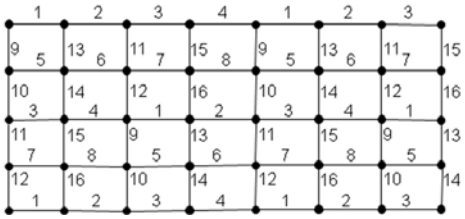

Figure 3.10 : *La coloration d'arêtes* 2*-distance de* $M_{5,8}$.

Cas $k = k'$. Deux arêtes qui appartiennent à une même séquence horizontale et qui ont la même couleur sont à distance $\ell + 2$.

Cas $k \neq k'$. Il suffit d'étudier les deux cas suivants $0 \leq k \leq \frac{\ell}{2}$ et $\frac{\ell}{2}+1 \leq k' \leq \ell+1$. D'après l'algorithme de coloration, $\mathcal{H}_k$ et $\mathcal{H}_{k'}$ ont le même ensemble de couleurs s'ils sont à distance $\frac{\ell}{2} + 1$, ainsi $k' = k + \frac{\ell}{2} + 1$. Nous avons aussi, pour tout $e_i \in \mathcal{H}_k$, $c(e_i) = 1+i+k(\ell+2)$, avec $0 \leq i \leq \ell+1$. Sans perte de généralité, nous considérons l'intervalle $\left[\frac{\ell}{2} + 1, \frac{3\ell}{2} + 3\right]$ pour avoir toutes les couleurs de Q_1. Dans ce cas, pour tout $e_i \in \mathcal{H}_k, c(e_j) = (k' - \frac{\ell}{2} - 1)(\ell + 2) + j - \frac{\ell}{2} - 1$, avec $\frac{\ell}{2} + 1 \leq j \leq \frac{3\ell}{2} + 3$. Si deux arêtes e_i et e_j ont la même couleur (i.e. $c(e_i) = c(e_j)$), cela signifie que $k(\ell + 2) + i = (k' - \frac{\ell}{2} - 1)(\ell + 2) + j - \frac{\ell}{2} - 1$, c'est à dire $i = j - \frac{\ell}{2} - 1$. Ainsi, $dist(e_i, e_j) = \ell + 1 > \ell$.

Pour la coloration des arêtes verticales $\mathcal{V}_k$, si nous considérons la grille $M_{m,n}$, alors le principe de coloration est similaire à celui de $\mathcal{H}_k$. Par conséquent, la coloration de $\mathcal{V}_k$ satisfait aussi la coloration d'arêtes ℓ-distance.

Le nombre de couleurs utilisées pour colorer les séquences $\mathcal{H}_k$ est égal à $(\frac{\ell}{2} + 1)$ $(\ell+2)$ couleurs et le nombre de couleurs de $c(\mathcal{V}_k)$ est égal à $\left(\frac{\ell}{2} + 1\right)(\ell + 2)$ couleurs. Le nombre total de couleurs est ainsi égal à $2\left(\frac{\ell}{2} + 1\right)(\ell + 2) = (\ell + 2)^2$ couleurs. Ainsi, $\chi'_\ell(M_{n,m}) \leq (\ell + 2)^2$ pour ℓ pair. De plus, d'après la valeur de la borne inférieure, nous avons $\chi'_\ell(M_{n,m}) \geq (\ell + 2)^2$. Par conséquent, $\chi'_\ell(M_{n,m}) = (\ell + 2)^2$ pour le cas ℓ pair.

Si ℓ est impair, la coloration c de $\mathcal{V}_k, 0 \leq k \leq n - 2$ est définie comme suit :

- Pour $0 \leq k \leq \left\lceil \frac{\ell}{2} \right\rceil$, soit $Q' = \{k(\ell+1)+1, k(\ell+1)+2, \cdots, (k+1)(\ell+1)\}$. Nous colorons d'une manière cyclique les arêtes de $\mathcal{V}_k$ avec les couleurs de Q'.
- Pour $\left\lceil \frac{\ell}{2} \right\rceil + 1 \leq k \leq \ell+2$, soit $Q'_1 = \{\left(k - \left\lceil \frac{\ell}{2} \right\rceil - 1\right)(\ell+1)+1, \left(k - \left\lceil \frac{\ell}{2} \right\rceil - 1\right)(\ell+1)+2, \cdots, \left(k - \left\lceil \frac{\ell}{2} \right\rceil\right)(\ell+1)\}$. Nous commençons la coloration des $\left\lceil \frac{\ell}{2} \right\rceil + 1$ premières arêtes de $\mathcal{V}_k$ avec les $\left\lceil \frac{\ell}{2} \right\rceil + 1$ dernières couleurs de Q'_1. Puis, nous colorons d'une manière cyclique les autres arêtes de $\mathcal{V}_k$ avec Q'_1.
- Pour $k > \ell + 2 : c(\mathcal{V}_k) = c(\mathcal{V}_{k \bmod (\ell+3)})$.

La coloration des arêtes horizontales $\mathcal{H}_k$ est similaire à celle des arêtes verticales $\mathcal{V}_k$ décrite précédemment. Il suffit seulement de considérer la grille $M_{m,n}$ au lieu de $M_{n,m}$ et d'utiliser des couleurs différentes. La Figure 3.11 montre un exemple de cette coloration. Nous procédons de la même manière que le cas pair. Comme $c(\mathcal{V}_k) = c(\mathcal{V}_{k \bmod (\ell+3)})$ pour $k > \ell + 2$, il suffit de s'intéresser dans cette preuve au cas $k \leq \ell + 2$. Nous devons montrer que pour tout $e_i = (x_{i,k}, x_{i+1,k}) \in \mathcal{V}_k$ et $e_j = (x_{j,k'}, x_{j+1,k'}) \in \mathcal{V}_{k'}$ telles que $c(e_i) = c(e_j)$ alors $dist(e_i, e_j) = |j - i - 1| + |k' - k| > \ell$. Selon $c(\mathcal{V}_k)$, Il y a deux cas en fonction de k et k'.

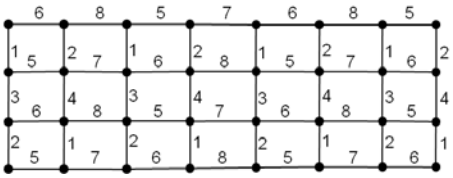

Figure 3.11 : *La coloration d'arêtes* 1*-distance de* $M_{4,8}$.

Cas $k = k'$. Deux arêtes qui appartiennent à une même séquence verticale et qui ont la même couleur sont à distance $\ell + 1$.

Cas $k \neq k'$. Il suffit d'étudier les deux cas suivants $0 \leq k \leq \left\lceil \frac{\ell}{2} \right\rceil$ et $\left\lceil \frac{\ell}{2} \right\rceil + 1 \leq k' \leq \ell + 2$. D'après l'algorithme de coloration, $\mathcal{V}_k$ et $\mathcal{V}_{k'}$ ont le même ensemble de couleurs s'ils sont à distance$\left\lceil \frac{\ell}{2} \right\rceil + 1$, ainsi $k' = k + \left\lceil \frac{\ell}{2} \right\rceil + 1$. Nous avons aussi, pour tout $e_i \in \mathcal{V}_k$,$c(e_i) = 1 + k(\ell+1) + i$, avec $0 \leq i \leq \ell + 2$. Sans perte de généralité, nous considérons l'intervalle $\left[\left\lceil \frac{\ell}{2} \right\rceil + 1, \left\lceil \frac{3\ell}{2} \right\rceil\right]$ pour avoir toutes les couleurs de Q'_1. Dans ce cas, pour tout $e_j \in \mathcal{V}_{k'}$,$c(e_j) = 1 + (k' - \left\lceil \frac{\ell}{2} \right\rceil - 1)(\ell+1) + j - \left\lceil \frac{\ell}{2} \right\rceil - 1$, avec $\left\lceil \frac{\ell}{2} \right\rceil + 1 \leq j \leq \left\lceil \frac{3\ell}{2} \right\rceil$. Si deux arêtes e_i et e_j ont la même couleur (i.e. $c(e_i) = c(e_j)$), cela signifie que $k(\ell+1)+i = (k' - \left\lceil \frac{\ell}{2} \right\rceil - 1)(\ell+1) + j - \left\lceil \frac{\ell}{2} \right\rceil$, c'est à dire $i = j - \left\lceil \frac{\ell}{2} \right\rceil$. Ainsi, $dist(e_i, e_j) = \ell + 1 > \ell$.

Pour la coloration des arêtes horizontales $\mathcal{H}_k$, si nous considérons la grille $M_{m,n}$, alors le principe de coloration est similaire à celui de $\mathcal{V}_k$. Par conséquent, la coloration de $\mathcal{H}_k$ satisfait aussi la coloration d'arêtes ℓ-distance.

Le nombre de couleurs utilisées pour colorer les séquences $\mathcal{V}_k$ est égal à $\left(\left\lceil \frac{\ell}{2} \right\rceil + 1\right)(\ell+1)$ couleurs et le nombre de couleurs de $c(\mathcal{H}_k)$ est égal à $\left(\left\lceil \frac{\ell}{2} \right\rceil + 1\right)(\ell+1)$ couleurs. Le nombre total de couleurs est ainsi égal à $(\ell+2)^2 - 1$ couleurs. Ainsi,

$\chi'_\ell(M_{n,m}) \leq (\ell+2)^2 - 1$ pour ℓ impair. De plus, d'après la valeur de la borne inférieure, nous avons $\chi'_\ell(M_{n,m}) \geq (\ell+2)^2 - 1$. Par conséquent, $\chi'_\ell(M_{n,m}) = (\ell+2)^2 - 1$ pour le cas ℓ impair. □

3.3.5 Grilles d'une dimension au moins $\leq \ell + 2$

Dans cette sous-section, nous supposons que $n \leq \ell + 2$ et nous varions la valeur de m. Différents cas sont présentés dans le théorème suivant.

Théorème 3.3.6. *Soit $M_{n,m}$ une grille à deux dimensions. Alors, l'indice ℓ-chromatique de $M_{n,m}$ est donné par :* $\chi'_\ell(M_{n,m}) =$

$$\begin{cases} |E(M_{n,m})| & \text{si } n+m < \ell+4 \\ & \quad ou\ (n+m = \ell+4 \ et\ n \neq 2), \quad (a) \\ |E(M_{n,m})| - (n+m-\ell-3)^2 & \\ -(\ell+n+m) \ \ mod\ 2+1 & \text{si } n+m > \ell+4 \ et\ n, m < \ell+2, \quad (b) \\ |E(M_{n,\ell+3})| - n^2 + n \ \ mod\ 2 & \text{si } n+m > \ell+4 \ et\ n < \ell+2,\ m > \ell+2\ , \quad (c) \\ |E(M_{n,m})| - (m-1)^2 + \ell \ \ mod\ 2-1 & \text{si } n, m = \ell+2, \quad (d) \\ |E(M_{n,m})| - (m-1)^2 & \text{si } 1 < m < \ell+2 \ et\ n = \ell+2, \quad (e) \\ |E(M_{\ell+2,\ell+3})| - (\ell+2)^2 + 1 & \text{si } m > \ell+2 \ et\ n = \ell+2. \quad (f) \end{cases}$$

Démonstration. Nous démontrons chaque cas séparément :

Cas (a) Si $n+m < \ell+4$ la plus grande distance entre les arêtes est la distance entre l'arête $e'_i = (x_{1,1}, x_{1,2})$ et $e'_j = (x_{n,m-1}, x_{n,m})$ si $n \leq m$. La distance est égale à $n+m-4 \leq \ell$. Si $n = 2$ et $m = \ell+2$, la distance entre toute paire d'arêtes dans la grille est au plus égale à ℓ sauf les arêtes $e_i = (x_{1,1}, x_{2,1})$ et $e_j = (x_{n-1,m}, x_{n,m})$. Par conséquent, si $n+m < \ell+4$ ou $(n+m = \ell+4$ et $n \neq 2)$, toutes les arêtes auront des couleurs différentes et $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| = n(m-1) + m(n-1)$.

Cas (b) $n+m > \ell+4$ et $n, m < \ell+2$. Si ℓ est pair et n, m ont la même parité ou ℓ est impair et n, m ont des parités différentes, alors $\chi'_\ell(M_{n,m}) \geq |E(M_{n,m})| - (n+m-\ell-3)^2$. Si ℓ est impair et n, m ont des parités différentes ou ℓ est pair et n, m ont la même parité, alors $\chi'_\ell(M_{n,m}) \geq |E(M_{n,m})| - (n+m-\ell-3)^2 + 1$. (voir Lemme 3.3.4). L'algorithme de coloration consiste à colorer tout d'abord les arêtes de $S"_\ell$ avec des couleurs différentes, puis nous colorons les arêtes des quatre *corners* $\Re_a$, $\Re_b$, $\Re_c$ et $\Re_d$ avec les couleurs de $S"_\ell$. Nous détaillons la colorations des arêtes qui appartiennent aux séquences horizontales $\mathcal{H}_k$, le même raisonnement peut être appliqué aux arêtes qui appartiennent aux séquences verticales $\mathcal{V}_k$. Nous développons la coloration du *corner* $\Re_a$ de la grille telle que ℓ, m et n sont pairs. Pour colorer chaque arête de $\Re_a$, nous utilisons les couleurs des arêtes de $S"_\ell$ qui sont à distance $\ell+1$. Pour $j \in [1, a]$, soient l'ensemble des arêtes

$\kappa = \{(x_{p,a-p-j}, x_{p,a-p-j+1})\}_{p=0\,..\,a-j}$ qui appartiennent à $\Re_a$ et l'ensemble des arêtes $\eta = \{(x_{n-d+q-j,m-q-2}, x_{n-d+q-j,m-q-1})\}_{q=0\,..\,d+j-1}$ qui appartiennent à $S"_\ell$. Il est facile de voir que pour tout $e \in \kappa$ et $e' \in \eta$, la distance entre toute paire d'arêtes est égale à

$$\begin{aligned} dist(e,e') &= n-d+q-j+m-q-3-(p+a-p-j)\,, \\ &= n+m+a-d-3, \\ &= n+m+\tfrac{n+m-\ell-4+\theta_a}{2}-\tfrac{n+m-\ell-4+\theta_d}{2}-3, \\ &= \ell+1. \end{aligned}$$

Pour tout $j \in [1,a]$, le nombre de couleurs des arêtes qui appartiennent à η est égal à $|\eta| = d+j$, qui est supérieur au nombre d'arêtes à colorer qui est égal à $|\kappa| = a-j+1$. Ainsi, nous affectons les couleurs des arêtes de l'ensemble η aux arêtes de l'ensemble κ. Par conséquent, $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| - (n+m-\ell-3)^2 + 1$.

Nous utilisons ce même principe de coloration dans le cas (d) et le cas (e).

Cas (d) Si $n,m = \ell+2$ et ℓ est pair, alors les couleurs de $S"_\ell$ sont suffisantes pour colorer la grille $M_{n,m}$ et $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| - (m-1)^2 - 1$. Si $n,m = \ell+2$ et ℓ est impair, nous ajoutons une seule couleur pour colorer la grille $M_{n,m}$ et $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| - (m-1)^2$.

Cas (e) Si $1 < m < \ell+2$ *et* $n = \ell+2$, alors si ℓ, m ont la même parité et $n = \ell+2$ nous ajoutons une seule couleur pour colorer la grille $M_{n,m}$. Pour les autre cas, les couleurs de $S"_\ell$ sont suffisantes. Par conséquent, $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| - (m-1)^2$.

Cas (c,f) Si $n+m > \ell+4$, $n < \ell+2$ et $m > \ell+2$ et selon le Lemme 3.3.4, $|E(S"_\ell)|$ dépend uniquement de n et ℓ. Par conséquent, si n est pair alors $\chi'_\ell(M_{n,m}) \leq |E(M_{n,\ell+3})| - n^2$ et si n est impair alors $\chi'_\ell(M_{n,m}) \leq |E(M_{n,\ell+3})| - n^2 + 1$.

Pour la borne supérieure, la coloration c de $\mathcal{H}_k$, $0 \leq k \leq n-1$ est la suivante :

$c(e_j) = c((x_{k,j}, x_{k,j+1})) = (1+j+k(2\ell-n+5-n \bmod 2)) \bmod \phi$, $0 \leq k \leq \left\lceil \frac{n}{2} \right\rceil - 1$, $0 \leq j \leq m-2$, et $c(e_j) = c((x_{k,j}, x_{k,j+1})) = (1+j+\varphi+k(2\ell-n+5-n \bmod 2)) \bmod \phi$, $\left\lceil \frac{n}{2} \right\rceil \leq k \leq n-1$, $0 \leq j \leq m-2$ tel que $\phi = n(\ell+2) - \frac{n^2}{2}$ et $\varphi = \ell - n + 2$ si n est pair et $\phi = n(\ell+2) - \frac{n^2}{2} + \frac{1}{2}$ et $\varphi = 0$ si n est impair.

Pour vérifier qu'il n'existe pas de conflit entre les arêtes des séquences horizontales, il suffit de prouver que pour tout $e_i = (x_{k,i}, x_{k,i+1}) \in \mathcal{H}_k$ et $e_j = (x_{k',j}, x_{k',j+1}) \in \mathcal{H}_{k'}$ tel que $c(e_i) = c(e_j)$ alors $dist(e_i,e_j) = |j-i-1| + |k'-k| > \ell$, $1 \leq d = k'-k < n$. Deux arêtes e_i et e_j ont la même couleur (i.e. $c(e_i) = c(e_j)$), signifie que $j + k'(2\ell-n+5-n \bmod 2) \equiv (i+\varphi+k(2\ell-n+5-n \bmod 2) \bmod \phi$. Ceci donne $j-i+d(2\ell-n+5-n \bmod 2)-\varphi \equiv 0 \bmod \phi$. Ainsi, si n est impair, alors $j-i+d(2\ell-n+4) = \alpha\phi$. Nous allons discuter selon la valeur de α. Il est évident que $\alpha \neq 0$, puisque $n < \ell+2$ et $d \geq 1$. Si $\alpha \geq 2$, alors nous avons $dist(e_i,e_j) + d(2\ell-n+3) \geq 2n(\ell+2) - n^2$. Ainsi, $dist(e_i,e_j) \geq (2\ell-n+3)(n-d)+n \geq 2\ell+3$, puisque $n > d$. Si $\alpha = 1$ alors pour $d \geq \frac{n}{2}$, $dist(e_i,e_j) = (2\ell-n+3)(d-\frac{n}{2}) + d - \frac{3}{2} > \ell$

et pour $d \leq \frac{n}{2}$ $dist(e_i, e_j) = (2\ell - n + 3)(\frac{n}{2} - d) + d - \frac{1}{2} > \ell$. Ainsi, pour $d < n$, $dist(e_i, e_j) > \ell$.

Si n est pair et $0 \leq k, k' \leq \left\lceil \frac{n}{2} \right\rceil - 1$ ou $\left\lceil \frac{n}{2} \right\rceil \leq k, k' \leq n-1$, alors $j - i + d(2\ell - n + 5) = \alpha\phi$. Nous allons discuter selon la valeur de α. Il est évident que $\alpha \neq 0$, puisque $n < \ell + 2$ et $d \geq 1$. Si $\alpha \geq 2$ alors nous avons $dist(e_i, e_j) + d(2\ell - n + 3) \geq 2\phi - 1$. Ceci donne $dist(e_i, e_j) \geq (2\ell - n + 3)(n - d) - 1 + n \geq 2\ell + 2$, puisque $n > d$. Si $\alpha = 1$ pour $d \geq \frac{n}{2}$ $dist(e_i, e_j) = (2\ell - n + 3)(d - \frac{n}{2}) + d - \frac{3}{2} > \ell$ et pour $d \leq \frac{n}{2}$ $dist(e_i, e_j) = (2\ell - n + 4)(\frac{n}{2} - d) - 1 > \ell$. Ainsi, pour $d < n$, nous avons $dist(e_i, e_j) > \ell$.

Si n est pair et $0 \leq k \leq \left\lceil \frac{n}{2} \right\rceil - 1$, $\left\lceil \frac{n}{2} \right\rceil \leq k' \leq n-1$, alors $j - i + d(2\ell - n + 5) - \varphi = \alpha\phi$. Nous allons discuter selon la valeur de α. Il est évident que $\alpha \neq 0$, puisque $n < \ell + 2$ et $d \geq 1$. Si $\alpha \geq 2$ alors nous avons $dist(e_i, e_j) + d(2\ell - n + 4) \geq \varphi + 2\phi - 1$. Ceci donne $dist(e_i, e_j) \geq (2\ell - n + 4)(n - d) + \ell - n + 1 \geq 3\ell - 2n + 5 \geq \ell + 1$, puisque $n > d$ et $n < \ell + 2$. Si $\alpha = 1$ alors pour $d \geq \frac{n}{2}$ $dist(e_i, e_j) = (2\ell - n + 4)(d - \frac{n}{2}) + 2d - \ell + n - 3 > \ell$ et pour $d \leq \frac{n}{2}$ $dist(e_i, e_j) = (2\ell - n + 4)(\frac{n}{2} - d) + \ell - n + 1 > \ell$. Ainsi, pour $d < n$, $dist(e_i, e_j) > \ell$.

La coloration c des arêtes de $\mathcal{V}_k$, $0 \leq k \leq n - 2$ est définie comme suit :

- Pour $0 \leq k \leq \left\lceil \frac{n}{2} \right\rceil - 2$, nous colorons d'une manière cyclique les arêtes de $\mathcal{V}_k$ avec les couleurs $\{1 + i + k(2\ell - n + 4 + n \bmod 2) : 0 \leq i \leq 2\ell - n + 3 + n \bmod 2\}$.
- Pour n pair et $k = \frac{n}{2} - 1$, nous colorons d'une manière cyclique les couleurs de $\mathcal{V}_{\frac{n}{2}-1}$ avec les couleurs $\{1 + i + (\frac{n}{2} - 1)(2\ell - n + 4) : 0 \leq i \leq \ell\}$.
- Pour $\left\lfloor \frac{n}{2} \right\rfloor \leq k \leq n - 2$, nous commençons la coloration de $\mathcal{V}_k$ avec les couleurs $\{1 + i + \ell + 2 - \left\lfloor \frac{n}{2} \right\rfloor + (k - \left\lfloor \frac{n}{2} \right\rfloor)(2\ell - n + 4 + n \bmod 2) : 1 \leq i \leq \ell + 1 - \left\lfloor \frac{n}{2} \right\rfloor\}$ puis nous colorons les autres arêtes d'une manière cyclique avec $\{1 + i + (k - \left\lfloor \frac{n}{2} \right\rfloor)(2\ell - n + 4 + n \bmod 2) : 0 \leq i \leq 2\ell - n + 3 + n \bmod 2\}$. Nous utilisons le même ensemble de couleurs que celui de $\mathcal{V}_{k, 0 \leq k \leq \lceil \frac{n}{2} \rceil - 2}$.

D'après la coloration de $\mathcal{V}_k$, $0 \leq k \leq n - 2$, nous distinguons deux cas selon la valeur de k et de k'.

Cas $k = k'$. Deux arêtes qui ont la même couleur sont à distance $2\ell - n + 4 + n \bmod 2$ ou $\ell + 1$.

Cas $k \neq k'$. Il suffit d'étudier les deux cas suivants $0 \leq k \leq \left\lceil \frac{n}{2} \right\rceil - 2$ et $\left\lfloor \frac{n}{2} \right\rfloor \leq k' \leq n-2$. D'après l'algorithme de coloration, $\mathcal{V}_k$ et $\mathcal{V}_{k'}$ ont le même ensemble de couleurs s'ils sont à distance$\left\lfloor \frac{n}{2} \right\rfloor$, ainsi, $k' = k + \left\lfloor \frac{n}{2} \right\rfloor$. Nous avons aussi, pour tout $e_i \in \mathcal{V}_k$, $c(e_i) = i + k(2\ell - n + 4 + n \bmod 2)$, avec $0 \leq i \leq 2\ell - n + 3 + n \bmod 2$. Sans perte de généralité, nous considérons l'intervalle qui nous permet d'avoir toutes les couleurs. Dans ce cas, pour tout $e_j \in \mathcal{V}_{k'}$, $c(e_j) = j - \ell - 2 + \left\lfloor \frac{n}{2} \right\rfloor + (k - \left\lfloor \frac{n}{2} \right\rfloor)(2\ell - n + 4 + n \bmod 2)$, avec $\ell + 1 - \left\lfloor \frac{n}{2} \right\rfloor \leq j \leq 3\ell - n - \left\lceil \frac{n}{2} \right\rceil + 4 + n \bmod 2$. $c(e_i) = c(e_j)$ signifie que $i + k(2\ell - n + 4 + n \bmod 2) = j - \ell - 2 + \left\lfloor \frac{n}{2} \right\rfloor + (k' - \left\lfloor \frac{n}{2} \right\rfloor)(2\ell - n + 4 + n \bmod 2)$. Ceci donne $i = j - \ell - 2 + \left\lfloor \frac{n}{2} \right\rfloor$. Ainsi, $dist(e_i, e_j) = |k' - k - 1| + |j - i| = \left\lfloor \frac{n}{2} \right\rfloor + \ell + 2 - \left\lfloor \frac{n}{2} \right\rfloor - 1 =$

$\ell + 1 > \ell$.

Si n est pair, alors le nombre de couleurs est égal à $\phi+(\left\lceil \frac{n}{2} \right\rceil -1)(2\ell-n+4)+\ell+1 = |E(M_{n,\ell+3})| - n^2$.

Si n est impair, alors le nombre de couleurs est égal à $\phi+(\left\lceil \frac{n}{2} \right\rceil -1)(2\ell-n+5) = |E(M_{n,\ell+3})| - n^2 + 1$. Le nombre total de couleurs est $|E(M_{n,\ell+3})| - n^2 + n \bmod 2$.

Par conséquent, $\chi'_\ell(M_{n,m}) = |E(M_{n,\ell+3})| - n^2 + n \bmod 2$.

En utilisant le même raisonnement, si $m > \ell + 2$ et $n = \ell + 2$, nous pouvons montrer que $\chi'_\ell(M_{n,m}) = |E(M_{n,m})| - |E(S"_\ell)| = |E(M_{\ell+2,\ell+3})| - (\ell + 2)^2 + 1$. □

3.4 Coloration ℓ-distance d'un arbre k-aire complet

3.4.1 Préliminaires

Soit T un arbre k-aire complet. Nous utilisons les notations suivantes : soit h la hauteur de T, les nœuds de T sont notés par $x_{i,j}$, où l'indice i, $0 \leq i \leq h$ indique le niveau du nœud et j, $0 \leq j \leq k^i - 1$, représente la position de ce nœud dans un niveau donné i (i.e. la racine de l'arbre est située au niveau 0 et est notée par $x_{0,0}$).

3.4.2 Construction du sous-graphe S_ℓ

Définition 3.4.1. Soient $\ell \geq 0$ et T un arbre k-aire complet de hauteur $h \geq \ell + 3$. Soient $v = x_{i,j} \in V(T)$ et $v' = x_{i',j'} \in V(T)$ tels que $\left\lfloor \frac{\ell}{2} \right\rfloor + 1 \leq i, i' \leq h - \left\lfloor \frac{\ell}{2} \right\rfloor - 1$ et $Q_v = \{u \in V(G) : dist(u, v) \leq \left\lfloor \frac{\ell}{2} \right\rfloor + 1\}$. Nous définissons le sous graphe induit $S_\ell \subset T$ comme $S_\ell = T[Q_v \cup Q_{v'}]$ tel que $v = v^{'}$ si ℓ est pair et $(v, v^{'}) \in E(T)$ si ℓ est impair. v et $v^{'}$ sont les centres de S_ℓ (voir Figure 3.12).

Lemme 3.4.2. *Soient $\ell \geq 0$, $k \geq 1$ et T un arbre k-aire complet de hauteur $h \geq \ell + 3$. Soit $S_\ell \subset T$ un sous-graphe de T construit selon la Définition 3.4.1. Alors, pour tout e, $e^{'} \in E(S_\ell)$, $dist(e, e^{'}) \leq \ell$ et*

$$|E(S_\ell)| = 1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1}.$$

Démonstration. Par construction, tous les sommets de S_ℓ à distance au plus $\left\lfloor \frac{\ell}{2} \right\rfloor$ des centres ont un même degré $1 + k$ et les feuilles ont un degré égal à 1.

Cas(a) Si ℓ est pair, il existe $(1 + (1 + k)\sum_{i=0}^{i=\frac{\ell}{2}-1} k^i)$ sommets ayant le même degré, égal à $(1 + k)$, et $((1 + k)\, k^{\frac{\ell}{2}})$ feuilles. Le nombre d'arêtes est égal à

$\frac{1}{2}(1 + k)\left(1 + (1 + k)\sum_{i=0}^{i=\frac{\ell}{2}-1} k^i + k^{\frac{\ell}{2}}\right) = \left(1 - k^{\frac{\ell}{2}+1}\right)\frac{1+k}{1-k}$.

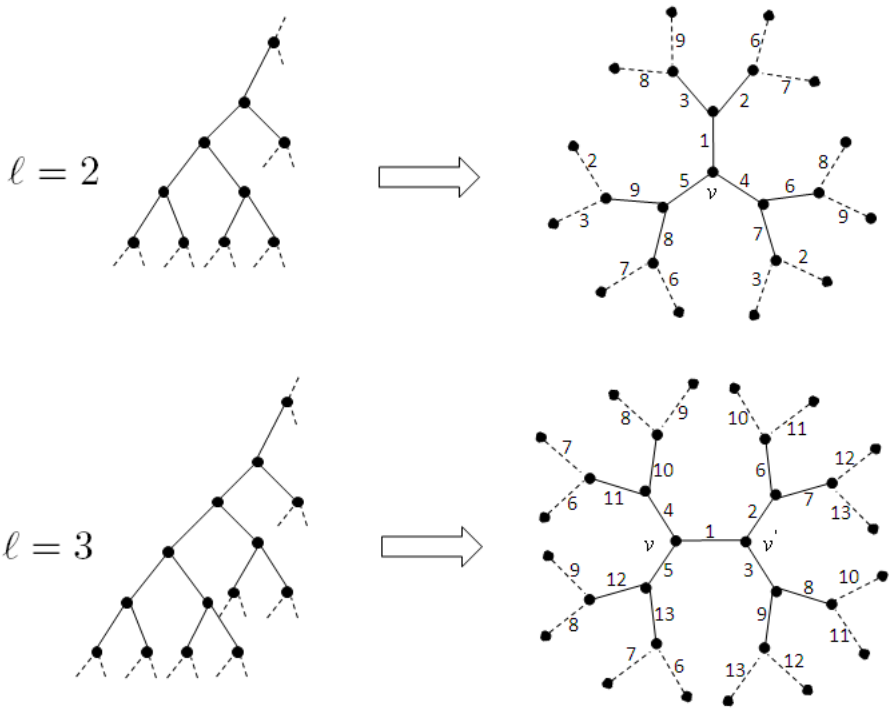

Figure 3.12 : *Les sous-graphes S_2 et S_3 d'un arbre binaire complet. S_2 et S_3 sont identifiés par les arêtes continues.*

Cas(b) Si ℓ est impair, il existe $(2\sum_{i=0}^{i=\lfloor \frac{\ell}{2} \rfloor} k^i)$ sommets ayant le même degré, égal à $(1+k)$, et $(2k^{\lfloor \frac{\ell}{2} \rfloor + 1})$ feuilles. Le nombre d'arêtes est égal à
$\frac{1}{2}\left(2\,(1+k)\sum_{i=0}^{i=\lfloor \frac{\ell}{2} \rfloor} k^i + 2k^{\lfloor \frac{\ell}{2} \rfloor + 1}\right) = \frac{1+k-2k^{\frac{\ell+3}{2}}}{1-k}$.

D'après la Définition 3.4.1, la distance entre deux arêtes de $E(S_\ell)$ est au plus ℓ. Ainsi, pour tout $e, e' \in E(S_\ell)$, nous avons $dist(e, e') \leq \ell$. □

3.4.3 Détermination de l'indice ℓ-chromatique

Théorème 3.4.3. *Soit $\ell \geq 0$ et T un arbre k-aire complet de hauteur $h \geq \ell + 3$, l'indice ℓ-chromatique de T est donné par :*

$$\chi'_\ell(T) = 1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1} = \begin{cases} \left(1 - k^{\frac{\ell}{2}+1}\right)\frac{1+k}{1-k} & \text{si } \ell \text{ est pair,} \quad (1) \\ \frac{1+k-2k^{\frac{\ell+3}{2}}}{1-k} & \text{si } \ell \text{ est impair.} \quad (2) \end{cases}$$

Démonstration. Soit T un arbre k-aire complet de hauteur h et S_ℓ un sous-graphe de T construit selon la Définition 3.4.1. Tout d'abord, nous prouvons qu'il est toujours possible de colorer les arêtes incidentes aux feuilles de S_ℓ (n'appartenant pas à S_ℓ) uniquement par des couleurs des arêtes du sous-graphe S_ℓ, noté $c(S_\ell)$.

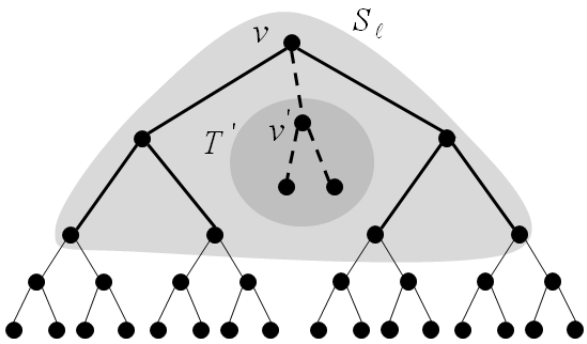

Figure 3.13 : *Le sous-graphe S_2 est inclus dans l'arbre T".*

Si ℓ est pair, S_ℓ possède $k+1$ branches, obtenues en supprimant le centre v de S_ℓ. Nous colorons l'ensemble des arêtes incidentes aux feuilles de S_ℓ qui appartiennent à une même branche i et n'appartiennent pas à S_ℓ, noté par Q_i, (les arêtes en pointillées dans la Figure 3.12) avec les couleurs des arêtes de S_ℓ qui sont à distance $\ell+1$ et qui appartiennent aux k autres branches. L'ensemble de ces arêtes est noté Q'_i. Nous avons les propriétés suivantes :

- La distance qui sépare les arêtes de Q'_i de celles de Q_i est égale à $\ell+1$. Nous remarquons que $|Q_i| = |Q'_i| = k^{\frac{\ell}{2}+1}$. Par conséquent, nous affectons les couleurs des arêtes colorées aux arêtes non-colorées.
- La distance entre les arêtes de Q_i et Q_j (avec $i \neq j$) est égale à $\ell+2$. Nous affectons ainsi des couleurs aux arêtes de Q_i indépendamment des couleurs des arêtes de Q_j.

Si ℓ est impair, alors S_ℓ possède $2k$ branches, obtenues en supprimant les centres v et v' de S_ℓ. Nous colorons l'ensemble des arêtes incidentes aux feuilles de S_ℓ qui appartiennent à une même branche i et n'appartiennent pas à S_ℓ, noté par P_i, (les arêtes en pointillées dans la Figure 3.12) avec les couleurs des arêtes de S_ℓ qui sont à distance $\ell+1$ et qui appartiennent aux k autres branches les plus éloignées. L'ensemble de ces arêtes est noté P'_i. Nous avons les propriétés suivantes :

- La distance qui sépare les arêtes de P'_i de ceux de P_i est égale à $\ell+1$. Nous remarquons que $|P| = |P'| = k^{\lfloor \frac{\ell}{2} \rfloor+1}\ k^{\lfloor \frac{\ell}{2} \rfloor+1}$ arêtes. Par conséquent, nous affectons les couleurs des arêtes colorées aux arêtes non-colorées.
- La distance entre les arêtes de P_i et P_j (avec $i \neq j$) est égale au moins $\ell+2$. Nous pouvons ainsi affecter des couleurs aux arêtes de P_i indépendamment des couleurs des arêtes de P_j.

Une telle coloration satisfait la coloration d'arêtes ℓ-distance. Par conséquent, pour $\ell \geq 0$, il est toujours possible de colorer les arêtes incidentes aux feuilles de S_ℓ en utilisant uniquement les couleurs de $c(S_\ell)$.

L'algorithme de coloration de l'arbre k-aire complet de hauteur h, T, est décrit comme suit (voir Algorithme 1), soient :

- $x_{i,j}$, $i: 0 \leq i \leq h-1$, $j: 0 \leq j \leq k^i - 1$ les sommets de T,
- $T' = (V(T'), E(T'))$ un arbre k-aire complet de hauteur $\left\lceil \frac{\ell}{2} \right\rceil + 1$, avec v' la racine de T',
- $T" = (V(T"), E(T"))$ un arbre tel que $T" = T \cup T'$, $V(T") = V(T) \cup V(T')$ et $E(T") = E(T) \cup E(T') \cup (v, v')$ avec $v = x_{0,0}$ (voir Figure 3.13),
- S_ℓ est un sous-graphe de $T"$ qui contient T' (*i.e.* $E(T') \subset E(S_\ell)$, v est le centre de S_ℓ si ℓ est pair et v, v' sont les centres de S_ℓ si ℓ est impair),
- (v, v') est une arête ancêtre de toutes les arêtes de $E(T")$.

L'algorithme consiste à colorer les arêtes du sous-graphe S_ℓ avec des couleurs différentes et par la suite colorer les autres arêtes en utilisant les mêmes couleurs de $c(S_\ell)$. Il suffit dans ce cas, à chaque étape, de colorer toutes les arêtes du niveau i de l'arbre $T" \supset T$ et qui sont adjacentes aux arêtes déjà colorées.

Soit $x \in V(T")$ un sommet incident à une arête colorée : si ℓ est pair alors soit $x' \in V(T")$ l'ancêtre de x tel que $dist(x, x') = \frac{\ell}{2} + 1$. Nous considérons S_ℓ tel que x' soit son centre, noté par $S_\ell^{x'}$, puis nous colorons les arêtes non-colorées, incidentes à x, en utilisant les couleurs de $c(S_\ell^{x'})$. Si ℓ est impair alors soit $e \in E(T")$ une arête colorée à l'étape précédente, x est une extrémité de e et $e' \in E(T")$ est l'ancêtre de e tel que $dist(e, e') = \left\lfloor \frac{\ell}{2} \right\rfloor$. Nous considérons S_ℓ tel que e' soit son centre, noté par $S_\ell^{e'}$, puis nous colorons les arêtes non-colorées, incidentes à x, en utilisant les couleurs de $c(S_\ell^{e'})$.

Comme T et $T"$ ont le même degré maximum $\Delta = k+1$, l'indice ℓ-chromatique de $T"$ est égal à l'indice ℓ-chromatique de T. Nous avons ainsi $\chi'_\ell(T") = \chi'_\ell(T)$. Alors, $\chi'_\ell(T) \leq 1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1}$. De plus, selon le Lemme 3.4.2, $\chi'_\ell(M_{n,m}) \geq 1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1}$. Par conséquent, $\chi'_\ell(M_{n,m}) = 1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1}$. □

Théorème 3.4.4. *Pour $\ell \geq 0$, soit T un arbre k-aire complet. Alors, la complexité de l'algorithme de coloration est $O(nk^{\lfloor \frac{\ell}{2} \rfloor + 1})$.*

Démonstration. Tout d'abord, pour calculer $|E(S_\ell)|$, nous parcourons $1 + \sum_{i=0}^{\ell} k^{\lfloor \frac{i}{2} \rfloor + 1}$ sommets. Nous déduisons la complexité de la recherche du sous-graphe : $O(k^{\lfloor \frac{\ell}{2} \rfloor + 1})$. Ensuite, pour colorer T, nous colorons au début S_ℓ. Puis, nous colorons les arêtes incidentes à chaque sommet. Nous traversons le plus proche sous-graphe maximum entièrement coloré. Par conséquent, la coloration de T est réalisée en $O(2k^{\lfloor \frac{\ell}{2} \rfloor + 1} + k^{\lfloor \frac{\ell}{2} \rfloor + 1}(n - k^{\lfloor \frac{\ell}{2} \rfloor + 1})) \cong O(nk^{\lfloor \frac{\ell}{2} \rfloor + 1})$. □

3.5 Arbres quelconques

Dans cette section, nous généralisons le résultat précédent aux arbres quelconques. Nous utilisons les mêmes notations que précédemment : Soit T un arbre,

Algorithme 1 La coloration d'arêtes ℓ-distance de l'arbre $T" = T \cup T'$.

Entrées : T un arbre k-aire complet de hauteur $h \geq 0$,
T' un arbre k-aire de hauteur $\left\lceil \frac{\ell}{2} \right\rceil + 1$, avec v' la racine de T',
Une coloration de $E(S_\ell) \subset E(T")$ telle que $E(T') \subset E(S_\ell)$.
Sortie : Une coloration d'arêtes ℓ-distance de $T"$.

---

1 : **Pour chaque** $i \in [0, h-2]$ **faire**
2 : **Pour chaque** $j \in [0, 2^{i-1}-2]$ **faire**
3 : **Si** $\{c(x_{i,j}, x_{i+1,kj+s}) \forall s \in \{0, ..., k-1\}\}$ est non-coloré
4 : *ColorDistantNodes*$(x_{i,j})$;
5 : *ChooseSetColor*() ;
6 : *AffectSetColor*$(x_{i,j})$;
7 : **Fin si**
8 : **Fin pour**
9 : **Fin pour**

ColorDistantNodes$(x_{i,j})$: Retourne l'ensemble des couleurs des arêtes à distance $\ell + 1$ des arêtes incidentes non-colorées, de $x_{i,j}$.
ChooseSetColor() : Choisir un ensemble de couleurs qui satisfait la coloration d'arêtes ℓ-distance.
AffectSetColor$(x_{i,j})$: Affecter l'ensemble de couleurs choisi aux arêtes incidentes non-colorées de $x_{i,j}$.

Algorithme 2 Recherche de l'arbre maximum S_ℓ.

Entrée : T un arbre de hauteur $h \geq 0$,
Sortie : Le sous-graphe maximum S_ℓ tel que pour toutes arêtes e et $e^{'}$, $dist\left(e, e^{'}\right) \leq \ell$.

---

1 : **Si** ℓ est pair **alors**
2 : **Pour chaque** sommet v
3 : *MaximalSubtree*(v, v) ;
4 : **Fin pour**
5 : **Sinon**
6 : **Pour chaque** arête $e = (v, v^{'})$
7 : *MaximalSubtree*$(v, v^{'})$;
8 : **Fin pour**
9 : **Fin si**
10 : *MaximumSubtree*() ;

MaximalSubtree(x, y) : Retourne le nombre d'arêtes du sous-graphe de hauteur $\left\lfloor \frac{\ell}{2} \right\rfloor + 2$, x et y sont les centres du sous-graphe.
MaximumSubtree() : Retourne le sous-graphe maximum S_ℓ ayant le nombre maximum d'arêtes.

$S_\ell = (V(S_\ell), E(S_\ell))$ est le sous-graphe maximum de T où $V(S_\ell)$ est l'ensemble des sommets et $E(S_\ell)$ est l'ensemble des arêtes, tel que pour $\ell \geq 0$, la distance entre deux arêtes de $E(S_\ell)$ est au plus ℓ. Dans la suite, nous prouvons que l'indice ℓ-chromatique d'un arbre quelconque T est égal au nombre d'arêtes de son sous-graphe maximum. Nous proposons une manière pour colorer T uniquement avec les couleurs des arêtes du sous-graphe S_ℓ.

Proposition 3.5.1. *Soit T un arbre de diamètre $diam(T)$ tel que $diam(T) \leq \ell+2$. Alors,*

$$\chi'_\ell(T) = |E(T)|.$$

Démonstration. Si $diam(T) \leq \ell+2$, alors toutes les arêtes de T sont à une distance au plus ℓ les unes des autres. Toutes les arêtes auront des couleurs différentes et $\chi'_\ell(T) = |E(T)|$. □

Théorème 3.5.2. *Soit T un arbre de diamètre $diam(T)$ tel que $diam(T) > \ell+2$. Alors,*

$$\chi'_\ell(T) = |E(S_\ell)|.$$

Démonstration. Soit $T = (V(T), E(T))$ un arbre. Pour satisfaire la coloration d'arêtes ℓ-distance de T, nous procédons en trois phases. La phase 1 recherche le sous-graphe maximum $S_\ell \subset T$. La phase 2 colore S_ℓ avec $|E(S_\ell)|$ couleurs. La phase 3 étend la coloration à l'ensemble de l'arbre T.

Phase 1 Algorithme de recherche du sous-graphe maximum S_ℓ.

L'algorithme considère deux cas en fonction de la parité de ℓ (voir Algorithme 2) (voir Figure 3.14).

Cas ℓ pair. Pour chaque sommet $v \in V(T)$, nous cherchons le sous-graphe maximum S_ℓ^v avec v est le centre de S_ℓ^v. S_ℓ^v est construit comme suit : $S_\ell^v = T[Q_v]$ avec $Q_v = \{u \in V(T) : dist(u,v) \leq \left\lfloor \frac{\ell}{2} \right\rfloor + 1\}$. Par la suite, nous déterminons le nombre d'arêtes de ce sous-graphe $|E(S_\ell^v)|$. Le sous-graphe maximum S_ℓ de T, pour ℓ pair, possède le nombre maximum d'arêtes : $S_\ell = \{S_\ell^v$ tel que $v \in V(T)$ et $|E(S_\ell)| = max\{|E(S_\ell^v)||v \in V(T)\}\}$. Supposons qu'il existe un sous-graphe S'_ℓ tel que $|E(S'_\ell)| > |E(S_\ell)|$, alors il existe $v \in V(S'_\ell)$: v est le centre de S'_ℓ. Cependant, S_ℓ^v est le sous-graphe maximal, d'où $|E(S'_\ell)| = |E(S_\ell^v)| \leq |E(S_\ell)|$, ce qui contredit l'hypothèse de départ.

Cas ℓ impair. Pour chaque arête $e \in E(T)$: $e = (v, v')$, nous cherchons le sous-graphe maximum S_ℓ^e avec v, v' sont les centres de S_ℓ^e. S_ℓ^e est construit comme suit : $S_\ell^e = T[Q_v \cup Q_{v'}]$ avec $Q_v = \{u \in V(T) : dist(u,v) \leq \left\lfloor \frac{\ell}{2} \right\rfloor + 1\}$ et $Q_{v'} = \{u \in V(T) : dist(u,v') \leq \left\lfloor \frac{\ell}{2} \right\rfloor + 1\}$. Par la suite, nous déterminons le nombre d'arêtes de ce sous-graphe $|E(S_\ell^e)|$. Le sous-graphe maximum S_ℓ de T, pour ℓ impair, possède le nombre maximum d'arêtes : $S_\ell = \{S_\ell^e$ tel que $e \in E(T)$ et $|E(S_\ell)| = max\{|E(S_\ell^e)||e \in$

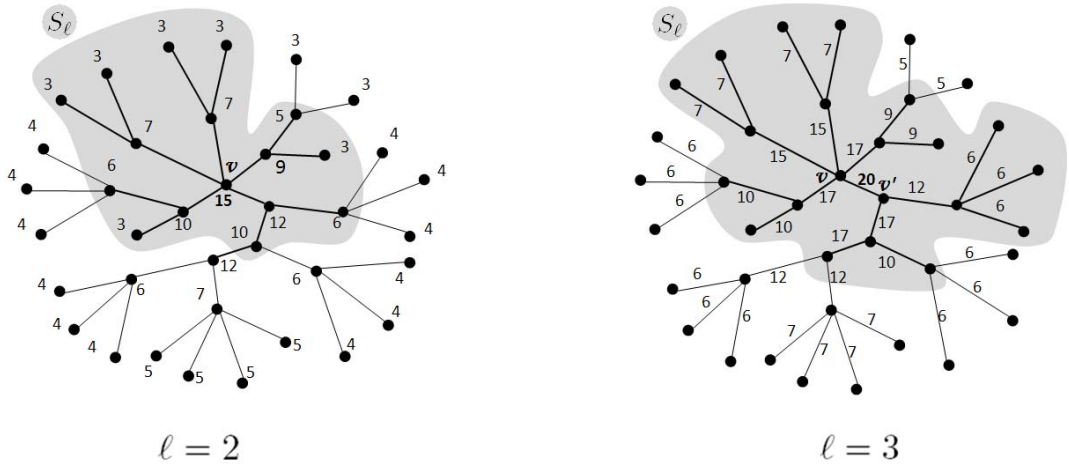

Figure 3.14 : *Recherche du sous-graphe maximum.*

$E(T)\}\}$. Supposons qu'il existe un sous-graphe S'_ℓ tel que $|E(S'_\ell)| > |E(S_\ell)|$, alors il existe $e \in E(S'_\ell) \ : \ e = (v, v')$ et v, v' sont les centres de S'_ℓ. Cependant, S^e_ℓ est le sous-graphe maximal, d'où $|E(S'_\ell)| = |E(S^e_\ell)| \leq |E(S_\ell)|$, ce qui contredit l'hypothèse de départ.

Pour ℓ pair, le centre de S_ℓ devient la racine de l'arbre T. Pour ℓ impair, l'arbre T possède deux racines v, v' telles que v, v' sont les centres de S_ℓ. L'arête $e = (v, v')$ est l'ancêtre de toutes les arêtes qui appartiennent à T. La borne inférieure de l'indice ℓ-chromatique de T est égale à $\chi'_\ell(T) \geq |E(S_\ell)|$.

Phase 2 Algorithme de coloration du sous-graphe S_ℓ.

Il suffit dans cette phase d'affecter des couleurs différentes à $E(S_\ell)$.

Phase 3. Algorithme de coloration de l'arbre T.

L'idée consiste à étendre l'algorithme utilisé précédemment dans le cas des arbres k-aires complets (Preuve du Théorème 3.4.3) aux arbres quelconques. La coloration des arêtes restantes se fait en plusieurs étapes. Les arêtes non-colorées adjacentes aux arêtes colorées durant l'étape $i-1$ seront colorées, comme suit :

Étape 0 : Nous colorons les arêtes incidentes aux feuilles de S_ℓ qui appartiennent à une même branche et n'appartenant pas à S_ℓ, en utilisant des couleurs de $c(S_\ell)$ des arêtes qui sont à distance $\ell + 1$.

Étape i, $i > 0$ (ℓ pair) : Nous colorons les arêtes adjacentes aux arêtes colorées durant l'étape $i-1$ comme suit : soit v' une extrémité d'une arête colorée durant l'étape $i-1$ et v un sommet ancêtre de v' tel que $dist(v, v') = \frac{\ell}{2} + 1$. Nous colorons toutes les arêtes, non-colorées, incidentes aux feuilles de S^v_ℓ en utilisant des couleurs de $c(S^v_\ell)$ des arêtes qui sont à distance $\ell + 1$. Les couleurs des arêtes à distance $\ell + 1$ sont parfois insuffisantes pour colorer ces arêtes. Nous utilisons les couleurs de $c(S_\ell) - c(S^v_\ell)$. En effet, nous pouvons toujours trouver une couleur à chaque arête pour satisfaire la coloration d'arêtes ℓ-distance car S_ℓ est le sous-graphe maximum

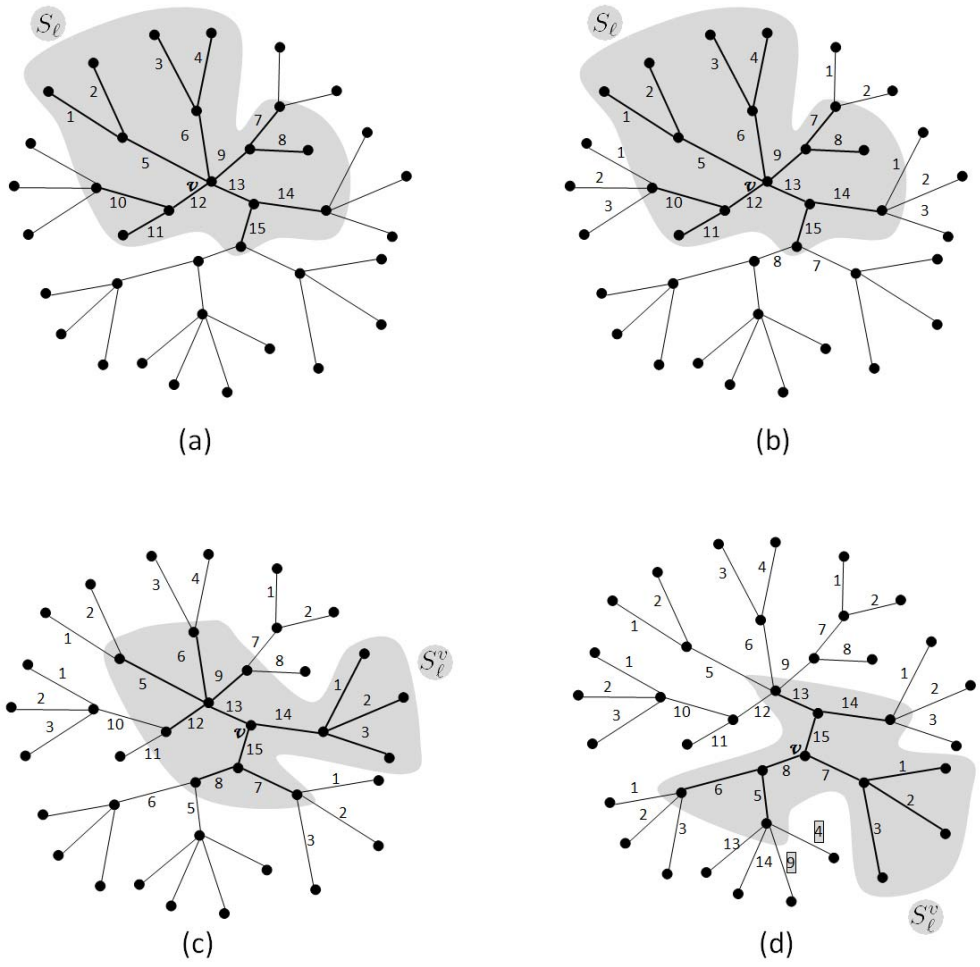

Figure 3.15 : *Exemple de la coloration d'arêtes ℓ-distance d'un arbre ($\ell = 2$).*

et $|E(S_\ell)| \geq |E(S_\ell^v)|$.

Étape i, $i > 0$ (ℓ impair) : Nous colorons les arêtes adjacentes aux arêtes colorées durant l'étape $i-1$ comme suit : soit e' une arête colorée durant l'étape $i-1$ et e une arête ancêtre de e' telle que $dist(e, e') = \left\lfloor \frac{\ell}{2} \right\rfloor$. Nous colorons toutes les arêtes, non-colorées, incidentes aux feuilles de S_ℓ^e en utilisant des couleurs de $c(S_\ell^e)$ des arêtes qui sont à distance $\ell+1$. Les couleurs des arêtes à distance $\ell+1$ sont parfois insuffisantes pour colorer ces arêtes. Nous utilisons les couleurs de $c(S_\ell) - c(S_\ell^e)$. Comme dans le cas pair, nous pouvons toujours trouver une couleur pour satisfaire la coloration d'arêtes ℓ-distance.

La Figure 3.15 montre un exemple de coloration dans le cas ℓ pair. La Figure 3.15(a) illustre et présente une coloration du sous-graphe maximum S_ℓ. La Figure 3.15(b) illustre l'étape 0 de l'algorithme : la coloration des arêtes incidentes aux feuilles de S_ℓ. La Figure 3.15(c) illustre l'étape 1 de l'algorithme : le sous-graphe S_ℓ^v est construit et ses couleurs sont utilisées pour colorer les arêtes adjacentes à celles colorées durant l'étape 0. La Figure 3.15(d) illustre l'étape 2 de l'algorithme. De la même manière, un autre S_ℓ^v est construit. Cependant, dans ce cas, le nombre de couleurs utilisées dans S_ℓ^v et qui sont à distance $\ell+1$ est insuffisant. Alors, deux couleurs additionnelles $\{4, 9\} \in c(S_\ell) - c(S_\ell^v)$ sont utilisées pour compléter la coloration. Ainsi, $\chi'_\ell(T) \leq |E(S_\ell)|$. Par conséquent, $\chi'_\ell(T) = |E(S_\ell)|$. □

Théorème 3.5.3. *Pour $\ell \geq 0$, soit T un arbre et Δ le degré maximum de T. Alors, la complexité de l'algorithme de coloration est $O(n(\Delta-1)^{\left\lfloor \frac{\ell}{2} \right\rfloor+1})$.*

Démonstration. Pour calculer le nombre d'arêtes du sous-graphe maximum, nous parcourons dans le pire des cas $1 + \sum_{i=0}^{\ell} (\Delta-1)^{\left\lfloor \frac{i}{2} \right\rfloor+1}$ sommets (le nombre d'arêtes d'un arbre $(\Delta-1)$-aire complet). En effet, nous parcourons tout l'arbre pour chercher le sous-graphe S_ℓ. La recherche de S_ℓ se fait en $O(n(\Delta-1)^{\left\lfloor \frac{\ell}{2} \right\rfloor+1})$. Ainsi la complexité de l'algorithme est $O(n(\Delta-1)^{\left\lfloor \frac{\ell}{2} \right\rfloor+1})$ sachant que pour colorer T, nous colorons, dans le pire des cas l'arbre $(\Delta-1)$-aire complet (Preuve du Théorème 3.4.4). □

3.6 Conclusion

Dans ce chapitre, nous avons étudié le paramètre de coloration d'arêtes ℓ-distance, une généralisation de la coloration propre d'arêtes. Nous avons donné des valeurs exactes de l'indice ℓ-chromatique du paramètre. Le chapitre suivant sera consacré à l'étude de la coloration d'arêtes ℓ-distance des graphes puissances.

Chapitre 4

Coloration d'arêtes ℓ-distance de graphes puissances

Résumé

*Une première étude du paramètre de la coloration d'arêtes ℓ-distance nous a permis de mieux connaître les propriétés du paramètre et ainsi de pouvoir l'étudier dans des classes de graphes plus complexes, à savoir les graphes puissances. Le problème de coloration de cette classe de graphes a attiré beaucoup d'attention. Dans la littérature, plusieurs classes de graphes puissances ont été étudiées [*Dahlhaus 87*] [*Brandstad 96*] [*Kheddouci 00*] [*Effantin 03*] [*Effantin 05b*].*

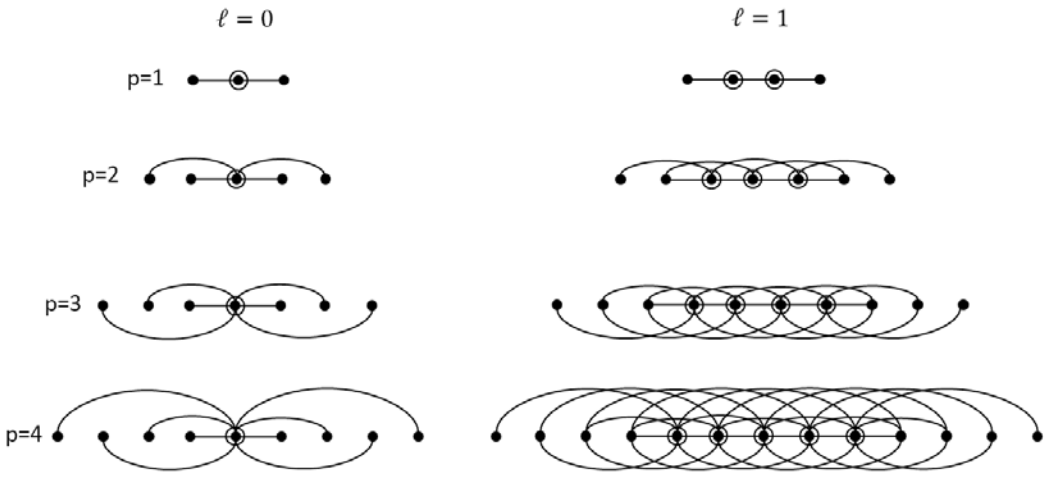

Figure 4.1 : *Exemples de sous-graphes S_ℓ du graphe puissance d'une chaîne pour $0 \leq \ell \leq 1$ et $1 \leq p \leq 4$. Les sommets encerclés sont les sommets de l'ensemble Q.*

4.1 Coloration d'arêtes ℓ-distance du graphe puissance d'une chaîne

Dans cette section, nous nous intéressons à la coloration d'arêtes ℓ-distance du graphe puissance d'une chaîne P_n^p. Nous définissons tout d'abord un sous-graphe particulier de P_n^p, noté S_ℓ. Ensuite, nous déduisons la valeur de l'indice ℓ-chromatique de P_n^p de la taille de S_ℓ.

4.1.1 Définition du sous-graphe S_ℓ

Définition 4.1.1. Pour tout $p \geq 1$ et $\ell \geq 0$, soit P_n une chaîne de sommets $x_1, x_2, \ldots, x_n$ avec $n \geq p(\ell+2)+1$. Soit $W = \left\{x_1, x_2, \ldots, x_{p(\ell+2)+1}\right\}$, $F_1 = \{x_1, x_2, \ldots, x_p\}$ et $F_2 = \{x_{p(\ell+1)+2}, x_{p(\ell+1)+3}, \ldots, x_{p(\ell+2)+1}\}$ des sous-ensembles de $V(P_n)$. Alors, nous définissons le sous-graphe $S_\ell = P_n^p[W] - E(P_n^p[F_1]) - E(P_n^p[F_2])$.

Nous avons ainsi $|V(S_\ell)| = |W| = 2p+1$ si $\ell = 0$ et $|V(S_\ell)| = |W| = p(\ell+2)+1$ si $\ell > 0$. En plus, par construction, chaque arête de S_ℓ est à distance au plus ℓ des autres arêtes de S_ℓ et toutes arêtes de $E(P_n^p) - E(S_\ell)$ est à distance au plus $\ell+1$ d'au moins une arête de S_ℓ. La Figure 4.1 illustre le sous-graphe obtenu S_ℓ pour différentes valeurs de ℓ et p.

Le lemme suivant donne la taille du sous-graphe S_ℓ.

Lemme 4.1.2. *Pour tout $p \geq 1$ et $\ell \geq 0$, soit P_n une chaîne de sommets $x_1, x_2, \ldots, x_n$ avec $n \geq p(\ell+2)+1$. Soit $S_\ell \subset P_n^p$ un sous-graphe de P_n^p construit selon la Définition 4.1.1. Alors,*

$$|E(S_\ell)| = \begin{cases} 2p & \text{si } \ell = 0\ , \quad (a) \\ p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p & \text{si } \ell > 0\ . \quad (b) \end{cases}$$

Démonstration. D'après la Définition 4.1.1, les arêtes du sous-graphe S_ℓ sont les arêtes incidentes à l'ensemble des sommets $Q = \left\{x_{p+1}, x_{p+2}, \ldots, x_{p(\ell+1)+1}\right\}$ dans P_n^p (avec $Q = W - F_1 - F_2$). Nous distinguons deux cas :

Cas (a) : Si $\ell = 0$, alors $|Q| = 1$. Le nombre d'arêtes incidentes est égal à $2p$.

Cas (b) : Si $\ell > 0$ et d'après la Définition 4.1.1, nous avons $|Q| = p\ell + 1$. Le degré de chaque sommet de Q dans S_ℓ est égal à $\Delta_{P_n^p} = 2p$. Pour les sommets $F_1 \cup F_2 = W - Q$, nous avons deux sommets de degré 1, deux autres sommets de degré $2, \ldots$, deux autres sommets de degré p (voir Figure 4.1). Le nombre d'arêtes de S_ℓ est ainsi égal à

$$\begin{aligned} |E(S_\ell)| &= \tfrac{1}{2}\textstyle\sum_{i=1}^{p(\ell+2)+1} d_{x_i}, \\ &= \tfrac{1}{2}\left(2\textstyle\sum_{j=1}^{j=p} j + 2p(p\ell+1)\right), \\ &= p^2\left(\ell + \tfrac{1}{2}\right) + \tfrac{3}{2}p. \end{aligned}$$

□

4.1.2 Détermination de l'indice ℓ-chromatique

Dans la suite, nous présentons le résultat principal de cette section.

Théorème 4.1.3. *Pour tout $p \geq 1$ et $\ell \geq 0$, soit P_n une chaîne de sommets $x_1, x_2, \ldots, x_n$ avec $n \geq p(\ell+2)+1$. Alors, l'indice ℓ-chromatique des graphes puissances P_n^p est donné par :*

$$\chi_\ell^{'}(P_n^p) = \begin{cases} 2p & \text{si } \ell = 0\ , \quad (a) \\ p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p & \text{si } \ell > 0\ . \quad (b) \end{cases}$$

Démonstration. Par la Définition 4.1.1, nous avons $\chi_\ell^{'}(P_n^p) \geq |E(S_\ell)|$ puisque deux arêtes différentes de S_ℓ sont à distance au plus ℓ. Nous prouvons la borne supérieure par construction. Nous décomposons P_n^p en p sous-ensembles d'arêtes tels que chaque sous-ensemble correspond à une puissance. Ainsi, $E(P_n^p) = \cup_{i=1}^{p}\xi_i$ avec $\xi_i = \{(x_1, x_{1+i}), (x_2, x_{2+i}), \ldots, (x_{n-i}, x_n)\}_{i\in\{1..p\}}$ (voir Figure 4.2).

Cas (a) : Si $\ell = 0$, pour une coloration propre de P_n^p, il est possible de colorer les arêtes du graphe puissance avec $\Delta_{P_n^p}$ couleurs. En effet, le graphe $G_i = (V(P_n), \xi_i)$ est un graphe biparti, avec $\Delta_{G_i} = 2$ [Konig 16]. Nous colorons chaque ξ_i, $i \in \{1..p\}$

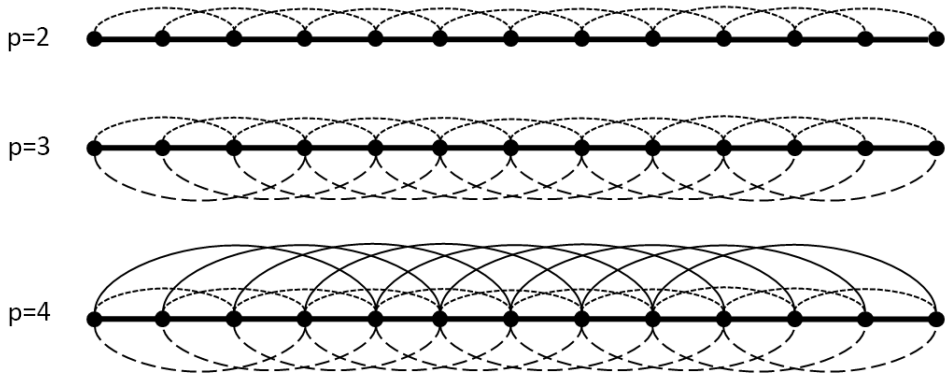

Figure 4.2 : *Quelques graphes puissances d'une chaîne P (P est marquée par les arêtes en gras). Les arêtes pointillées, les arêtes tiretées et les arêtes continues représentent respectivement les arêtes de puissance 2, de puissance 3 et de puissance 4.*

avec deux couleurs différentes, ce qui implique $\chi_0^{'}(P_n^p) \leq 2p$. Par conséquent, l'indice ℓ-chromatique est égal à $2p$.

Cas (b) : Si $\ell > 0$, nous définissons la coloration de chaque sous-ensemble d'arêtes comme suit : nous colorons d'une manière cyclique les arêtes de ξ_i avec $1 + i + p\ell$ couleurs. Les couleurs de ξ_i sont différentes des couleurs de ξ_j avec $i \neq j$. De plus, la distance entre les arêtes, qui partagent la même couleur et qui appartiennent à un même sous-ensemble ξ_i, est égale à $\left\lceil \frac{1+i+p\ell}{p} \right\rceil > \ell$. Ainsi, la coloration est une coloration d'arêtes ℓ-distance. Le nombre de couleurs du graphe est égal au nombre de couleurs utilisées pour colorer chaque sous-ensemble d'arêtes $\sum_{i=1}^{p}(1 + i + p\ell) = p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p$. Ainsi, $\chi_\ell^{'}(P_n^p) \leq p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p$. □

4.2 Coloration d'arêtes ℓ-distance du graphe puissance d'un cycle

Dans cette section, nous étudions la coloration d'arêtes ℓ-distance du graphe puissance du cycle C_n, avec $p \geq 1$.

4.2.1 Résultats de certains cas particuliers

Nous donnons tout d'abord un résultat de l'indice ℓ-chromatique de C_n^p pour $\ell = 0$.

Théorème 4.2.1. *Pour tout $p \geq 1$ et $\ell = 0$, soit C_n un cycle d'ordre n avec $n \geq 3$. Alors, l'indice ℓ-chromatique du graphe puissance C_n^p est donné par*

$$\begin{array}{ll} \chi'_0(C_n^p) = 2p+1 & \text{si } n \text{ est impair }, \quad (a) \\ 2p \leq \chi'_0(C_n^p) \leq 2p+1 & \text{si } n \text{ est pair }. \quad (b) \end{array}$$

Démonstration. ***Cas (a)*** : Puisque C_n^p est un graphe régulier avec un nombre impair de sommets, alors C_n^p est de classe 2 [Wilson 78]. Par conséquent, $\chi'_0(C_n^p) = 2p+1$.

Cas (b) : Voir la preuve de Vizing [Vizing 64]. □

Pour les petits cycles C_n^p d'ordre $n \leq 2p\ell + 3$, la topologie du graphe implique le résultat suivant.

Théorème 4.2.2. *Pour tout $p \geq 1$ et $\ell > 0$, soit C_n un cycle de sommets $x_1, x_2, \ldots, x_n$, avec $3 \leq n \leq 2p\ell + 3$. Alors,*

$$\chi'_\ell(C_n^p) = n.$$

Démonstration. D'après la définition de la coloration d'arêtes ℓ-distance, la distance entre deux arêtes ayant la même couleur est au moins égale à $\ell + 1$. Puisque $3 \leq n \leq 2\ell p + 3$, la distance entre deux arêtes du graphe puissance C_n^p est inférieure à $\ell + 1$. Par conséquent, aucune couleur ne peut être répétée et $\chi'_\ell(C_n^p) = n$. □

4.2.2 C_n^p avec $n > 2\ell p + 3$

Le résultat suivant donne des valeurs de l'indice ℓ-chromatique du graphe puissance C_n^p avec $n > 2\ell p + 3$. Nous montrons tout d'abord la borne supérieure de l'indice ℓ-chromatique. Ensuite, pour quelques cas particuliers nous avons la valeur exacte de ce paramètre.

Théorème 4.2.3. *Pour tout $p \geq 1$ et $\ell > 1$, soit C_n un cycle de sommets $x_1, x_2, \ldots, x_n$, avec $n > 2p\ell + 3$. Alors,*

$$\chi'_\ell(C_n^p) \leq p^2 \left(\ell + \frac{1}{2}\right) + \frac{3}{2}p + \sum_{i=1}^{p} \left\lceil \frac{r_i}{q_i} \right\rceil,$$

avec $q_i = \left\lfloor \frac{n}{1+i+p\ell} \right\rfloor$ et $r_i = n - q_i\,(1 + i + p\ell)$ pour tout $1 \leq i \leq p$.

Démonstration. Nous prouvons la borne supérieure par construction. La coloration du graphe puissance C_n^p est similaire à celle du graphe puissance P_n^p. Nous utilisons cet algorithme de coloration car $2p\ell + 4 > p\,(\ell + 2) + 1$ pour tout $p \geq 1$ et $\ell > 1$. Nous avons $E(C_n^p) = \{\xi_i\}_{i \in \{1..p\}}$ avec $\xi_i = \{(x_1, x_{1+i}), (x_2, x_{2+i}), \ldots, (x_{n-i}, x_n), (x_{n-i+1}, x_1), \ldots, (x_n, x_i)\}$ l'ensemble des arêtes de la puissance i dans C_n^p. Nous

colorons d'une manière cyclique, $q_i - b_i$ fois (avec $b_i = r_i - q_i \left\lfloor \frac{r_i}{q_i} \right\rfloor$), les arêtes de ξ_i avec $1 + i + p\ell + \left\lfloor \frac{r_i}{q_i} \right\rfloor$ différentes couleurs. Pour les arêtes restantes (si $b_i \neq 0$), nous colorons d'une manière cyclique, b_i fois, avec $1 + i + p\ell + \left\lceil \frac{r_i}{q_i} \right\rceil$ couleurs (le même ensemble de couleurs précédent plus une nouvelle couleur). Notons que les couleurs des arêtes d'un sous-ensemble ξ_i sont différentes de celles de ξ_j. La distance entre les arêtes colorées par une même couleur et appartenant au même ξ_i est supérieure à ℓ car nous colorons soit avec $1 + i + p\ell + \left\lfloor \frac{r_i}{q_i} \right\rfloor$ couleurs soit avec $1 + i + p\ell + \left\lceil \frac{r_i}{q_i} \right\rceil$ couleurs. De plus, toutes les arêtes sont colorées car $n = q_i (1 + i + p\ell) + r_i = (q_i - b_i) \left(1 + i + p\ell + \left\lfloor \frac{r_i}{q_i} \right\rfloor\right) + b_i \left(1 + i + p\ell + \left\lceil \frac{r_i}{q_i} \right\rceil\right)$. Le nombre de couleurs utilisées est égal à $\sum_{i=1}^{p}(1 + i + p\ell + \left\lceil \frac{r_i}{q_i} \right\rceil) = \chi_\ell^{'}(P_n^p) + \sum_{i=1}^{p} \left\lceil \frac{r_i}{q_i} \right\rceil$. □

À la suite des conditions particulières, nous donnons une égalité entre $\chi'_\ell(C_n^p)$ et $\chi'_\ell(P_n^p)$.

Théorème 4.2.4. *Pour tout $p \geq 1$ et $\ell > 1$, soit C_n un cycle de sommets $x_1, x_2, \ldots, x_n$, avec $n > 2p\ell+3$. Alors, l'indice ℓ-chromatique du graphe puissance C_n^p est donné par*

$$\chi'_\ell(C_n^p) = \chi'_\ell(P_n^p) \quad \begin{array}{l} \text{si } n \bmod \left(p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p\right) = 0, \\ \text{ou } n \bmod \prod_{i=1}^{i=p}(1 + i + p\ell) = 0. \end{array}$$

Démonstration. Puisque P_n^p est un graphe partiel de C_n^p, alors $\chi'_\ell(C_n^p) \geq \chi'_\ell(P_n^p) = p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p$. Nous prouvons la borne supérieure par construction pour chaque cas séparément. Nous avons $E(C_n^p) = \{\xi_i\}_{i \in \{1..p\}}$ avec $\xi_i = \{(x_1, x_{1+i}), (x_2, x_{2+i}), \ldots, (x_{n-i}, x_n), (x_{n-i+1}, x_1), \ldots, (x_n, x_i)\}$ l'ensemble des arêtes de la puissance i dans C_n^p. Premièrement, si n est un multiple du produit du nombre de couleurs de tous les ξ_i (i.e. $n \bmod \prod_{i=1}^{i=p}(1 + i + p\ell) = 0$) alors nous utilisons le même algorithme que dans le Théorème 4.1.3.b. En effet, chaque ξ_i est coloré avec $1+i+p\ell$ différentes couleurs. Aucune autre couleur n'est ajoutée car $n \bmod \prod_{i=1}^{i=p}(1 + i + p\ell) = 0$ et toutes les arêtes de ξ_i sont colorées.

Deuxièmement, si n est un multiple du nombre total de couleurs (i.e. $n \bmod (p^2(\ell + \frac{1}{2}) + \frac{3}{2}p) = 0$) alors chaque sous-ensemble ξ_i est coloré d'une manière cyclique. Les couleurs utilisées appartiennent à cet ensemble de couleurs $\{1, 2, \ldots, p^2(\ell + \frac{1}{2}) + \frac{3}{2}p\}$ et plus précisément, chaque ξ_i est coloré avec les couleurs $\{p\ell(i-1) + \frac{i}{2}(i+3) - 1, p\ell(i-1) + \frac{i}{2}(i+3), \ldots, p^2\left(\ell + \frac{1}{2}\right) + \frac{3}{2}p, 1, 2, \ldots, p\ell(i-1) + \frac{i}{2}(i+3) - 2\}$.

Par exemple, pour $\ell = 2$, l'ensemble de couleurs utilisées pour colorer C_{26}^2 est $\{1, 2, \ldots, 13\}$. Tout d'abord, Nous colorons toutes les arêtes de ξ_1 avec les couleurs suivantes $\{1, 2, \ldots, 13, 1, 2, \ldots, 13\}$ telles que $c(x_1, x_2) = 1$, $c(x_2, x_3) = 2$,..., et $c(x_n, x_1) = 13$. Ensuite, Nous colorons toutes les arêtes de ξ_2 avec les couleurs suivantes $\{8, 9, \ldots, 13, 1, 2, \ldots, 13, 1, 2, \ldots, 7\}$ telles que $c(x_1, x_3) = 8$, $c(x_2, x_4) = 9$, ..., et $c(x_n, x_2) = 7$.

Aucune nouvelle couleur n'est ajoutée car $n \bmod \left(p^2\left(\ell+\frac{1}{2}\right)+\frac{3}{2}p\right) = 0$ et toutes les arêtes de ξ_i sont colorées. La distance entre les arêtes ayant une même couleur du même ξ_i est supérieure à ℓ car le nombre de couleurs est égal à $p^2\left(\ell+\frac{1}{2}\right)+\frac{3}{2}p$. Dans la suite de la preuve, nous montrons que la distance entre les arêtes de même couleur et appartenant à différents ξ_i est supérieure à ℓ.

Soit $e_i = (x_a, x_{a+i}) \in \xi_i$, $e_j = (x_b, x_{b+j}) \in \xi_j$ et $e_k = (x_c, x_{c+k}) \in \xi_k$, avec $j < i < k \leq p$. D'après l'algorithme de coloration précédent, $c(e_i) = c(e_j)$ signifie que $a + p\ell(i-1) + \frac{i}{2}(i+3) - 1 = b + p\ell(j-1) + \frac{j}{2}(j+3) - 1$. Puisque $j < i < p$, alors $b - a \geq p\ell + j + 2$ et $dist_{C_n^p}(e_i, e_j) = \left\lceil \frac{b-a-i}{p} \right\rceil \geq \ell + 1$. D'une manière similaire, nous montrons que si $c(e_i) = c(e_k)$, alors $dist_{C_n^p}(e_i, e_k) \geq \ell + 1$. □

Pour de grands cycles, l'indice ℓ-chromatique de C_n^p est borné par $\chi'_\ell(P_n^p) + 1$.

Lemme 4.2.5. *Pour tout $p \geq 1$ et $\ell > 1$, soit C_n un cycle de sommets $x_1, x_2, \ldots, x_n$, avec $n > 2p\ell + 3$. Alors, l'indice ℓ-chromatique est donné par*

$$\chi'_\ell(C_n^p) \leq \chi'_\ell(P_n^p) + 1 \quad \text{si} \quad n \geq p^3\left(\ell^2+\ell+\frac{1}{3}\right) + 3p^2\left(\ell+\frac{1}{2}\right) + \frac{7}{6}p.$$

Démonstration. La technique de coloration est similaire à celle donnée dans le Théorème 4.2.3. Dans ce lemme, nous prouvons que pour de grandes valeurs de n une seule couleur supplémentaire suffit pour colorer les r_i arêtes restantes pour tout ξ_i avec $r_i = n - q_i(1 + i + p\ell)$.
Pour tout ξ_i, soit Q_i un ensemble de couleurs qui contient $1 + i + p\ell$ différentes couleurs (avec $Q_i \neq Q_j$ pour tout $j \neq i$). Soit c une couleur telle que $c \notin Q_i$ pour tout i. Les arêtes de ξ_i sont colorées d'une manière cyclique, r_i fois, avec les couleurs $Q_i \cup \{c\}$ et les autres arêtes sont colorées d'une manière cyclique avec les couleurs de Q_i. La distance entre les arêtes qui partagent une même couleur du même ξ_i est supérieure à ℓ.

Soit $\mathcal{P}_i$, $i \in [1, p]$ l'ensemble des sommets incidents aux arêtes des r_i périodes de ξ_i colorées avec les couleurs $Q_i \cup \{c\}$. Dans notre algorithme de coloration, ces ensembles de sommets ne se chevauchent pas, ce qui signifie que $\mathcal{P}_1 \cap \mathcal{P}_2 \cap \cdots \cap \mathcal{P}_p = \varnothing$ car n est suffisamment grand. En effet,

$$\begin{aligned} n &= p^3\left(\ell^2+\ell+\tfrac{1}{3}\right) + 3p^2\left(\ell+\tfrac{1}{2}\right) + \tfrac{7}{6}p, \\ &= \textstyle\sum_{i=1}^{i=p} (i + p\ell)(2 + i + p\ell), \\ &= \textstyle\sum_{i=1}^{i=p} |\mathcal{P}_i|. \end{aligned}$$

Par conséquent, la distance entre les arêtes ayant la même couleur et appartenant au même ξ_i est supérieure à ℓ. □

4.3 Coloration d'arêtes ℓ-distance du graphe puissance des arbres k-aire complets

4.3.1 Préliminaires

Soit T un arbre k-aire complet. Nous utilisons les notations suivantes : soit h la hauteur de T. Les nœuds de T sont notés par $x_{i,j}$, avec l'indice i, $0 \leq i \leq h$ indique le niveau du nœud et j, $0 \leq j \leq k^i - 1$ représente la position de ce nœud dans un niveau donné i (i.e. la racine de l'arbre est située au niveau 0 et notée par $x_{0,0}$).

Soit T^p, avec $p \geq 1$, le graphe puissance d'un arbre k-aire complet T. Les notations précédentes (hauteur, niveau et position) restent les mêmes, car l'arbre k-aire complet T est inclus dans T^p. Tout au long de cette section, dans chaque figure les arêtes en gras et les arêtes fines seront utilisées pour représenter respectivement les arêtes de l'arbre k-aire complet T et les arêtes puissances de son graphe puissance T^p.

4.3.2 Construction du sous-graphe S_ℓ

Définition 4.3.1. Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$ et $h \geq p\,(\ell + 2)$, soit T un arbre k-aire complet de hauteur h. Soit $v = x_{\left\lfloor \frac{p\ell}{2} \right\rfloor + p,0} \in V(T)$ et $v' = x_{\left\lfloor \frac{p\ell}{2} \right\rfloor + p+1,0} \in V(T)$ deux sommets. Soit $Q_z = \{u \in V(T) : dist_T(u,z) \leq \left\lfloor \frac{p\ell}{2} \right\rfloor + p\}$ et $\mathcal{L}_z = \{u \in V(T) : \left\lceil \frac{p\ell}{2} \right\rceil + 1 \leq dist_T(u,z) \leq \left\lceil \frac{p\ell}{2} \right\rceil + p\}$ deux ensembles de sommets de $V\,(T^p)$. Alors, le sous-graphe partiel $S_\ell \subset T^p$ est défini de la manière suivante : $S_\ell = T^p\,[Q_v \cup Q_{v'}] - E\,(T^p\,[\mathcal{L}_z])$, avec $(v, v') \in E(T)$ si $p\ell$ est impair et $v = v'$ sinon. Les sommets v et v' sont les centres de S_ℓ.

La Figure 4.3 illustre deux exemples du sous-graphe $S_\ell \subset T^p$. Le premier cas (a) illustre S_ℓ pour $p\ell$ impair et le deuxième cas (b) donne un exemple du sous-graphe S_ℓ pour $p\ell$ pair.

Les centres de S_ℓ sont situés entre les niveaux $\left\lfloor \frac{p\ell}{2} \right\rfloor + p$ et $h - \left\lceil \frac{p\ell}{2} \right\rceil - p$. Par conséquent, les sommets qui sont à distance au plus $\left\lfloor \frac{p\ell}{2} \right\rfloor$ dans T des centres, ont un degré maximum. S_ℓ est le sous-graphe maximum tel que pour tout $e, e' \in E(S_\ell)$, $dist_{S_\ell}(e, e') \leq \ell$. En effet, d'après la Définition 4.3.1, nous choisissons les centres de S_ℓ et par la suite nous incluons toutes les arêtes telles que la distance entre deux arêtes ne dépasse pas la valeur ℓ.

Si $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p\,(\ell + 2)$, il est impossible d'appliquer la Définition 4.3.1 pour construire le sous-graphe S_ℓ. Nous définissons un autre sous-graphe S'_ℓ, comme le sous-graphe maximum de T^p tel que la distance entre deux arêtes de S'_ℓ est au plus égale à ℓ. S'_ℓ est une partie de S_ℓ. Soit H_ℓ, avec $H_\ell = S_\ell - S'_\ell$, le sous-graphe qui contient les arêtes manquantes de S_ℓ. Par conséquent, si $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p\,(\ell + 2)$,

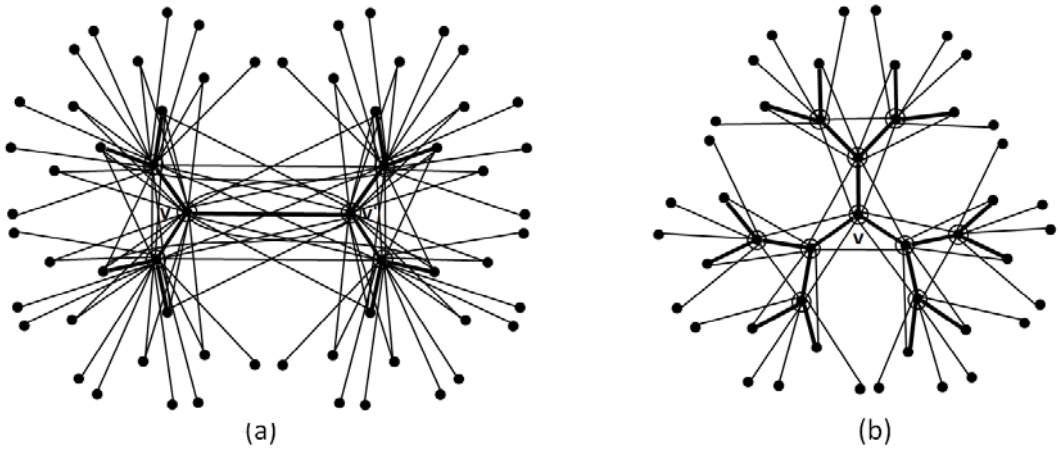

Figure 4.3 : *Deux exemples de sous-graphe S_ℓ si $h \geq p(\ell+2)$. (a) $k=2$, $\ell=1$, $p=3$ (b) $k=2$, $\ell=2$, $p=2$.*

le nombre d'arêtes du sous-graphe maximum S'_ℓ est égal à $|E(S'_\ell)| = |E(S_\ell)| - |E(H_\ell)|$. Une construction détaillée du sous-graphe S'_ℓ est donnée dans les Lemmes 4.3.6, 4.3.7, 4.3.8 et 4.3.9.

Dans la suite, nous présentons quelques caractéristiques du sous-graphe H_ℓ (voir Figure 4.4).

Définition 4.3.2. Soit H_ℓ un sous-graphe S_ℓ, H_ℓ contient trois types de sommets :

- *Les sommets internes* sont les sommets ayant le même degré maximum D que les centres : $D = (1+k)\sum_{i=0}^{i=p-1} k^i = \frac{1+k}{1-k}(1-k^p)$. La distance entre ces sommets et les centres est au plus égale à $\left\lfloor \frac{p\ell}{2} \right\rfloor$ dans T.
- *Les sommets périphériques* sont des extrémités de S_ℓ. Ils ont un même degré $\widetilde{D_j}$, qui dépend de la distance par rapport aux centres : $\widetilde{D_j} = 1+\sum_{i=0}^{i=p-j-2} k^j = 1 + \frac{1-k^{p-j-1}}{1-k}$.
- *Les sommets connecteurs* connectent le sous-graphe H_ℓ au reste du sous-graphe. Leur degré est égal à : $\overline{D_j} = \sum_{i=0}^{i=j-1} k^i = \frac{1-k^j}{1-k}$.

Nous notons que les sommets du sous-graphe S_ℓ ont un degré égal à soit D soit $\widetilde{D_j}$.

4.3.3 Résultats préliminaires

Le lemme suivant donne la taille du sous-graphe S_ℓ.

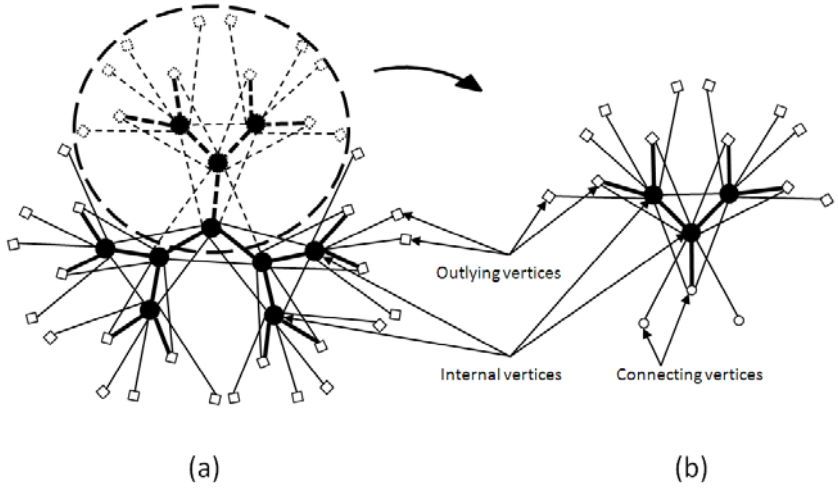

Figure 4.4 : *Ces exemples montrent trois types de sommets : internes, périphériques et connecteurs, pour un sous-graphe S_ℓ donné (a) et un sous-graphe $H_\ell \subset S_\ell$ (b).*

Lemme 4.3.3. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$ et $h \geq p(\ell+2)$. Soit T un arbre k-aire complet de hauteur h. Soit S_ℓ un sous-graphe construit selon la Définition 4.3.1. Le nombre d'arêtes du sous-graphe S_ℓ est donné par*

$$|E(S_\ell)| = \begin{cases} \frac{D}{1-k}\left(1-k^{\lfloor \frac{p\ell}{2} \rfloor+1}\right) + \sum_{i=0}^{i=p-1} \widetilde{D}_i k^{\lfloor \frac{p\ell}{2} \rfloor+i+1} & \text{si } p\ell \text{ est impair ,} \quad (a) \\ \frac{D}{2}\left(1+\frac{1+k}{1-k}\left(1-k^{\frac{p\ell}{2}}\right)\right) + \frac{1+k}{2}\sum_{i=0}^{i=p-1} \widetilde{D}_i k^{\frac{p\ell}{2}+i} & \text{sinon ,} \quad (b) \end{cases}$$

avec D et $\widetilde{D}_i$ sont donnés par la Définition 4.3.2.

Démonstration. D'après la Définition 4.3.1, pour calculer le nombre d'arêtes de S_ℓ, il y a deux cas selon la parité de $p\ell$. En effet, si $p\ell$ est impair (cas (a)), le sous-graphe S_ℓ possède deux centres $v = x_{i,j}$ et $v' = x_{i+1,j'}$ avec $\left\lfloor \frac{p\ell}{2} \right\rfloor + p \leq i \leq h - \left\lceil \frac{p\ell}{2} \right\rceil - p$ et possède un centre $v = x_{i,j}$ sinon (cas (b)). Nous présentons chaque cas séparément.

Cas (a) : Le nombre de sommets internes de S_ℓ est $N_D = 2\sum_{i=0}^{i=\lfloor \frac{p\ell}{2} \rfloor} k^i = \frac{2}{1-k}\left(1-k^{\lfloor \frac{p\ell}{2} \rfloor+1}\right)$. Le degré des sommets périphériques dépend de la distance au centre le plus proche. Il existe $N_{\widetilde{D_i}} = 2k^{\lfloor \frac{p\ell}{2} \rfloor+i+1}$ sommets à distance $\left\lfloor \frac{p\ell}{2} \right\rfloor + i + 1$ dans T des centres, et chaque sommet a un degré $\widetilde{D}_i$, avec $0 \leq i \leq p-1$. Puisque $\sum_{x_i \in V(G)} d(x_i) = 2|E(G)|$, alors

$$\begin{aligned} |E(S_\ell)| &= \tfrac{1}{2}\left(D.N_{\widetilde{D_i}} + \sum_{i=0}^{i=p-1} \widetilde{D}_i.N_{\widetilde{D_i}}\right), \\ &= \tfrac{D}{1-k}\left(1-k^{\lfloor \frac{p\ell}{2} \rfloor+1}\right) + \sum_{i=0}^{i=p-1} \widetilde{D}_i k^{\lfloor \frac{p\ell}{2} \rfloor+i+1}. \end{aligned}$$

Cas (b) : Nous utilisons le même raisonnement que le cas (a). Le nombre de sommets internes de S_ℓ est $N_D = 1 + (1+k)\sum_{i=0}^{i=\frac{p\ell}{2}-1} k^i = 1 + \frac{1+k}{1-k}\left(1-k^{\frac{p\ell}{2}}\right)$ et il existe $N_{\widetilde{D_i}} = (k+1)\,k^{\frac{p\ell}{2}+i}$ sommets périphériques à distance au moins $\frac{p\ell}{2} + i + 1$

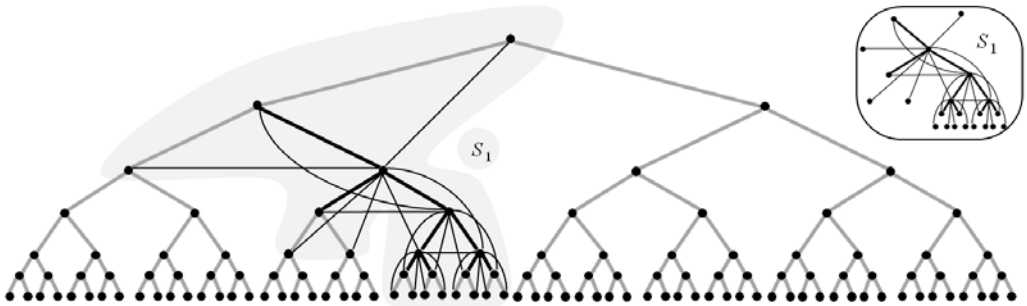

Figure 4.5 : *Exemple du sous-graphe $S_\ell \subset T^p$ avec $k = 2$, $\ell = 1$, $p = 2$ et $h = 6$. Le centre de S_ℓ est le sommet $v = x_{3,3}$. Les arêtes noires en gras et fines sont les arêtes du sous-graphe S_ℓ. Pour la lisibilité de la figure, nous gardons uniquement les arêtes puissances de S_ℓ.*

dans T au centre, tels que chacun a un degré $\widetilde{D_i}$, avec $0 \leq i \leq p-1$. Alors,

$$\begin{aligned} |E(S_\ell)| &= \tfrac{1}{2}\left(D.N_{\widetilde{D_i}} + \textstyle\sum_{i=0}^{i=p-1} \widetilde{D_i}.N_{\widetilde{D_i}}\right), \\ &= \tfrac{D}{2}\left(1 + \tfrac{1+k}{1-k}\left(1 - k^{\frac{p\ell}{2}}\right)\right) + \tfrac{1+k}{2} \textstyle\sum_{i=0}^{i=p-1} \widetilde{D_i} k^{\frac{p\ell}{2}+i}. \end{aligned}$$

□

4.3.4 Résultat du cas général

Dans le théorème suivant, nous déduisons la valeur de l'indice ℓ-chromatique à partir de la taille de S_ℓ.

Théorème 4.3.4. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$ et $h \geq p(\ell+2)$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est égal à*

$$\chi_\ell^{'}(T^p) = |E(S_\ell)|.$$

Démonstration. D'après la Définition 4.3.1, nous avons $\chi'_\ell(T^p) \geq |E(S_\ell)|$ car deux arêtes de S_ℓ sont à distance au plus ℓ. Pour prouver la valeur de la borne supérieure de l'indice ℓ-chromatique de T^p, par construction, nous colorons tout d'abord $E(S_\ell)$ avec $|E(S_\ell)|$ couleurs différentes. Pour toute arête non-colorée e adjacente à une arête colorée dans T^p, e est adjacente au plus à $|E(S_\ell)| - 1$ couleurs. En effet, S_ℓ est le sous-graphe maximum, il existe au moins une couleur possible pour colorer le graphe et vérifier ainsi la coloration d'arêtes ℓ-distance. Ainsi, $\chi'_\ell(T^p) \leq |E(S_\ell)|$. Nous déduisons le résultat suivant : $\chi'_\ell(T^p) = |E(S_\ell)|$. □

4.3.5 Résultats des cas particuliers

Soient v, v' les centres de S_ℓ. Alors, nous définissons par *branches* de S_ℓ les sous-graphes obtenus en supprimant les centres de S_ℓ ainsi que leurs arêtes incidentes. Les ensembles $N(v) = \{v', v_1, v_2, \ldots, v_{k-1}\}$ et $N(v') = \{v, v'_1, v'_2, \ldots, v'_{k-1}\}$ sont

respectivement les voisins de v et v'. Alors, les branches de S_ℓ sont $branch(S_\ell) = \{T_{v_1}, T_{v_2}, \ldots, T_{v'_1}, T_{v'_2}, \ldots\}$ où T_r est un arbre enraciné en r. Si ℓ est pair, S_ℓ possède $k+1$ branches, sinon il a $2k$ branches.

Nous étudions la valeur de l'indice ℓ-chromatique pour tous les cas particuliers en fonction de la hauteur de l'arbre. Nous présentons dans cette sous-section les résultats obtenus. Nous commençons par des valeurs faibles de la hauteur.

Lemme 4.3.5. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$ et $h \leq \left\lfloor \frac{p\ell}{2} \right\rfloor + 1$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est égal à*

$$\chi'_\ell(T^p) = |E(T^p)|.$$

Démonstration. D'après la définition de la coloration d'arêtes ℓ-distance, la distance entre deux arêtes ayant la même couleur est au moins égale à $\ell + 1$. Ainsi, si $h \leq \left\lfloor \frac{p\ell}{2} \right\rfloor + 1$, la distance entre toute paire d'arêtes du graphe puissance de T est inférieure à $\ell + 1$. Par conséquent, aucune couleur ne peut être répétée et $\chi'_\ell(T^p) = |E(T^p)|$. □

Si $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p(\ell + 2)$, alors le sous-graphe maximum est le sous-graphe S'_ℓ, sachant que S'_ℓ est une partie du sous graphe S_ℓ, et le sous-graphe H_ℓ, avec $H_\ell = S_\ell - S'_\ell$, contient les arêtes manquantes de S_ℓ. Pour trouver S'_ℓ, l'idée consiste à trouver le niveau approprié du centre de S'_ℓ. Ce niveau du centre de S'_ℓ doit garantir un nombre maximum des arêtes dans S'_ℓ. Comme nous pouvons le remarquer, si ℓ est pair, les feuilles de k branches de S_ℓ sont au même niveau. Cependant, pour la dernière branche, ses feuilles sont situées à des niveaux inférieurs (voir Figure 4.5). Ainsi, étant donné une hauteur h, nous devons choisir le niveau du centre de S'_ℓ tel que le nombre de feuilles de k branches de S'_ℓ soit maximum. Le même raisonnement s'applique pour ℓ impair. En effet, si ℓ est impair, les feuilles de k branches sont situées au niveau lev et les feuilles de $k-1$ branches sont situées au niveau $lev - 1$. Cependant, pour la dernière branche, ses feuilles sont situées à des niveaux inférieurs. Ainsi, étant donné une hauteur h, nous devons choisir le niveau du centre de S'_ℓ tel que le nombre de feuilles de $2k-1$ branches de S'_ℓ soit maximum.

Dans la suite, pour chaque valeur de h, nous calculons le nombre d'arêtes du sous-graphe S'_ℓ que nous déduisons de S_ℓ. Le nombre d'arêtes de S'_ℓ est égal à $|E(S'_\ell)| = |E(S_\ell)| - |E(H_\ell)|$.

Lemme 4.3.6. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$, $p\ell$ pair, et $\frac{p\ell}{2}+1 \leq h \leq \frac{p\ell}{2}+p$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est donné par*

$$\begin{aligned}\chi'_\ell(T^p) \geq\ & |E(S_\ell)| - \tfrac{1-k^{\frac{p\ell}{2}}}{2(1-k)}D - \tfrac{1}{2}\sum_{j=0}^{j=p-1}\widetilde{D_j}k^{\frac{p\ell}{2}+j} - \tfrac{1}{2}\sum_{j=1}^{j=p}\overline{D_j}k^{p-j} \\ & - \sum_{j=h-\frac{p\ell}{2}}^{j=p-1}\widetilde{D_j}k^{\frac{p\ell}{2}+j+1},\end{aligned}$$

avec D, $\widetilde{D_j}$, et $\overline{D_j}$ sont donnés par la Définition 4.3.2.

Démonstration. Pour maximiser le nombre de feuilles de k branches de S'_ℓ, nous devons choisir le centre de T^p (i.e. $x_{0,0}$) comme centre de S'_ℓ. Dans ce cas, en comparaison avec le sous-graphe S_ℓ, nous notons que :

- S_ℓ a $k+1$ branches et S'_ℓ a k branches. Alors, une seule branche B de S_ℓ appartient au sous-graphe H_ℓ (i.e $B \subset H_\ell$) avec $H_\ell = S_\ell - S'_\ell$. Ainsi, le nombre d'arêtes de B est égal à
  $\Theta = \frac{1}{2}\left(\sum_{j=0}^{j=\lfloor\frac{p\ell}{2}\rfloor-1} Dk^j + \sum_{j=0}^{j=p-1}\widetilde{D_j}k^{\lfloor\frac{p\ell}{2}\rfloor+j} + \sum_{j=1}^{j=p}\overline{D_j}k^{p-j}\right)$ où :
  - $\sum_{j=0}^{j=\lfloor\frac{p\ell}{2}\rfloor-1} Dk^j$ est la somme des degrés des sommets internes,
  - $\sum_{j=0}^{j=p-1}\widetilde{D_j}k^{\lfloor\frac{p\ell}{2}\rfloor+j}$ est la somme des degrés des sommets périphériques,
  - $\sum_{j=1}^{j=p}\overline{D_j}k^{p-j}$ est la somme des degrés des sommets connecteurs.
- L'excentricité du sommet u de S_ℓ est $ecc_{S_\ell}(u) = \frac{p\ell}{2}+p$ et l'excentricité du centre v de S'_ℓ est $ecc_{S'_\ell}(v) = h = \frac{p\ell}{2}+i$ avec $1 \leq i \leq p$. Alors, les arêtes incidentes des sommets périphériques de S_ℓ situées aux niveaux $\left[\frac{p\ell}{2}+i+1, \frac{p\ell}{2}+p\right]$ appartiennent à H_ℓ. Le nombre d'arêtes incidentes à ces derniers sommets est égal à $\Theta' = \sum_{j=i}^{j=p-1}\widetilde{D_j}k^{\frac{p\ell}{2}+j+1}$.

Par conséquent, le nombre d'arêtes est égal à $|E(H_\ell)| = \Theta + \Theta'$ (voir Figure 4.6.a). Nous obtenons ainsi la borne inférieure de l'indice ℓ-chromatique de T^p, $\chi'_\ell(T^p) \geq |E(S'_\ell)| = |E(S_\ell)| - \Theta - \Theta'$. □

Lemme 4.3.7. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$, $p\ell$ impair, et $\left\lfloor\frac{p\ell}{2}\right\rfloor+1 \leq h \leq \left\lfloor\frac{p\ell}{2}\right\rfloor+p$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est donné par*

$$\begin{aligned}\chi'_\ell(T^p) \geq\ & |E(S_\ell)| - \tfrac{1-k^{\lfloor\frac{p\ell}{2}\rfloor}}{2(1-k)}D - \tfrac{1}{2}\sum_{j=0}^{j=p-1}\widetilde{D_j}k^{\lfloor\frac{p\ell}{2}\rfloor+j} - \tfrac{1}{2}\sum_{j=1}^{j=p}\overline{D_j}k^{p-j} \\ & - \sum_{j=i-1}^{j=p-1}\widetilde{D_j}k^{\lfloor\frac{p\ell}{2}\rfloor+j+2} - (k-1)\sum_{j=i}^{j=p-1}\widetilde{D_j}k^{\lfloor\frac{p\ell}{2}\rfloor+j+1},\end{aligned}$$

avec D, $\widetilde{D_j}$, et $\overline{D_j}$ sont donnés par la Définition 4.3.2.

Démonstration. Les centres de S'_ℓ sont les sommets v et v' tels que v est la racine de T^p (i.e $x_{0,0}$) et $(v,v') \in E(T)$. Pour calculer le nombre d'arêtes de S'_ℓ, nous utilisons

le même raisonnement que dans le Lemme 4.3.6 : Une branche B de S_ℓ appartient au sous-graphe H_ℓ. Le nombre d'arêtes de B est égal à Θ (voir preuve du Lemme 4.3.6).

Pour les autres branches, les arêtes incidentes aux sommets périphériques de S_ℓ situées du niveau $\left\lfloor \frac{p\ell}{2} \right\rfloor + i + 1$ au niveau $\left\lfloor \frac{p\ell}{2} \right\rfloor + p$ appartiennent à H_ℓ. Le nombre de ces arêtes incidentes est égal à $\Theta" = \sum_{j=i-1}^{j=p-1} \widetilde{D_j} k^{\left\lfloor \frac{p\ell}{2} \right\rfloor + j + 1} + (k-1) \sum_{j=i}^{j=p-1} \widetilde{D_j} k^{\left\lfloor \frac{p\ell}{2} \right\rfloor + j}$.

Par conséquent, le nombre d'arêtes de H_ℓ est égal à $|E(H_\ell)| = \Theta + \Theta"$ (voir Figure 4.6.b). Nous obtenons ainsi la borne inférieure de l'indice ℓ-chromatique de T^p, $\chi'_\ell(T^p) \geq |E(S'_\ell)| = |E(S_\ell)| - \Theta - \Theta"$. □

Lemme 4.3.8. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$, et $\left\lfloor \frac{p\ell}{2} \right\rfloor + p < h \leq p(\ell+1)$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est donné par*

$$\chi'_\ell(T^p) \geq |E(S_\ell)| - \frac{1 - k^{\left\lceil \frac{p\ell}{2} \right\rceil - i}}{2(1-k)} D - \frac{1}{2} \sum_{j=0}^{j=p-1} \widetilde{D_j} k^{\left\lceil \frac{p\ell}{2} \right\rceil - i + j + 1} - \frac{1}{2} \sum_{j=1}^{j=p} \overline{D_j} k^{p-j},$$

avec D, $\widetilde{D_j}$, et $\overline{D_j}$ sont donnés par la Définition 4.3.2.

Démonstration. Si $h = \left\lfloor \frac{p\ell}{2} \right\rfloor + p + i, 1 \leq i < \left\lceil \frac{p\ell}{2} \right\rceil$, alors le centre de S'_ℓ est localisé au niveau i si $p\ell$ est pair et les centres de S'_ℓ sont localisés aux niveaux $i-1$ et i si $p\ell$ est impair. Dans ce cas, le sous-graphe H_ℓ constitue uniquement une partie d'une branche de S_ℓ, car $\left\lfloor \frac{p\ell}{2} \right\rfloor + p < h < p(\ell+1)$ et $ecc_{S'_\ell}(u) = \left\lceil \frac{p\ell}{2} \right\rceil + p$ avec u est un centre de S'_ℓ. Pour calculer le nombre d'arêtes de S'_ℓ, nous notons que H_ℓ contient des sommets internes, des sommets périphériques et des sommets connecteurs, tels que :

- $\sum_{j=0}^{j=\left\lceil \frac{p\ell}{2} \right\rceil - i - 1} D k^j$ est la somme des degrés des sommets internes,
- $\sum_{j=0}^{j=p-1} \widetilde{D_j} k^{\left\lceil \frac{p\ell}{2} \right\rceil - i + j}$ est la somme des degrés des sommets périphériques,
- $\sum_{j=1}^{j=p} \overline{D_j} k^{p-j}$ est la somme des degrés des sommets connecteurs.

Le nombre d'arêtes du sous-graphe maximum S'_ℓ est ainsi égal à $|E(S_\ell)| - \frac{1}{2} \sum_{j=0}^{j=\left\lceil \frac{p\ell}{2} \right\rceil - i - 1} D k^j - \frac{1}{2} \sum_{j=0}^{j=p-1} \widetilde{D_j} k^{\left\lceil \frac{p\ell}{2} \right\rceil - i + j} - \frac{1}{2} \sum_{j=1}^{j=p} \overline{D_j} k^{p-j}$ (voir Figure 4.6.c). □

Lemme 4.3.9. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$, et $p(\ell+1) \leq h < p(\ell+2)$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est*

$$\chi'_\ell(T^p) \geq |E(S_\ell)| - \sum_{j=i}^{j=p-1} \widetilde{D_j} k^j,$$

avec $\widetilde{D_j}$ est donné par la Définition 4.3.2.

Démonstration. Si $h = p(\ell+1)+i, 0 \leq i \leq p-1$, alors les arêtes de H_ℓ sont les arêtes incidentes à $\sum_{j=i}^{j=p-1} k^j$ sommets périphériques (voir Figure 4.6.d). Nous obtenons ainsi la borne inférieure de l'indice ℓ-chromatique de T^p, $\chi'_\ell(T^p) \geq |E(S'_\ell)| = |E(S_\ell)| - \sum_{j=i}^{j=p-1} \widetilde{D_j} k^j$. □

Dans le théorème suivant, nous avons la valeur exacte de l'indice ℓ-chromatique pour $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p(\ell+2)$.

Théorème 4.3.10. *Pour tout $p \geq 1$, $\ell \geq 0$, $k \geq 2$, et $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p(\ell+2)$, soit T un arbre k-aire complet de hauteur h. Alors, l'indice ℓ-chromatique de T^p est égal à*

$$\chi'_\ell(T^p) = |E(S'_\ell)|.$$

Démonstration. D'après les Lemmes 4.3.6, 4.3.7, 4.3.8, et 4.3.9, nous avons $\chi'_\ell(T^p) \geq |E(S_\ell)|$. Si $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p(\ell+2)$ et pour chaque cas particulier, nous trouvons le sous-graphe maximum S'_ℓ.

Pour prouver la borne supérieure de l'indice ℓ-chromatique, nous utilisons le même raisonnement que dans le Théorème 4.3.4. En effet, nous colorons au début $E(S'_\ell)$ avec $|E(S'_\ell)|$ couleurs différentes. Pour toute arête non-colorée e adjacente à une arête colorée dans T^p, e est adjacente à au plus $|E(S'_\ell)| - 1$ couleurs car S'_ℓ est le sous-graphe maximum. Ainsi, il existe au moins une couleur possible pour colorer le graphe et vérifier la coloration d'arêtes ℓ-distance. Alors, $\chi'_\ell(T^p) \leq |E(S'_\ell)|$ et par conséquent $\chi'_\ell(T^p) = |E(S'_\ell)|$.

□

4.4 Coloration d'arêtes ℓ-distance du graphe puissance des arbres quelconques

Selon le résultat précédent de l'indice ℓ-chromatique du graphe puissance des arbres k-aires complets, nous déduisons le résultat pour le graphe puissance des arbres quelconques.

Corollaire 4.4.1. *Pour tout $p \geq 1$, $\ell \geq 0$, soit T un arbre quelconque de hauteur h et T_k un arbre k-aire complet de hauteur h. Alors, nous avons : $\chi'_\ell(T^p) \leq \chi'_\ell\left(T^p_{\Delta_{T^p}-1}\right)$ avec Δ_{T^p} est le degré maximum de T^p.*

Démonstration. Ce corollaire est déduit des Théorèmes 4.3.4, 4.3.10 et du Lemme 4.3.5 car T^p est un graphe partiel de $T^p_{\Delta_{T^p}-1}$. □

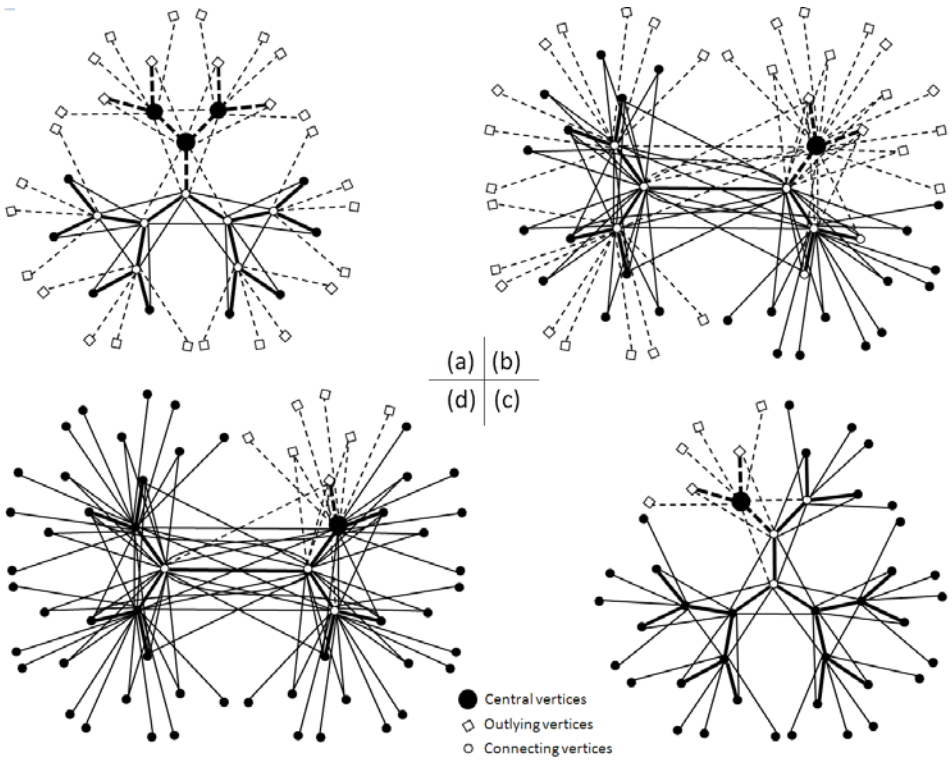

Figure 4.6 : *Chaque graphe représente un exemple du sous-graphe S_ℓ (les arêtes continues + les arêtes interrompues). Pour une hauteur h donnée $\left\lfloor \frac{p\ell}{2} \right\rfloor + 1 \leq h < p(\ell+2)$, le sous-graphe maximum S'_ℓ est représenté uniquement par les arêtes continues. (a) $\ell = 2$, $p = 2$ et $h = 3$ (b) $\ell = 1$, $p = 3$ et $h = 4$ (c) $\ell = 2$, $p = 2$ et $h = 5$ (d) $\ell = 1$, $p = 3$ et $h = 6$.*

4.5 Conclusion

Dans ce chapitre, nous avons étudié la coloration d'arêtes ℓ-distance des graphes puissances. Nous nous sommes basés sur l'étude précédemment effectuée du paramètre. Nous avons défini un sous-graphe maximum pour chaque graphe puissance pour déterminer la borne inférieure du paramètre de la coloration d'arêtes ℓ-distance ainsi qu'un algorithme de coloration pour chacun.

Deuxième partie

Algorithmes auto-stabilisants dans les graphes

Chapitre 5

Algorithme auto-stabilisant pour la coloration propre d'arêtes

Résumé

*La plupart des résultats de la théorie des graphes ont été revus dans le but de proposer des versions auto-stabilisantes. Dans ce chapitre, nous commençons par présenter les définitions nécessaires à la compréhension de ce paradigme d'auto-stabilisation. Pour une présentation détaillée du sujet, de ses enjeux et des problèmes que l'auto-stabilisation permet de résoudre, on peut se référer au livre de Dolev [*Dolev 00*]. Une bibliographie en ligne accompagnée d'un guide d'accès à la littérature est proposée par Ted Herman [*Herman 02*] et des surveys dans ce domaine de recherche ont été écrits par Marco Schneider [*Schneider 93*], Mitchell Flatebo, Ajoy Datta et Sukumar Ghosh [*Flatebo 94*] et Mohamed Gouda [*Gouda 95*]. Ensuite, nous présentons un algorithme auto-stabilisant pour la coloration propre d'arêtes. Notre solution assure une coloration avec au plus* $(\Delta + 1)$ *couleurs et une complexité en* $O(m(\Delta + n))$ *, où* Δ*,* n *et* m *désignent respectivement le degré maximum, le nombre de sommets et le nombre d'arêtes dans le graphe.*

5.1 Introduction à l'auto-stabilisation

Avec la multiplication du nombre de nœuds, les réseaux sont de plus en plus vulnérables aux défaillances (ou pannes). L'étude des algorithmes s'oriente depuis des années vers la prise en compte des défaillances potentielles des éléments du réseau. Les systèmes auto-stabilisants font l'hypothèse que les défaillances sont limitées dans le temps. Ils garantissent qu'après l'occurrence de toutes les pannes transitoires, le réseau retrouve un fonctionnement (ou comportement) correct en un temps fini. L'un des avantages de ces systèmes est qu'ils atteignent d'eux-mêmes, sans intervention extérieure, une configuration correcte à partir d'une configuration quelconque.

La Figure 5.1 illustre le fonctionnement des systèmes auto-stabilisants.

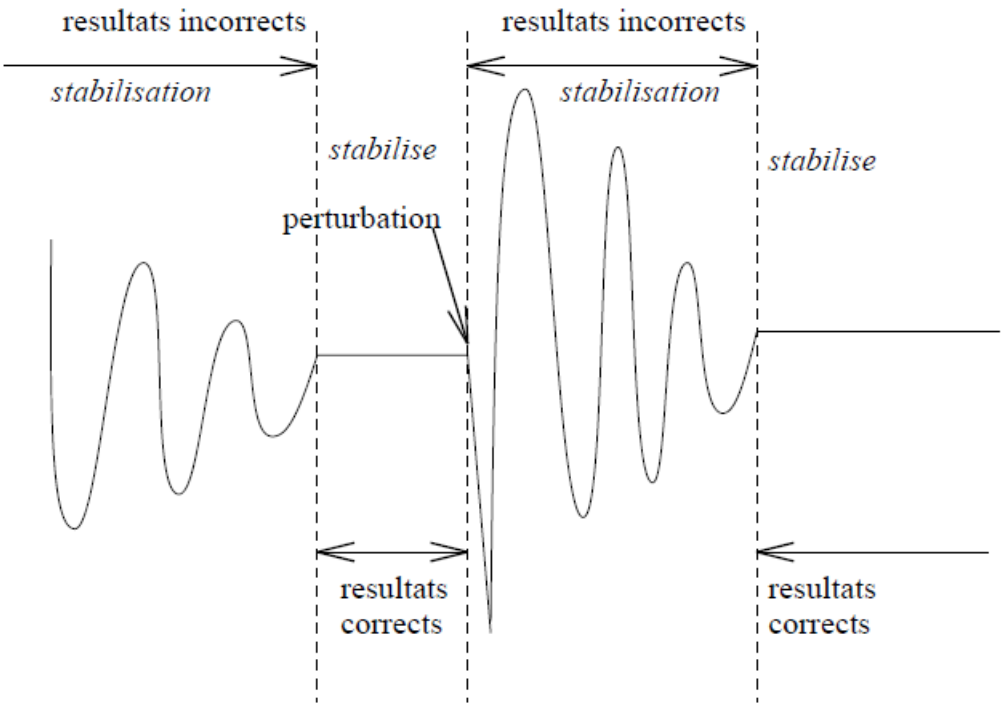

Figure 5.1 : *L'auto-stabilisation selon Dijkstra.*

5.2 Classification des pannes

Une panne peut provoquer une erreur, qui est un état incorrect, ce qui peut conduire à un échec, qui est un écart par rapport à la spécificité du système. Dans la littérature, les pannes sont classifiées en trois grandes catégories [Anderson 08] [Cristian 91] :

- **Panne temporelle** : Une sortie correcte associée à une requête entrante se manifeste de façon incohérente avec les spécifications [1] temporelles (trop tard, trop tôt)
- **Panne transitoire ou intermittente** : En réponse à un événement en entrée un composant ne délivre jamais la réponse attendue (le composant ne peut fournir son service habituel pendant une certaine période d'où perte de quelques données). Ultérieurement, il peut fonctionner à nouveau de façon correcte.
  - une panne transitoire apparaît une seule fois puis disparaît.
  - une panne intermittente apparaît plus ou moins périodiquement.
- **Panne quelconque ou byzantine** : Tout comportement s'écartant des spécifications (principalement lorsque les résultats sont non conformes) est qualifié de comportement byzantin [Lamport 83]. Nous distinguons quelquefois :
  - Fautes byzantines "naturelles", par exemples :
    - Erreur physique non détectée (sur une transmission de messages, en mémoire, sur une instruction).
    - Erreur logicielle.
  - Fautes byzantines "malicieuses", par exemple : Comportement visant à faire échouer le système (virus, etc.).

Les systèmes auto-stabilisants ont la capacité de gérer des événements de natures diverses qui perturbent leur fonctionnement. En particulier, ils permettent de tolérer toute défaillance transitoire.

5.3 Les algorithmes auto-stabilisants

Les algorithmes auto-stabilisants ont la propriété de retrouver un comportement correct après une corruption du système. Un algorithme auto-stabilisant est défini autour de deux propriétés : la convergence et la correction du système. La convergence garantit qu'à partir d'une configuration quelconque le système retrouvera au bout d'un temps fini un comportement correct, tandis que la propriété de correction assure que partant d'une configuration correcte, le système reste dans une configuration correcte. Ces deux propriétés garantissent que le système retrouvera un comportement correct à partir d'une configuration quelconque, tant qu'aucune panne ne vient perturber le système.

Dans cette section, nous présentons un aperçu de la théorie de l'auto-stabilisation et ses importants concepts : la convergence et la correction.

1. Résoudre une tâche implique d'être en mesure de la décrire avec précision, c'est à dire de la spécifier. Une spécification est une famille d'ensembles d'exécutions possédant une propriété commune. Chaque ensemble d'exécutions se rapporte à un système particulier résolvant la tâche spécifiée.

5.3.1 Formalisation

Définition 5.3.1. Un système est l'ensemble $S = (C, \rightarrow)$ tel que C est l'ensemble des états du système et $\rightarrow$ est une relation de transition binaire sur C. Une exécution E de S est une suite alternée infinie de configurations et de pas $E = (c_0, a_1, c_1, a_2, c_2, \ldots)$ telle que pour tout $i > 0$, l'action a_i fait passer le système de la configuration c_{i-1} à la configuration c_i.

Un algorithme auto-stabilisant est un algorithme caractérisé par un ensemble de transitions sur l'ensemble des états. Le caractère auto-stabilisant de l'algorithme signifie que, quel que soit l'état du système dont on part et quelle que soit la séquence de transitions effectuées, le système aboutit toujours à un état légitime. Dans un algorithme auto-stabilisant, chaque sommet maintient ses propres variables locales, et il peut prendre des décisions basées uniquement sur son état et les états de ses voisins. L'état global du système est l'union de tous les états locaux de ses sommets.

Au cours d'une transition, une entité peut recevoir un message, changer d'état, et envoyer des messages. Elle est dite équitable si elle ne contient pas de séquence infinie de pas au cours de laquelle une certaine transition pourrait être exécutée, mais ne l'est jamais. En d'autres termes, dans une exécution équitable, aucun nœud n'est privé de la possibilité d'effectuer une certaine transition.

Nous allons d'abord définir plus précisément le comportement correct d'un système auto-stabilisant, appelé configuration légitime.

Définition 5.3.2. (Configuration légitime) Étant donné une spécification P sur les exécutions, une configuration légitime pour un système est une configuration du système qui vérifie la spécification P. Nous notons $\mathcal{L}$ l'ensemble des configurations légitimes du système.

Il faut noter qu'une configuration qui n'est pas légitime est appelée configuration illégitime. Nous pouvons maintenant introduire la définition d'un système auto-stabilisant.

Définition 5.3.3. (Auto-stabilisation) Un système S est dit auto-stabilisant vers la spécification P si et seulement s'il existe un sous-ensemble $\mathcal{L} \subset C$ de configurations du système, dit ensemble de configurations légitimes, tel que les propriétés suivantes soient vérifiées :

- Convergence : toute exécution de l'algorithme conduit vers une configuration appartenant à $\mathcal{L}$.
- Correction : toute exécution dont la configuration initiale est dans $\mathcal{L}$ vérifie la spécification P [Tel 94].

Ces deux propriétés assurent bien un retour vers un état légitime, puisque, partant d'un état quelconque, une exécution conduit vers un état légitime, à partir duquel la specification P est vérifiée.

Le temps (nombre de mouvements) mis par le système pour atteindre un comportement quelconque à la suite d'une panne est appelé temps de stabilisation (ou temps de convergence).

Définition 5.3.4. (temps de stabilisation) Étant donné un protocole auto-stabilisant destiné à accomplir une certaine tâche, le temps de stabilisation est le temps maximum mis pour qu'en partant d'une configuration quelconque, l'ensemble du réseau retrouve un fonctionnement correct.

La mesure dans le pire des cas du temps de stabilisation constitue évidemment un critère essentiel de comparaison des performances de protocoles destinés à accomplir une même tâche.

5.3.2 Notion de démon

Dans un système auto-stabilisant, chaque entité dispose d'un protocole lui indiquant les actions à accomplir. Du point de vu de la modélisation, une telle multitude d'exécutions rend la conception et la démonstration d'algorithmes particulièrement difficile à établir. D'où l'introduction du *démon.*

Un démon est une hypothèse simplificatrice faite sur l'ensemble des exécutions d'un système auto-stabilisant. Classiquement, il limite le nombre d'exécutions possibles d'un système, rendant ainsi possible la formalisation et la preuve d'algorithmes. Il se présente sous forme d'un prédicat sur les exécutions du système. Concrètement, la majorité des démons fixe un ensemble de contraintes que les exécutions des systèmes doivent suivre. Les démons fixent l'atomicité des actions à exécuter, c'est à dire que l'action ne peut pas être interrompue par une autre action. Ils déterminent aussi si plusieurs entités peuvent agir en même temps ou bien si les actions seront exécutées les unes après les autres.

Les contraintes d'atomicité concernent l'ordre des actions apparaissant dans l'ensemble des traces [Tixeuil 00] [Dolev 93] [Dijkstra 74] [Tsai 94] [Israeli 93] [Ducourthial 00].

- Un démon à lecture-écriture n'impose pas d'ordre particulier sur les actions [Dolev 93].
- Un démon totalement réparti autorise une entité à lire l'ensemble des variables de ses voisins ou à écrire dans toutes ses variables propres en une action. A part cela, aucun ordre n'est imposé [Tsai 94].
- Un démon réparti choisit un ensemble d'entités parmi celles qui peuvent agir. Puis toutes ces entités doivent simultanément lire les variables de leurs voi-

sines. Enfin, elles doivent toutes agir. L'ensemble constitue une action atomique [Dijkstra 74].
- Un démon est synchrone si et seulement si, il est réparti et toute séquence de phases implique toutes les entités du système [Dijkstra 74].
- Un démon est centralisé si et seulement si, il est réparti et toute séquence de phase implique une unique entité du système [Dijkstra 74] [Herman 90].

A cela viennent s'ajouter des propriétés sur la manière de choisir l'ensemble des entités à agir :
- Un démon équitable est totalement libre.
- Un démon non-équitable interdit qu'une entité pouvant agir soit systématiquement exclue de l'ensemble des entités à agir.
- Un démon probabiliste choisit d'une manière aléatoire l'ensemble des entités à agir.

5.3.3 Avantages

Les algorithmes auto-stabilisants offrent les avantages suivants par rapport aux algorithmes classiques :
- Tolérance aux fautes : un algorithme auto-stabilisant offre une protection automatique contre toutes les pannes transitoires.
- Initialisation : aucun besoin d'initialisation du système n'est nécessaire puisque le système garantit d'atteindre une configuration légitime partant d'une configuration quelconque.
- Dynamisme : La propriété d'auto-stabilisation est également en rapport avec la dynamicité des réseaux (réseau dont la topologie change au cours du temps). Un protocole est bien adapté aux réseaux dynamiques si son code n'a pas besoin d'être modifié en cas d'ajout ou de retrait de nœuds ou d'arêtes dans le réseau, et s'il tolère des changements de la topologie pendant son fonctionnement.

5.3.4 Notation et terminologie

Dans la suite de ce chapitre et pour le chapitre suivant, nous nous basons sur les notations et terminologie suivantes. Un système auto-stabilisant peut être modélisé par un graphe simple connecté non orienté $G = (V, E)$ avec V représente l'ensemble des sommets et E l'ensemble des arêtes représentent les liens de communication entre les sommets. Chaque sommet exécute un algorithme qui contient un ensemble fini d'actions sous la forme suivante : $\langle label \rangle :: \langle garde \rangle \longrightarrow \langle action \rangle$, avec *garde* est une expression booléenne. Une *action* met à jour l'état des sommets. L'exécution d'une action est appelée *mouvement*. Il est supposé que les règles sont exécutées

d'une manière automatique, i.e. l'évaluation des gardes et des mouvements est effectuée en une seule étape. Un sommet de G est soit *stable* soit *actif*. Le graphe G est stable lorsque tous ses sommets sont stables. Il suffit d'avoir un seul sommet actif pour dire que le système n'est pas stable. Un sommet est dit actif lorsque toutes les gardes d'une règle sont évaluées à vrai.

Un démon centralisé choisit un sommet pour exécuter une action. Dans chaque mouvement, il choisit un sommet arbitraire parmi les sommets actifs et la règle qui le rend actif. Par la suite, l'action associée à cette règle est exécutée. Le système est stable, lorsqu'aucun mouvement n'est possible. Afin de mesurer la complexité en temps des algorithmes auto-stabilisants, nous utilisons souvent le nombre de mouvements.

5.4 Travaux existants liés à la théorie des graphes

C'est Edsger Wybe Dijkstra [Dijkstra 74] qui a introduit la notion de système auto-stabilisant. Dans [Dijkstra 74], l'auteur présente un algorithme auto-stabilisant d'exclusion mutuelle. Pendant neuf ans, ce résultat resta marginal jusqu'à une intervention de Leslie Lamport [Lamport 83] qui fit remarquer l'importance et la puissance potentielle que présentait l'auto-stabilisation. Depuis, l'intérêt pour le domaine ne fait que croître [Wuu 95] [Schneider 93] [Flatebo 94] [Gouda 95] [Tel 94].

Un algorithme auto-stabilisant ne nécessite aucune initialisation pour fonctionner correctement, peut récupérer toutes défaillances passagères de types arbitraires survenant à n'importe quel moment et il est insensible aux topologies dynamiques. Cela rend l'auto-stabilisation une solution intéressante pour un grand nombre d'applications, notamment la théorie des graphes [Butelle 95] [Huang 98] [Israeli 90] [Itkis 92] [Mayer 92] [Umemoto 98], le routage et les protocoles de communication [Awerbuch 95] [Dolev 97] [Ghosh 97] [Gouda 91] [Katz 93], et autres.

Les problèmes de la théorie des graphes ne cessent de recevoir une grande attention dans le domaine des systèmes auto-stabilisants. Plusieurs travaux ont été étudiés. Par exemples, la coloration de graphes est utilisée dans les problèmes d'exclusion mutuelle et l'allocation de ressources. L'ensemble dominant est utilisé dans le clustering. Les algorithmes du plus court chemin sont utilisés dans le routage, etc. La théorie des graphes est un outil naturel pour modéliser ces problèmes. Un survey récent détaillant une bibliographie très riche des algorithmes auto-stabilisants est disponible dans le papier de Guellati et Kheddouci [Guellati 10].

Nous nous sommes intéressés à l'étude des algorithmes auto-stabilisants pour la coloration de graphes.

Les algorithmes auto-stabilisants pour la coloration de sommets ont été intensi-

vement étudiés dans la littérature, par exemples dans [Gradinariu 00] [Ghosh 93] [Sur 93] [Hedetniemi 03], etc.

Quelques travaux sur les algorithmes auto-stabilisants pour la coloration d'arêtes ont été présentés dans la littérature. Les auteurs dans [Sakurai 04] proposent un algorithme auto-stabilisant pour la coloration propre d'arêtes résistant aux pannes byzantines et non uniquement aux pannes transitoires dans les réseaux sous forme d'arbres. L'algorithme utilise $(\triangle + 1)$ couleurs et chaque nœud atteint son état légitime dans trois tours. Plus tard, Masuzawa et Tixeuil, proposent un algorithme auto-stabilisant pour la coloration d'arêtes de graphes quelconques [Masuzawa 05] [Masuzawa 07]. L'algorithme suppose un démon centralisé, et utilise $(2\triangle - 1)$ couleurs. Cet algorithme garantit que toute arête (u, v) entre les nœuds u et v aura une couleur dans $(2\triangle + 2)$ mouvements et sa couleur reste inchangée par la suite. Dans cet article, les auteurs supposent que l'ensemble des exécutions de l'algorithme est géré par un démon non-équitable. Dans [Masuzawa 09], ils utilisent les concepts de la coloration de sommets et d'arêtes afin de construire une connaissance locale de la topologie : Chaque nœud reçoit une couleur, puis, le nœud qui reçoit le plus de couleurs décide de la couleur d'une arête. Un nœud acquiert les connaissances locales de la topologie en collectant les couleurs des nœuds voisins et les couleurs des arêtes voisines. Bien que l'algorithme soit un algorithme probabiliste qui permet de sélectionner au hasard les couleurs et entraîne une coloration non-optimale de sommet/arête, il peut fonctionner dans les réseaux anonymes par un démon distribué qui permet l'exécution simultanée des nœuds.

L. Kuszner et al. présentent un autre algorithme auto-stabilisant pour la coloration propre d'arêtes qui fonctionne dans un modèle où le démon choisit un sommet quelconque parmi les actifs. Dans [Kuszner 06], les auteurs attribuent des couleurs aux arêtes du graphe de la même manière que dans l'algorithme de Hsu-Huang [Hsu 92]. Leur coloration est un $(2\triangle - 1)$-coloration d'arêtes de G et s'exécute en $O(m\triangle)$ mouvements, où m est le nombre d'arêtes du graphe.

Une autre borne supérieure est présentée dans [Tzeng 07]. Cet article propose un algorithme auto-stabilisant pour la coloration propre d'arêtes en utilisant $(\triangle+4)$ couleurs pour un graphe planaire. Sa complexité est $O(n^2)$ et les exécutions de l'algorithme sont gérées par un démon centralisé. Les auteurs se basent sur l'étiquetage des nœuds du graphe pour définir une priorité aux arêtes. L'idée d'étiquetage est la même que celle de Ghosh et Karaata [Ghosh 93], elle se base sur la propriété suivante des graphes planaires.

Proposition 5.4.1. *Tout graphe planaire contient au moins un sommet de degré au plus 5.*

Ensuite, les couleurs sont assignées aux arêtes en fonction de leurs priorités. Dans [Huang 09], Huang et Tzeng proposent un autre algorithme auto-stabilisant pour

la coloration propre d'arêtes des graphes bipartis avec $\triangle$ couleurs et s'exécute en $O(n^2k + m)$ mouvements, avec k est le nombre d'arêtes qui ne sont pas proprement colorées dans une configuration initiale. Si $k = O(1)$, le temps de stabilisation est $O(n^2)$ et si $k = O(m)$, le temps de stabilisation devient $O(n^2m)$.

Le tableau 5.2 résume l'ensemble des algorithmes auto-stabilisants pour la coloration propre d'arêtes.

5.5 Algorithme auto-stabilisant pour la coloration propre d'arêtes

Dans cette section, nous décrivons notre algorithme auto-stabilisant pour la coloration propre d'arêtes. Nous utilisons les mêmes notations et terminologies que dans la section 5.3.4.

5.5.1 Préliminaires

Soit $G = (V, E)$ un graphe simple connecté non orienté avec V représente l'ensemble des sommets et E l'ensemble des arêtes représentent les liens de communication entre les sommets. L'ensemble des voisins d'un sommet $u \in V$, c'est à dire les sommets adjacents à u n'incluant pas u lui-même, forment un sous-graphe induit appelé voisinage de u. Soit $N.u$ l'ensemble des voisins de u et $deg.u$ son degré. Nous utilisons les couleurs de l'ensemble suivant $\{1, 2, .., \Delta + 1\}$. Pour exprimer la couleur d'une arête (u, v) d'une manière distribuée, le sommet u (respectivement le sommet v) maintient une variable $C(v).u$ (respectivement $C(u).v$) qui contient la couleur associée à cette arête. Chaque arête possède ainsi deux variables de couleurs. Pour éviter les éventuels conflits de couleurs, un seul sommet décide de la couleur d'une arête et l'autre extrémité de l'arête copie cette couleur, i.e pour une arête (u, v) le sommet u choisit une couleur propre pour $C(v).u$ et le sommet v met à jour sa variable $C(u).v = C(v).u$. Le sommet qui décide de la couleur de l'arête est choisi par le démon centralisé pour exécuter une règle. Nous notons par $C(u, v)$ la couleur de (u, v), c'est à dire que la même couleur est partagée entre les sommets u et v (i.e. $C(u).v = C(v).u$). Il est défini uniquement s'il y a égalité entre les variables de couleurs propres à chaque sommet.

Nous appelons couleur incidente à un sommet, une arête incidente à ce sommet colorée avec cette couleur. Si aucune des arêtes incidentes n'est colorée avec une couleur donnée, alors cette couleur est libre (ou manquante) sur ce sommet. Nous pouvons déduire qu'il existe au moins une couleur libre au niveau de chaque sommet.

Références	Topologie	Nombre de couleurs	Démon centralisé	Complexité
Y. Sakurai et al.	Arbre	$\triangle + 1$	non-équitable	$O(3)$ mouvements
T. Masuzawa et al.	Quelconque	$2\triangle - 1$	non-équitable	$O(2\triangle + 2)$ mouvements
Ł. Kuszner et al.	Quelconque	$2\triangle - 1$	équitable	$O(m\triangle)$ mouvements
C. H. Tzeng et al.	Planaire	$\triangle + 4$	équitable	$O(n^2)$ mouvements
S. T. Huang et al.	Biparti	$\triangle$	équitable	$O(n^2m)$ mouvements

Figure 5.2 : *Algorithmes auto-stabilisants pour la coloration propre d'arêtes.*

Algorithme 3 Elimination du conflit.

$(R0)$: : if $\exists v, v^{'} \in V(G)$ such that $v \neq v^{'}$
and $C(u,v) = C(u,v^{'})$
then $C(v).u = Null$

Algorithme 4 Coloration propre de l'arête.

$(R1)$: : if $Free(u) \cap Free(v) \neq \emptyset$ and $C(u,v) = Null$
then $Color(u,v)$

En effet, il y a $\Delta + 1$ couleurs possibles et le degré de chaque sommet est inférieur strictement à $\Delta + 1$. Soit $Used(u) = \{C(v).u | v \in N.u\}$ l'ensemble des couleurs incidentes à u et $Free(u)$ l'ensemble des couleurs libres au niveau du sommet u. Nous avons ainsi pour tout $u \in V(G)$:

$$Free(u) \cup Used(u) = \{1, 2, .., \Delta + 1\}$$

Soit la fonction $Color(u,v)$ qui permet au sommet u de décider de la valeur de $C(v).u$:

$$Color(u,v) \equiv \text{pour fixer } C(v).u = min\{1, ..., \Delta + 1\} \setminus (Used(u) \cup Used(v))$$

Soit la fonction $Proper(u)$: $Proper(u) = vrai$ si toutes les arêtes, colorées et adjacentes entre elles, ont des couleurs différentes et $Proper(u) = faux$ s'il existe deux arêtes incidentes à u et ayant la même couleur.

Soit un fan F centré au niveau du sommet u et commençant à partir du sommet v. F contient une séquence de voisins de u ($F = \{v = x_0, x_1, ..., x_k\}$), tels que tous les x_i sont différents, et $C(u, x_{i+1}) \in Free(x_i)$ pour $i = 0, ..., k-1$. Si c, d sont des couleurs différentes, alors le (c,d)-path P représente la chaîne maximale contenant les deux couleurs alternées c, d. En inversant (ou alternant) P, signifie que nous échangeons la couleur c par d et la couleur d par c (voir sous-section 2.3.3).

5.5.2 Description de notre algorithme auto-stabilisant

Dans cette section, nous donnons les différentes règles qui composent notre algorithme auto-stabilisant, et les conditions requises pour les exécuter. Notre algorithme auto-stabilisant est composé de quatre parties de règles :

(Partie1) $(R0)$: cette règle est nécessaire pour éliminer les conflits potentiels. En effet, s'il existe deux arêtes adjacentes qui partagent la même couleur, $(R0)$ supprime ce conflit (voir Algorithme 3). Cette règle est privilégiée par rapport aux règles $(R1)$, $(R2)$ et $(R3)$.

(Partie 2) $(R1)$: cette règle est utilisée pour affecter une couleur à une arête.

Algorithme 5 Rotation du fan.

$(R2)$: : if $Test(u,v) = true$ and $C(u,v) = Null$
then $Rotate(u,v)$

Notation :
Fonction $Test(u,v)$ (voir algorithme 7)
Fonction $Rotate(u,v)$ (voir algorithme 8)

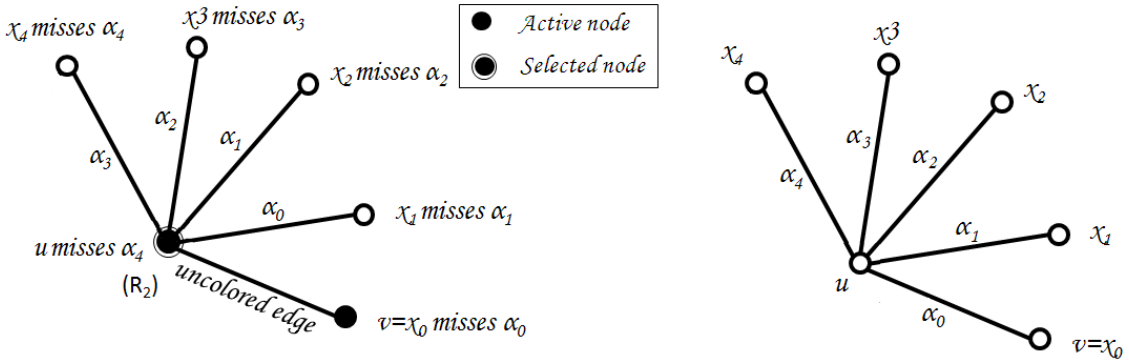

Figure 5.3 : *Rotation du fan.*

Elle propose une couleur possible s'il n'y a pas eu encore de couleur proposée (voir Algorithme 4). En effet, pour un sommet actif u, nous pouvons affecter la couleur c à une arête (u,v) sans changer la couleur des arêtes incidentes à u, car $Free(u) \cap Free(v) \neq \emptyset$ et $c \in Free(u) \cap Free(v)$. Si c n'existe pas, i.e $Free(u) \cap Free(v) = \emptyset$, alors la fonction *Test* (voir Algorithme 7) retourne *vrai* ou *faux*. Cette fonction *Test* vérifie respectivement si le sommet u effectue une rotation du fan (voir Partie (3)) ou inversion du (c,d)-path (voir Partie (4)).

(Partie 3) $(R2)$: cette règle est basée sur la technique de la rotation du fan. Faire une rotation du fan F, pour colorer une arête non-colorée $e = (u, x_0)$, signifie que nous colorons (u, x_i) pour tout i, $i = 0, ..., k-1$, et écrasons la couleur de (u, x_k). Une rotation de F engendre une autre coloration non-propre de G dans laquelle (u, x_k) est non-colorée (au lieu de (u, x_0)). Par conséquent, un sommet est actif pour exécuter cette règle, si la fonction *Test* retourne *vrai*. Soit (u,v) une arête telle que $Free(u) \cap Free(v) = \emptyset$, nous construisons un fan pour effectuer une rotation du fan (voir Algorithme 5). La fonction *Rotate* (voir Algorithme 8) exécute l'opération de la rotation du fan. Dans la Figure 5.3, la fonction *Test* renvoie *vrai* au niveau du sommet u. Alors, u peut faire une rotation du fan.

(Partie 4) $(R3) \rightarrow (R6)$: ces règles sont utiles pour colorer une arête si la technique de la rotation du fan ne peut pas être appliquée. La fonction *Test* retourne *faux*. Dans ce cas, la rotation du fan F est bloquée, c'est à dire pour tout $\alpha \in Free(x_k)$, nous avons $\alpha \in \{C(u,x_1), C(u,x_2), ..., C(u,x_{k-1})\}$. De plus, α n'est pas une couleur manquante de u et $Free(u) \cap Free(x_k) = \emptyset$. Alors, il existe un sommet

Algorithme 6 Inversion du (c, d)-path.

$(R3)$: : if $Test(u, v) = false$ and $C(u, v) = Null$
then $Path(u, v)$
$u.StartPath = true$

$(R4)$: : if $\exists v \in N.u | v.StartPath = true$
and $\exists w \in N.u \,|\, C(u, w) = v.col_1$ and $w \neq v$
and $C(u, v) = v.col_1$
then $C(w).u = v.col_2$
$u.StartPath = true$
$u.col_1 = v.col_1$ and $u.col_2 = v.col_2$

$(R5)$: : if $\exists v \in N.u | v.StartPath = true$
and $\exists w \in N.u \,|\, C(u, w) = v.col_2$ and $w \neq v$
and $C(u, v) = v.col_2$
then $C(w).u = v.col_1$
$u.StartPath = true$
$u.col_1 = v.col_1$ and $u.col_2 = v.col_2$

$(R6)$: : if $u.StartPath = true$
and $\exists w \in N.u \,|\, C(u, w) = u.col_1$ or $u.col_2$
and $\forall x \in N.u \, Proper(x) = true$
then $u.StartPath = false$
$u.col_1 = u.col_2 = null$

Notation :
Fonction $Path(u, v)$ (voir Algorithme 9)

x_j voisin de u tel que $C(u, x_j) = \alpha$.

Pour débloquer la rotation d'un fan, soit P un (c, d)-path. c est une couleur manquante au niveau du sommet u telle que $c \neq d$ et c n'est pas une couleur manquante au niveau du sommet x_k. Il existe ainsi une arête incidente au sommet x_k telle que la couleur de cette arête est c. L'inversion du (c, d)-path P débloque la rotation du fan (voir Algorithme 6).

Les règles $(R4)$ et $(R5)$ sont privilégiées en comparaison avec les règles $(R0) \rightarrow (R2)$. En effet, en inversant le (c, d)-path, les sommets qui appartiennent à P inversent les couleurs c et d. Un nœud actif appartenant à P exécute ainsi $(R4)$ et $(R5)$ avant $(R0) \rightarrow (R2)$.

Dans la suite, nous expliquons le fonctionnement de chaque règle de l'Algorithme 6 qui consiste à inverser le (c, d)-path P. Nous définissons tout d'abord trois nouvelles variables au niveau de chaque sommet : $StartPath$, col_1 et col_2.

La variable $u.StartPath$ du sommet u (u appartient à P) est une variable booléenne. Elle est égale à *faux* quand l'inversion des couleurs du (c, d)-path P est réalisée avec succès (règle $(R6)$). Si $u.StartPath = true$, alors le sommet actif doit effectuer une inversion des couleurs de P, il met à jour ses variables $u.col_1$ et $u.col_2$, telles que la variable $u.col_1$ contient la première couleur manquante $c \in Free(u)$ du

Algorithme 7 $Test(u, v)$

$f = Free(v)$
$i = 1$
while $i < d(u)$
    if $\forall c \in f \, \exists e = (u, u') \,|\, C(e) = c$
    then $f = Free(u') \,|\, C(e) = Min\,\{c, c \in f\}$
      $i++$
    else return $true$
    endif
endwhile
return $false$

Algorithme 8 $Rotate(u, v)$

$f = Free(v)$
$x = v$
$i = 1$
while $i \leq d(u)$
    if $\forall c \in f \, \exists e = (u, u') \,|\, C(e) = c$
    then $C(x).u = Min\,\{c, c \in f\}$
      $f = Free(u') \,|\, C(e) = Min\,\{c, c \in f\}$
      $i++$
      $x = u'$
    else $C(x).u = Min\,\{c, c \in Free(x) \cap Free(u)\}$
      break
    endif
endwhile
return $false$

Algorithme 9 $Path(u, v)$

$f = Free(v)$
$x = v$
$bool = true$
$S = \{(u, v)\}$
$i = 1$
while $bool = true$ or $i \leq d(u)$
  if $\forall c \in f \; \exists e = (u, u^{'}) | C(e) = c$ and $(u, u^{'}) \notin S$
  then $C(x).u = Min\{c, c \in f\}$
    $f = Free(u^{'}) \,|\, C(e) = Min\{c, c \in f\}$
    $i++$
    $x = u^{'}$
    $S = S \cup e$
  else $C(x).u = Min\{c, c \in Free(u)\}$
    $bool = false$
  endif
endwhile
$k_1 = Min\{k, k \in Free(u)\}$
$k_2 = Min\left\{k, k \in Free(u^{'})\right\}$
$col_1 = k_1$
$col_2 = k_2$

(c, d)-path P et la variable $u.col_2$ contient la deuxième couleur manquante d de P $(d \neq c)$.

Les règles utiles pour effectuer une inversion du (c, d)-path P sont :

$(R3)$: La fonction *Test* renvoie *faux*, signifie que le sommet u ne peut pas exécuter la rotation du fan. Une operation d'inversion du (c, d)-path est nécessaire pour débloquer la rotation du fan. La règle $(R3)$ est utilisée pour engendrer l'inversion du (c, d)-path P. La variable $u.StartPath$ est ainsi mise à jour et égale à $vrai$ et la procédure *Path* (voir Algorithme 9) est exécutée et en conséquence la variable col_1 contient la couleur manquante $c = Min\{k, k \in Free(u)\}$ et col_2 contient la couleur manquante $d = Min\{k, k \in Free(v)\}$. L'exécution de la règle $(R3)$ va activer le prochain sommet $v \in N.u$ du (c, d)-path. Nous notons que dans cette règle la rotation du fan est exécutée avant l'inversion des couleurs du (c, d)-path. En effet, si nous commençons par l'inversion de P dans le but de débloquer F, un sommet actif w voisin à u peut changer la couleur de (u, w) ce qui peut modifier les couleurs manquantes de u et la fonction $Test$ du sommet u peut garder la valeur $faux$ même après l'inversion de P. La rotation de F reste toujours bloquée. Pour résoudre ce problème, nous appliquons tout d'abord la rotation du fan et par la suite l'inversion du (c, d)-path. En effet, d'après le Lemme 5.5.1, s'il y a inversion du (c, d)-path P, alors il y a nécessairement rotation du fan F.

Lemme 5.5.1. *Soit F un fan et $c \in Free(u)$, $d \in Free(x_k)$. Supposons qu'il existe un éventuel (c,d)-path P qui commence de u. Alors, après inversion de P, il existe une rotation du fan F.*

Démonstration. Soit (u,v) une arête non-colorée et F un fan bloqué. Pour débloquer la rotation du fan, la solution consiste à inverser le (c,d)- path P tel que $c \in Free(u)$, $d \in Free(x_k)$. Par conséquent, après inversion, c devient une couleur manquante aux deux extrémités de (u,x_k). Nous changeons ainsi la coloration des arêtes du fan par : colorer (u,x_i) avec $C(u,x_{i+1})$ pour $i = 0,...,k-1$ et colorer (u,x_k) avec c. Il est facile de vérifier que la coloration de (u,v) est ainsi une coloration propre.

□

La rotation et l'inversion dans ce cas sont bien liées, nous pouvons ainsi exécuter la rotation du fan avant l'inversion de P. Par conséquent, si *Test* retourne *faux*, le sommet u change les couleurs du fan bloqué par :

- colorer (u,x_i) avec $C(u,x_{i+1})$ pour $i = 0,...,k-1$
- colorer (u,x_k) avec c, telle que c est une couleur manquante au niveau du sommet u mais n'est pas manquante au niveau du sommet x_k. Le sommet x_k devient actif et il supprime le conflit, en exécutant l'inversion du (c,d)-path.

Nous détaillons maintenant les autres règles de l'inversion de P. Pour les règles $(R4)$ et $(R5)$, nous distinguons trois différents états de sommets. Ces états dépendent des conditions d'inversion des couleurs. Certaines conditions sont communes aux sommets, c'est à dire qu'un sommet u exécute l'inversion de P s'il existe un voisin $v \in N.u$ tel que sa variable $v.StartPath$ est égale à *vrai.* Les autres conditions sont différentes, chacune est spécifique à une règle. Dans ce cas , les différents états de sommets sont :

- Le sommet u est incident à deux arêtes (u,v) et (u,w) qui partagent la même couleur $v.col_1$. Le sommet u exécute ainsi la règle $(R4)$.
- Le sommet u est incident à deux arêtes (u,v) et (u,w) qui partagent la même couleur $v.col_2$. Le sommet u exécute ainsi la règle $(R5)$.

Pour chacune des règles $(R4)$ et $(R5)$, le sommet u exécute les actions suivantes :

- le sommet u change la couleur de l'arête (u,w),
- il met à jour ses variables, $u.col_1 = v.col_1$ et $u.col_2 = v.col_2$, et
- $u.StartPath = vrai$, cette condition est utile pour propager l'inversion du (c,d)-path P.

La dernière règle $(R6)$ est nécessaire pour mettre à jour les différentes variables $StartPath$, col_1 et col_2 : $u.StartPath = faux$ et $u.col_1 = u.col_2 = null$. En effet, un sommet est actif lorsque tous ses voisins qui appartiennent au (c,d)-path P ont déjà exécuté l'inversion de P, i.e la fonction *Proper* est égale à *vrai.* Par conséquent, il peut mettre à jour ses variables : $u.StartPath = faux$ et $u.col_1 = u.col_2 = null$.

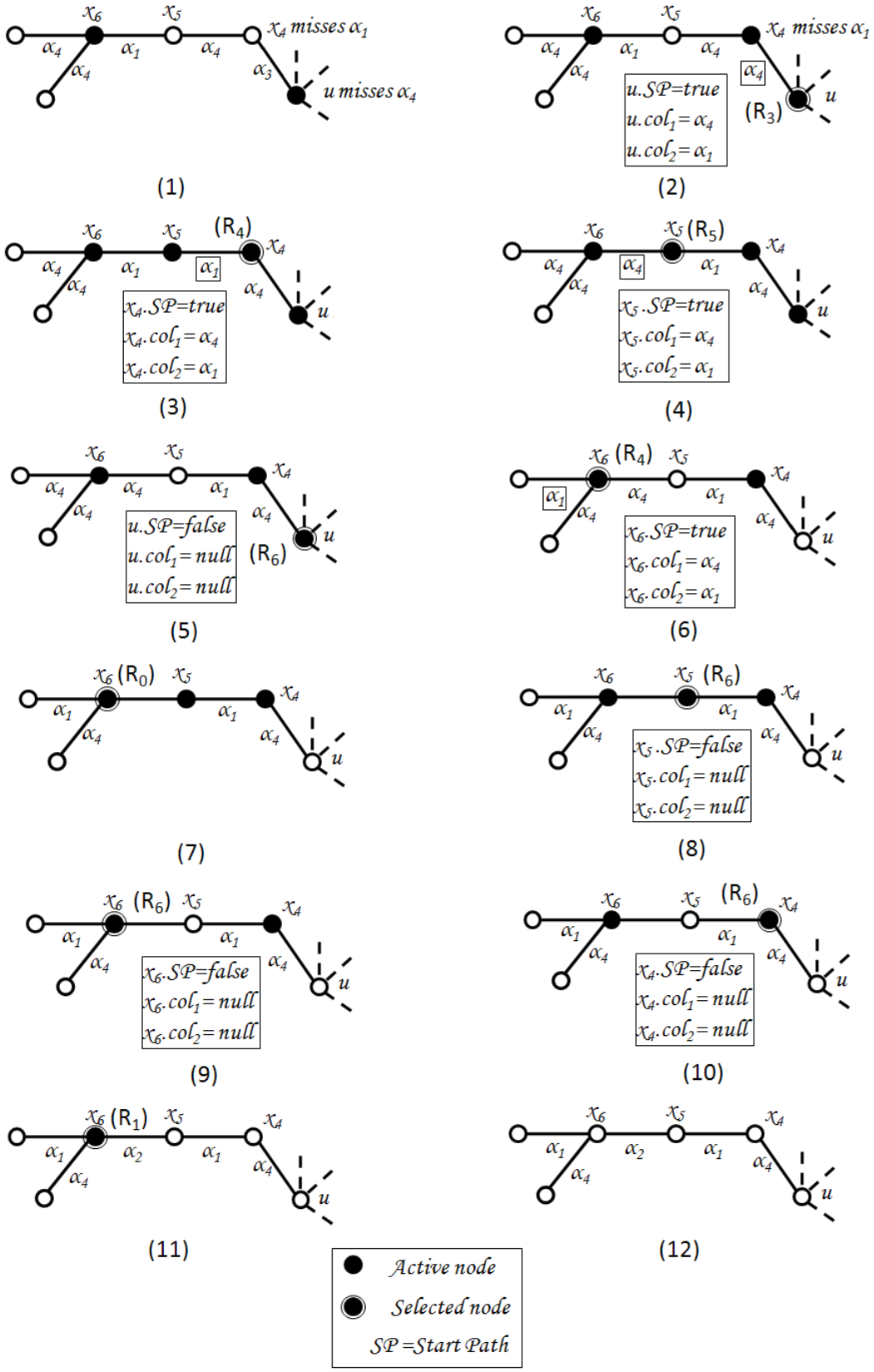

Figure 5.4 : *Exemple détaillé.*

5.5.3 Exemple d'exécution de l'algorithme

Dans la Figure 5.4, un exemple détaillé de la rotation du fan et l'inversion du (c, d)-path est présenté. Chacune des figures illustre l'état des sommets dans des mouvements consécutifs. Les sommets stables sont marqués par des cercles vides et les sommets actifs sont marqués par des cercles pleins. Un nœud qui est susceptible de faire le prochain mouvement est encerclé et est suivi du nom de la règle qui le rend actif.

Dans chacune des Figures 5.4 $(i)_{1\leq i\leq 12}$, le cadre contient l'action du nœud sélectionné. Nous expliquons le mouvement de chaque sommet dans la suite :

Dans la Figure 5.4 (2), le sommet u est actif car sa fonction $Test$ retourne $faux$ et il ne peut pas exécuter $(R1)$ ni $(R2)$. Dans ce cas, la rotation du fan est bloquée. Le sommet u exécute ainsi la règle $(R3)$, sa variable $StartPath$ est ainsi égale à $vrai$ pour déclencher l'inversion du (α_4, α_1)-path, et exécute la rotation du fan. L'arête (u, x_4) devient non-colorée et par conséquent $C(x_4).u = \alpha_4$. Le sommet x_4 devient actif.

Dans la Figures 5.4 (3), la Figure 5.4 (4) et la Figure 5.4 (6), le sommet sélectionné inverse les couleurs α_4, α_1. Il exécute respectivement $(R4))$ et $(R5)$. Il ne peut pas exécuter la règle $(R0)$ car les règles $(R4)$ et $(R6)$ sont privilégiées en premier temps aux règles $(R0) \rightarrow (R2)$.

Le sommet u, de la Figure 5.4 (5), met à jour ses variables $StartPath$, col_1 et col_2.

Dans la Figure 5.4 (6), le sommet x_5 n'est pas encore actif pour mettre à jour ses variables. En effet, la variable $Proper(x_6)$ de son voisin x_6 est égale à $faux$ $(Proper(x_6) = faux)$. x_6 est le sommet sélectionné dans la Figure 5.4 (7), il exécute la règle $(R0)$ et supprime le conflit. Par conséquent, x_5 devient actif, il peut ainsi mettre à jour ses variables (voir Figure 5.4 (8)).

Dans la Figure 5.4 (9), le sommet sélectionné x_6 met à jour ses variables, car son voisin x_5 qui appartient au (α_4, α_1)-path inverse avec succès ses couleurs. Le même raisonnement s'applique à la Figure 5.4 (10) pour le sommet sélectionné.

Pour colorer l'arête (x_5, x_6), le sommet x_6 -ayant l'identité la plus grande- applique la règle (R_1) de la Figure 5.4 (11).

Notons que, pour tous ces mouvements, si un sommet décide de la couleur d'une arête, l'autre extrémité de l'arête copie cette couleur. La Figure 5.4 (12) est l'état légitime obtenu.

5.5.4 Convergence et exactitude de l'algorithme

Pour montrer que notre algorithme converge vers un état légitime, nous introduisons les deux notions suivantes : mouvement *ascendant* et mouvement *descendant*.

Une exécution d'une règle est dite mouvement ascendant si et seulement si elle augmente le nombre d'arêtes colorées. Une exécution d'une règle est dite mouvement descendant si et seulement si elle diminue le nombre d'arêtes colorées. Ci-dessous, nous montrons que les mouvements descendants d'un sommet ne peuvent pas dépasser ses mouvements ascendants.

Lemme 5.5.2. *Après avoir exécuté l'inversion du (c, d)-path, un sommet u ne peut engendrer ni un mouvement ascendant ni un mouvement descendant et après l'exécution de la rotation du fan, il engendre un mouvement ascendant pour colorer $C(v).u$.*

Démonstration. Il est évident que pour inverser le (c, d)-path, les règles $(R4)$ et $(R5)$ sont utiles pour uniquement inverser les couleurs des arêtes déjà colorées. D'où, le nombre de couleurs ne va pas changer. Pour une rotation de fan bloquée (respectivement non-bloquée), la règle $(R2)$ (respectivement règle $(R3)$) est utilisée pour faire une rotation des couleurs du fan, dans le but de colorer l'arête non-colorée. Dans ce cas, nous avons un mouvement ascendant. □

Lemme 5.5.3. *Pour une arête (u, v), le mouvement du sommet u dans les règles $(R0)$,$(R1)$ et $(R2)$ ne peut jamais engendrer une coloration non-propre d'arêtes.*

Démonstration. Soit (u, v) une arête. Pour prouver ce lemme, nous montrons que lorsqu'un sommet exécute une règle, une arête ne peut jamais avoir la couleur utilisée par une autre arête adjacente. La règle (R_0) est nécessaire pour supprimer les conflits de couleurs du graphe initial. La règle $(R1)$ propose une couleur appropriée s'il n'y a pas eu encore de couleur proposée et la règle $(R2)$ exécute la rotation du fan (voir Lemme 5.5.2). Par conséquent, quand un sommet exécute $(R0)$, $(R1)$ ou $(R2)$, il ne peut jamais produire une coloration non-propre. □

Lemme 5.5.4. *Après inversion du (c, d)-path, le graphe engendre une coloration propre d'arêtes.*

Démonstration. D'après [Misra 92], si la coloration initiale du graphe est propre alors la coloration après l'inversion reste toujours propre. Dans notre algorithme, quand nous inversons le (c, d)-path P, le nombre de couleurs ne change pas, nous faisons seulement une inversion de couleurs locales sans changer l'état réel des sommets. Les règles $(R4)$ et $(R5)$ sont nécessaires pour l'inversion de P. Ces règles sont exécutées lorsque le sommet u déclenche l'inversion du (c, d)-path, i.e. lorsque u exécute $(R3)$. Les règles $(R3) \rightarrow (R5)$ peuvent être considérées comme une boîte noire. En effet, le sommet sélectionné pour faire un mouvement change sa variable $StartPath$ à $vrai$, i.e. il est ainsi verrouillé et il ne peut faire que l'inversion du (c, d)-path. De plus, les sommets qui appartiennent à P privilégient l'exécution de

l'inversion du (c, d)-path aux autres règles. Par exemple, lorsqu'un sommet exécute la règle $(R3)$ pour déclencher l'inversion de P, il génère une coloration non-propre. Cependant, ce conflit sera résolu par un autre sommet actif qui appartient à P. Par conséquent, nous aurons une coloration propre après l'exécution de l'inversion du (c, d)-path. □

Lemme 5.5.5. *Une arête effectue uniquement un seul mouvement descendant.*

Démonstration. Soit (u, v) une arête, le sommet u exécute $(R0)$ pour supprimer les conflits du graphe initial. D'après les Lemmes 5.5.3 et 5.5.4, les mouvements de u engendrent une coloration propre. Le sommet u exécute $(R0)$ une seule fois. D'où, (u, v) effectue uniquement un seul mouvement descendant. □

Lemme 5.5.6. *Soit (u, v) une arête, après exécution de $(R1)$, le sommet u effectue uniquement un seul mouvement ascendant.*

Démonstration. Pour une arête non-colorée (u, v), u exécute une seule fois $(R1)$ pour colorer (u, v), car (u, v) effectue uniquement un mouvement descendant (voir Lemme 5.5.5) et les mouvements de u engendrent une coloration propre. □

Lemme 5.5.7. *G effectue $m - 1$ mouvements descendants.*

Démonstration. Ce lemme est déduit directement du Lemme 5.5.5. □

Théorème 5.5.8. *Le système entre dans un état légitime.*

Démonstration. Ce théorème est une conséquence directe des Lemmes 5.5.6 et 5.5.7. Si G effectue $m - 1$ mouvements descendants, les sommets actifs exécutent $(R1)$, $(R2)$ ou $(R3)$ pour colorer l'arête non-colorée. Lorsqu'un sommet u exécute $(R1)$ ou $(R2)$, le mouvement de u engendre une coloration propre. Lorsque $(R3)$ et l'inversion du (c, d)-path sont exécutées, le graphe résultant est un graphe proprement coloré. De plus, pour chaque mouvement descendant un sommet effectue un mouvement ascendant et chaque mouvement ascendant engendre une coloration propre d'arêtes. Par conséquent, le système converge vers un état légitime en utilisant $(\Delta + 1)$ couleurs. □

Théorème 5.5.9. *A partir d'un état initial arbitraire, une arête satisfait la coloration propre d'arêtes en $O(\Delta + n)$.*

Démonstration. Soit (u, v) une arête, le sommet u exécute $(R0)$ puis la rotation du fan et l'inversion du (c, d)-path. Dans le pire des cas, u exécute $(R0)$ en $O(\Delta)$, il exécute la rotation du fan en $O(\Delta)$ et exécute l'inversion du (c, d)-path en $O(n)$. Par conséquent, le nombre total de mouvements, pour stabiliser une arête, est $O(n+\Delta)$. □

5.6 Conclusion

L'auto-stabilisation est un paradigme général de programmation qui permet à un système de retrouver un comportement correct à partir de n'importe quel état. L'auto-stabilisation rend possible la résistance des systèmes à certaines types de défaillances. Les protocoles auto-stabilisants tolèrent des pannes qui n'avaient pas été prévues lors de la conception du système et permettent ainsi de construire des systèmes dont la résistance aux fautes est pérenne. Nous avons proposé un algorithme auto-stabilisant pour la coloration propre d'arêtes en utilisant au maximum $\Delta + 1$ couleurs et un temps de convergence en $O(m(\Delta + n))$. Notre solution constitue une première étape. Nous envisageons en perspectives d'étendre notre algorithme pour la coloration d'arêtes ℓ-distance en se basant sur le théorème de Vizing.

Chapitre 6

Algorithme auto-stabilisant de clustering dans les réseaux mobiles ad hoc

Résumé

L'auto-stabilisation prend naturellement une importance fondamentale dans les réseaux mobiles ad hoc (MANETs). L'essentiel de ce chapitre sera axé sur la conception d'un algorithme auto-stabilisant de clustering dans les MANETs. La topologie que nous proposons est un partitionnement basé sur les relations de confiance entre les membres d'un groupe. Elle forme une structure capable de s'adapter dynamiquement aux changements de la topologie. La première section de ce chapitre définit la notion des réseaux mobiles ad hoc et leurs contextes d'utilisation. Une synthèse des solutions d'organisation des MANETs fait l'objet des sections suivantes. Ensuite, nous présentons notre approche de clustering basée sur la confiance et destinée à des applications dans le domaine de la sécurité. Nous décrivons par la suite un algorithme de gestion de clés de groupe dans les réseaux mobiles ad hoc qui s'appuie sur la topologie de clusters préalablement construite afin d'assurer une meilleure confidentialité du réseau.

6.1 Introduction aux réseaux mobiles ad hoc

6.1.1 Histoire

Dès le début des années 80 et avec l'apparition des ordinateurs personnels, les réseaux filaires ont pris une grande dimension dans le monde informatique. Tout le monde cherche à s'échanger des données sans se déplacer, à faire coopérer des équipes de chercheurs ou de programmeurs sans que la distance ne soit une contrainte. En 1985, le gouvernement des États Unis a introduit une de ses technologies militaires présentant une nouvelle notion, celle des réseaux mobiles ad hoc, pour la mettre à la disposition du public, et il a accordé trois plages de fréquences partagées pour les usages de l'Industrie, de la Science et de la Médecine. Ce concept a été très vite adopté par les géants de l'industrie informatique (IBM, Apple, Nokia, etc.). Au début des années 90, le concept débarque en Europe mais fut moins apprécié des militaires européens qui se voyaient contraints d'abandonner des bandes de fréquences qui leur étaient réservées. En 1998, les premiers standards supportant le mode ad hoc apparaissent et permettent ainsi aux réseaux mobiles ad hoc de se déployer en toute légalité.

6.1.2 Définition

Un Réseau mobile ad hoc ou *MANET* (*Mobile Ad hoc Network*) est une collection d'entités mobiles interconnectées par une technologie sans fil formant un réseau temporaire indépendamment de toute infrastructure ou d'administration centralisée. A un instant donné, la topologie du réseau (la position des nœuds, la configuration des antennes d'émission/réception, la puissance du signal et les interférences entre les canaux) change avec le temps en fonction des mouvements des nœuds ou tout autre changement du paramètre de configuration.

6.1.3 Applications possibles

Les applications des réseaux mobiles ad hoc sont nombreuses. Nous pouvons citer :

- La communication tactique : Tout ce qui concerne les applications militaires, partage des informations de positions sur les cibles ou sur l'ennemi, synchronisation, coordination et guidage des troupes sans passer par des stations fixes, ce qui à petite échelle, minimise le risque d'interception des messages.
- Opérations de secours : en cas de catastrophes naturelles (Incendies, inondations, etc.), les réseaux mobiles ad hoc peuvent résoudre le problème de communication là où une installation filaire ne peut être réalisée qu'après de très longs délais d'attente.

- Les conférences : Les réseaux mobiles ad hoc facilitent l'échange et le partage d'informations entre les participants d'une conférence.
- Les sites de patrimoine : Dans les sites archéologiques, musées (anciens), châteaux ou tout autre lieu semblable, où dans la plupart des lieux où il est interdit d'installer des réseaux filaires vu le risque de nuisance sur de tels environnements. Les réseaux mobiles ad hoc peuvent très bien résoudre le problème.
- En privé : Libre échange de données, connexions en peer-to-peer, etc.

6.1.4 Contraintes et Défis

Nous distinguons les contraintes fortes suivantes :

- L'absence d'infrastructure : Un réseau mobile ad hoc est caractérisé par l'absence d'infrastructure préexistante ou toute autre administration centralisée. Cette caractéristique est fondamentale car elle définit la manière dont les unités mobiles communiquent entre elles, c'est à dire sans passer par aucune antenne relais ou tout autre dispositif fixe.
- Topologie dynamique : Les mouvements des unités mobiles étant aléatoires, des changements de connectivité surviennent d'une manière imprévisible. En effet, un lien de communication existant entre deux stations peut se briser à n'importe quel moment. Le mouvement des stations peut aussi conduire à la création de nouveaux liens et ainsi avoir la possibilité de remplacer les routes brisées par de nouvelles.
- Contraintes sur la bande passante : Les réseaux sans fil se basent sur le partage des médiums de communication entre les entités. Ainsi, la bande passante réservée à une entité sera relativement modeste. Ceci implique des liens sans fil à capacité variable.
- Contraintes d'énergie : Les unités mobiles sont, d'une part, alimentées par des sources d'énergie autonomes (exemple : batteries), ce qui réduit le temps de disponibilité des unités mobiles. Mais, d'autre part, l'utilisation de nouvelles technologies (exemple : processeurs à fréquence variable) permet l'optimisation de la consommation d'énergie.
- Passage à l'échelle : le réseau peut comprendre un grand nombre de terminaux. Cependant, les performances ne doivent pas pour autant chuter lorsque le nombre de participants augmente.
- Sécurité limitée : Vues les contraintes précédentes, les méthodes de sécurité (chiffrement, etc.) sont réduites ce qui augmente le risque d'attaques ou de piratage. En effet, les réseaux mobiles ad hoc sont considérés comme étant très fragiles en matière d'attaques en tout genre. Les pirates informatiques peuvent intercepter les données d'une manière directe en utilisant des antennes pirates

(car les données circulent par voie hertzienne) ou bien obliger une station à consommer ses ressources d'énergie en l'inondant de toutes sortes de requêtes inutiles.

En raison des caractéristiques énumérées ci-dessus, les réseaux mobiles ad hoc souffrent de plus en plus de comportements malveillants.

6.1.5 Graphes et MANET

Un réseau mobile ad hoc peut être représenté par un graphe orienté ou non orienté, selon la nature des liens, $G(t) = (V(t), E(t))$ où $V(t)$ désigne l'ensemble de nœuds (unités mobiles) à l'instant t et $E(t)$ désigne l'ensemble des liens qui existent entre les nœuds à l'instant t. La mobilité des nœuds induit des changements fréquents de la topologie, ce qui entraîne des connexions et des déconnexions répétitives.

6.2 Organisation des MANETs

La flexibilité des réseaux mobiles ad hoc associée avec la vulnérabilité des liens sans fils rend difficile la mise en place de protocoles de communication. La solution la plus appropriée doit passer par une meilleure organisation des nœuds. En effet, la topologie du réseau change fréquemment. Il serait intéressant dans ce cas de pouvoir maintenir une topologie adéquate en maîtrisant les liens à inclure dans le réseau. Ceci permet de créer une vue différente de la topologie radio, introduisant un ou plusieurs niveaux de hiérarchie pour faciliter la configuration des nœuds et la mise en place de protocoles de communication. Nous parlons dans ce cas du contrôle de la topologie des MANETs [Wattenhofer 01].

Le contrôle de la topologie est fondamental pour résoudre plusieurs problèmes des réseaux mobiles ad hoc, nous pouvons citer : assurer une meilleure stabilité du réseau, masquer une partie de la dynamique des nœuds, réduire la consommation d'énergie, optimiser la diffusion d'information, augmenter la capacité du réseau.

Notre étude se focalise sur les techniques d'organisation des MANETs, nous nous intéressons plus particulièrement aux techniques de clustering. Le partitionnement du réseau en zones appelées clusters permet de l'organiser pour des problématiques de routage [Jiang 99] [Drira 06], de découverte de services [Haddad 07], de prédiction de la mobilité [Dekar 08], etc. De plus, une organisation en clusters peut former une topologie virtuelle utile pour l'agrégation d'informations et pour cacher les changements de la topologie physique du réseau. Elle est aussi utilisée pour la qualité de services [Cardei 04], la couche MAC [Hou 01], la gestion de clés [Tseng 07a], etc.

Les clusters sont identifiés par un chef (appelé aussi cluster-head). Si le cluster-head disparaît, le cluster n'est plus valide. Les cluster-heads et les nœuds qui connectent deux clusters voisins (appelés aussi passerelles) sont des nœuds spéciaux qui ont quelques responsabilités supplémentaires par rapport aux nœuds ordinaires dans un réseau mobile ad hoc. Un cluster-head sauvegarde l'information requise pour des communication efficaces entre clusters. Les passerelles sont les nœuds à la frontière d'un cluster et communiquent avec les passerelles des clusters voisins.

Un algorithme de clustering est basé sur les étapes suivantes :

- Formation (élection) des cluster-heads : le réseau est ainsi divisé en plusieurs clusters. Cette phase utilise des heuristiques comme le plus grand/plus petit *ID* dans le voisinage, le degré de connectivité, la zone géographique, la puissance de transmission ou la vitesse de déplacement, ou bien en utilisant un poids pour chaque nœud qui représente une combinaison des derniers attributs.
- Communication entre les cluster-heads : cette phase est assurée par les nœuds passerelles.
- Maintenance des clusters : pour s'adapter aux changements fréquents de la topologie dans le réseau, une mise à jour des clusters est dynamiquement réalisée.

Dans la littérature, il existe de nombreuses propositions pour construire des clusters dans les réseaux mobiles ad hoc [Chen 04] [Erciyes 07]. Les premiers algorithmes de clustering *Lowest-ID algorithm (LID)* de Baker et Ephremides [Baker 81] et *High-Connectivity Clustering* (*HCC*) de Gerla [Gerla 95] ont des mécanismes assez proches. Dans *LID*, les auteurs se sont basés sur un critère particulier pour le choix des cluster-heads, qui est l'identifiant d'un nœud. Cet algorithme permet de former des clusters à un seul saut, où chaque membre est voisin direct de son cluster-head. Dans la phase de construction des clusters, les nœuds communiquent avec leurs voisins pour avoir une connaissance locale et fixer ainsi le cluster-head. Cette phase se répète périodiquement pour tout changement de topologie. L'algorithme cité dans [Lin 97] est une version modifiée de l'algorithme *Lowest-ID algorithm*. Les auteurs proposent un algorithme de clustering pour réduire le trafic de contrôle du clustering. Un nœud ne diffuse qu'un seul message contenant sa décision de clustering. Selon sa connaissance locale de la topologie, chaque nœud prend la décision de devenir un cluster-head ou non. Cette décision est diffusée au voisinage, forçant les voisins du nouveau cluster-head qui ne sont pas encore affiliés à un cluster de le choisir comme un cluster-head.

Les algorithmes qui se basent sur l'identifiant des nœuds permettent de stabiliser les cluster-heads. Cependant, un tel choix peut être loin de l'optimum si un nœud à faible identifiant est très mobile et si sa réserve d'énergie est faible. De plus,

l'élection n'est pas égalitaire, les nœuds à identifiant faible consomment en moyenne plus d'énergie que les autres, puisqu'ils restent plus longtemps des cluster-heads.

Dans l'algorithme *High-Connectivity Clustering* (*HCC*) [Gerla 95], l'élection du cluster-head est basée sur le degré (nombre de voisins du nœud) au lieu des identités des nœuds. Un nœud est élu comme un cluster-head s'il a la plus haute connectivité parmi tous ses voisins. Cet algorithme souffre de fréquents changements de cluster-heads. Dans [Chiang 96], les auteurs ajoutent une étape de la maintenance des clusters : les clusters ne sont reconstruits que si deux cluster-heads se retrouvent voisins ou si un nœud ordinaire n'a plus aucun cluster-head dans son voisinage. De cette façon, l'algorithme améliore la stabilité de la structure. Cependant, l'algorithme peut engendrer une maintenance totale de la structure. De plus, un nœud peut appartenir à plusieurs clusters à la fois.

[Basu 01] propose un algorithme de clustering basé sur le degré de mobilité. En effet, un nœud avec une faible mobilité gardera un voisinage plus stable au cours du temps, condition idéale pour être un cluster-head pour favoriser la stabilité des clusters. Un nœud u mesure le niveau de signal avec chacun de ses voisins. La mobilité d'un nœud est calculée à partir des rapports entre ce niveau de signal et celui mesuré à l'étape précédente pour chaque voisin du nœud u. L'atténuation du signal dépend de la distance qui sépare les nœuds. Le nœud ayant la mobilité la plus faible devient un cluster-head. Pour la maintenance de la topologie, les auteurs se basent sur l'algorithme de Chiang et al. [Chiang 96] décrit ci-dessus et ils ajoutent une autre condition : si deux cluster-heads u et v deviennent voisins, le cluster-head v ayant le plus grand identifiant n'abandonne son rôle que si le cluster-head u reste son voisin pour une certaine période de temps. Ceci évite de reconstruire la structure si deux cluster-heads se retrouvent voisins pour une période courte. L'algorithme de Basu et al. permet de former des clusters à un seul saut. Par contre, ce clustering garde les mêmes inconvénients de l'algorithme de la maintenance de la topologie proposé dans [Chiang 96]. De plus, cette méthode est complexe, les nœuds doivent estimer les puissances du signal, ils ne considèrent pas d'autres phénomènes physiques qui peuvent atténuer le signal.

Bouassida et al. proposent l'algorithme *Efficient Clustering for Multicast Key Distribution in MANETs* [Bouassida 05], une architecture de sécurisation des communications de diffusion, dédiée aux réseaux mobiles ad hoc. Elle se base sur l'algorithme de clustering dynamique OMCT (*Optimized Multicast Clustering Tree*) qui tient compte de la mobilité. Le critère de décision de la formation des clusters est le paramètre de mesure de la proximité des membres du cluster. Plus le nombre de membres locaux atteignables en un seul saut est grand, plus le facteur de cohésion tend vers 1. *Min-cohesion* est le seuil minimal en dessous duquel la cohésion est perdue. Un système GPS (*Global Positioning System*) est présent au sein du réseau

mobile ad hoc. Cependant, le coût d'utilisation du système *GPS* dans les réseaux mobiles ad hoc reste extrêmement élevé.

D'autres solutions prennent en compte plusieurs critères de clustering à la fois : la mobilité, la connectivité et l'énergie. L'algorithme décrit dans [Chatterjee 00] utilise la somme pondérée entre le degré, la distance entre les nœuds, la vitesse relative et une constante contenant le nombre de nœuds qu'un cluster-head peut servir. L'élection se fait en utilisant le même principe que l'algorithme de [Baker 81], le nœud dont la somme pondérée est la plus petite devient un cluster-head. La distance est calculée en utilisant un *GPS*. L'algorithme de clustering est exécuté lorsqu'un nœud se trouve dans une zone sans cluster-head. Dans cet algorithme, un nœud peut aussi appartenir à plusieurs clusters à la fois, ce qui entraîne un recouvrement de clusters.

Dans [Tseng 07a], les auteurs proposent *SGCP* (*Secure Group communication Protocol*). *SGCP* est un protocole de communication sécurisé qui se base sur le protocole *Virtual Dynamic Backbone Protocol* (*VDBP*) introduit dans [Jaikaeo 02] [Kozat 01] pour construire des clusters. Afin de réaliser le clustering, tous les nœuds dans *VDBP* créent une table de voisinage. Cette table contient le degré, l'identité, l'autonomie restante et la capacité de calcul. La table est appelée le poids du nœud. Chaque nœud met à jour ses informations et les diffuse d'une manière périodique à ses voisins dans le message *hello* (*hello* est un message périodique échangé entre un nœud et ses nœuds voisins). Un nœud ayant le meilleur poids parmi les poids reçus est optimal. S'il est optimal, il change son état à un *multicast router* (*MR*). Les *MRs* doivent diffuser leur poids et leur identité dans le réseau. Un nœud qui n'est pas *MR* choisit le meilleur poids et envoie un message au *MR* correspondant pour appartenir au même sous-groupe. Chaque *MR* constitue un sous-groupe avec tous les nœuds qui ne sont pas des *MRs* et qui lui ont envoyé un message. Après la création du sous-groupe, l'étape suivante de *SGCP* consiste à établir une connexion entre les sous-groupes. Dans cette étape, tous les nœuds doivent diffuser l'identité du sous-groupe à leur voisinage. Chaque nœud qui a au moins un voisin dans un autre sous-groupe est appelé connecteur. Chaque connecteur recueille les informations de tous les connecteurs adjacents dans une table et l'envoie à son *MR*. Ce processus de construction de connecteurs est lancé périodiquement pour assurer un chemin de connexion correct.

L'algorithme de clustering présenté dans [Basagni 99] [Siddiqui 02] introduit la notion de poids générique pour la sélection de cluster-heads utilisant le principe de l'algorithme de [Baker 81]. C'est un mécanisme qui réagit face aux changements de la topologie, mais les clusters formés sont aussi à un saut.

D'autres méthodes de clustering ont été présentées pour augmenter la taille du cluster ou le nombre de sauts entre deux cluster-heads. Dans [Yu 03], le nœud

qui a le plus grand degré devient un cluster-head, ses voisins rejoignent ce cluster et deviennent les "membres du cluster". Les voisins des membres du cluster ne peuvent plus être des cluster-heads. Par la suite, la procédure de clustering peut être exécutée en parallèle. Pour la maintenance de cet algorithme de clustering, les cluster-heads doivent être toujours au moins à distance trois. Lorsque deux cluster-heads deviennent voisins, celui qui a le plus grand identifiant n'est plus cluster-head. Lorsqu'un nœud se déplace et il n'est dans le voisinage d'aucun cluster-head, il peut rejoindre un cluster comme un membre invité s'il est au moins voisin à membre du cluster. Par conséquent, il n'est pas nécessaire de former de nouveaux clusters afin de couvrir un tel nœud comme dans l'algorithme de [Baker 81] et la topologie du cluster ne change pas. Cela peut réduire le nombre de clusters et élimine les petits clusters.

Dans [Drira 10], l'algorithme construit des clusters dont le diamètre est au plus égal à quatre. Dans les clusters, il y a trois niveaux de hiérarchie : le cluster-head, les cluster-heads secondaires et les nœuds ordinaires. L'idée consiste à construire un ensemble de nœuds dominants (nœuds de diffusion) dans le réseau. Le caractère dominant caractérise les nœuds dont le nombre de chemins, pour atteindre les nœuds à deux sauts, est supérieur à celui de ses voisins. Un nœud ne peut choisir qu'un seul nœud de diffusion. La moyenne de choix d'un nœud u désigne le rapport entre le nombre de nœuds qui ont choisi u comme un nœud de diffusion et le degré de u. Un nœud devient un cluster-head s'il est un nœud dominant et possède une moyenne de choix supérieure à celle de ses voisins. Les cluster-heads secondaires sont des nœuds dominants voisins au cluster-head et les nœuds ordinaires sont des nœuds non-dominants.

L'algorithme de [Amis 00] construit des clusters à k sauts. Cet algorithme est le Max-Min k-clustering. Cet algorithme permet de construire des clusters disjoints de rayon au plus k. Chaque cluster a un cluster-head tel que chaque nœud dans le cluster ne peut être à plus de k sauts de son cluster-head. L'algorithme Max-Min k-clustering s'execute en $2k$ étapes d'échange d'informations. A la fin des $2k$ étapes, le nœud ayant le plus grand ID est élu comme un cluster-head dans son k-voisinage. En outre, chaque nœud choisit un cluster-head et ainsi un cluster auquel il appartiendra. Une fois qu'un nœud a choisi son cluster, il détermine s'il est un nœud passerelle ou pas, et informe son cluster-head qu'il est membre du cluster. Un nœud passerelle est un nœud qui est adjacent à un autre cluster que le sien, et qui permet d'accéder à celui-ci. Pour déterminer les nœuds passerelles, chaque nœud envoie l'ID de son cluster-head à tous ses voisins. Après avoir écouté tous ses voisins, un nœud détermine s'il est nœud passerelle et les clusters auxquels il est adjacent. Si tous les voisins d'un nœud ont le même cluster-head que lui, alors ce nœud n'est pas un nœud passerelle. Sinon, s'il y a des voisins qui ont un cluster-head

différent, alors ce nœud est bien un nœud passerelle. Lorsqu'un nœud a déterminé s'il est un nœud passerelle ou pas, il envoie à son cluster-head son ID et l'ID de ses nœuds voisins. A la fin, le cluster-head connaît tous les nœuds de son cluster et la topologie physique de celui-ci. La structure résultante s'avère robuste, cependant la latence induite par l'algorithme est non négligeable.

Dans [Chen 02a], l'algorithme combine *LID* et *HCC*. Afin de sélectionner les cluster-heads, la connectivité est considérée comme un premier critère et l'identifiant comme un second critère. En effet, utiliser l'identifiant peut générer plus de clusters que nécessaire. Le but étant de minimiser le nombre de clusters formés dans le réseau pour obtenir des ensembles dominants de plus petites tailles. Les clusters dans [Chen 02a] sont formés par un cluster-head et tous les nœuds qui sont à distance au plus k-sauts du cluster-head. Au début de l'algorithme, un nœud inonde le réseau par une demande de clustering. Dans *HCC*, la connectivité mesure le degré du nœud pour des clusters à 1-saut. [Chen 02a] généralise la connectivité pour un voisinage à k-sauts. Ainsi, lorsque $k = 1$ la connectivité est la même que le degré du nœud. Chaque nœud du réseau possède une paire $did = (d, ID)$. d est la connectivité d'un nœud et l'ID est l'identifiant du nœud. Un nœud devient un cluster-head s'il a la plus grande connectivité. En cas d'égalité de connectivité, un nœud est prioritaire s'il a la plus faible ID.

D'autres solutions cherchent dans un premier temps à déterminer un ensemble dominant connecté sur lequel les clusters vont être par la suite construits. Par définition, un graphe est dit connexe si tous les nœuds sont joignables, c'est-à-dire qu'il existe toujours un chemin constitué d'arcs reliant deux nœuds du graphe. Cette propriété est très importante dans le cas des ensembles dominants. En effet, elle garantit que chaque nœud de l'ensemble dominant peut joindre n'importe quel autre nœud de ce même ensemble. L'algorithme de diffusion est alors très simple : seuls les nœuds de l'ensemble dominant ré-émettent le message. Chaque nœud de l'ensemble dominant couvre alors l'ensemble de ses voisins, garantissant ainsi une couverture complète du réseau. Les ensembles dominants peuvent être utilisés dans les domaines du routage pour d'une part déterminer les nœuds appartenant à l'espace de la diffusion et de recherche des routes et pour d'autre part déterminer les nœuds qui vont retransmettre les messages. Quand un nœud diffuse un message aux nœuds appartenant au CDS, tout le réseau est couvert. L'intérêt ici est de minimiser la taille du *CDS* de manière à limiter le nombre de nœuds qui retransmettent le message. De même pour le routage, diminuer la taille du *CDS* réduit la complexité de la recherche de routes.

Certaines méthodes de construction d'un ensemble de nœuds dominants connectés se basent sur les *MIS* (*maximal independent set*). Un *MIS* est l'ensemble formé par le maximum de nœuds non-connectés dans un graphe G. Ensuite, les som-

mets d'un *MIS* sont connectés entre eux pour former un *CDS*. Les auteurs dans [Alzoubi 02] [Li 02] proposent de construire un *CDS* en passant par la construction des *MIS*. Dans [Li 02], un nœud est MIS s'il possède l'identité la plus faible en comparaison avec ses voisins. L'étape suivante consiste à interconnecter l'ensemble des MIS pour former le backbone virtuel (CDS). Il suffit dans ce cas d'appliquer l'algorithme de choix des connecteurs entre les MIS. Cet algorithme est basé sur un échange de messages régulier entre les nœuds pour connecter les MIS à deux et à trois sauts

Dans [Fernandess 02], les auteurs proposent de trouver un arbre couvrant du réseau qui se base sur un CDS. Les auteurs utilisent l'algorithme de [Alzoubi 02] pour construire le CDS. Ensuite, l'arbre couvrant est divisé en des sous-arbres, tels que le diamètre de chaque sous-arbre est égal au plus à $2k$-sauts. Chaque sous-arbre est un k-cluster. Les auteurs ne donnent pas l'algorithme de la maintenance de cette structure, qui ne semble pas triviale.

La structure d'arbre dans les réseaux mobiles ad hoc peut s'avérer efficace. Elle permet d'éliminer les boucles de routage, déterminer le chemin le plus court, etc. Cependant, il est important de maintenir la structure d'arbre au cours du temps. En effet, le réseau mobile ad hoc est très dynamique, les nœuds peuvent disparaître ou apparaître à n'importe quel moment. Il faut ainsi mettre à jour les informations de chaque nœud pour conserver l'arbre.

Comme nous venons de le voir, il existe de nombreux algorithmes de clustering. Certaines solutions ne répondent pas à la totalité des exigences des protocoles en terme de persistance, de robustesse, de passage à l'échelle, etc. Par exemple, les algorithmes de clustering qui forment des clusters à un saut engendrent des reconstructions assez fréquentes de la topologie. Dans d'autres cas, la charge d'un cluster-head est très importante ce qui peut épuiser ses ressources. Dans ce cas :

- il serait intéressant de répartir la charge d'un cluster-head avec d'autres nœuds pour apporter une certaine équité entre les nœuds et pour rester assez longtemps.
- le clustering doit naturellement être distribué et les clusters doivent être les plus stables possibles, i.e. les nœuds du clusters doivent garder leur rôle suffisamment longtemps afin d'être efficaces.
- selon la nature du problème des mécanismes de clustering doivent être adéquats. Les clusters doivent se former d'une manière dynamique où les cluster-heads sont choisis parmi les nœuds les plus performants du réseaux. Le critère de clustering doit ainsi être adapté au problème afin de rendre les protocoles de communication plus efficaces.

Dans la suite de ce chapitre, nous proposons un nouvel algorithme de clustering destiné à des applications dans le domaine de la sécurité qui s'adapte aux différentes

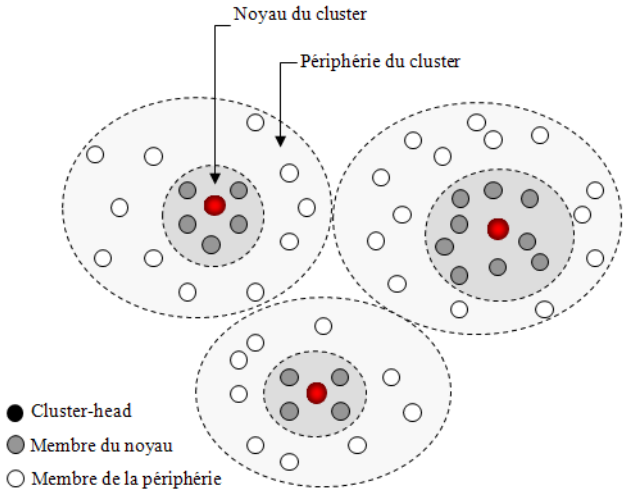

Figure 6.1 : *Structure générale de notre topologie.*

modifications de la topologie.

6.3 Algorithme de clustering pour les MANETs

6.3.1 Architecture globale

Cette section introduit notre topologie. Nous supposons tout d'abord que tous les nœuds diffusent périodiquement un message *hello* à leurs voisins à un seul saut pour obtenir les informations des nœuds qui les entourent. Notre topologie est organisée en clusters. Chaque cluster est composé d'un cluster-head, un noyau et une périphérie.

- Le cluster-head est le nœud qui identifie le cluster. Il est responsable de la communication entre les clusters. Le cluster-head est la racine d'un sous-arbre construit durant le processus de clustering et couvre tous les membres du cluster.
- Le noyau est au centre du cluster. Le cluster-head est l'un des membres du noyau de son cluster. Tous les membres du noyau sont voisins au cluster-head.
- La périphérie est composée des membres du cluster qui ne sont pas dans le noyau.

La Figure 6.1 illustre les principales caractéristiques et éléments de notre structure.

6.3.2 Critère de clustering

Nous nous sommes intéressés au clustering dans le domaine de la sécurité, pour cela notre algorithme utilise la confiance comme critère de partitionnement. La

confiance est fondamentale pour maintenir un certain niveau de sécurité. C'est un aspect important dans la conception et l'analyse des réseaux comme il est un élément essentiel par lequel les relations entre les nœuds peuvent se développer ou cesser. Toutefois, dans un environnement dynamique et mobile sans autorité de confiance centralisée, il n'est pas facile d'évaluer la confiance. Plusieurs solutions existantes [Li 05] [Pirzada 04] [Theodor 06a] [Theodor 06b] proposent de calculer la confiance dans les MANETs sur la base des informations qu'un nœud peut collecter des autres nœuds d'une manière passive. En fait, les informations vitales concernant les autres nœuds peuvent être recueillies par l'analyse des paquets reçus, des paquets transmis et des paquets écoutés [Pirzada 04]. Comme ces interactions sont fréquentes dans le comportement coopératif des nœuds d'un MANET, il ne sera pas difficile d'établir rapidement une première estimation du degré de confiance entre voisins directs. Si les interactions ordinaires ne sont pas suffisantes pour évaluer un rapport de confiance/méfiance, les nœuds peuvent générer un trafic supplémentaire, afin d'évaluer à quel point ils ont confiance en leurs voisins. Ainsi, pour gérer ses relations de confiance, chaque nœud i maintient la valeur de la confiance $tv(i,j)$ pour tout j voisin au nœud i (et éventuellement d'anciens voisins et les nœuds non-voisins pour lesquels le nœud i reçoit des recommandations). Cette valeur reflète le degré de confiance ou méfiance que le nœud i possède sur son voisin j. Plusieurs fonctions de confiance et quantifications des relations de confiance ont été proposées dans la littérature [Theodor 06a] [Theodor 06b]. Nous utilisons la quantification de la confiance proposée dans [Li 05] et nous l'étendons de manière à tenir compte des recommandations de confiance développées dans [Liu 04]. Ainsi, nos valeurs de confiance sont des nombres réels compris entre -1 et $+1$. Un nombre négatif représente le degré de méfiance. -1 indique une méfiance totale. Un nombre positif représente le degré de confiance. $+1$ représente la confiance absolue. Quand un nouveau nœud ou un nœud inconnu j entre dans le voisinage du nœud i, le nœud i initialise $tv(i,j)$ à une première valeur *init_trust*, ($tv(i,j) = init\_trust$). Cette valeur initiale est utile pour les deux nœuds qui n'ont pas encore communiqué ensemble. Par exemple, s'ils sont totalement inconnus *init_trust* est égale à -1, s'ils peuvent s'authentifier alors *init_trust* est égale à $+1$. La valeur de $tv(i,j)$ évolue dans le temps en fonction des interactions des deux nœuds dans le cadre des protocoles des différentes couches du réseau. Ces interactions peuvent être positives ou négatives. Une interaction positive augmente la valeur de la confiance et une interaction négative diminue la valeur de la confiance en conséquence. Notez que deux voisins i et j peuvent avoir des interprétations différentes de leurs échanges. Ainsi, $tv(i,j)$ peut être différente de $tv(j,i)$. En considérant qu'un nœud peut avoir diverses interactions avec le nœud j, peut être un nombre illimité d'interactions, et chaque interaction différente peut avoir un effet différent sur l'évaluation de la

confiance, nous utilisons la fonction suivante [Li 05] pour calculer la valeur de la confiance du nœud i au nœud j :

$$tv(i,j) = tanh\left(\sum_{k=1}^{n_1} \mu_k \omega_k\right) \tag{6.1}$$

où n_1 est le nombre d'interactions entre les deux nœuds. ω_k est le poids de l'interaction numéro k. μ_k est $+1$ si l'interaction k est positive et -1 si elle est négative. La fonction $tanh()$ sert à projeter la somme des différentes interactions dans l'intervalle $[-1, +1]$. Plusieurs exemples d'interactions qui peuvent être utilisés pour calculer les valeurs de confiance sont décrits dans [Pirzada 04]. Toutes ces interactions sont basées sur des informations de routage. Nous développons ici les interactions que nous utilisons dans notre algorithme de clustering (voir tableau 6.2 pour une synthèse).

1. *Connaissance passive* : Un nœud peut obtenir des informations importantes d'un voisin lors de la construction de route par exemple. En fait, si un nœud se met en mode "promiscuous" après la transmission de tous les paquets de manière à entendre la retransmission par le nœud destinataire, il peut obtenir l'information suivante à propos de ce voisin [Pirzada 04] :
   - Il agit comme un trou noir si le paquet n'est pas retransmis.
   - Il procède à une attaque de modification si le contenu a été modifié.
   - Il procède à une attaque de fabrication si un paquet auto-produit est transmis.
   - Il procède à une attaque par vol d'identité si les adresses *IP* ont été falsifiées.
   - Il induit des retards en retardant la retransmission du paquet.
2. *Exactitude/Erreur de paquets* : Quand un nœud reçoit un paquet correct, il peut augmenter la valeur de confiance qu'il attribue à celui qui a envoyé le paquet, ainsi que les autres nœuds dans le chemin de la source (si le protocole de routage permet de connaître les nœuds qui composent un itinéraire à partir d'une source vers une destination). De la même façon, si le paquet reçu est erroné, le récepteur peut diminuer la valeur de la confiance qu'il attribue à l'expéditeur du paquet ainsi que la valeur de la confiance qu'il attribue aux nœuds intermédiaires de la route.
3. *Comportement altruiste* : Si un nœud intermédiaire, sur une route à une destination donnée, reçoit un paquet pour lequel son prochain saut n'est pas disponible, il peut supprimer le paquet et en informer l'expéditeur. Toutefois, s'il existe une route vers le destinataire final il peut utiliser cette route à partir de son cache, envoyer le paquet sur la nouvelle route et informer l'expéditeur du lien brisé. Si la nouvelle route est jugée exacte, cela révèle que l'expéditeur de l'erreur à un comportement altruiste. Par conséquent, cette information

peut être utilisée pour augmenter les relations de confiance entre les deux nœuds.

4. *Appréciation globale* : Un nœud i attribue une plus grande confiance à un voisin qui lui offre toujours des paquets corrects et des routes efficaces et une baisse de confiance à un voisin qui lui offre toujours des paquets incorrects ou des itinéraires inefficaces. Pour ce faire, le nœud i doit enregistrer le nombre d'interactions positives, $pos(i,j)$, et le nombre d'interactions négatives, $neg(i,j)$, qu'il a avec un voisin j. Ensuite, il examine périodiquement le ratio $pos(i,j)/total(i,j)$ (resp. $neg(i,j)/total(i,j)$), où $total(i,j)$ est le nombre total d'interactions qu'un nœud i possède avec le nœud j. Le nœud i compare ce ratio avec un seuil prédéfini $\varepsilon_{positive}$ (resp. $\varepsilon_{negative}$). Si $pos(i,j)/total(i,j) > \varepsilon_{positive}$, le nœud i augmente $tv(i,j)$ avec une notation globale prédéfinie ga_{pos}. De même, si $neg(i,j)/total(i,j) > \varepsilon_{negative}$, le nœud i diminue $tv(i,j)$ avec une notation globale prédéfinie ga_{neg}. Cela favorise la fidélité des nœuds au fil du temps.

5. *Respect/non-respect des règles de clustering* : un nœud qui ne respecte pas les règles de clustering est évidemment un nœud malveillant. Ses voisins peuvent détecter ce problème en observant comment il établit ses variables de clustering, décrites plus loin dans ce chapitre, dans ses messages *hello*.

6. *Incohérence du rapport de confiance* : un nœud qui distribue de faux rapports de confiance ou des mensonges au sujet de ses relations de confiance est malveillant. Ce comportement peut être détecté en comparant le rapport de confiance reçu et la surveillance des communications de ses voisins.

7. *Communication avec les nœuds malveillants* : quand un nœud échange régulièrement des messages avec un nœud malveillant, il est considéré comme un nœud malveillant.

Ces appréciations de confiance directes peuvent être renforcées par les rapports de confiance distribués. Les rapports de confiance permettent aux nœuds de partager les informations de confiance et de les diffuser dans le réseau. Une approche simple pour distribuer les rapports de confiance consiste pour chaque nœud à diffuser uniquement les rapports de confiance à ses voisins immédiats [Liu 04]. Un rapport de confiance initié par le nœud k contient la liste des valeurs de confiance qu'il a des autres nœuds, à savoir $\{tv(k,j)\}$. Lorsque le nœud i reçoit des rapports de confiance d'un certain nœud j, il les utilise pour améliorer sa valeur de confiance $tv(i,j)$ comme suit :

$$tv(i,j) = tanh\left(\sum_{k=1}^{n_1} \mu_k \omega_k + \sum_{k=1}^{n_2} tv(i,k) * tv(k,j)\right) \tag{6.2}$$

Interaction	Type
Attaque du trou noir	Négative
Attaque de modification	Négative
Attaque de fabrication	Négative
Usurpation d'identité	Négative
Comportement égoïste	Négative
Temps de latence	Négative
Paquet correct	Positive
Paquet erroné	Négative
Comportement altruiste	Positive
Appréciation périodique globale positive	Positive
Appréciation périodique globale négative	Négative
Authentification	Positive
Recommandation positive	Positive
Recommandation négative	Positive
Aucun contact préalable	Négative
Non-respect des règles de clustering	Négative
Respect des règles de clustering	Positive
Incohérence du rapport de confiance	Négative
Communication avec les nœuds malveillants	Négative

Figure 6.2 : *Exemples d'interactions qui évaluent la confiance.*

où n_2 est le nombre de nœuds qui ont envoyé des rapports de confiance sur le nœud j au nœud i. Si le rapport reçu est celui d'un nœud inconnu, ce rapport de confiance n'est pas considéré. En outre, l'utilisation de $tv(i,k)$ comme un poids pour un rapport de confiance initié par un nœud k favorise les considérations directes de confiance. Certains nœuds malveillants peuvent mentir sur les rapports de confiance. Toutefois, ces faux rapports peuvent être détectés par les voisins. Une surveillance mutuelle des nœuds évite les incohérences des rapports de confiance reçus.

Pour utiliser les valeurs de confiance calculées par les nœuds comme un critère de clustering, nous définissons deux seuils de confiance S_{min} et S_{max} où $S_{min} \leq S_{max}$, $S_{max} \in [0,1]$ et $S_{min} \in [-1,0]$. Ces seuils servent à classer les relations de confiance entre les nœuds en trois catégories selon les valeurs de confiance :

- *Confiance totale (TT)* : Une relation entre deux nœuds i et j est une relation de confiance totale ($Relation(i,j) = TT$) si et seulement si $tv(i,j) \in [S_{max}, 1]$ et $tv(j,i) \in [S_{max}, 1]$.
- *Confiance partielle (PT)* : Une relation entre deux nœuds i et j est une relation de confiance partielle ($Relation(i,j) = PT$) si et seulement si :
  - $tv(i,j) \in [S_{max}, 1]$ et $tv(j,i) \in [S_{min}, S_{max}]$; ou
  - $tv(j,i) \in [S_{max}, 1]$ et $tv(i,j) \in [S_{min}, S_{max}]$; ou
  - $tv(i,j) \in [S_{min}, S_{max}]$ et $tv(j,i) \in [S_{min}, S_{max}]$.

– *Méfiance (DT)* : Une relation entre deux nœuds i et j est une relation de méfiance ($Relation(i,j) = DT$) si et seulement si $tv(i,j) \in [-1, S_{min}]$ ou $tv(j,i) \in [-1, S_{min}]$.

Notons que les trois relations sont symétriques. Par exemple, si un nœud i a une relation de confiance totale avec un nœud j alors le nœud j a aussi une relation de confiance totale avec le nœud i.

Dans la suite, nous développons les différentes étapes de notre algorithme de clustering.

6.3.3 Algorithme de clustering basé sur la confiance

L'algorithme de clustering ajoute aux messages *hello* les champs suivants :

– $TT - edge_i$: Nombre de relations TT qu'un nœud i a avec ses voisins.
– CH_i : Le cluster-head du cluster est rattaché le nœud i. CH_i est égal à *null* si le nœud i n'appartient pas encore à un cluster.
– $Core_i$: Le membre du noyau auquel appartient le nœud i. Si i est un cluster-head ou un membre du noyau alors $Core_i$ est fixé à i. Si i n'est pas encore attaché à un noyau alors $Core_i$ est mis à *null*.
– $Hop - CH_i$: nombre de sauts à partir du nœud i au cluster-head.
– $tv(i,j)$: Pour chaque voisin j, ce champ représente la valeur de la confiance du nœud i envers le nœud j.
– $Parent_i$: ce champ exprime le père du nœud i dans le sous-arbre correspondant.

Le clustering est effectué en trois phases totalement dynamiques et réparties :

Phase 1 : Election du cluster-head. Chaque nœud qui a le plus grand nombre $TT - edge$ parmi ses voisins qui n'appartiennent pas encore à un cluster se déclare comme un cluster-head. En cas d'égalité, le nœud ayant le plus de voisins est élu. En cas d'égalité des valeurs $TT - edge$ et égalité du nombre de voisins, le nœud qui a la plus grande identité est privilégié. Une fois qu'un nœud devienne un cluster-head, il met son identité dans le champ CH de ses messages *hello* et change la valeur du champ $Hop - CH$ à zéro.

Phase 2 : Formation du noyau. Tous les voisins qui ont des relations TT avec un cluster-head forment le noyau du cluster. Leurs messages *hello* contiennent l'identité du cluster-head dans le champ CH et la valeur 1 dans le champ $Hop-CH$. Un nœud peut avoir des relations TT avec plusieurs cluster-heads. Dans ce cas, le nœud choisit le cluster-head qui a le plus grand $TT - edge$.

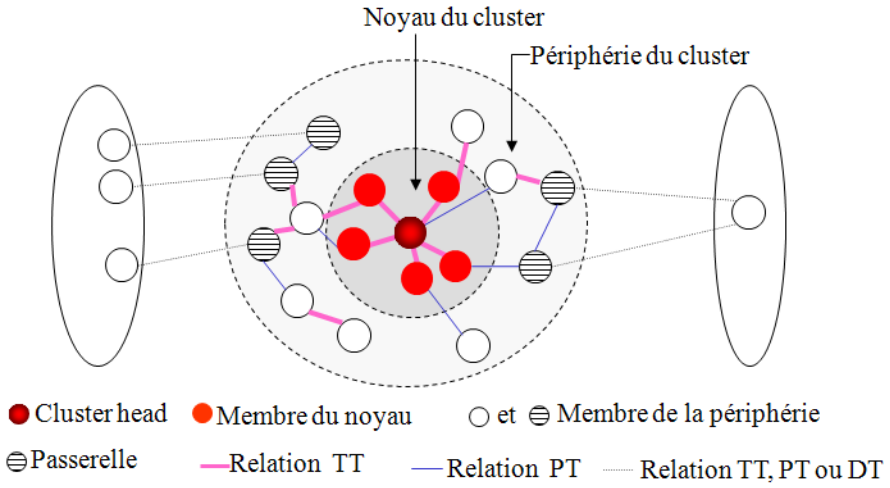

Figure 6.3 : *Le clustering obtenu.*

Phase 3 : Formation de la périphérie du noyau. Après la phase 1 et la phase 2, les nœuds qui entourent le noyau rejoignent le cluster selon les deux étapes suivantes (une relation TT est privilégiée à une relation PT qui est privilégiée à une relation DT, nous la notons par $TT > PT > DT$) :

- *Étape 1 : membres de la périphérie ayant des relations TT.* Après la constitution des noyaux, s'il y a des nœuds qui n'ont pas encore adhéré à un cluster et ont des relations TT avec au moins un noyau, ils rejoignent le cluster avec lequel ils ont la plus grande valeur de confiance.
- *Étape 2 : membres de la périphérie ayant des relations PT.* Cette dernière étape consiste à ajouter les nœuds qui partagent des relations PT avec au moins un nœud dans le cluster. Un nœud de cette catégorie privilégie le cluster avec lequel il a la plus faible distance (nombre de sauts) au cluster-head. Le voisin avec lequel il a une relation PT et la plus faible distance au cluster-head devient son père dans le sous-arbre enraciné. Le cluster-head constitue la racine du sous-arbre. Ce sous-arbre permet de simplifier la communication entre clusters. Un nœud de la périphérie avec au moins un voisin appartenant à un autre cluster est appelé *passerelle*. Lorsqu'un nœud rejoint un cluster, il met à jour les champs CH et $CH - Hop$ du message *hello*. Le clustering obtenu est illustré dans la Figure 6.3.

Pour réussir le clustering, en dépit de la présence des nœuds malveillants, les nœuds honnêtes coopèrent étroitement. Ils ne communiquent pas les messages de clustering aux nœuds malveillants et ignorent tous les messages de clustering provenant de ces nœuds. Ainsi, les messages de clustering et les données de diffusion ne passent que par les relations TT ou les relations PT. Toutefois, même si tous les nœuds malicieux ont été détectés, le clustering peut être perturbé. Une condition

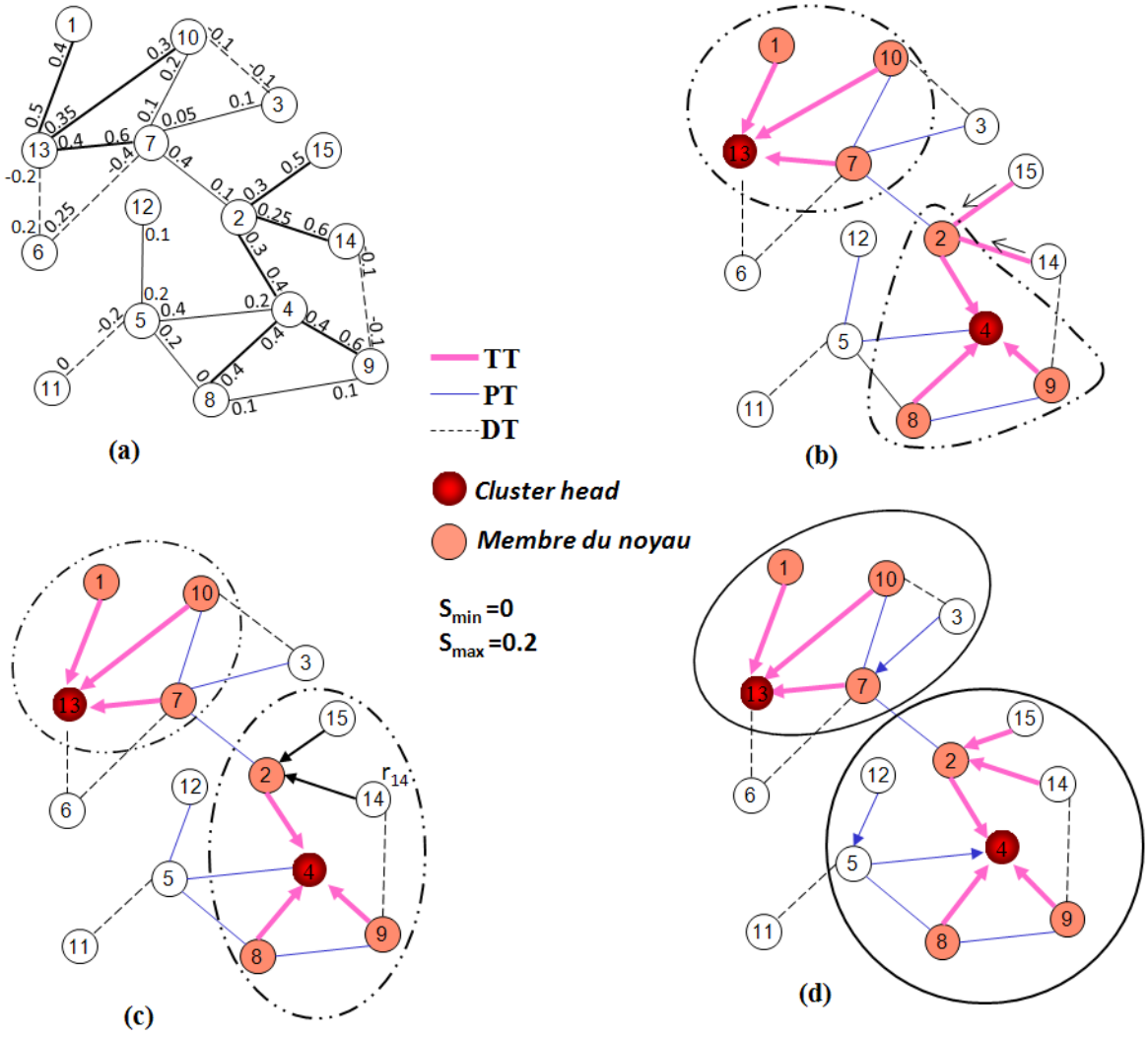

Figure 6.4 : *Exemple de clustering.*

sur le nombre de nœuds malveillants et leur dispersion dans le réseau est nécessaire. En fait, si le réseau n'est pas suffisamment dense et les nœuds malveillants sont dispersés de sorte qu'ils empêchent les nœuds honnêtes de participer au clustering, le protocole ne parviendra pas à atteindre un clustering complet. Comme résultat, des nœuds isolés et des clusters déconnectés peuvent apparaître.

6.4 Exemple de clustering

Nous expliquons notre algorithme de clustering en l'appliquant à l'ensemble de nœuds décrit dans la Figure 6.4. Dans cet exemple, nous supposons que les nœuds ont déjà calculé leurs valeurs de confiance de leurs voisins directs. Chaque nœud possède un identifiant unique et il est libellé par la confiance qu'il attribue à ses

voisins. les seuils de confiance sont fixés à $S_{min} = 0$ et $S_{max} = 0.2$. La Figure 6.4(a) illustre les différentes relations TT, PT et DT dans le réseau en fonction des valeurs de S_{min} et S_{max}. Dans la Figure 6.4(b), les nœuds 4 et 13 se déclarent des cluster-heads en fonction du nombre de relations TT qu'ils ont avec leurs voisins. Rappelons que cette information est diffusée dans les messages *hello*. Chacun des nœuds 4 et 13 met son identité dans le champ CH et met à jour le champ $Hop-CH$ à 0. Ensuite, les nœuds 1, 7 et 10 (respectivement 2, 8 et 9) qui partagent une relation TT avec le cluster-head 13 (respectivement 4) forment le noyau de celui-ci (voir Figure 6.4(c)). Les nœuds 1, 7 et 10 (respectivement 2, 8 et 9) changent leurs champs CH et $Hop - CH$ de leurs messages *hello* à respectivement 13 et 1 (respectivement 4 et 1). Les deux nœuds 14 et 15 ont des relations TT avec le nœud 2. Ainsi, ils rejoignent le cluster 4. Ils sont attachés au membre 2 du noyau (voir Figure 6.4(c)). Ils mettent à jour leurs champs CH et $Hop-CH$ en conséquence. Dans la dernière étape, les nœuds 12 et 5 (respectivement 3) qui partagent une relation PT avec un nœud appartenant au cluster 4 (respectivement 13), rejoignent ce cluster et mettent à jour les champs CH et $Hop - CH$ de leurs messages *hello*. Les arêtes orientées dans l'exemple illustrent les sous-arbres construits durant le processus de clustering. Lorsqu'un nœud rejoint un cluster, il choisit un père dans le plus court chemin vers le cluster-head. Les nœuds 6 et 11 ne partagent aucunes relations TT ou PT avec d'autres nœuds du réseau. Ainsi, l'algorithme de clustering ne tient pas compte de ces nœuds dans les clusters obtenus.

6.5 Maintenance de la topologie

L'algorithme de clustering est auto-stabilisant. Il s'exécute en continu et réajuste les clusters en fonction des relations de confiance entre les nœuds. Les relations de confiance évoluent dans le temps en fonction des interactions entre les nœuds. La mobilité peut également changer la situation des clusters. En fait, quand un nœud acquiert de nouveaux voisins ou perd certains d'entre eux, en raison de la mobilité, plusieurs changements peuvent apparaître dans la situation du nœud à l'intérieur du cluster :

- Le nombre de relations TT ($TT-edge$) du nœud peut changer. Par exemple, si le nœud est un cluster-head, il ne peut plus l'être. Si le nœud est un membre de la périphérie, il peut devenir un membre du noyau ou un cluster-head.
- Les relations PT ou TT qui lient un nœud à un cluster peuvent se briser et le nœud n'appartient plus au cluster.
- Un nœud qui n'a que des relations DT avec ses voisins peut acquérir des relations PT ou TT et rejoint ainsi un cluster.
- Etc.

La Phase 1, la Phase 2 et la Phase 3 de l'algorithme de clustering sont ré-exécutées aussi souvent que nécessaire pour former de nouveaux clusters ou mettre à jour les clusters existants.

Election d'un nouveau cluster-head

Plusieurs situations peuvent impliquer l'élection d'un nouveau cluster-head. Nous nous concentrons sur deux de ces situations : la panne du cluster-head actuel et la modification de la valeur $TT - edge$ du cluster-head actuel de sorte qu'il n'est plus le nœud ayant la plus grande valeur $TT - edge$ parmi ses voisins.

- *Panne du cluster-head* : Si un cluster-head tombe en panne, ses fils (c'est à dire les autres membres du même noyau) ne reçoivent plus ses messages *hello* périodiques. Dans ce cas, les membres du noyau ré-initialisent le champ CH de leurs messages *hello* à *null*. Lors de la réception de ces modifications du message *hello*, les membres de la périphérie sous-jacente mettent également le champ CH de leurs messages *hello* à *null*. Ceci lance les phases de clustering de ces nœuds et réorganise le cluster.
- *Modification de la valeur* $TT - edge$: Un cluster-head dont un de ses voisins a une plus grande valeur $TT - edge$ ré-initialise le champ CH de ses messages *hello* à *null*. Lors de la réception des mises à jour des messages *hello*, les voisins du cluster-head qui sont membres du cluster propagent cette ré-initialisation à tous les membres du cluster. Cela conduit à une situation de panne du cluster-head.

Rupture des relations de confiance

Lorsqu'un nœud est authentifié comme membre du groupe et possède au moins une relation TT ou PT avec un membre du cluster, il demeure un membre du cluster. Lorsque les relations TT ou PT qui lient le nœud au cluster sont cassées (à cause de la mobilité ou changement de la valeur de confiance), le nœud doit être exclu du cluster : il est considéré comme un nœud malveillant. A ce niveau, l'exclusion n'a pas un impact pratique. En fait, si le nœud est malveillant, il ne respectera pas les règles de clustering. Le nœud ensuite rejoint un autre cluster s'il a des relations TT ou PT avec les autres nœuds du réseau. Cette modification peut engendrer d'autres modifications dans la constitution des clusters. Notons ici que toutes les étapes de clustering sont basées sur la confiance et le strict respect des règles de clustering des membres. Les nœuds malveillants peuvent ne pas respecter ces règles. Dans ce cas, à l'aide des valeurs de confiance, il est possible de détecter les nœuds malveillants. Comme décrit un peu plus haut dans cette section, un nœud

peut détecter qu'un voisin ne respecte pas les règles de clustering en contrôlant les valeurs de ses variables de clustering (contenues dans ses messages *hello*).

Défaillance d'un membre du noyau ou d'un membre de la périphérie

Lorsqu'un membre du noyau ou un membre de la périphérie tombe en panne, les autres membres de la périphérie qui en dépendent sont isolés du cluster. Ces membres ré-exécutent la Phase 3 du clustering pour sélectionner un nouveau père dans le cluster ou pour rejoindre un autre cluster.

Gestion de nouveaux nœuds

Lorsqu'un nouveau membre j devient voisin d'un membre i , i affecte la valeur 1 à sa relation de confiance avec j ($tv(i,j) = 1$). Cela signifie qu'un nouveau membre lui est accordé une relation TT à son arrivée et, par conséquent, a accès aux différentes étapes de clustering. Toutefois, si i et j ne parviennent pas à s'authentifier, ils attribuent l'un à l'autre une relation de méfiance.

6.6 Description des différentes règles de l'algorithme

Dans cette section, nous allons présenter les règles de notre algorithme de clustering en utilisant les mêmes notations et terminologies que dans la section 5.3.4. Nous commençons par caractériser un état légitime.

Définition 6.6.1. Un état légitime est un clustering stable avec formation des cluster-heads, des membres du noyau, des membres de la périphérie et des membres exclus.

L'algorithme se compose de cinq règles détaillées dans les Algorithmes 10, 11 et 12, chacune détermine le rôle du nœud selon les messages *hello* qu'il reçoit de ses voisins.

- La règle ($R0$) est nécessaire pour mettre à *null* les variables du sommet qui ne partage que des relations de méfiance avec ses voisins. Par conséquent, il n'appartient à aucun cluster.
- La règle ($R1$) est nécessaire pour l'élection des cluster-heads. Un nœud i devient cluster-head si :
  - Il possède un nombre de relations TT supérieur à celui de ses voisins j avec qui, il partage une relation TT (i.e. $TT - edge_i > TT - edge_j$).

- Il n'est voisin à aucun cluster-head et son nombre de relations TT est supérieur à celui de ses voisins j avec qui, il partage une relation TT et qui n'appartiennent pas encore à un cluster.

Le nœud i devient ainsi cluster-head, son champ CH_i contient son identité i et il met à jour les champs $Hop-CH_i$ et $Parent_i$ respectivement à 0 et i. Dans cette règle, nous utilisons la fonction $Compare(TT-edge_i, TT-edge_j)$. En effet, dans certains cas, il peut exister deux voisins ayant la même valeur maximale du nombre de relations TT (il existe plusieurs j tels que $TT-edge_i = TT-edge_j$). Dans ce cas, le sommet i possède le nombre de voisins le plus élevé. S'il y a égalité du nombre de voisins, le sommet i possède l'identité la plus grande par rapport aux sommets j (i>j). La comparaison se fait ainsi selon l'ordre suivant : $TT-edge \rightarrow degré \rightarrow identité$. Les conflits sont bien résolus car chaque nœud possède un identifiant unique.

– La règle ($R2$) permet de ré-initialiser les variables d'un cluster-head en cas d'incohérences. Si un cluster-head, c'est à dire $CH_i = i$, possède une valeur erronée des champs $Hop-CH_i$ ou $Parent_i$ alors le sommet i met les champs CH_i, $Hop-CH_i$ et $Parent_i$ à $null$. D'un autre coté, des changements topologique peuvent survenir suite à l'ajout, la suppression ou le déplacement des entités mobiles. Par conséquent, un cluster-head peut ne plus avoir le plus grand nombre de relations TT en comparaison avec ses fils j (les nœuds j qui ont choisi le nœud i comme père : $Parent_j = i$). Le nœud i exécute ainsi la règle ($R2$) pour initialiser son état et ainsi mettre les champs CH_i, $Hop-CH_i$ et $Parent_i$ à $null$.

– La règle ($R3$) permet à un nœud qui n'est pas un cluster-head de choisir son cluster. Un nœud va intégrer soit le noyau du cluster soit la périphérie du cluster. Cette règle utilise la fonction $Compare(Relation(i,j), Relation(i,k))$ qui permet à un nœud de privilégier le cluster avec lequel il est lié par une relation TT à un autre cluster. Chaque nœud décide en local selon ses relations avec ses voisins. Un nœud i choisit le nœud j comme père, car :
  - j est un cluster-head avec qui, il partage une relation TT. S'il y a plusieurs cluster-heads voisins, le nœud i choisit le nœud j selon l'ordre de comparaison suivant : $TT-edge \rightarrow degré \rightarrow identité$ (nous suivons le même raisonnement que précédemment).
  - i n'a pas un cluster-head voisin et j appartient déjà à un cluster et i partage une relation TT avec j. S'il y a plusieurs voisins, le nœud i choisit le nœud j selon le même ordre de comparaison que précédemment : $TT-edge \rightarrow degré \rightarrow identité$
  - i n'a pas de relations TT et le champ $Hop-CH_j$ du nœud j contient la valeur la plus faible par rapport aux autres voisins. S'il y a plusieurs voisins

Algorithme 10 Les membres exclus.

$(R0)$: : if $\forall j \in N.iRelation(i,j) = DT$
then $CH_i = null$
$\wedge Parent_i = null$

Algorithme 11 Choix et mise à jour du cluster-head.

$(R1)$: : if $\forall j \in N.i[Relation(i,j) = TT \Rightarrow Compare(TT - edge_i, TT - edge_j)]$
$\vee \forall j \in N.i[CH_j \neq j \wedge (CH_j = null \wedge Relation(i,j) = TT) \Rightarrow$
$Compare(TT - edge_i, TT - edge_j)]$
then $CH_i = i$
$\wedge Parent_i = i$
$\wedge Hop - CH_i = 0$

$(R2)$: : if $CH_i = i \wedge (\exists j \in N.i[(Relation(i,j) = TT \wedge Parent_j = i) \Rightarrow$
$Comapre(TT - edge_j, TT - edge_i)] \vee Parent_i \neq i \vee Hop - CH_i \neq 0)$
then $CH_i = null$
$\wedge Parent_i = null$
$\wedge Hop - CH_i = null$

Notation :
$Compare(x,y)$: est une fonction booléenne qui renvoie vrai si $x > y$,
si $x = y$ selon la valeur de x et y, la comparaison dans ce cas se fait selon l'ordre suivant : *degré* $\rightarrow$ *identité*

ayant cette même valeur, le nœud i choisit le nœud j selon le même ordre de comparaison que précédemment : $TT - edge \rightarrow$ *degré* $\rightarrow$ *identité*.

Après avoir choisi son père j, le nœud i met à jour ses champs $Parent_i$ et $Hop - CH_i$ respectivement à j et $Hop - CH_j + 1$.

– Comme pour le cas du cluster-head, en cas de changement de la topologie, si un nœud i ne trouve plus son père alors il exécute la règle $(R4)$ et il initialise ainsi ses champs CH_i, $Parent_i$ et $Hop - CH_i$ à $null$.

Dans notre algorithme, les règles $(R0)$, $(R2)$ et $(R4)$ sont prioritaires par rapport aux autres règles $(R1)$ et $(R3)$. En effet, un sommet actif qui présente une valeur erronée de ses variables initialise son état à $null$. Ensuite, il exécute la règle correspondante pour devenir un cluster-head, un membre du noyau, un membre de la périphérie ou un membre exclu.

6.7 Convergence et exactitude de l'algorithme

Nous montrons dans cette section que notre algorithme converge vers un état légitime, c'est à dire à partir d'un état initial quelconque, l'algorithme converge vers un état stable composé de cluster-heads, membres du noyau, membres de la périphérie et membres exclus. Nous allons supposer pour cela que les nœuds ne sont

Algorithme 12 Choix et mise à jour des autres nœuds du cluster.

$(R3)$: : if $\forall k \in N.i \exists j \in N.i\ Compare(Relation(i,j), Relation(i,k))$
$\wedge CH_i \neq i$
then $CH_i = CH_j$
$\wedge Parent_i = j$
$\wedge Hop - CH_i = Hop - CH_j + 1$

$(R4)$: : if $CH_i \neq CH_{Parent_i} \vee Hop - CH_i \neq Hop - CH_{Parent_i} + 1$
then $CH_i = null$
$\wedge Parent_i = null$
$\wedge Hop - CH_i = null$

Notation :
$Compare(x, y)$: est une fonction booléenne qui renvoie vrai si $x > y$,
si $x = y$ selon la valeur de x et y, la comparaison dans ce cas se fait selon l'ordre suivant : $TT - edge \rightarrow degré \rightarrow identité$

pas malveillants.

Lemme 6.7.1. *Le nœud qui exécute la règle* $(R0)$ *ne change son état qu'une seule fois et reste stable après.*

Démonstration. Un nœud qui ne partage que des relations DT avec ses voisins ne va appartenir à aucun cluster, il exécute ainsi la règle $(R0)$. De plus, aucun de ses voisins ne va le considérer, il sera ainsi ignoré. Par conséquent, il ne va plus changer d'état. □

Lemme 6.7.2. *Le nœud qui exécute la règle* $(R1)$ *se stabilise au plus après* $(\Delta-1)^2$ *mouvements.*

Démonstration. Le nœud qui possède le nombre maximum de relations TT en comparaison avec tous ses voisins avec qui, il partage une relation TT, va devenir un cluster-head. Il ne changera pas d'état par la suite, car la décision est locale et il a la meilleure valeur (le nombre maximum de relations TT). Cette comparaison dépend uniquement du nombre de relations TT qui est déjà mis à jour par le message *hello*. Par contre, dans certains cas de la règle $(R1)$, le nœud se déclare comme un cluster-head car il a le nombre maximum de relations TT en comparaison avec tous ses voisins avec qui, il partage une relation TT et qui n'appartiennent pas encore à un cluster. Sa décision ne dépend que de ses voisins qui se stabilisent au plus après $\Delta - 1$ mouvements. D'où, un cluster-head se stabilise au plus après $(\Delta - 1)^2$ mouvements

□

Lemme 6.7.3. *Le système entre dans un état légitime en* $O(n(\Delta - 1)^2)$.

Démonstration. Les autres sommets du graphe, qui ne sont pas des cluster-heads, choisissent toujours le parent avec qui, ils partagent la relation la plus solide. Ce parent existe et il est unique car la fonction $Compare(x, y)$ choisit un seul sommet. La meilleure relation c'est la relation avec un cluster-head qui est la racine du sous-arbre. D'après le lemme précédent, un cluster-head converge vers un état stable au plus après $(\Delta - 1)^2$ mouvements. L'algorithme converge ainsi vers un état légitime en $O(n(\Delta - 1)^2)$.

□

6.8 Application à la gestion de clés de groupe

Dans cette section, nous détaillons les différentes étapes de l'algorithme de gestion de clés. La gestion de clés se base sur la topologie de clusters construite ci-dessus.

La gestion de clés de groupe permet d'assurer la confidentialité des communications au sein d'un groupe. Il s'agit d'établir un secret partagé entre les membres du groupe. Ce secret appelé *"traffic encryption key"* (*TEK*) sert à chiffrer/déchiffrer les messages échangés dans un groupe. D'après le modèle de [Kim 00] [Kim 04] un protocole de gestion de clés de groupe doit satisfaire les propriétés suivantes :

- La **confidentialité de la clé de groupe** consiste à garantir l'impossibilité de calculer ou de découvrir la clé du groupe par un utilisateur n'appartenant pas au groupe.
- L'**indépendance de la clé de groupe** garantit qu'un intrus passif, qui connaît un ensemble de clés, ne peut pas découvrir la clé de groupe
- La **confidentialité future** garantit que les nouvelles clés doivent rester hors de portée des anciens membres de groupe.
- La **confidentialité passée** garantit que les clés de groupe déjà utilisées ne peuvent pas être découvertes par un nouveau membre du groupe.

Pour garantir la confidentialité future et passée, il est nécessaire de mettre à jour la clé du groupe suite à l'arrivée ou le départ de membres. Ainsi, le re-calcul de la clé peut induire un coût de communication important en particulier dans le cas de changements fréquents de la composition du groupe. Par conséquent, une propriété appréciée de tout mécanisme de re-calcul de la clé est son **facteur d'échelle 1-affecte-n** (***1-affects-n scalability***) [Mittra 97]. Ce critère permet d'évaluer l'efficacité d'un système de gestion de clés de groupe qui traduit la capacité du système à s'adapter aux variations de la taille du groupe et à prendre en charge les événements qui modifient cette taille (départ, ajout, partition). Il mesure l'effet d'une entrée ou sortie d'un membre du groupe sur les autres membres du groupe.

Dans notre algorithme de gestion de clés nous supposons que :

- tous les nœuds possèdent des certificats de clé publique avant de rejoindre le réseau. C'est à dire, chaque nœud i à une paire de clés : une clé publique k_i et une clé privée k_i^{-1}. Cette paire de clés permettra de sécuriser les communications entre les paires et d'assurer l'authentification et l'intégrité des données transmises.
- Lorsque deux nœuds veulent communiquer les uns avec les autres, ils doivent être authentifiés mutuellement. Notre protocole omet les étapes de cette authentification mutuelle.
- Les nœuds qui composent le noyau du cluster sont responsables de la génération de la clé de chiffrement du cluster. Cette clé est obtenue en utilisant un protocole d'accord de clés entre les membres du noyau.

Chaque membre du noyau est responsable de la distribution de la clé dans sa périphérie. Ainsi, les membres du noyau forment une zone d'accord de clé. Cela signifie que tous les membres du noyau contribuent au calcul de la clé TEK du cluster. La périphérie est une zone de distribution de clés, c'est à dire que les membres de la périphérie obtiennent la clé TEK des membres du noyau du cluster. Tout cluster c maintient une clé de chiffrement du trafic TEK_c qui sert à chiffrer/déchiffrer les données de diffusion par tous les membres du cluster. Par conséquent, le facteur d'échelle 1-affecte-n est améliorée car un re-calcul de la clé après une opération arrivée/départ ne concerne que les membres du cluster concerné. TEK_c est calculée par les membres du noyau en utilisant un protocole d'accord de clé. Puisque les principaux membres sont des voisins, TEK_c est rapidement disponible. Pour distribuer TEK_c, dans la périphérie du cluster, une clé de chiffrement de clés (KEK) est nécessaire. Pour ne pas perdre dans la vitesse de déploiement de TEK_c, nous partageons cette tâche entre les membres du noyau. Cela renforce également le facteur d'échelle 1-affecte-n car mettre à jour une nouvelle clé KEK ne concerne qu'un sous-ensemble des membres du cluster. Ainsi, chaque membre q d'un noyau partage une clé, KEK_q, avec les membres de la périphérie qui appartiennent au sous-arbre dont-il est la racine. KEK_q sert à transmettre à ces membres de la périphérie la clé TEK_c. La communication sécurisée entre les clusters est assurée par le calcul d'une clé TEK entre chaque paire de clusters adjacents. Les détails du calcul de ces clés sont indiqués ci-dessous. Nous introduisons avant les notations et les messages utilisés dans l'algorithme.

6.8.1 Notation et messages

Cinq types de messages sont utilisés dans le protocole.

$NEW - TEK$: Ce type de message est utilisé par un membre du noyau pour distribuer une nouvelle clé TEK aux membres de sa périphérie. Il contient l'identité

du CH, l'identité du membre du noyau et la clé de chiffrement TEK chiffrée avec la clé KEK du membre du noyau.

$NEW-TEK-RQ$: Ce type de message est utilisé par un membre du noyau pour demander au cluster-head de mettre à jour la clé actuelle du cluster.

$JOIN/LEAVE$: Ces messages sont envoyés par les membres pour adhérer ou quitter le groupe de diffusion. Une expulsion d'un membre en raison d'une relation DT est considérée comme une réception d'un message $LEAVE$. Ce type de message spécifie le type de changement adhésion/expulsion, les informations nécessaires à l'authentification du membre, la contribution du membre à la clé KEK dans le cas d'un $JOIN$ et le membre du noyau auquel il choisit d'être attaché.

$Contact-Adj-CH$: Ce type de message est utilisé par un cluster-head pour calculer une clé commune avec chaque cluster-head adjacent. Il précise la contribution du cluster-head et ses informations d'authentification. Ce message est pris en charge par les passerelles. Une passerelle transmet à une passerelle adjacente si et seulement si elle fait confiance à cette passerelle.

$Cluster-Diffuse$: Ce type de message est utilisé par un cluster-head ou par tout membre du cluster pour propager les données de diffusion chiffrées dans le cluster. Il doit être transmis à tous les membres du cluster.

$CH-to-CH$: Ce type de message est utilisé par un cluster-head pour propager les données de diffusion à un cluster-head adjacent. Il contient l'identité du cluster-head source, l'identité du cluster-head destinataire et les données chiffrées par leur clé commune. Ce message a besoin d'établir une route entre les deux cluster-heads. La mise en place de cet itinéraire peut être simplifiée par l'utilisation des nœuds passerelles. En fait, le message traverse le sous-arbre vers les feuilles jusqu'à atteindre une passerelle. Chaque passerelle doit transmettre le message aux passerelles des clusters adjacents. Ensuite, le message traverse le sous-arbre en destination du cluster-head destinataire.

6.8.2 Gestion de clés intra-cluster

a. L'établissement des clés

Les membres du noyau du même cluster c calculent TEK_c en utilisant le protocole d'accord de clés de groupe (TRP) introduit dans [Augot 07]. TRP ne nécessite que deux tours pour calculer la clé. Ainsi, les membres du noyau procèdent comme suit :

- Tour 1 : Chaque membre du noyau i génère un secret aléatoire r_i et envoie sa version publique $br_i = g^{r_i} \, mod \, p$ au cluster-head. Rappelons que tous les membres du noyau sont des voisins du cluster-head.
- Tour 2 : Le cluster-head ℓ élève à la puissance r_ℓ (son propre secret) la version publique du secret de chaque membre du noyau et diffuse à ses voisins $g^{r_i r_\ell} \, mod \, p$ pour tous les i dans $M - \{\ell\}$, où M est l'ensemble des membres du noyau. Chaque membre du noyau supprime son secret de $g^{r_i r_\ell}$ pour obtenir g^{r_ℓ}. Tous les membres du noyau du même cluster calculent la même clé :

$$TEK_c = g^{r_i} * \prod_{i \in M - \{\ell\}} g^{r_i r_\ell} = g^{r_\ell \left(1 + \sum\limits_{i \in M - \{\ell\}} r_i \right)} \tag{6.3}$$

Une fois la clé TEK_c calculée, chaque membre q du noyau distribue cette clé aux membres de la périphérie sous-jacente chiffrée avec KEK_q. En fait, le membre q du noyau du cluster c partage une clé de chiffrement de clés KEK_q avec tous les membres de la périphérie qui en dépendent. KEK_q est également calculée durant l'algorithme de clustering comme suit :

- Chaque fois qu'un nœud i de la périphérie rejoint un cluster c, il génère un secret r_i et envoie sa version publique g^{r_i}, dans un message $JOIN$, à son père dans le cluster en tant que contribution pour calculer KEK_q. Les ancêtres du nœud i diffusent la contribution dans le sous-arbre du cluster jusqu'à atteindre un membre du noyau.
- Le membre du noyau exécute le tour 2 du protocole TRP pour calculer une clé avec tous les membres de la périphérie qui lui sont attachés.

b. Re-calcul de la clé

Afin d'assurer la confidentialité future et passée, la clé de chiffrement doit être mise à jour à chaque fois qu'un nouveau membre rejoint le groupe ou un ancien membre est exclu de celui-ci. Rappelons que, en plus de la liste des membres autorisés, les relations de confiance sont également considérés comme un critère d'adhésion en particulier pour l'exclusion des nœuds malveillants. Ainsi, il suffit qu'un membre du groupe n'ait plus de relations de confiance avec les autres membres du groupe pour être expulsé de la session de diffusion. Dans le cas d'arrivée d'un membre, une nouvelle clé KEK est calculée pour le membre du noyau auquel le nouveau membre est attaché et ainsi une nouvelle clé TEK est calculée pour l'ensemble du cluster. Le nouveau membre donne sa contribution à son père dans le sous-arbre du cluster. Cette contribution est transmise au membre du noyau dans un message $JOIN$. Le membre du noyau génère un nouveau secret et exécute le tour 2 du protocole TRP. Tous les nœuds concernés calculent la nouvelle clé KEK. Le membre du

noyau envoie un message $NEW - TEK - RQ$ au cluster-head. Ce dernier génère un nouveau secret et exécute le deuxième tour du protocole TRP pour calculer une nouvelle clé TEK pour le cluster. Chaque membre du noyau distribue la nouvelle clé TEK dans sa périphérie chiffrée avec la clé KEK correspondante.

Le re-calcul de la clé est également lancé lorsqu'un membre quitte volontairement le groupe ou est exclu de celui-ci, en raison de l'apparition de relations DT. Nous considérons ces deux hypothèses séparément :

- Cas d'un nœud qui quitte volontairement le groupe : Ce nœud envoie un message $LEAVE$ à son père dans le cluster. Ce message est transmis dans le sous-arbre jusqu'à atteindre le membre du noyau responsable du départ du nœud.
- Cas d'un nœud exclu : rappelons qu'un nœud d est lié à un cluster soit par une relation TT soit une relation PT. Cette relation peut l'attacher à plusieurs nœuds dans le cluster. Toutefois, selon l'algorithme de clustering, il existe un nœud particulier que le nœud d choisit en tant que père dans le sous-arbre du cluster. Ainsi, le re-calcul n'est effectué que lorsque la relation TT ou la relation PT qui lie le nœud d à son père est rompue. Le père envoie un message $LEAVE$ au noyau.

Dans les deux cas, le message $LEAVE$ contient l'identité du membre exclu/sortant. Chaque nœud sur la route vers la racine reçoit un message $LEAVE$ et transmet cette information au membre du noyau concerné. Ensuite, le membre du noyau envoie un message $NEW - TEK - RQ$ au cluster-head et met à jour la clé KEK comme suit :

- Chaque membre du noyau maintient une liste qui contient l'identité des nœuds de la périphérie, qui contribuent au calcul de la clé KEK, avec leurs contributions correspondantes.
- Le membre du noyau, qui reçoit un message $LEAVE$, génère un nouveau secret r'_ℓ.
- Il se débarrasse de l'identité et de la contribution des membres exclus de sa liste de contributions.
- Il élève ensuite chaque contribution de la liste à la puissance r'_ℓ (son nouveau secret) et transmet les valeurs obtenues aux membres rattachés à son sous-arbre.
- Tous les membres de la périphérie de ce sous-arbre calculent la nouvelle clé KEK selon le protocole TRP.

Lorsque les membres du noyau calculent une nouvelle clé TEK pour le cluster, ils la distribuent aux membres de leur périphérie chiffrée avec la clé KEK correspondante.

6.8.3 Gestion de clés inter-clusters

Le chiffrement du trafic dans le groupe ne se fait pas avec une même clé. Les différents cluster-heads assurent la communication entre les clusters. Chaque cluster-head calcule une clé de chiffrement du trafic avec le cluster-head de chaque cluster adjacent en utilisant le protocole d'échange de clés de Diffie-Hellman [Diffie 76], comme suit :

- Chaque cluster-head h génère un secret r_h et envoie sa version publique $g^{r_h} \, mod \, p$ dans un message $Contact - Adj - CH$ aux passerelles rattachées aux feuilles de son sous-arbres.
- Lorsque le message atteint une passerelle, il est transmis à la passerelle correspondante du cluster adjacent si la relation entre les deux passerelles est une relation TT ou une relation PT.
- Le message est ensuite envoyé au cluster-head du cluster adjacent.
- Chaque paire de cluster-heads x et y peut alors calculer une clé TEK, selon le protocole d'accord de clé de Diffie-Hellman,

$$K = g^{r_x r_y} \, mod \, p \tag{6.4}$$

6.8.4 Le transfert de données

Afin d'assurer la communication de données au sein d'une session sécurisée, les membres du groupe suivent les règles suivantes :

- Les messages $Cluster - diffuse$, $CH - à - CH$, etc. ne suivent que les relations TT ou PT.
- Pour diffuser les données, un membre les chiffre avec la clé TEK de son cluster et les transmet à ses voisins dans un message $Cluster - Diffuse$.
- Lors de la réception du message $Cluster - Diffuse$ pour la première fois, chaque nœud le diffuse à ses voisins. Si le nœud est un cluster-head, il déchiffre et re-chiffre avec la clé TEK qu'il partage avec chaque cluster-head adjacent et la transmet à chacun d'eux dans un message $CH - to - CH$. Un message $CH - to - CH$ traverse le sous-arbre du cluster jusqu'à atteindre une passerelle.
- Une passerelle, qui reçoit un message $CH - to - CH$ provenant d'un autre cluster pour la première fois, l'envoie à son cluster-head.
- Lorsqu'un cluster-head reçoit un message $CH - to - CH$, il le déchiffre, re-chiffre avec la clé TEK de son cluster et le transmet à ses voisins dans un message $Cluster - Diffuse$. Il le re-chiffre également avec la clé de chiffrement de trafic qu'il partage avec chaque cluster-head adjacent, sauf celui d'où il reçoit le message, et le transmet à chacun d'eux dans un message $CH - to - CH$.

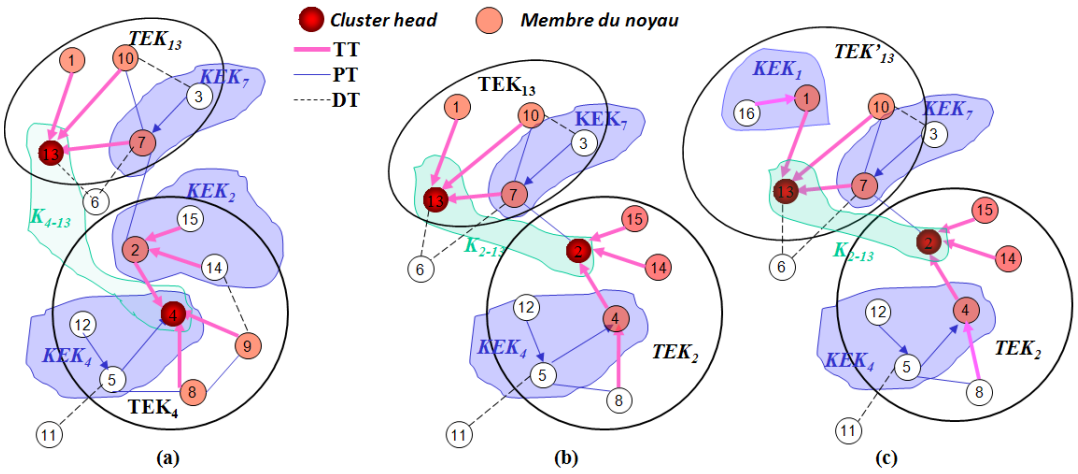

Figure 6.5 : *Exemple de calcul des clés.*

6.9 Exemple de gestion de clés

Dans cette section, nous illustrons le schema du calcul/re-calcul de la clé ainsi que la maintenance de la topologie sur l'exemple présenté dans la Figure 6.4(d). Notez que les deux protocoles : le clustering et l'établissement de la clé s'exécutent simultanément. Dans l'exemple, nous avons deux clusters c_{13} et c_4. Ainsi, nous avons besoin de deux clés $TEKs$, une clé TEK par cluster : TEK_{13} et TEK_4. Comme l'illustre la Figure 6.5(a), le cluster c_{13} ne contient qu'un seul membre dans sa périphérie : le nœud 3. Ce dernier est attaché au membre 7 du noyau et partage avec lui une clé KEK : KEK_7. Le cluster 4 contient quatre membres de la périphérie : 5, 12, 14 et 15. Les nœud 5 et 12 sont attachés au cluster-head et partagent avec lui une clé KEK : KEK_4. Les nœuds 14 et 15 sont attachés aux membres 2 du noyau et partagent avec lui une clé KEK : KEK_2. Enfin, les deux cluster-heads se partagent une clé commune : K_{4-13}. Pour calculer ces clés, les nœuds opèrent comme suit :

- Le nœud 13, le cluster-head du cluster c_{13}, génère deux secrets : une contribution pour la clé TEK_{13} et une contribution pour la clé K_{4-13}. Le nœud 4, le cluster-head du cluster c_4, génère trois secrets : une contribution pour la clé TEK_4, une contribution pour la clé KEK_4 et une contribution pour la clé K_{4-13}.
- Chacun des nœuds 1, 7 et 10 génère sa contribution pour la clé TEK_{13}. De même, les nœuds 2, 8 et 9 génèrent leurs contributions pour la clé TEK_4. Le nœud 2 (respectivement 7) génère également une contribution pour la clé KEK_2 (respectivement KEK_7).
- Les membres de la périphérie 14 et 15 (respectivement 5 et 12) génèrent leur contribution pour la clé KEK_2 (respectivement KEK_4).

– Le membre 3 de la périphérie génère sa contribution pour la clé KEK_7.

Supposons maintenant que le nœud 9 quitte le groupe (voir Figure 6.5(b)). Le nœud 9 envoie un message $LEAVE$ au cluster-head. TEK_4 doit être mise à jour. Toutefois, avant de lancer cette opération, le nœud 4 met à jour le champ $TT-edge$ de son message *hello* : il a perdu une relation TT et ainsi il n'est plus le nœud ayant le plus grand nombre de relations TT en comparaison avec ses voisins. Le nœud 2 est le nouveau cluster-head. Il génère une contribution à la nouvelle clé TEK : TEK_2. Chaque membre du noyau, à savoir les nœuds 4, 14 et 15, génère une contribution pour la clé TEK_2 et l'envoie au nouveau cluster-head. Le nœud 4 envoie la clé TEK aux nouveaux nœuds 5 et 12 chiffrée avec la clé TEK_4. Le nœud 2 envoie un message $Contact - Adj - CH$ et calcule une clé commune K_{2-13} avec le nœud 13. La Figure 6.5(c) illustre l'arrivée d'un nouveau membre du groupe, le nœud 16, qui partage une relation TT avec le nœud 1. Lors de la réception du message $JOIN$ du nœud 16, le nœud 1 envoie un $NEW - TEK - RQ$ au cluster-head. Ce dernier génère une nouvelle contribution et exécute le second tour de TRP. Tous les membres du noyau peuvent alors calculer la nouvelle clé TEK et la distribuer aux membres de la périphérie. Le nœud 1 calcule également une clé KEK : KEK_1 avec le nœud 16. Il chiffre la nouvelle clé TEK avec KEK_1 et l'envoie au nœud 16.

6.10 Analyse de complexité

Dans cette section, nous analysons les coûts de communication et de stockage de notre protocole et nous les comparons avec les coûts de certains protocoles de gestion de clés de groupe existants dans la littérature. Le Tableau 6.6 résume les résultats de la comparaison. Dans ce tableau, nous utilisons les notations suivantes :

n : nombre des membres du groupe.

g : nombre de cluster-heads.

c : nombre moyen des membres d'un cluster.

x(relayé) : x est le nombre de messages relayés, i.e. qui ont dûs être transmis par des nœuds intermédiaires pour atteindre la destination. En fait, une communication directe n'existe qu'entre un nœud et ses voisins à 1 saut.

CDS : abréviation de l'ensemble dominant connecté.

Tous les protocoles décrits dans le tableau 6.6 construisent des clusters à l'exception de $\mu TGDH$ [Manulis 05] qui utilise un arbre. Dans cet arbre le re-calcul de la clé nécessite $log_2 n$ messages. Les membres du groupe sont les feuilles de l'arbre. Les autres nœuds de l'arbre sont les clés $KEKs$ calculées par deux nœuds à chaque étape. En outre, tous les membres du groupe doivent connaître la structure de l'arbre [Kim 04]. Outre le protocole de [Li 02], les approches à base de clusters représentées dans le Tableau 6.6 utilisent des messages *hello* afin de réaliser le clustering.

Ainsi, ces approches ne génèrent pas de nouveaux trafics. Toutefois, elles étendent les messages *hello* périodiques avec de nouveaux champs. Pour économiser la bande passante, ces champs ne peuvent être inclus que s'ils sont modifiés. Ainsi, le coût en communication est principalement dû aux communications entre cluster-heads. Ces communications doivent trouver une route entre chaque paire de cluster-heads ou utiliser des nœuds passerelles. Le clustering dans [Li 02] construit un CDS et utilise un ordre entre les cluster-heads afin d'exécuter le protocole GDH_2 [Steiner 00]. Les auteurs montrent que chacune de ces deux tâches est $O(n)$ [Li 02]. $SGCP$, LID et notre protocole partagent une propriété commune : l'auto-organisation. Cela signifie que le processus de clustering est entièrement distribué et n'a pas besoin d'être initié par un nœud particulier comme dans les protocoles [Li 02] [Bhaskar 05]. Notre protocole est moins performant par rapport aux autres protocoles au niveau du nombre de messages transmis. En fait, chaque groupe est organisé selon un arbre et toutes les communications au sein du cluster sont transmises dans l'arbre pour atteindre tous les membres de l'arbre.

6.11 Evaluation

Nous avons simulé notre algorithme de clustering pour évaluer ses performances et le comparer avec les performances d'autres algorithmes de clustering. Dans cette section, nous donnons un aperçu de notre modèle de simulation et les résultats que nous avons obtenus. Nous avons simulé notre approche de clustering au sein de la plate-forme de simulation de réseau NS_2 [Fall 03]. Notre simulation modélise un $MANET$ au maximum de 100 nœuds se déplaçant au hasard dans une zone de 1000x1000 m^2 conformément au modèle *Random Waypoint Model* [Bettstetter 04]. Chaque nœud est équipé d'un émetteur-récepteur radio capable de transmettre jusqu'à 250 m. Nous utilisons 802.11 comme protocole de la couche MAC dans nos expériences. Nous avons évalué la stabilité de notre système de clustering par l'étude de la variation du nombre de clusters qu'il engendre. Nous considérons ce paramètre dans différents scénarios de mobilité et de connectivité des nœuds. Pour la mobilité, nous considérons trois scénarios :

- Une mobilité faible qui correspond à une vitesse de nœud variant entre $5m/s$ et $10m/s$.
- Une mobilité moyenne qui correspond à une vitesse de nœud variant entre $10m/s$ et $15m/s$.
- Une mobilité importante qui correspond à une vitesse de nœud variant entre $15m/s$ et $20m/s$.

Pour mesurer la mobilité, nous utilisons l'approche introduite dans [Larson 98] où la mobilité est la variation moyenne de la distance entre tous les nœuds sur une

	[Li 02]	$\mu TGDH$ [Manulis 05]	$SGCP$ [Tseng 07a]	Notre protocole	LID [Baker 81]
Type de la topologie	Clusters avec des CDS +ordre	Arbres	Clusters	Clusters	Clusters
Complexité de la construction de la topologie	$O(n)$	$O(n)$+ communication entre cluster-heads	$O(n)$+ communication entre cluster-heads	$O(n)$+ communication entre cluster-heads	$O(n)$+ communication entre cluster-heads
Maintenance de la topologie	Doit être reconstruit	Doit être reconstruit	Auto-organisé	Auto-organisé	Auto-organisé
Augmentation des messages *hello*	Non	Non	Degré, liens défaillants, énergie	$TT-edge$, CH, core, $tv(i,j)$	liste des voisins
Couplage de clustering et re-calcul de la clé	Non	Oui	Non	Oui	-
Nombre de messages pour le calcul de la clé	g(relayé)+ 1+g+c	$log_2 n$ (relayé)	$g+c+g$ (relayé)	$2g$(relayé)+2	-
Stockage Max	M_i :1M_n :$c+2+g$	$log_2 n$	M_i :1 cluster-head :$1+g$	M_i :2 cluster-head : $2+g$	-
Nombre de messages d'arrivée/départ	1(relayé)+1 diffusion+c	$log_2 n$ (relayé)	c	$(g+1)$ (relayé)+2	-

Figure 6.6 : *Analyse de complexité*

période de temps (période de simulation dans notre cas).

$A_x\,(t)$ est la distance moyenne du nœud x aux autres nœuds à l'instant t et M_x définit la mobilité moyenne du nœud x par rapport aux autres nœuds durant tout le temps de la simulation. M_x est égal à la variation de la distance moyenne $A_x(t)$ d'un nœud x pendant l'intervalle de temps $T - \triangle t$ (T : le temps de simulation et $\triangle t$: la granularité). Enfin, la mobilité pour l'ensemble du scénario est la somme de la mobilité pour tous les nœuds divisée par le nombre de nœuds. Le facteur de la mobilité décrit la vitesse moyenne de la variation de la distance entre les nœuds. L'unité de la mobilité est m/s. Lorsque la mobilité augmente cela signifie que la topologie du réseau est plus dynamique et les connexions/déconnexions sont fréquentes.

$$A_x\,(t) = \frac{1}{n-1}\sum_{i=1}^{n} dist\,(n_x - n_i) \tag{6.5}$$

$$M_x = \frac{1}{T - \triangle t}\sum_{t=0}^{T-\triangle t} |A_x\,(t) - A_x\,(t + \triangle t)| \tag{6.6}$$

$$M = \frac{1}{n}\sum_{i=1}^{n} M_i \tag{6.7}$$

Le taux de connectivité ($crate$) est estimé par le nombre moyen de voisins comme suit :

$$crate = \frac{1}{n^2(T - \triangle t)}\sum_{i=0}^{n}\sum_{t=0}^{T-\triangle t} |neighbors(i)_t - neighbors(i)_{T-\triangle t}| \tag{6.8}$$

avec $neighbors(i)$ est le nombre des voisins du nœud i, n est le nombre de nœuds et T est la période de simulation.

Tout d'abord, nous nous sommes intéressés à la validation de la confiance en tant que critère de clustering en évaluant la variation du nombre de clusters en fonction de la confiance. Nous étudions la variation du nombre de clusters en fonction de la valeur moyenne de la confiance calculée par chaque nœud et obtenue comme suit :

$$avg - tv(i) = \frac{1}{|neighbors(i)|}\sum_{j \in neighbors(i)} avg - tv(i,j) \tag{6.9}$$

$$avg - tv = \frac{1}{n}\sum_{i=1}^{n} avg - tv(i) = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_{j \in neighbors(i)} avg - tv(i,j)}{|neighbors(i)|} \tag{6.10}$$

avec $neighbors(i)$ est le nombre des voisins du nœud i, n est le nombre de nœuds et $avg - tv(i)$ est la moyenne de $tv(i,j)$ du nœud i. Rappelons que $tv(i,j)$ est la

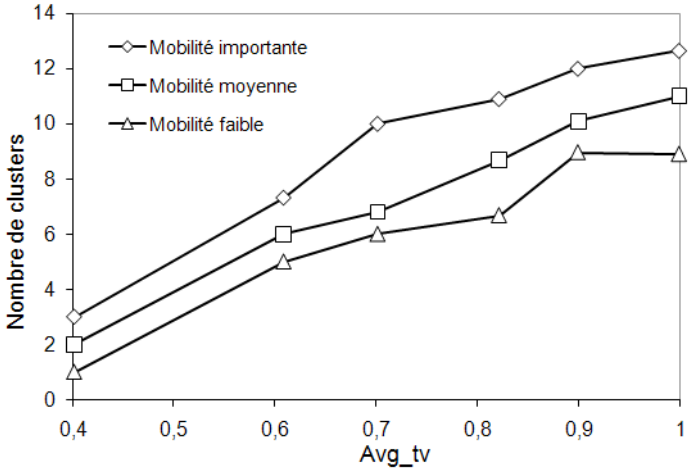

Figure 6.7 : *Variation du nombre de clusters en fonction de la confiance.*

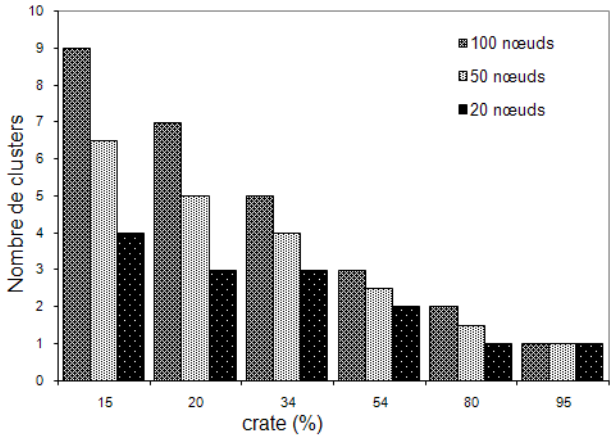

Figure 6.8 : *Variation du nombre de clusters en fonction de la connectivité.*

valeur de la confiance qu'un nœud i affecte au nœud j.

La Figure 6.7 montre que lorsque les valeurs de la confiance augmentent le nombre de clusters augmente également. Cela peut s'expliquer par le fait qu'il y a plus de relations TT que de relations PT si les valeurs de la confiance sont élevées. Selon notre algorithme de clustering, le nombre de clusters dépend du nombre de relations TT. La Figure 6.7 montre également que le nombre de clusters suit la même tendance dans les trois scénarios de mobilité, même s'il est important lorsque la mobilité est élevée. La Figure 6.8 illustre la variation du nombre de clusters en fonction de la connectivité dans trois scénarios de réseau : 100 nœuds, 50 nœuds et 25 nœuds. Elle montre que plus la connectivité est élevée plus le nombre de clusters est faible. Elle montre également que lorsque la connectivité est supérieure à 50% le nombre de clusters est indépendant du nombre de nœuds dans le réseau.

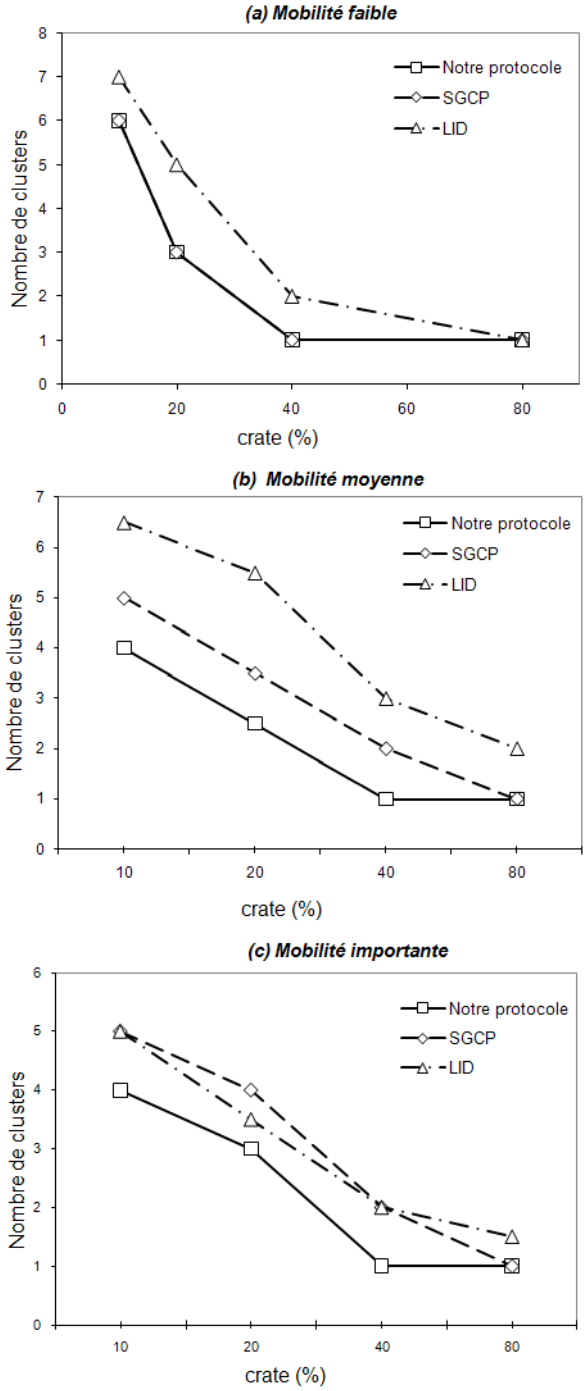

Figure 6.9 : *Comparaisons du nombre de clusters de notre protocole avec LID, SGCP.*

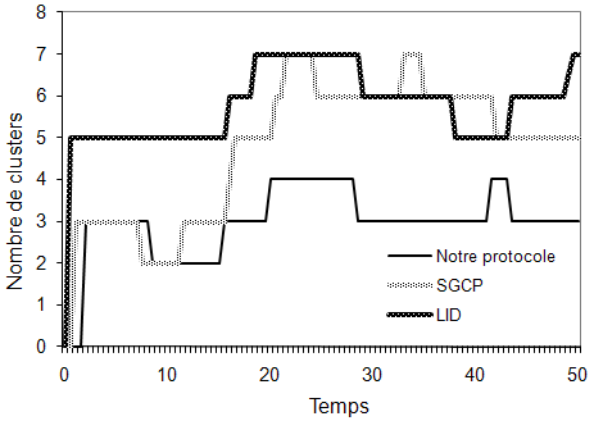

Figure 6.10 : *Evolution du nombre de clusters.*

Nous avons comparé notre algorithme de clustering avec deux schémas existants : $SGCP$ [Tseng 07a] et LID [Baker 81] [Baker 81a]. $SGCP$ (*Secure Communication Group Protocol*) [Tseng 07a] et LID (*Lowest IDentifier*) sont décrits dans la sous-section 6.2. LID est l'un des protocoles les plus connus de clustering. LID est habituellement utilisé comme un protocole de référence pour l'évaluation des performances des algorithmes de clustering. Nous avons simulé les trois protocoles dans les trois scénarios de mobilité décrites précédemment : mobilité faible, mobilité moyenne et mobilité importante. Nous avons considéré différents taux de connectivité et avons mesuré le nombre de clusters realisé par chacun des trois protocoles. La Figure 6.9 montre que notre algorithme donne de bons résultats. Il a le même nombre de clusters que LID dans un scénario de mobilité faible et a le plus faible nombre de clusters dans les scénarios de mobilité moyenne et importante. Nous avons également étudié la variation du nombre de clusters des trois protocoles au cours du temps. Pour cela, nous également exécuté les trois protocoles dans les mêmes conditions de mobilité et de connectivité pendant $50s$. La Figure 6.10 représente nos résultats. Elle montre clairement que la variation du nombre de clusters n'est pas importante et est stable avec LID et notre protocole. Ce n'est pas le cas pour $SGCP$ où le nombre de clusters augmente de façon considérable après $20s$ de simulation. Cela montre que par rapport à $SGCP$ et LID, notre protocole de clustering est stable : il génère un nombre raisonnable de clusters dans tous les scénarios de mobilité.

6.12 Conclusion

Mis à part les problèmes connus dans les réseaux statiques, les réseaux mobiles ad hoc se caractérisent principalement par la mobilité et par l'absence d'infrastructures de communication. Notre algorithme de clustering permet de gérer la dynamique

du réseau, il se base sur la construction d'une topologie permettant de minimiser la mobilité du réseau, optimiser le passage à l'échelle, faciliter et sécuriser les protocoles de communication. Notre proposition se base sur la définition d'un modèle de confiance dynamique et distribué pour les réseaux mobiles ad hoc. Notre modèle de confiance constitue un mécanisme flexible permettant le suivi continu des changements qui interviennent au niveau de la confiance de chaque nœud. Les métriques utilisées pour la construction de la topologie sont fondées essentiellement sur cette notion de confiance. Notre approche consiste à diviser le réseau en clusters organisés sous forme de sous-arbres liés et supervisés par des cluster-heads. Par conséquent, l'ajout ou la suppression d'un membre n'affecte que le cluster auquel il appartient. Nous avons donné par la suite un exemple d'application à la gestion de clés de groupe. La gestion de clé est une brique de base dans toute application sécurisée. La sécurité de notre protocole est renforcée par son critère de clustering qui surveille en permanence les relations de confiance entre les nœuds et expulse les nœuds malveillants de la session de diffusion.

Chapitre 7

Conclusion et perspectives

Résumé de nos contributions

Dans ce travail, nous avons présenté quelques contributions dans le domaine de la théorie des graphes et celui de l'algorithmique dans le cadre de l'auto-stabilisation. Nous avons structuré notre travail en deux parties. Dans la première partie, nous nous sommes intéressés à l'étude du paramètre de coloration d'arêtes ℓ-distance. Nous avons mené une étude combinatoire et algorithmique à travers l'étude du paramètre pour quelques classes de graphes. Nous avons donné des valeurs exactes du paramètre pour les chaînes, les hypercubes à d dimensions, les grilles à deux dimensions et les arbres k-aires complets. A partir du résultat des arbres k-aires complets nous avons pu déduire une borne supérieure de l'indice ℓ-chromatique pour des arbres quelconques. Nous avons aussi étudié ce paramètre pour des classes de graphes plus complexes telles que le cas des graphes puissances. Nous avons commencé par les graphes puissances des chaînes, nous avons trouvé une valeur exacte de l'indice ℓ-chromatique. L'étude que nous avons menée sur les graphes puissances des cycles nous a permis de trouver une borne supérieure de l'indice ℓ-chromatique. Pour certaines valeurs particulières du nombre de sommets n, nous avons réduit la valeur de cette borne supérieure. De plus, lorsque n est suffisamment grand, nous avons trouvé une valeur exacte de l'indice ℓ-chromatique. Nous avons donné un algorithme de coloration d'arêtes ℓ-distance pour chaque classe de graphes du cas général ainsi que des cas particuliers que nous avons étudiés.

La deuxième partie de la thèse a été consacrée à la proposition de deux algorithmes de graphes auto-stabilisants. L'auto-stabilisation est une approche particulièrement intéressante pour la conception d'algorithmes résistants aux pannes. Ces algorithmes tolérants aux pannes sont des directions de recherche très intéressantes, ce qui nous a motivé en premier lieu de travailler sur les aspects auto-stabilisants de coloration d'arêtes. De plus, nous avons remarqué qu'aucun des résultats dans littérature ne donne la valeur de la borne supérieure donnée par Vizing. Nous avons

donc proposé un algorithme polynomial auto-stabilisant de coloration d'arêtes en se basant sur le résultat de Vizing. Notre coloration est assurée avec au plus $(\Delta+1)$ couleurs et une complexité en temps de $O(m(\Delta + n))$, où Δ, n et m désignent respectivement le degré maximum, le nombre de sommets et le nombre d'arêtes dans le graphe.

Nous avons aussi proposé un algorithme auto-stabilisant de clustering dans les MANETs. Notre algorithme de clustering auto-stabilisant est très intéressant dans le cadre des MANETs. En effet, la propriété de l'auto-stabilisation lui permet de s'adapter aux changements fréquents de la topologie du réseau. De plus, notre algorithme utilise un partitionnement en clusters basé sur les relations de confiance qui existent entre les membres d'un groupe, ceci oriente son application à des domaines sécuritaires. En effet, notre algorithme présente l'avantage d'ignorer la présence des nœuds malveillants. En effet, les nœuds honnêtes coopèrent étroitement. Ils ne communiquent pas les messages de clustering aux nœuds méfiants et ignorent tous les messages de clustering provenant de ces nœuds. L'algorithme de clustering s'exécute en continu et réajuste les clusters en fonction des relations de confiance entre les nœuds. Les relations de confiance évoluent dans le temps en fonction des interactions entre les nœuds. La sécurité de notre protocole est donc renforcée par son critère de clustering qui surveille en permanence les relations de confiance et expulse les nœuds malveillants de la session de diffusion.

Nous avons choisi d'appliquer notre algorithme de clustering à la gestion de clés de groupe. La structure que nous avons défini a deux principaux avantages sur la gestion de clés : c'est une structure robuste. En fait, chaque groupe peut être considéré comme une zone de distribution de clés. La distribution de clés n'est pas effectuée par un seul nœud, au contraire, tous les membres du noyau du cluster ont cette responsabilité. Cette redondance renforce la tolérance aux pannes de la distribution des clés au sein du cluster, puisque le cluster continue de fonctionner tant qu'il y a au moins un nœud qui fonctionne dans le noyau. Deuxièmement, comme les principaux membres du noyau sont des voisins, le protocole d'accord de clés utilisé pour générer la clé du cluster est d'un faible coût et se termine dans un délai raisonnable. Cela contribue à la mise en place rapide de la clé du cluster.

Nous avons comparé notre approche de clustering avec deux schémas existants. A partir des simulations faites, nous avons pu constater que notre protocole génère un nombre raisonnable de clusters dans tous les scénarios de mobilité. Ceci est important dans notre exemple d'application à la gestion de clés parce que le nombre de clusters a un impact direct sur le nombre d'opérations de chiffrement/déchiffrement. En outre, notre protocole ne change pas de manière significative le nombre de clusters au cours du temps, ce qui est aussi très important pour les coûts de chiffrement/déchiffrement de la gestion de clés.

Perspectives de recherche

Nos différents travaux de recherche dans cette thèse, que ce soit les études combinatoires et algorithmiques de la première partie ou les algorithmes auto-stabilisants de la deuxième partie, nous ont amené à envisager de nombreuses perspectives de recherche qui semblent intéressantes pour chaque type de problème étudié. Nous citons quelques unes de ces perspectives dans ce qui suit.

Concernant le paramètre de coloration d'arêtes ℓ-distance, il serait intéressant d'étudier le paramètre pour d'autres classes de graphes telles que les graphes réguliers, les graphes distants, les graphes circulants.

L'étude algorithmique proposée dans cette thèse peut être aussi étendue à des environnements distribués en vue d'éventuelles applications. Ainsi, des algorithmes distribués ou auto-stabilisants peuvent être proposés pour construire une coloration d'arêtes ℓ-distance. Il serait intéressant d'étendre l'algorithme auto-stabilisant de coloration propre d'arêtes que nous avons proposé dans le chapitre 5 pour la coloration d'arêtes ℓ-distance. Nous pouvons nous baser sur les résultats de Goddard et Hedetniemi concernant les algorithmes auto-stabilisants utilisant les informations du k-voisinage [Goddard 08].

En ce qui concerne notre notre algorithme auto-stabilisant de clustering, d'autres applications peuvent être éventuellement envisagées. Par exemple, il est possible d'appliquer notre algorithme de clustering sur d'autres protocoles de communication afin d'évaluer ses performances.

Bibliographie

[Alzoubi 02] K. Alzoubi, P. J. Wan, F. Ophir. *New Distributed Algorithm for Connected Dominating Set in Wireless Ad hoc Networks*, Proceedings of the 35th Hawaii International Conference on System Sciences (2002)

[Amis 00] A.D. Amis, R. Prakash, T.H.P. Vuong, D.T. Huynh, *Max Min D-cluster formation in wireless ad hoc networks*, in : Infocom (2000)

[Anderson 08] R. Anderson. *Security Engineering : A Guide to Building Dependable Distributed Systems* Wiley (2008)

[Appel 77] K. Appel, W. Haken, J. Koch. *Every Planar Map is Four Colorable*, Illinois Journal of Mathematics 21 : 439-567 (1977)

[Augot 07] D. Augot, R. Bhaskar, V. Issarny, D. Sacchetti, A three round authenticated group key agreement protocol for ad hoc networks, Pervasive and Mobile Computing 3 (1) 36-52 (2007)

[Awerbuch 95] B. Awerbuch, B. Patt-Shamir, G. Varghese, *Self Stabilizing End-to-End Communication*, Journal of High Speed Networks, (1995)

[Baker 81] D. Baker, A. Ephremides. *The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm*, Communications, IEEE Transactions, vol. 29-11,, pp. 1694-1701 (1981)

[Baker 81a] D.J. Baker, A. Ephremides, *A distributed algorithm for organizing mobile radio telecommunication networks*, in : Proceedings of the 2nd International Conference on Distributed Computer Systems, pp. 476-483 (1981)

[Balister 02] P.N. Balister, E. Gyori, J. Lehel, R.H. Schelp, *Adjacent vertex distinguishing edge-colorings*, J. Graph Theory (2002)

[Balister 03] P.N. Balister, O.M. Riordan, R.H. Schelp, *Vertex-distinguishing edge colorings of graphs*, J. Graph Theory 42 95-109 (2003)

[Barrett 06] C. L. Barrett, V. S. Anil Kumar, M. V. Marathe, S. Thite, G. Istrate. *Strong Edge Coloring for Channel Assignment in*

Wireless Radio Networks, Pervasive Computing and Communications Workshops, IEEE International Conference, 106-110 (2006)

[Basagni 99] S. Basagni. *Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks*. In Proceedings of the IEEE 50th vehicular technology conference (1999)

[Basu 01] P. Basu, N. Khan, T.D.C. Little, *A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks*, Distributed Computing Systems Workshops (2001)

[Beineke 73] L. W. Beineke, R. J.Wilson. *On the edge-chromatic number of a graph*, In Discrete Mathematics, 15-20, 5 (1973)

[Berge 57] C. Berge. *Théorie des graphes et ses Applications*, Ed. Dunod (1957)

[Berge 70] C. Berge. *Graphes et hypergraphes*, Ed. Dunod (1970)

[Bettstetter 04] C. Bettstetter, H. Hartenstein, X. Pérez-Costa Stochastic properties of the random waypoint mobility model, in : ACM/Kluwer Wireless Networks, (2004)

[Bhaskar 05] R. Bhaskar, D. Augot, V. Issarny, D. Sacchetti, *An efficient group key agreement protocol for ad hoc networks*, in : IEEE Workshop on Trust, Security and Privacy in Ubiquitous Computing, Taormina, Italy, 12-16 (2005)

[Bouassida 05] M. S. Bouassida, I. Chrisment et O. Festor. *Efficient Clustering for Multicast Key Distribution in MANETs*. Networking'05 (2005)

[Brandstad 96] A. Brandstäd, V. D. Chepoi, F. F. Dragan, *Perfect elimination orderings of chordal powers of graphs*, Discrete Mathematics 158, 273-278 (1996)

[Brooks 41] R. L. Brooks. *On colouring the nodes of a network*. Mathematical Proceedings of the Cambridge Philosophical Society, 37 , pp 194-197 (1941)

[Burris 93] A. C. Burris. *Vertex-Distinguishing edge colourings*. PhD Dissertation, Memphis state university, (1993)

[Butelle 95] F. Butelle, C. Lavault, M. Bui, *A Uniform Self-Stabilizing Minimum Diameter Spanning Tree Algorithm*, In Proceedings of 9th International Workshop WDAG'95 in Lecture Notes in Computer Science, vol. 972, Springer, , pp.257-272 (1995)

[Cairnie 97] N. Cairnie and K. Edwards. *Some results on the achromatic number*. Journal of Graph Theory, 26(3) :129-136, (1997)

[Cardei 04] I. Cardei, S. Varadarajan, A. Pavan, L. Graba, M. Cardei, M. Min. *Resource management for ad hoc wireless networks with cluster organization*. Cluster Computing, 7(1) :91 103 (2004)

[Chartrand 08] G. Chartrand, P. Zhang. *Chromatic Graph Theory*, DISCR. MATH. & ITS APPL.(2008)

[Chatterjee 00] M. Chatterjee, S.K. Das, D. Turgut. *A weight based distributed clustering algorithm for mobile Ad hoc networks*. In International Conference on High Performance Computing, pages 511-521, Bangalore, India, IEEE, ACM (2000)

[Chen 02a] G. Chen, F. Garcia, J. Solano, and I. Stojmenovic. *Connectivity-based k- hop clustering in wireless networks*. In HICSS (2002)

[Chen 04] Y.P. Chen, A.L. Liestman, J. Liu. *Clustering Algorithms for Ad Hoc Wireless Networks*, Ad Hoc and Sensor Networks. Nova Science Publishers, (2004)

[Chiang 96] C. C. Chiang, H. K.Wu, W. Liu, M. Gerla. *Routing in clustered multihop, mobile wireless networks with fading channel*. In The IEEE SingaporeInternational Conferenceon Networks, pages 197- 211. IEEE, (1996)

[Christen 79] C. A. Christen and S. M. Selkow. *Some perfect coloring properties of graphs*. Journal of Combinatorial Theory B27, pages 49-59, (1979)

[Cook 71] S.A. Cook, .*The Complexity of Theorem-Proving Procedures*. Proceedings of the Third Annual ACM Symposium on Theory of Computing : 151-158. Consulté le 2007-06-11 (1971)

[Cristian 91] F. Cristian. *Understanding fault-tolerent distributed system*, COMMUNICATIONS OF THE ACM/February /Vo1,34, No.2 (1991)

[Dahlhaus 87] E. Dahlhaus, P. Duchet, *On strongly chordal graphs*, Ars Combinatoria 24B23-30 (1987)

[Dekar 08] L. Dekar, H. Kheddouci, *A Cluster Based Mobility Prediction Scheme for Ad hoc networks*, Ad hoc Networks, Volume 6, Issue 2, Pages 168-194, (2008).

[Diffie 76] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory (1976)

[Dijkstra 74] E. W.Dijkstra, *Self stabilizing systems in spite of distributed control*, Communications of the ACM, vol. 17, n o 11, p. 643-644 (1974)

[Dijkstra 86] E. W.Dijkstra, *A belated proof of self stabilization*, Distrubted Computing 1 :5-6 (1986)

[Drira 06] K. Drira, H. Kheddouci and N. Tabbane. *Virtual Dynamic Topology for Routing in Mobile Ad Hoc Networks*. Proceedings of the International Conference on Late advances in Networks, Paris, France (ICLAN'2006), pp 129-134 (2006)

[Drira 10] K. Drira and H. Kheddouci, *A New Clustering Algorithm for MANETs*, 14th International Telecommunications Network Strategy and Planning Symposium (Networks'10), Warsaw-Poland, September 27-30 (2010)

[Dolev 93] S. Dolev, A. Israeli, S. Moran. *Self-stabilization of dynamic systems assuming only read/write atomicity*, Distrib. Comput. 7, pp. 3-16 (1993)

[Dolev 97] S. Dolev, A. Israeli, S. Moran. *Resource bounds for self-stabilizing message-driven protocols*, SIAM Journal on Computing, no. 26, pp.273-290 (1997)

[Dolev 00] S. Dolev. *Self-stabilization*, MIT Press (2000)

[Ducourthial 00] B. Ducourthial, S. Tixeuil. *Self-stabilization with path algebra*, SIROCCO, 95-109 (2000)

[Edwards 99] K. Edwards. *The harmonious chromatic number of complete r-ary trees*. Discrete Mathematics, 203 :83-99, (1999)

[Effantin 05a] B. Effantin and H. Kheddouci. *Exact values for the b-chromatic number of power complete k-ary tree*. Journal of Discrete Mathematical Sciences and Cryptography, vol. 8(1), pp. 117-129 (2005)

[Effantin 03] B. Effantin and H. Kheddouci. *The b-chromatic number of some power graphs*. Discrete Mathematics and Theoretical Computer Science, 6 :45-54, (2003)

[Effantin 05b] B. Effantin, *The b-chromatic number of power graphs of complete caterpillars*, Journal of Discrete Mathematical Sciences and Cryptography, vol. 8(3), pp. 483-502 (2005)

[Effantin 07] B. Effantin and H. Kheddouci. *Grundy number of graphs*. Discussiones Mathematicae Graph Theory, 27(1) :5-18, (2007)

[Elghazel 08] H. Elghazel, H. Kheddouci, V. Deslandres, A. Dussauchoy. *A Graph b-coloring Framework for Data Clustering*, Journal of Mathematical Modelling and Algorithms, Ĭ Springer Verlag, Volume 7, Number 4, pages 389-423. (2008)

[Erciyes 07] K. Erciyes, O. Dagdeviren, D. Cokuslu, D. Ozsoyeller. *Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks-survey*. Appl. Comput. Math. 6(2), 162-180 (2007)

[Erdos 77] P. Erdös, R.J. Wilson. *On the chromatic index of almost all graphs*, J. Combin. Th. pp. 255-257, 23 B (1977).

[Fall 03] K. Fall, K. Varadhan, The ns Manual (formerly ns Notes and Documentation, The VINT Project (2003), available at http ://www.isi.edu/nsnam/ns/doc/.

[Farber 86] M. Farber, G. Hahn, P. Hell, and D. Miller. *Concerning the achromatic number of graphs*. Journal of Combinatorial Theory, B40(1) :21-39 (1986)

[Fernandess 02] Y. Fernandess. *K-clustering in wireless ad hoc networks*, POMC '02 : Proceedings of the second ACM international workshop on Principles of mobile computing (2002)

[Fertin 03] G. Fertin, E. Godard, A. Raspaud, *Acyclic and k-distance coloring of the grid*, Inform. Process. Lett. 87 (1) 51-58 (2003)

[Flatebo 94] M Flatebo, AK Datta, and S Ghosh. *Self-stabilization in distributed systems*. In Readings in Distributed Computing Systems, pages 100-114, TL Casavant and M Singal, Editors (1994)

[Fiorini 77] S. Fiorini and R.J. Wilson, *Edge-Colourings of Graphs*, Research notes in. Mathematics, Pitman, (1977)

[Gandham 05] S. Gandham, M. Dawande, R. Prakash. *Link scheduling in sensor networks : distributed edge coloring revisited*, in Proceedings of the IEEE 24th Annual Joint Conference of Computer and Communications Societies (INFOCOM'05), vol. 4, pp. 2492-2501 (2005)

[Gerla 95] M. Gerla, J.T. Tsai. *Multicluster, mobile, multimedia radio network*, Wireless Networks, vol. 1(3), pp. 255-265 (1995)

[Germain] C. Germain and H. Kheddouci. *Grundy numbers of powers of graphs*. Discrete Mathematics, (to appear)

[Ghosh 93] S. Ghosh, M. H. Karaata. *A self-stabilizing algorithm for coloring planar graphs*. Distributed Computing, 7(1) :55-59, (1993)

[Ghosh 97] S. Ghosh, A. Gupta, S. V. Pemmaraju. *Fault-containing network protocols*, Proceedings of 12th ACM Symposium on Applied Computing, (1997)

[Goddard 08] W. Goddard, S. T. Hedetniemi, D. P. Jacobs, V. Trevisan : *Distance- k knowledge in self-stabilizing algorithms*. Theor. Comput. Sci. 399(1-2) : 118-127 (2008)

[Gondran 86] M. Gondran , M. Minoux. *Graphes et algorithmes*, 2e édition, Eyrolles (1986)

[Gonthier 08] G. Gonthier. *Formal Proof–The Four-Color Theorem*, Notices of the American Mathematical Society 55 : 11 (2008)

[Gouda 91] M. G. Gouda, N. Multari, *Self-Stabilizing Communication Protocols*, IEEE Transactions on Computers, vol.40, no.4,. pp. 448-458 (1991)

[Gouda 95] M.G. Gouda. *The triumph and tribulation of system stabilization*. In WDAG95 Distributed Algorithms 9th International Workshop Proceedings, Springer LNCS :972, pages 1-18, (1995)

[Gradinariu 00] M. Gradinariu and S. Tixeuil. *Self-stabilizing vertex coloration and arbitrary graphs*. In Proceedings of the 4th International Conference on Principles of Distributed Systems, OPODIS 2000, Paris, France, December 20-22, 2000, pages 55-70, (2000)

[Guellati 10] N. Guellati, H. Kheddouci. A Survey on Self-Stabilizing Algorithms for Independence, Domination, Coloring, and Matching in Graphs. Journal of Parallel and Distributed Computing (2010)

[Haddad 07] M. Haddad and H. Kheddouci. *A Virtual dynamic topology for service discovery in mobile ad hoc networks*. In the Proceedings of IEEE Wireless Communications and Networking Conference 2007 - Services and applications - Hong Kong 11-15, (2007)

[Haddad 08] M. Haddad, L. Dekar, H. Kheddouci, *A distributed strict strong coloring algorithm for broadcast applications in ad hoc networks*. 8th annual international conference on New Technologies of Distributed Systems NOTERE'2008 ACM Digital Library Lyon, FRANCE - June 23-27, 2008.

[Haddad 09] M. Haddad and H. Kheddouci, *A strict strong coloring of graphs*, Information Processing Letters - IPL, Volume : 109, Issue : 18, Pages : 1047-1054, (2009)

[Hakimi 86] S. L. Hakimi and O. Kariv. *On a generalization of edge-coloring in graphs*. Journal of Graph Theory, 10 :139-154, (1986)

[Harary 70] F. Harary, S. Hedetniemi. *The achromatic number of a graph*. Journal of Combinatorial Theory, 8 :154-161, (1970)

[Han 07] B. Han, W. Jia. *Clustering wireless ad hoc networks with weakly connected dominating set*, Journal of Parallel and Distributed Computing Volume 67, Issue 6 Pages : 727-737 (2007)

[Harary 69] F. Harary. *Graph theory*, Ed. Addison-Wesley (1969)

[Haynes 98] T. W. Haynes, S. T. Hedetniemi, P. J. Slater. *Domination in graphs*, Advanced Topics, Marcel Dekker, Inc., (1998)

[Hedetniemi 82] S. M. Hedetniemi, S. T. Hedetniemi, and T. Beyer. *A linear algorithm for the grundy (coloring) number of a tree*. Congressus Numerantium, 36 :351-363, (1982)

[Hedetniemi 03] S. T. Hedetniemi, D. P. Jacobs, P. K. Srimani. *Linear time self-stabilizing colorings*. Inf. Process. Lett., 87(5) :251-255 (2003)

[Herman 90] T. Herman. *Probabilistic self-stabilization*, Inform. Proc. Lett. 35 63-67 (1990)

[Herman 02] T. Herman. *Self-Stabilization Bibliography : Access. Guide*, University of Iowa, (2002). See ftp ://ftp.cs.uiowa.edu/pub/selfstab/bibliography/

[Holyer 81] I. Holyer. *The NP-completeness of edge-coloring*, SIAM J. Comput, pp. 718-720, 10 (1981)

[Hopcroft 83] J. Hopcroft and M. S. Krishnamoorthy. *On the harmonious coloring of graphs*. SIAM Journal on Algebraic Discrete Methods, 4 :306-311 (1983)

[Hou 01] T.C. Hou, T.J. Tsai. *An access-based clustering protocol for multihop wireless ad hoc networks*. IEEE Journal on Selected Areas in Commnications, 19(7) :12011210 (2001)

[Hsu 92] S.C. Hsu, S. T. Huang. *A self-stabilizing algorithm for maximal matching*, Inform. Process. Lett. 43 , 77-81 (1992)

[Huang 98] S. T. Huang, T. J. Liu. *Four-state stabilizing phase clock for unidirectional rings of odd size*, Information Processing Letters, 65(6), pp.325-329. 41 (1998)

[Huang 09] S.T. Huang, C.H. Tzeng. *Distributed edge coloration for bipartite networks*. Distributed Computing 22(1) : 3-14 (2009)

[Irving 99] W. Irving and D. F. Manlove. *The b-chromatic number of a graph*. Discrete Mathematics, 91 :127-141 (1999)

[Israeli 90] A. Israeli, M. Jalfon, *Self-Stabilizing Ring Orientation*, Proceedings of 4th Workshop on Distributed Algorithms, vol. 486 of Lecture Notes in Computer Science, Springer-Verlag, pp.1-14, (1990)

[Israeli 93] A. Israeli, M. Jalfon. *Uniform Self-Stabilizing Ring Orientation*, Inf. Comput. 104(2) : 175-196 (1993)

[Itkis 92] G. Itkis. *Self-stabilizing distributed computation with constant space per edge*, Presentation at MIT, (1992)

[Ito 07] T. Ito, A. Kato, X. Zhou and T. Nishizeki. *Algorithms for finding distanceedge-colorings of graphs*, Journal of Discrete Algorithms, pp. 304-322, (2007)

[Jakobsen 73] I. T. Jakobsen. *Some remarks on the chromatic index of a graph*, Archiv der Mathematik, 440-448, 24 (1973)

[Jaikaeo 02] C. Jaikaeo, C.-C. Shen, *Adaptive backbone-based multicast for ad hoc networks*, in : Proc. of the IEEE Int'l Conf. Comm. (2002)

[Jarvis 01] M. Jarvis and B. Zhou. *Bounded vertex coloring of trees*. Discrete Mathematics, 232 :145-151 (2001)

[Jiang 99] M. Jiang, J. Li, Y. C. Tay. *Cluster based routing protocol (CBRP)*. Internet draft version 01, IETF (1999)

[Kaklamanis 97] C. Kaklamanis, P. Persiano, T. Erlebach, K. Jansen. *Constrained Bipartite Edge Coloring with Applications to Wavelength Routing*. Automata, Languages and Programming, 24th International Colloquium, ICALP'97, 493-504 (1997)

[Kang 09] R. Kang and P. Manggala. *On distance edge-colourings and matchings*, Electronic Notes in Discrete Mathematics 34 : 301-306 (2009)

[Karp 72] R. Karp, *Reducibility Among Combinatorial Problems*. Proceedings of a Symposium on the Complexity of Computer Computations, Plenum Press (1972)

[Katz 93] S. Katz, K. Perry. *Self-stabilizing extensions for message-passing systems*, Distributed Computing, vol.7 no.1„ pp.17-26 (1993)

[Kheddouci 00] H. Kheddouci, J.F. Saclé AND M. Wozniak, *Packing of two copies of a tree into its fourth power*, Discrete Mathematics 213 (1-3)169-178 (2000)

[Kim 00] Y. Kim, A. Perrig, G. Tsudik. *Simple and fault-tolerant key agreement for dynamic collaborative groups*, in : Proc. Seventh ACM Conf. Computer and Comm. Security, pp. 235-244 (2000)

[Kim 04] Y. Kim, A. Perrig, G. Tsudik, *Tree-based group key agreement*, ACM Transactions on Information System Security 7 (1) 60-96 (2004)

[Kocay 05] W. Kocay, D.L. Kreher. *Graphs, Algorithms, and Optimization*, Discrete Mathematics CRC Press (2005)

[Konig 16] D. König. *Uber Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre*,Math. Ann. 77, 453-465 (1916)

[Kouider 02] M. Kouider and M. Mahéo. *Some bounds for the b-chromatic number of a graph*. Discrete Mathematics, 256(Issues 1-2) :267-277, (2002)

[Kouider 06] M. Kouider and M. Zaker. *Bounds for the b-chromatic number of some families of graphs*. Discrete Mathematics, 306 :617-623, 2006.

[Kozat 01] U.C. Kozat, G. Kondylis, B. Ryu, M.K. Marina, *Virtual dynamic backbone for mobile ad hoc networks*, in : Proc. IEEE Int'l Conf. Comm (2001)

[Kramer 69] F. Kramer and H. Kramer. *Un probleme de coloration des sommets d'un graphe*, C.R. Acad. Sci., pp. 46-48, Paris A 268 (1969)

[Kramer 72] F. Kramer. *Sur le nombre chromatique K(p,G) des graphes*, RAIRO, pp. 67-70, R-1 (1972)

[Kramer 08] F. Kramer and H. Kramer. *A survey on the distance-colouring of graphs*, Discrete Mathematics, pp. 422-426, 308 (2008)

[Kubale 04] M. Kubale. *History of graph coloring*, in Kubale (2004)

[Kuratowski 30] C. Kuratowski. *Sur le problème des courbes gauches en topologie*, Fund. Math., 15, 271-283 (1930)

[Kuszner 06] L. Kuszner, A. Nadolski. *Self-stabilizing algorithm for edge-coloring of graphs*, Foundations of Computing and Decision Sciences , Vol. 31 No. 2, 157-167 (2006)

[Lamport 83] L. Lamport. Solved Problems, Unsolved Problems and Non-Problems in Concurrency. PODC84, pages 63-67 (1983)

[Larson 98] T. Larson, N. Hedman, Routing protocols in wireless ad-hoc networks – a simulation study, Mater's Thesis, Lulea University of Technology Stockholm, (1998)

[Lewis 10] R. Lewis, J. Thompson. *On the application of graph colouring techniques in round-robin sports scheduling*. Computers and Operations Research, vol. In Press, Corrected Proof, pp (2010)

[Li 02] X. Li, Y. Wang, O. Frieder, *Efficient hybrid key agreement protocol for wireless ad hoc networks*, Computer Communications and Networks (2002)

[Li 05] X. Li, J. Slay, S. Yu, *Evaluating trust in mobile ad hoc networks*, in : Proc. of the International Conference on Computational Intelligence and Security (2005)

[Lin 97] C.R. Lin, M Gerla. *Adaptive clustering for mobile wireless networks*. IEEE Journal of Selected Areas in Communications, 15(7) : 1265-1275 (1997)

[Liu 94] G. Liu. *On (g, f)-factors and factorizations*. Acta Math. Sin. 37, 230-237 (1994)

[Liu 04] Z. Liu, A. Joy, R.A. Thompson. *A dynamic trust model for mobile ad hoc networks*, in : Proc. of the 10th IEEE Int'l Workshop on Future Trends of Distributed Computing Systems, (2004)

[Liu 05] G. Liu, X. Deng. *A polynomial algorithm for finding (g, f)-colorings orthogonal to stars in bipartite graphs*. Sci. China Ser. A 35(3), 334-344 (2005)

[Lu 97] Z. Lu. *The exact value of the harmonious chromatic number of a complete binary tree*. Discrete Mathematics, 172 :93-101, (1997)

[Malaguti 09] E. Malaguti, P. Toth. *A survey on vertex coloring problems*. International Transactions in Operational Research, pp. 1-34 (2009)

[Manulis 05] M. Manulis, *Contributory group key agreement protocols, revisited for mobile ad-hoc groups*, in : Proc. of the 2nd IEEE Int'l Conference on Mobile Sensor Systems, MASS'05, IEEE Computer Society, pp. 811-818 (2005)

[Masuzawa 05] T. Masuzawa and S. Tixeuil. *A Self-stabilizing Link Coloring Algorithm Resilient to Unbounded Byzantine Faults in Arbitrary Networks*, Proceedings of OPODIS, LNCS, Springer-Verlag, 118-129, (2005)

[Masuzawa 07] T. Masuzawa and S. Tixeuil. *Stabilizing link-coloration of arbitrary networks with unbounded Byzantine faults*. Int. J. Princ. Appl. Inf. Sci. Technol. 1(1), 1-13 (2007)

[Masuzawa 09] T. Masuzawa, and S. Tixeuil. *On bootstrapping topology knowledge in anonymous networks*. ACM Trans. Autonom. Adapt. Syst. 4, 1, Article 8, 27 pages (2009)

[Mayer 92] A. Mayer, Y. Ofek, R. Ostrovsky, M. Yung. *Self-Stabilizing Symmetry Breaking in Constatnt-Space*, STOC'92, pp.667-678 (1992)

[Misra 92] J.Misra, D. Gries. *A constructive Proof of Vizing's Theorem*. Inf. Proc. Lett., 41 : 131-133 (1992)

[Mittra 97] S. Mittra. *Iolus :a framework for scalable secure multicasting*, in : Proc. of the ACM SIGCOMM, Cannes, France, pp. 277-288 (1997)

[Molloy 02] M. Molloy , M. R. Salavatipour. *Frequency channel assignment on planar networks*, Proceedings of ESA'02, LNCS Vol. 2461, 736-747, Springer (2002)

[Pardalos 98] P. Pardalos, T. Mavridou, J. Xue. *The graph coloring problem : A bibliographic survey*, vol. 2, pp. 331-395. Kluwer Academic Publishers, Boston (1998)

[Paschos 07] V. Paschos. *Optimisation combinatoire 5, Problèmes paradigmatiques et nouvelles problématiques*, Hermès Science : Lavoisier (2007)

[Pirzada 04] A.A. Pirzada, C. McDonald, *Establishing trust in pure adhoc networks*, in : Proc. of the 27th Conference on Australasian Computer Science, Australian Computer Society, Dunedin, New Zealand, (2004)

[Prim 57] R. C. Prim, *Shortest Connection Matrix Network and Some Generalizations*, Bell System Technical Journal, Vol. 36, 1389-1401, (1957)

[Sache 74] A. Sache. *La théorie des graphes*, Presses Universitaires de France - Que sais-je ? (1974)

[Sakurai 04] Y. Sakurai, F. Ooshita, and T. Masuzawa. *A self-stabilizing link-coloring protocol resilient to byzantine faults in tree networks*, In 8th International Conference on Principles of Distributed Systems, 283-298 (2004)

[Sanders 01] D.P. Sanders, Y. Zhao. *Planar Graphs of Maximum Degree Seven are Class I*. Journal of Combinatorial Theory, Series B. 83, Issue 2, 201-212 (2001)

[Schneider 93] M Schneider. *Self-stabilization*. ACM Computing Surveys, 25 :45-67 (1993)

[Siddiqui 02] A. Siddiqui, , R. Prakash, *Effect of Availability Factor Threshold and Clustering Gap on Performance of Clustering Mechanisms for Multi-cluster Mobile Ad Hoc Networks*, ICC (2002)

[Skiena 08] S. S. Skiena. *The Algorithm Design Manual*, Springer (2008)

[Speranza 75] F. Speranza. *Colorazioni di specie superiore d'un grafo*, Boll. Un. Mat. Ital., pp. 53-62, (4) 12 (1975)

[Steiner 00] M. , G. Tsudik, M. Waidner, *Key agreement in dynamic peer groups*, IEEE Transactions on Parallel and Distributed Systems 11 (8) 769-779 (2000)

[Sur 93] S. Sur, K. Srimani. *A self-stabilizing algorithm for coloring bipartite graphs*. Inf. Sci., 69(3) :219-227, (1993)

[Tel 94] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, (1994)

[Tixeuil 00] S. Tixeuil. *Auto-stabilisation Efficace*. PhD thesis, University of Paris Sud XI, January (2000)

[Theodor 06a] G. Theodorakopoulos, J.S. Baras, *A testbed for comparing trust computation algorithms*, in : Proc. of the 25th Army Science Conference, Orlando, FL, November 27-30, (2006)

[Theodor 06b] G. Theodorakopoulos, J.S. Baras, *On trust models and trust evaluation metrics for ad-hoc networks*, Journal of Selected Areas in Communications, Security in Wireless Ad-Hoc Networks 24 (2) 318-328 (2006)

[Tsai 94] M. S. Tsai, S. T. Huang. *A self-stabilizing algorithm for the shortest paths problem with a fully distributed demon*, Parallel Processing Letters 4 ,65-72 (1994)

[Tzeng 07] C.H. Tzeng, J.R. Jiang, S.T. Huang. *A self-stabilizing ($\triangle$ + 4)- edge-coloring algorithm for planar graphs in anonymous uniform systems*. Inf. Process. Lett. 101(4), 168-173 (2007)

[Tseng 07a] Y.M. Tseng, C.C. Yang, D.R. Liao, *A secure group communication protocol for ad hoc wireless networks*, in : Advances in Wireless Ad Hoc and Sensor Networks, Signals and Communication Technology Series, Springer, (2007)

[Umemoto 98] N. Umemoto, H. Kakugawa, M. Yamashita. *A Self-Stabilizing Ring Orientation with a Smaller Number of Processor States*, IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 6, (1998)

[Vizing 64] V.G. Vizing. *On an estimate of the chromatic class of a graph*, Diskret. Anal.,pp. 25-30, 3 (1964)

[Vizing 65] V.G. Vizing. *Critical graphs with a given chromatic class*, Diskret. Anal., pp.9-17, 5 (1965)

[Wattenhofer 01] R. Wattenhofer, L. Li, P. Bahl, et Y.-M. Wang. *Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad hoc Networks*. Infocom (2001)

[West 00] D. B. West. *Introduction to Graph Theory*, 2nd ed. Englewood Cliffs, NJ : Prentice-Hall, (2000)

[Wilson 04] R. J. Wilson. *History of Graph Theory,Handbook of Graph Theory*, Section 1.3, pp. 29-49 (2004)

[Wilson 78] R. J. Wilson. Edge-colorings of graphs : A survey. Theory and Applications of Graphs. pp.608-619, (1978)

[Wuu 95] L. C. Wuu, S. T. Huang. *Distributed self-stabilizing systems*, Journal of Information Science and Engineering 11, 307-319 (1995)

[Yannakakis 80] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. SIAM Journal on Applied Mathematics, 38(3) :364-372, (1980)

[Yu 03] J. Y. Yu and P. H. J. Chong, *3hBAC (3-hop between Adjacent Clusterheads) : a Novel Non-overlapping Clustering Algorithm for Mobile Ad Hoc Networks*, in proceedings of IEEE Pacrim'03, vol. 1, pp. 318-21 (2003)

[Zhang 02] Z. Zhang, L. Liu and J. Wang, *Adjacent strong edge coloring of graphs*, Appl. Math. Lett. 15, pp. 623Ű626 (2002)

[Zhou 95] X. Zhou, T. Nishizeki. *Simple reduction of f-colorings to edge-colorings*. COMPUTING AND COMBINATORICS, Lecture Notes in Computer Science, 1995, Volume 959, 223-228 (1995)

[Zverovich 06] I. E. Zverovich, *A new kind of graph coloring*, Journal of Algorithms, 58(2), pp. 118-133 (2006)