N° d'ordre : 3257

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Jérémie CHALOPIN**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes

---

Soutenue le : 24 novembre 2006

Après avis des rapporteurs :

Joffroy Beauquier	Professeur
Jacques Mazoyer	Professeur
Antoni Mazurkiewicz	Professeur

Devant la commission d'examen composée de :

Joffroy Beauquier	Professeur	Rapporteur
Bruno Courcelle	Professeur	Président
Jacques Mazoyer	Professeur	Rapporteur
Yves Métivier	Professeur	Directeur de Thèse

– 2006 –

Remerciements

Mes remerciements s'adressent en premier lieu à mon directeur de thèse, Yves Métivier. Il m'a dans un premier temps encadré lors de mon stage de DÉA et c'est à cette occasion qu'il m'a fait découvrir le domaine de l'algorithmique distribuée. Par la suite, il m'a encadré durant ma thèse et je tiens le remercier pour l'écoute et la disponibilité dont il a fait preuve au cours de ces années : cela a été un réel plaisir de travailler avec lui.

Je tiens à remercier chaleureusement Joffroy Beauquier, Jacques Mazoyer et Antoni Mazurkiewicz qui m'ont fait l'honneur de relire mon mémoire. Leurs commentaires ont été très enrichissants et je tiens à exprimer ma gratitude pour l'intérêt qu'ils ont porté à mon travail.

Je remercie également Bruno Courcelle pour avoir accepté de présider mon jury et pour l'intérêt qu'il a porté à mes travaux au cours de ma thèse.

Je remercie également mes collègues, amis et parfois coauteurs du LaBRI. Parmi les plus jeunes, je tiens à remercier particulièrement Alexandre, Pascal, Olivier, Bilel, Maxime, Julien, Séverine, Daniel, Omer, Jocelyn, Antoine, Jean-François, Brahim, Rodrigue, Abelaaziz.

Je souhaite également remercier les personnes extérieures au LaBRI avec qui j'ai eu l'occasion de travailler durant ces trois dernières années : Daniel Paulusma, Shantanu Das, Paola Flocchini, Nicola Santoro, Wiesław Zielonka, Emmanuel Godard.

Finalement, je tiens à remercier ma famille pour leur soutien.

Introduction

Un système distribué est un environnement où plusieurs processus collaborent pour réaliser un objectif commun. Dans un réseau, les différents processus ne peuvent communiquer directement qu'avec un nombre limité d'autres processus, leurs «voisins». L'algorithmique distribuée a pour but de décrire quelles sont les tâches qui peuvent être réalisées dans de tels systèmes. Un élément du réseau est un «noeud» et ce qui permet de distinguer un noeud d'un autre peut être un identifiant (comme une adresse IP sur Internet), mais plus généralement, c'est la position de chaque noeud dans le réseau qui permet de le distinguer. Ce qui caractérise les systèmes distribués est qu'il n'existe pas a priori de système de centralisation qui peut coordonner globalement les différents processus. L'algorithmique distribuée cherche ainsi à déterminer quels sont les comportements globaux qui peuvent être obtenus dans de tels systèmes où les comportements des différents processus ont des effets locaux.

Quelques Problématiques de l'Algorithmique Distribuée

Dans ce mémoire, on étudie principalement des problèmes où la «symétrie» initiale du réseau doit être brisée. Le problème de l'*élection* est un problème de ce type, qui a été pour la première fois étudié par Lelann [LeL77]. Le but d'un algorithme d'élection est de distinguer un noeud du réseau qui est dans l'état ÉLU à la fin de l'exécution de l'algorithme, alors que tous les autres noeuds sont dans l'état NON-ÉLU. Si les noeuds possèdent des identifiants uniques, alors on peut élire le noeud qui a le plus petit identifiant. Si de tels identifiants n'existent pas, le réseau est dit *anonyme* et on va voir qu'il est impossible de résoudre l'élection dans certains réseaux anonymes qui sont trop «symétriques». Un autre problème qui nécessite de briser la symétrie initiale du réseau est le problème du *nommage*. Le but d'un algorithme de *nommage* est d'arriver dans une configuration finale où un identifiant unique est associé à chaque noeud. Les problèmes du nommage et de l'élection sont équivalents dans certains modèles (si on sait résoudre l'un, on sait résoudre l'autre), mais ce n'est pas toujours le cas. D'autres problèmes classiques en algorithmique distribuée sont aussi basés sur la rupture de la «symétrie» comme par exemple le calcul d'un arbre couvrant ou la reconstruction d'une carte du réseau où chaque noeud doit connaître sa position dans le réseau.

Ces problèmes sont très importants en algorithmique distribuée puisque de nombreux algorithmes existants ne fonctionnent correctement que sous l'hypothèse qu'il existe un noeud distingué ou que les processus ont des identifiants uniques. Disposer d'un noeud distingué permet par exemple, de centraliser ou de diffuser de l'information, d'initialiser l'exécution d'un algorithme, de prendre une décision de manière centralisée, etc. De même,

disposer d'une carte du réseau où chaque processeur connaît sa position dans le réseau permet de diffuser de l'information à partir d'un noeud en minimisant le nombre de messages utilisés.

Ces problèmes ont été très étudiés dans la littérature, et de nombreux algorithmes ont été proposés pour des topologies particulières : les arbres, les anneaux, les tores, etc. On pourra se référer aux livres de Tel [Tel00], d'Attiya et Welch [AW04] ou de Lynch [Lyn96] pour plus de détails sur ces résultats. Dans les sections suivantes, on présente les résultats existants pour les réseaux de topologies arbitraires obtenus par Yamashita et Kameda [YK96b], par Boldi, Codenotti, Gemmell, Shammah, Simon et Vigna [BCG$^+$96] et par Mazurkiewicz [Maz97].

Dans ce mémoire, dans tous les modèles qu'on considère, un réseau est modélisé par un graphe simple dont les sommets correspondent aux processus et dont les arêtes correspondent aux liens de communication. On considère des réseaux de topologie arbitraire qui sont fiables : il ne se produit aucune défaillance, ni sur les noeuds, ni sur les liens de communication.

Des Résultats Généraux

Caractériser les graphes admettant des algorithmes d'élection ou de nommage dans les différents modèles est un problème important pour l'étude de ces modèles. D'une part, on va voir que cela permet de mettre en évidence les différences entre les modèles et cela permet de pouvoir montrer qu'un modèle a une puissance de calcul strictement plus forte qu'un autre. Ces caractérisations permettent de décrire de manière combinatoire quelles sont les «symétries» d'un graphe qui empêchent de le distinguer d'un autre.

D'autre part, la compréhension des caractéristiques des graphes admettant un algorithme d'élection ou de nommage dans les différents modèles est une étape importante dans la compréhension de ce qui est calculable de manière distribuée. Ainsi, les travaux de Yamashita et Kameda [YK96a, YK98] sur les fonctions qui pouvaient être calculées dans un modèle où les processus communiquent par échange de messages utilisent les concepts introduits dans [YK96b] où ils caractérisent les réseaux admettant un algorithme d'élection dans ce même modèle.

Dans [BCG$^+$96], Boldi, Codenotti, Gemmell, Shammah, Simon et Vigna caractérisent les graphes qui admettent un algorithme d'élection dans un modèle où en un pas de calcul, un sommet peut modifier son état en fonction de l'état de ses voisins. Ils considèrent un modèle *synchrone* où en un pas de calcul tous les sommets du graphe appliquent une transition de ce type et un modèle *entrelacé*, où à chaque étape, il y a un unique sommet qui applique une transition. À partir des résultats présentés dans [BCG$^+$96], Boldi et Vigna ont caractérisé quelles fonctions pouvaient être calculées de manière distribuée dans le même modèle [BV99, BV01]. Toujours en utilisant les mêmes outils, ils ont aussi caractérisé quelles tâches pouvaient être effectuées de manière auto-stabilisante [BV02b] dans un système synchrone.

Dans [Maz97], Mazurkiewicz caractérise les graphes admettant un algorithme de nommage dans un modèle où en un pas de calcul, un sommet peut modifier son état et l'état de ses voisins en fonction de son propre état et de l'état de ses voisins. Ce modèle est *entrelacé* puisque deux voisins ne peuvent pas appliquer simultanément une transition de ce type. Dans [GM02], Godard et Métivier utilisent l'algorithme de Mazurkiewicz ainsi que

d'autres outils pour caractériser quelles sont les familles de graphes qui admettent un algorithme universel d'élection dans ce modèle. Cet algorithme est aussi utilisé dans [MT00] par Métivier et Tel afin de déterminer quelles conditions permettent de détecter la terminaison globale d'un algorithme distribué. Dans les travaux de Godard, Métivier et Muscholl [GMM04, GM03], l'algorithme de Mazurkiewicz est aussi l'un des outils utilisés afin de caractériser les classes de graphes qui peuvent être reconnues de manière distribuée dans ce modèle.

Importance des Connaissances Initiales

Une fois qu'on sait qu'un réseau admet un algorithme d'élection dans un modèle, on cherche à savoir quelle caractéristique du réseau est nécessaire pour obtenir un algorithme d'élection pour ce réseau. L'algorithme peut être tout à fait spécifique : pour un réseau différent, il faudra utiliser un autre algorithme. On peut toutefois souhaiter obtenir des algorithmes *universels* pour certaines familles de graphes.

Si on considère l'ensemble des graphes de taille n qui admettent un algorithme d'élection, est-il possible de trouver un algorithme universel qui permette de résoudre le problème de l'élection sur tous ces graphes ? Autrement dit, est-il possible de résoudre le problème de l'élection en connaissant seulement la taille du graphe ? De même, on peut se demander s'il est possible de résoudre le problème de l'élection en connaissant seulement une borne sur la taille du graphe ou sans aucune connaissance sur le graphe.

Puisque tous les graphes n'admettent pas d'algorithme d'élection, il est assez naturel de se demander quelles sont les connaissances «minimales» qui permettent à un algorithme de détecter qu'on ne peut pas résoudre le problème sur le graphe sur lequel l'algorithme est exécuté. Il est clair que si on connaît la topologie du graphe et qu'on sait caractériser les graphes n'admettant pas d'algorithme d'élection, il est possible de détecter qu'on ne peut pas résoudre le problème sur le graphe avant même de commencer l'exécution. Cependant, est-il possible de détecter que le problème ne peut pas être résolu sur un graphe de manière distribuée en connaissant seulement la taille du graphe ou une borne sur la taille du graphe ?

La recherche de solutions générales de ce type permet d'obtenir des algorithmes plus «fiables» : les conditions que doivent vérifier un réseau pour que l'algorithme fonctionne correctement sont moins contraignantes. Ainsi, on préférera un algorithme d'élection qui nécessite la connaissance d'une borne sur la taille du graphe à un algorithme qui nécessite de connaître la topologie du graphe, puisque dans le premier cas, on peut ajouter ou enlever un certain nombre de sommets et continuer à utiliser le même algorithme.

Travaux Existants

Depuis le travail original d'Angluin [Ang80], il est connu que certains réseaux n'admettent pas d'algorithme d'élection en raison de leur «symétrie» trop importante. Dans le modèle d'Angluin, l'outil pour exprimer ces symétries est la notion de *revêtements simples* qui sont des homomorphismes de graphes simples localement bijectifs. Par ailleurs, le raisonnement d'Angluin permet de montrer aussi qu'il existe des graphes qu'on ne peut pas distinguer les uns des autres. Ainsi, si un graphe G est un revêtement simple d'un graphe H, il n'existe pas d'algorithme permettant aux sommets du graphe sur lequel l'algorithme

est exécuté de décider si le graphe auquel ils appartiennent est le graphe G ou le graphe H.

Ce résultat d'impossibilité a ensuite été adapté dans le modèle considéré par Mazurkiewicz [Maz97] et dans le modèle étudié par Boldi et al. [BCG$^+$96]. Mazurkiewicz [Maz97] a donné un algorithme qui permet de résoudre nommage et élection dans tous les graphes pour lesquels le raisonnement d'Angluin ne permettait pas d'obtenir un résultat d'impossibilité. L'algorithme de Mazurkiewicz repose seulement sur les connaissances locales que chaque sommet peut collecter à propos de ses voisins et permet de résoudre les problèmes de l'élection et du nommage sur tout graphe qui n'est un revêtement d'aucun autre graphe que lui-même.

La caractérisation obtenue par Yamashita et Kameda [YK96b] dans un modèle où les processus communiquent par échange de messages est très différente de celle de Mazurkiewicz et les techniques utilisées sont elles aussi totalement différentes. La caractérisation de Yamashita et Kameda [YK96b] est exprimée en termes de «vues», où la vue d'un sommet v est un arbre infini dont les sommets sont les chemins issus de v dans le graphe. Yamashita et Kameda montrent que pour tout algorithme, si deux sommets ont la même vue, il existe une exécution de l'algorithme où ces deux sommets restent toujours dans le même état. L'algorithme de Yamashita et Kameda repose sur un résultat de Norris [Nor95] qui a montré que deux sommets d'un graphe de taille n avaient la même vue si leurs vues tronquées à la hauteur n étaient les mêmes. Ainsi, dans l'algorithme de Yamashita et Kameda, chaque sommet construit sa vue tronquée à la hauteur n et la compare ensuite avec celle des autres sommets du graphe ; le sommet ayant la plus «petite» vue est ensuite élu.

Les travaux de Boldi et al. [BCG$^+$96] permettent d'établir un premier lien entre les travaux de Yamashita et Kameda et ceux de Mazurkiewicz. En effet, Boldi et al. utilisent un raisonnement «à la Angluin» pour établir des résultats d'impossibilité mais leurs algorithmes d'élection fonctionnent sur le même principe que l'algorithme de Yamashita et Kameda. Les résultats d'impossibilité présentés par Boldi et al. reposent sur la notion de fibrations qui sont des homomorphismes de graphes dirigés qui induisent une bijection entre les arcs entrants de chaque sommet et les arcs entrants de son image. Dans [BCG$^+$96], Boldi et al. montrent comment obtenir des fibrations (qui ont des propriétés similaires aux revêtements) à partir de la vue de chaque sommet.

Cependant, les algorithmes utilisés par Yamashita et Kameda et par Boldi et Vigna restent très différents de l'algorithme de Mazurkiewicz. En effet, les algorithmes de Yamashita et Kameda et de Boldi et Vigna nécessitent des connaissances initiales sur le graphe afin de savoir jusqu'à quelle hauteur chaque sommet doit calculer sa vue. Au contraire, l'algorithme de Mazurkiewicz ne nécessite aucune connaissance initiale et il termine toujours, même si sans connaissance initiale, il est impossible aux sommets de détecter si l'exécution de l'algorithme est terminée ou non. Cette différence est importante puisque cette propriété de l'algorithme de Mazurkiewicz est utilisée dans [GMM04] afin de caractériser les classes de graphes reconnaissables de manière distribuée dans le modèle considéré par Mazurkiewicz (dans ce cadre, les sommets ne doivent pas nécessairement détecter la terminaison de l'algorithme). Dans [God02b], Godard présente une version auto-stabilisante de l'algorithme de Mazurkiewicz et cette propriété de terminaison permet d'assurer que le temps de stabilisation est fini.

Par ailleurs, pour exécuter l'algorithme de Mazurkiewicz, les sommets d'un graphe G ont besoin d'une mémoire polynomiale en la taille du graphe, alors que les algorithmes

de Yamashita et Kameda et de Boldi et Vigna nécessitent que chaque sommet ait une mémoire exponentielle en la taille du graphe. Cette propriété est importante puisque dans l'algorithme de Yamashita et Kameda, la taille de la mémoire de chaque processus est liée à la taille des messages échangés, qui sont aussi de taille exponentielle.

Techniques et Résultats

Étudier des Modèles Intermédiaires

Une première motivation des travaux présentés dans ce mémoire est d'étudier des modèles «intermédiaires» entre le modèle de Mazurkiewicz et le modèle de Yamashita et Kameda. Le but de ce travail est de comprendre quelles sont les limites de chaque modèle et d'essayer de déterminer si un résultat d'impossibilité ou de possibilité est spécifique à un modèle ou s'il est plus général.

Dans ce but, on considère différents modèles utilisant des *calculs locaux* dont le modèle étudié par Mazurkiewicz est le plus puissant. Dans de tels modèles, l'état global du système distribué est représenté par un graphe simple étiqueté où le graphe correspond au réseau sous-jacent et où l'étiquette de chaque sommet représente l'état du processus correspondant.

Dans le modèle de Mazurkiewicz [Maz97], un pas de calcul permet à un sommet de modifier son étiquette et l'étiquette de ses voisins en fonction de sa propre étiquette et des étiquettes de ses voisins. Ainsi, dans ce modèle, un algorithme peut être décrit à l'aide de règles de réétiquetage qui permettent de modifier les étiquettes des sommets d'une étoile du graphe (un sommet et ses voisins) en fonction seulement des étiquettes apparaissant sur cette étoile. Ce modèle correspond au modèle (7) de la Figure 1. Les résultats de Mazurkiewicz [Maz97] et des corollaires de résultats plus généraux obtenus par Godard et Métivier [GM02] sont présentées au Chapitre 2.

Dans le modèle entrelacé de Boldi et al. [BCG$^+$96], un pas de calcul permet à un sommet de modifier seulement son étiquette en fonction de sa propre étiquette et de l'étiquette de ses voisins. Ce modèle peut être décrit à l'aide de règles de réétiquetage de graphes qui permettent de modifier l'étiquette d'un sommet en fonction des étiquettes de ses voisins. Ce modèle correspond au modèle (5) de la Figure 1. On étudie ce modèle au Chapitre 4 : on présente de nouvelles preuves des résultats de Boldi et al. [BCG$^+$96]. On présente en particulier des algorithmes de nommage et d'élection qui utilisent certaines idées de l'algorithme de Mazurkiewicz et qui nécessitent que chaque sommet ait une mémoire polynomiale en la taille du graphe, et non exponentielle comme les algorithmes de Boldi et al.

Les modèles étudiés par Boldi et al. [BCG$^+$96] et par Mazurkiewicz [Maz97] sont des modèles plus abstraits que le modèle étudié par Yamashita et Kameda, mais dans lesquels les caractérisations présentées s'expriment de manière élégante. Ces modèles correspondent à différents niveaux de synchronisation entre voisins. En effet, bien que l'exécution soit globalement asynchrone, un pas de calcul nécessite une synchronisation entre un sommet et ses voisins.

Dans ce mémoire, on étudie aussi des modèles où en un pas de calcul, une synchronisation a lieu entre deux sommets voisins du graphe. Ou bien, les étiquettes des deux sommets peuvent être modifiées en un pas de calcul, ou alors un seul sommet peut modifier son étiquette en fonction de l'étiquette d'un de ses voisins.

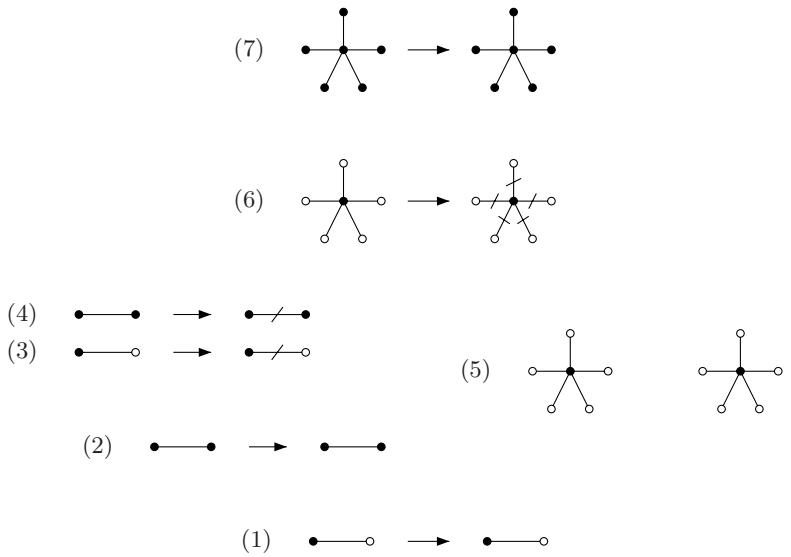


FIG. 1 – Les différents modèles de calculs locaux étudiés dans ce mémoire décrits par le type de règles utilisées. Un sommet noir est un sommet *actif* dont l'étiquette peut être modifiée par l'application de la règle, alors qu'un sommet blanc est un sommet *passif* qui permet d'appliquer la règle mais dont l'étiquette n'est pas modifiée. Dans les modèles (3), (4) et (6), les étiquettes des arêtes peuvent être modifiées alors que ce n'est pas possible dans les autres modèles.

On se rend rapidement compte que dans ces modèles où les synchronisations ont lieu sur des arêtes, on obtient des modèles strictement plus puissants lorsque les règles de réétiquetages peuvent modifier les étiquettes des arêtes sur lesquelles elles sont appliquées. De même, dans le modèle de Boldi et al., on obtient un modèle plus puissant si les règles de réétiquetage peuvent modifier les étiquettes des arêtes incidentes au sommet dont l'étiquette est modifiée. On verra au Chapitre 2 que ce n'est pas le cas pour le modèle de Mazurkiewicz.

Dans le modèle (6) de la Figure 1, un pas de calcul permet à un sommet de modifier son étiquette et les étiquettes des arêtes qui lui sont incidentes en fonction de son étiquette, des étiquettes des arêtes qui lui sont incidentes et des étiquettes de ses voisins. La seule différence entre ce modèle et le modèle (5) de la Figure 1 est le fait que les arêtes du graphe peuvent être étiquetées et réétiquetées.

Dans les modèles (2) et (4) de la Figure 1, un pas de calcul permet à deux sommets voisins de modifier leurs étiquettes en fonction de leurs étiquettes précédentes. La différence entre ces deux modèles est que dans le modèle (4), les arêtes du graphes peuvent être étiquetées et réétiquetées et lorsqu'une règle de réétiquetage est appliquée sur une arête, la règle peut dépendre de l'étiquette de l'arête et peut modifier cette étiquette.

Dans les modèles (1) et (3) de la Figure 1, un pas de calcul permet à un sommet de modifier son étiquette en fonction de l'étiquette d'un de ses voisins. Dans le modèle (3), comme dans le modèle (4), les arêtes du graphes peuvent être étiquetées et lorsqu'une règle de réétiquetage permet de modifier l'étiquette d'un sommet v en fonction de l'étiquette de l'un de ses voisins v', la règle peut dépendre de l'étiquette de l'arête $\{v, v'\}$ et peut modifier cette étiquette.

Dans les Chapitres 2, 3, 4, 5 et 6, on étudie les différents modèles de la Figure 1 à

travers les problèmes de l'élection et du nommage. Pour chaque modèle, on caractérise les graphes dans lesquels on peut résoudre les problèmes de l'élection et du nommage. Pour obtenir des conditions nécessaires, il faut d'abord trouver quels sont les homomorphismes qui permettent d'obtenir un lemme de relèvement «à la Angluin» pour la classe de graphes la plus large possible pour chaque modèle considéré. Pour obtenir des conditions suffisantes, il faut ensuite trouver un algorithme qui permette de nommer dans tous les graphes pour lesquels on n'a pas obtenu de résultat d'impossibilité. La recherche de ces homomorphismes et de ces algorithmes sont très liés, puisqu'il faut réussir à déterminer exactement quelles sont les informations que peut obtenir chaque sommet à propos de ses voisins. Par exemple, dans certains modèles, un sommet peut réussir à détecter s'il a un ou plusieurs voisins qui ont la même étiquette, alors que ce n'est pas possible dans d'autres modèles. Les algorithmes qu'on propose sont inspirés de l'algorithme de Mazurkiewicz, au sens où, un sommet essaie d'obtenir le plus d'information possible à propos de son voisinage. Il faut toutefois noter que les algorithmes proposés ne sont pas des adaptations triviales de l'algorithme de Mazurkiewicz et qu'il faut dans chaque modèle utiliser des méthodes particulières afin de s'assurer que chaque sommet réussisse à collecter toute l'information à propos de ses voisins dont il a besoin. Les caractérisations obtenues permettent de mettre en évidence les relations entre les modèles et la différence entre leurs puissances de calcul respectives.

On cherche ensuite à déterminer quelles connaissances initiales sont nécessaires pour résoudre ces problèmes. Les résultats obtenus nous permettent de mettre en évidence quels sont les résultats existants dans le modèle de Mazurkiewicz [MT00, GM02, GM03, GMM04] qu'on peut espérer pouvoir étendre dans les différents modèles considérés et quels sont les résultats qui sont spécifiques à certains modèles. Cela permet de présenter sous un angle différent les différences entre les modèles considérés.

Dans le Chapitre 3, on étudie les modèles (3), (4) et (6) de la Figure 1. Dans ces trois modèles, les arêtes peuvent être étiquetées (et réétiquetées). On montre que les modèles (3) et (4) ont la même puissance de calcul et que si chaque sommet connaît initialement son degré (i.e., son degré fait partie de son étiquette initiale), ces modèles sont aussi puissant que le modèle (6). La caractérisation des graphes admettant un algorithme de nommage ou d'élection dans ces modèles est basée sur la notion de revêtements, qui sont des homomorphismes de graphes localement bijectifs, où les graphes considérés peuvent avoir des arêtes multiples. On montre que les modèles considérés au Chapitre 3 ont la même puissance de calcul que le modèle étudié par Angluin [Ang80] et que le modèle de communication synchrone dans un système asynchrone. On obtient ainsi la première caractérisation des graphes admettant un algorithme de nommage ou d'élection dans ces modèles à partir des résultats obtenus pour les modèles (3), (4) et (6) de la Figure 1. Une partie des résultats présentés au Chapitre 3 a été publiée dans [CM04].

Lorsque les arêtes ne peuvent pas être étiquetées, on ne peut pas obtenir de résultats d'équivalence correspondant à ceux présentés au Chapitre 3. Dans le Chapitre 4, on étudie le modèle entrelacé de Boldi et al. [BCG$^+$96] qui correspond au modèle (5) de la Figure 1. Les caractérisations des graphes admettant un algorithme d'élection ou de nommage présentées au Chapitre 4 permettent de montrer que le modèle (5) a une puissance de calcul strictement plus faible que le modèle (6).

Dans le Chapitre 5, on étudie le modèle (2) de la Figure 1. Afin de caractériser les graphes admettant un algorithme d'élection ou de nommage dans ce modèle, on introduit

la notion de *pseudo-revêtements*. Grâce à cette caractérisation, on peut montrer que le modèle (2) a une puissance de calcul strictement plus faible que les modèles (3) et (4) et que les puissances de calcul des modèles (2) et (5) sont incomparables. Une partie des résultats présentés au Chapitre 5 a été publiée dans [Cha05].

Dans le Chapitre 6, on étudie le modèle (1) de la Figure 1. La caractérisation des graphes admettant un algorithme de nommage dans ce modèle est basée sur la notion de *submersions* qui sont des homomorphismes localement surjectifs de graphes simples. Cette caractérisation permet de montrer que le modèle (1) a une puissance de calcul strictement plus faible que les modèles (2) et (5). La caractérisation des graphes admettant un algorithme d'élection dans ce modèle repose aussi sur la notion de submersions mais la caractérisation obtenue et l'algorithme d'élection présenté sont assez techniques. Une partie des résultats présentés au Chapitre 6 a été publiée dans [CMZ04] et une version complète est parue dans [CMZ06].

Des Algorithmes plus Efficaces et de Nouvelles Caractérisations

Un autre objectif de ce mémoire est de fournir un nouvel algorithme d'élection dans le modèle où les processus communiquent par échange de messages. Il s'agit d'obtenir un algorithme «à la Mazurkiewicz» dans le modèle de Yamashita et Kameda ; le but étant d'obtenir un algorithme plus efficace et «implémentable».

Dans le Chapitre 7, on considère comme Yamashita et Kameda [YK96b], les systèmes où les processus communiquent par échange de messages. On présente d'abord un codage du réseau qui permet d'exprimer toutes les transitions possibles d'un tel système par des règles de réétiquetages du même type que celles du modèle (4) de la Figure 1. À partir de ce codage et des résultats d'impossibilité du Chapitre 3, on déduit une condition nécessaire que doivent vérifier les graphes qui admettent un algorithme d'élection et de nommage.

On présente ensuite un algorithme de nommage «à la Mazurkiewicz» qui a des propriétés intéressantes que l'algorithme de Yamashita et Kameda n'a pas. Ainsi, notre algorithme est «totalement» polynomial, au sens où le temps d'exécution, le nombre de messages et les tailles des messages échangés sont polynomiaux en la taille du graphe, alors que l'algorithme de Yamashita et Kameda nécessite un nombre polynomial de messages de tailles exponentielles. Par ailleurs, pour tout graphe, on montre qu'il existe une exécution de notre algorithme qui permet d'élire et de nommer dans ce graphe. Ainsi, notre algorithme permet parfois d'exploiter l'asymétrie qui provient de l'exécution et non du graphe.

On explique ensuite comment ces résultats peuvent être généralisés à des modèles plus faibles où les processus communiquent par échange de messages qui ont été étudiés par Yamashita et Kameda dans [YK99].

Une partie des résultats présentés au Chapitre 7 a été publiée dans [CM05]. D'autres résultats ont été obtenus dans ce modèle [CDS06, CGMT07] et sont présentés dans la conclusion du Chapitre 7.

Dans le Chapitre 8, on considère les modèles distribués où les différents sommets du réseau sont passifs et où des agents mobiles qui se déplacent sur le réseau sont en charge de l'exécution de l'algorithme distribué. On montre qu'un tel système à agents mobiles est équivalent à un système distribué où les processus communiquent par échange de messages si les graphes sous-jacents sont identiques. On en déduit une caractérisation des systèmes à

agents mobiles dans lesquels on peut résoudre le problème du rendez-vous, et on généralise ainsi des résultats existants.

Les résultats présentés au Chapitre 8 ont été publiés dans [CGMO06]. D'autres résultats obtenus dans un modèle légèrement différent, qui ont été publiés dans [Cha06], sont présentés dans la conclusion du Chapitre 8.

Complexité

Dans le Chapitre 9, on étudie la complexité de décider si un graphe admet un algorithme d'élection ou de nommage dans les différents modèles considérés dans ce mémoire. On montre qu'à l'exception de deux modèles où les processus communiquent par échange de messages, il est co-NP-complet de décider si un graphe donné admet un algorithme d'élection ou de nommage dans chaque modèle. Ce résultat et le fait que les algorithmes présentés dans les Chapitres 2,3, 4 et 7 sont des algorithmes distribués polynomiaux permettent de mettre en évidence l'aspect non-déterministe d'une exécution distribuée. Les résultats présentés au Chapitre 9 ont été publiés dans [CP06].

Table des matières

Chapitre 1

Préliminaires

Sommaire

Dans ce chapitre, on rappelle d'abord quelques définitions élémentaires sur les graphes non-dirigés et dirigés et on introduit les notations qu'on utilisera par la suite. On présente ensuite les relations de réétiquetage de graphes et on explique comment elles permettent de coder des algorithmes distribués. On présente finalement les problèmes que l'on considère dans ce mémoire.

1.1 Graphes non-dirigés

1.1.1 Définitions

Définition 1.1 *Un* graphe non-dirigé sans boucle *est défini par un ensemble de sommets $V(G)$, un ensemble d'arêtes $E(G)$ et par une fonction ext qui associe à chaque arête deux éléments distincts de $V(G)$, appelés ses* extrémités.

Pour toute arête $e \in E(G)$ et tous sommets $u, v \in E(G)$ tels que $ext(e) = \{u, v\}$, on dit que l'arête e est incidente *à u et à v. Les sommets u et v sont dits* adjacents *ou* voisins.

Par la suite, sauf précision contraire, le terme *graphe* désignera un graphe non-dirigé sans boucle.

Définition 1.2 *Un* graphe simple non-dirigé *est un graphe non-dirigé sans boucle tel qu'il y ait au plus une arête entre deux sommets, i.e., pour tous sommets $u, v \in V(G)$, $|\{e \in E(G) \mid ext(e) \in \{u, v\}\}| \leq 1$.*

Chaque arête $e \in E(G)$ peut alors être vue comme une paire de sommets distincts qui sont ses extrémités.

Par la suite, sauf précision contraire, le terme *graphe simple* désignera un graphe simple non-dirigé.

Définition 1.3 *Un graphe H est un sous-graphe partiel d'un graphe G si $V(H) = V(G)$ et $E(H) \subseteq E(G)$.*

Un graphe H est un sous-graphe de G si $V(H) \subseteq V(G)$ et $E(H) \subseteq E(G)$.

Un graphe H est un sous-graphe induit de G si $V(H) \subseteq V(G)$ et $E(H)$ est l'ensemble des arêtes de G dont les extrémités sont dans $V(H)$, i.e., $\forall e \in E(G), ext(e) \subseteq V(H) \iff e \in E(H)$. Pour tout graphe G et tout ensemble $S \subseteq V(G)$, on note $G[S]$ le sous-graphe induit de G dont les sommets sont les éléments de S.

Définition 1.4 *Dans un graphe G, le* voisinage *d'un sommet u, noté $N_G(u)$, est l'ensemble des voisins de u, i.e., $N_G(u) = \{v \in V(G) \mid \exists e \in E(G)$ telle que $ext(e) = \{u, v\}\}$. On note $I_G(u)$ l'ensemble des arêtes incidentes au sommet u, i.e., $I_G(u) = \{e \in E(G) \mid \exists e \in E(G)$ telle que $u \in ext(e)\}$.*

Dans un graphe G, le degré *d'un sommet u, noté $\deg_G(u)$, est le nombre d'arêtes incidentes à u, i.e., $\deg_G(u) = |I_G(u)|$. En particulier, si G est un graphe simple, alors le degré de u est son nombre de voisins, i.e., $\deg_G(u) = |N_G(u)|$.*

Un graphe G est *d-régulier* si tous les sommets de G sont de degré d ; on dit aussi que G est *régulier*. Dans le cas particulier où tous les sommets ont un degré 0 (le graphe ne contient pas d'arête), on dit que G est un *stable*. Un graphe G est *biparti* si on peut partitionner les sommets de $V(G)$ en deux ensembles V_1 et V_2 tels que $G[V_1]$ et $G[V_2]$ sont des stables. Un graphe biparti G est *(k, l)-semi-régulier* si tous les sommets de V_1 sont de degré k et tous les sommets de V_2 sont de degré l ; on dit aussi que G est *biparti semi-régulier*. Un graphe G admet un *couplage parfait* s'il existe un ensemble d'arêtes $E' \in E(G)$ tel que chaque sommet $v \in V(G)$ soit incident à exactement une arête de E'.

Définition 1.5 *Étant donnés un graphe simple G et un sommet $u \in V(G)$, l'*étoile *de centre u est le sous-graphe de G défini par $V(B_G(u)) = N_G(u)$ et $E(B_G(u)) = I_G(u)$.*

Définition 1.6 *Dans un graphe G, un* chemin Γ *entre deux sommets u et v de G est une suite alternée $(u_0, e_1, u_1, e_2, \ldots, e_n, u_n)$ telle que :*

- *pour tout $i \in [0, n]$, $u_i \in V(G)$,*
- *pour tout $i \in [1, n]$, $e_i \in E(G)$,*
- *pour tout $i \in [1, n]$, $ext(e_i) = \{u_{i-1}, u_i\}$,*
- *$u_0 = u$ et $u_n = v$.*

Les sommets u_0 et u_n sont les extrémités *du chemin Γ et les sommets $u_1, \ldots, u_{n-1}$ sont les* sommets internes *du chemin ; la* longueur *n du chemin est son nombre d'arêtes.*

Un chemin dont toutes les arêtes sont distinctes est dit simple *et un chemin qui ne contient pas deux fois le même sommet est dit* élémentaire.

Lorsque les extrémités d'un chemin simple Γ sont confondues, on dit que Γ est un cycle. *Un* cycle élémentaire *est un cycle dont tous les sommets internes sont distincts.*

Dans un graphe simple, un chemin sera généralement décrit par la suite de ses sommets $(u_0, u_1, \ldots, u_n)$, puisque pour tout $i \in [1, n]$, l'arête e_i est nécessairement l'arête $\{u_{i-1}, u_i\}$. Ainsi, dans un graphe simple G, une suite de sommets $(u_0, u_1, \ldots, u_n)$ est un chemin si pour tout $i \in [1, n]$, $\{u_{i-1}, u_i\} \in E(G)$.

Définition 1.7 *Un graphe G est* connexe *si pour tous sommets $u, v \in V(G)$, il existe un chemin entre u et v dans G.*

Tous les graphes considérés dans ce mémoire seront connexes.

Un *arbre* est un graphe connexe sans cycle. Un *anneau* est un graphe constitué d'un cycle simple. Un arbre couvrant A d'un graphe G est un arbre qui est un graphe partiel de G.

Définition 1.8 *Étant donnés un graphe connexe G et deux sommets $u, v \in V(G)$, la* distance *de u à v dans G, notée $\mathrm{dist}_G(u, v)$, est la longueur du plus court chemin de u à v. Le* diamètre *de G, noté $D(G)$, est la plus grande distance entre deux sommets de G, i.e., $D(G) = \max\{\mathrm{dist}_G(u, v) \mid u, v \in V(G)\}$.*

Définition 1.9 *Un* homomorphisme *φ d'un graphe G dans un graphe H est une application de $V(G)$ dans $V(H)$ et de $E(G)$ dans $E(H)$ qui préserve les relations d'incidence, i.e., pour toute arête $e \in E(G)$, $ext(\varphi(e)) = \varphi(ext(e))$.*

Définition 1.10 *Un homomorphisme φ d'un graphe G dans un graphe H est un* isomorphisme *si φ est bijective.*

On dit alors que G et H sont isomorphes *et on note $G \simeq H$.*

Une *famille* (ou *classe*) de graphes est un ensemble de graphes clos par isomorphisme.

Dans les définitions suivantes, on considère les homomorphismes entre graphes simples. Il est facile de voir que dans ce cas particulier, ces définitions sont équivalentes avec les précédentes.

Définition 1.11 *Un* homomorphisme *φ d'un graphe simple G dans un graphe simple H est une application de $V(G)$ dans $V(H)$ qui préserve les relations d'adjacence entre sommets, i.e., pour toute arête $\{v, w\} \in E(G)$, $\{\varphi(v), \varphi(w)\} \in E(G)$.*

Définition 1.12 *Un homomorphisme φ d'un graphe simple G dans un graphe simple H est un* isomorphisme *si φ est bijective et si φ^{-1} est aussi un isomorphisme.*

1.1.2 Graphes étiquetés

On travaille sur des graphes dont les sommets et les arêtes sont étiquetés par un ensemble d'étiquettes L, qui peut être fini ou infini. On supposera l'ensemble L muni d'un ordre total, noté $<_L$.

Un graphe étiqueté, noté (G, λ), est un graphe G muni d'une fonction d'étiquetage $\lambda : V(G) \cup E(G) \to L$ qui associe une étiquette à chaque sommet $v \in V(G)$ et à chaque arête $e \in E(G)$. Lorsque la fonction d'étiquetage n'aura pas à être explicitée, les graphes $(G, \lambda_G), (H, \lambda_H), \ldots$ seront dénotés $\mathbf{G}, \mathbf{H}, \ldots$; on dit que le graphe G est le graphe *sous-jacent* de $\mathbf{G}$.

On dit que l'étiquetage d'un graphe (G, λ) est *uniforme* s'il existe deux étiquettes $\alpha, \beta \in L$ telles que pour tout sommet $v \in V(G)$, $\lambda(v) = \alpha$ et pour toute arête $e \in E(G)$, $\lambda(v) = \beta$.

Les notions de sous-graphes, sous-graphes induits, sous-graphes partiels s'étendent aux graphes étiquetés en demandant que l'étiquetage soit préservé.

Définition 1.13 *Un graphe $\mathbf{H} = (H, \eta)$ est un graphe partiel (resp. sous-graphe, sous-graphe induit) d'un graphe $\mathbf{G} = (G, \lambda)$ si H est un graphe partiel (resp. sous-graphe, sous-graphe induit)de G et pour tout $x \in V(H) \cup E(H)$, $\lambda(x) = \eta(x)$.*

Un homomorphisme entre deux graphes étiquetés est un homomorphisme entre les graphes sous-jacents qui préserve l'étiquetage.

Définition 1.14 *Un* homomorphisme *φ d'un graphe étiqueté $\mathbf{G} = (G, \lambda)$ dans un graphe $\mathbf{H} = (H, \eta)$ est un homomorphisme de G dans H tel que pour tout $x \in V(G) \cup E(G)$, $\lambda(x) = \eta(\varphi(x))$.*

L'homomorphisme φ est un isomorphisme entre $\mathbf{G}$ et $\mathbf{H}$ si φ définit un isomorphisme entre G et H.

Une *occurence* de $\mathbf{G} = (G, \lambda)$ dans $\mathbf{G}' = (G', \lambda')$ est un isomorphisme entre (G, λ) et un sous-graphe (H, η) de (G', λ').

Remarque 1.15 *On peut assimiler tout graphe non-étiqueté G au graphe étiqueté (G, λ_ϵ) où λ_ϵ est un étiquetage tel que pour tout $x \in V(G) \cup E(G)$, $\lambda_\epsilon(x) = \epsilon$, où ϵ est le mot vide. Par conséquent, le terme « graphe » désignera indistinctement un graphe étiqueté et un graphe non-étiqueté.*

1.1.3 Étiquetages particuliers

Dans les Chapitres 2, 3, 4, 5, 6 et 7, on va introduire des notions d'étiquetages et de colorations qui vont nous permettre d'obtenir des résultats de complexité dans le Chapitre 9.

Les fonctions d'étiquetages considérées n'attribuent des étiquettes qu'aux sommets (les arêtes ne sont pas étiquetées). Une coloration d'un graphe simple G est un étiquetage tel que deux sommets adjacents dans G ont deux couleurs distinctes.

Définition 1.16 *Une* coloration *ℓ d'un graphe simple G est un étiquetage de G tel que pour toute arête $\{u, v\} \in V(G)$, $\ell(u) \neq \ell(v)$.*

Les colorations et étiquetages qu'on introduit dans les chapitres suivants sont définies par des propriétés que doivent vérifier les graphes induits par les sommets de chaque couleur. On considère un graphe simple G et un étiquetage ℓ de G. Pour toute couleur $i \in \ell(V(G))$, on note $G[i]$ le graphe $G[\ell^{-1}(i)]$ qui est le graphe induit par tous les sommets étiquetés i. De plus, pour toutes couleurs distinctes $i, j \in \ell(V(G))$, on note $G[i, j]$ le graphe biparti obtenu à partir du graphe induit $G[\ell^{-1}(i) \cup \ell^{-1}(j)]$ dans lequel on a supprimé toutes les arêtes $\{u, v\}$ telles que $\ell(u) = \ell(v)$.

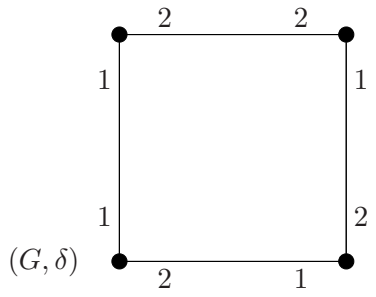


Fig. 2 – Un graphe simple G avec un étiquetage de ports δ.

Par exemple, une coloration d'un graphe G est un étiquetage ℓ d'un graphe G telle que pour toute couleur $i \in \ell(V(G))$, $G[i]$ est un stable, i.e., $G[i]$ ne contient pas d'arêtes. Un graphe G est un couplage parfait s'il admet une coloration ℓ tel que $\ell(V(G)) = \{1, 2\}$ et telle que $G[1, 2]$ soit un graphe 1-régulier.

Un étiquetage est dit propre si au moins deux sommets du graphe ont la même couleur.

Définition 1.17 *Un étiquetage ℓ d'un graphe G est un étiquetage* propre *si $|\ell(V(G))| < |V(G)|$.*

1.1.4 Étiquetage des ports

Dans les chapitres suivants, un système distribué va être représenté par un graphe étiqueté dont les sommets correspondent aux processus et les arêtes aux liens de communication. Dans certains modèles, chaque processus peut distinguer ses voisins, i.e., il peut distinguer les canaux de communication par lesquels il reçoit ou envoie des messages. Pour cela, on suppose que le graphe correspondant au système distribué a un étiquetage des ports défini de la manière suivante.

Dans un graphe simple, un étiquetage des ports est un ensemble de fonctions locales qui permettent à chaque sommet de distinguer ses voisins.

Définition 1.18 *Étant donné un graphe simple* **G***, un étiquetage des ports δ est un ensemble de fonctions $\{\delta_u \mid u \in V(G)\}$ tel que pour tout sommet u, δ_u est une bijection entre $N_G(u)$ et $[1, \deg_G(u)]$.*

On représente un graphe **G** avec un étiquetage δ en représentant le numéro $\delta_u(v)$ sur l'incidence entre le sommet u et l'arête $\{u, v\}$ comme représenté sur la Figure 2.

1.2 Graphes dirigés

Les définitions et notations utilisées ici proviennent du travail de Boldi et Vigna sur les fibrations [BV02a].

1.2.1 Définitions

Définition 1.19 *Un* graphe dirigé *D est défini par un ensemble de* sommets *$V(D)$, un ensemble d'*arcs *$A(D)$ et deux fonctions s_D et t_D de $A(D)$ dans $V(D)$.*

Pour chaque arc $a \in A(D)$, $s_D(a)$ est la source *de l'arc a et $t_D(a)$ est la* cible *de a. On dit que l'arc a est* incident *à $s_D(a)$ et à $t_D(a)$. Si $s_D(a) = t_D(a)$, l'arc a est une* boucle.

Pour tous sommets $u, v \in V(D)$, s'il existe un arc $a \in A(D)$ tel que $s_D(a) = u$ et $t_D(a) = v$, u est un prédecesseur *de v et v est un* successeur *de u.*

Lorsqu'il n'y aura pas d'ambiguité, les fonctions s_D et t_D seront respectivement notées s et t.

Définition 1.20 *Dans un graphe dirigé D, le* voisinage sortant *d'un sommet u, noté $N_D^+(u)$ est l'ensemble des successeurs de u, i.e., $N_D^+(u) = \{v \in V(D) \mid \exists a \in A(D)$ tel que $s(a) = u$ et $t(a) = v\}$. On note $I_D^+(u)$ l'ensemble des arcs sortants de u, i.e., $I_D^+(u) = \{a \in A(D) \mid s(a) = u\}$. Le* degré sortant *de u, noté $d_D^+(u)$ est le nombre d'arcs sortants de u, i.e., $d_D^+(u) = |I_D^+(u)|$.*

Le voisinage entrant *d'un sommet u dans un graphe dirigé D noté $N_D^-(u)$ est l'ensemble des prédecesseurs de u, i.e., $N_D^-(u) = \{v \in V(D) \mid \exists a \in A(D)$ tel que $s(a) = v$ et $t(a) = u\}$. On note $I_D^-(u)$ l'ensemble des arcs entrants de u, i.e., $I_D^-(u) = \{a \in A(D) \mid t(a) = u\}$. Le* degré entrant *de u, noté $d_D^-(u)$ est le nombre d'arcs entrant de u, i.e., $d_D^-(u) = |I_D^-(u)|$.*

Définition 1.21 *Dans un graphe dirigé D, un* chemin *de u à v est une suite d'arcs $(a_0, a_2, \ldots, a_n)$ telle que $s(a_0) = u$, $t(a_n) = v$ et pour tout $i \in [1, n]$, $t(a_{i-1}) = s(a_i)$.*

Définition 1.22 *Un graphe dirigé D est fortement connexe si pour tous sommets u et v, il existe un chemin de u à v dans D.*

Définition 1.23 *Un* graphe dirigé symétrique *est un graphe dirigé D muni d'une involution $Sym : A(D) \to A(D)$ qui à chaque arc $a \in A(D)$ associe son arc symétrique $Sym(a)$ tel que $s_D(Sym(a)) = t_D(a)$.*

Définition 1.24 *Un* homomorphisme *φ d'un graphe dirigé D dans un graphe dirigé D' est une application de $V(D)$ dans $V(D')$ et de $A(D)$ dans $A(D)$ qui commute avec les fonctions source et cible, i.e., pour toute arc $a \in A(D)$, $s_{D'}(\varphi(a)) = \varphi(s_D(a))$ et $s_{D'}(\varphi(a)) = \varphi(s_D(a))$.*

Définition 1.25 *Un homomorphisme φ d'un graphe dirigé D dans un graphe dirigé D' est un* isomorphisme *si φ est bijective.*

On dit alors que D et D' sont isomorphes.

1.2.2 Graphes dirigés étiquetés

Comme les graphes non-dirigés, les graphes dirigés que l'on considère sont étiquetés par un ensemble d'étiquettes L, qui peut être fini ou infini. On supposera l'ensemble L muni d'un ordre total, noté $<_L$.

Comme pour les graphes non-dirigés, un graphe dirigé étiqueté, noté (D, λ) est un graphe dirigé D muni d'une fonction d'étiquetage $\lambda : V(D) \cup A(D) \to L$ qui associe une étiquette à chaque sommet $v \in V(D)$ et à chaque arc $a \in A(D)$. Comme dans le cas des graphes non-dirigés, lorsque les fonctions d'étiquetages n'auront pas à être explicitées, le graphe (D, λ) sera dénoté $\mathbf{D}$; on dira que le graphe dirigé D est le graphe dirigé *sous-jacent* de $\mathbf{D}$.

Un homomorphisme entre deux graphes dirigés étiquetés est un homomorphisme entre les graphes dirigés sous-jacents qui préserve l'étiquetage.

Définition 1.26 *Un* homomorphisme φ *d'un graphe dirigé étiqueté* $\mathbf{D} = (D, \lambda)$ *dans un graphe dirigé* $\mathbf{D'} = (D', \lambda')$ *est un homomorphisme de* D *dans* D' *tel que pour tout* $x \in V(D) \cup A(D)$, $\lambda(x) = \lambda'(\varphi(x))$.

L'homomorphisme φ *est un isomorphisme entre* $\mathbf{D}$ *et* $\mathbf{D'}$ *si* φ *définit un isomorphisme entre* D *et* D'.

Remarque 1.27 *Comme pour les graphes non-dirigés, on peut assimiler tout graphe dirigé non-étiqueté* D *au graphe étiqueté* (D, λ_ϵ) *où* λ_ϵ *est la fonction d'étiquetage qui associe à chaque sommet et à chaque arc de* D *l'étiquette* ϵ *qui est le mot vide.*

On présente maintenant quelques propriétés que peuvent avoir l'étiquetage des arcs d'un graphe dirigé. Les termes employés sont similaires à ceux utilisés en théorie des automates.

Définition 1.28 *Étant donné un graphe dirigé* D*, on dit qu'une fonction d'étiquetage* λ *est* déterministe *(resp.* codéterministe*), si pour tous arcs* $a, a' \in A(D)$ *tels que* $s(a) = s(a')$ *(resp.* $t(a) = t(a')$*),* $\lambda(a) \neq \lambda(a')$.

Lorsqu'un graphe dirigé est symétrique, l'étiquetage des arcs peut refléter cette propriété.

Définition 1.29 *Étant donné un graphe dirigé symétrique* D*, on dit qu'une fonction d'étiquetage* λ *est* symétrique*, s'il existe une fonction* $Sym : L \to L$ *telle que pour tout arc* a, $Sym(\lambda(a)) = \lambda(Sym(a))$.

Par la suite, lorsqu'on considérera des graphes dirigés symétriques ayant un étiquetage symétrique, on supposera que tout arc a a une étiquette de la forme (p, q) telle que $Sym(a) = (q, p)$. Étant donnés un graphe dirigé symétrique D et un étiquetage symétrique λ de D, il est toujours possible d'obtenir un étiquetage λ' de cette forme, en posant pour tout arc $a \in A(D)$, $\lambda'(a) = (\lambda(a), Sym(\lambda(a)))$.

1.2.3 Des Graphes non-dirigés aux Graphes Dirigés

À partir d'un graphe $\mathbf{G} = (G, \lambda)$, on peut construire un graphe dirigé symétrique, noté $Dir(\mathbf{G}) = (Dir(G), \lambda')$, défini comme suit. L'ensemble des sommets de $V(Dir(G))$ est l'ensemble $V(G)$ et pour chaque arête $e \in E(G)$ dont les extrémités sont u et v, il existe deux arcs $a_{e,u,v}$ et $a_{e,v,u}$ dans $A(Dir(G))$ tels que $s(a_{e,u,v}) = t(a_{e,v,u}) = u$, $s(a_{e,v,u}) = t(a_{e,u,v}) = v$ et $Sym(a_{e,u,v}) = a_{e,v,u}$. Pour tout sommet $u \in V(G) = V(Dir(G))$, $\lambda'(u) = \lambda(u)$ et pour toute arête $e \in E(G)$ dont les extrémités sont u et v, $\lambda'(a_{e,u,v}) = \lambda'(a_{e,v,u}) = \lambda(e)$

En général, on manipulera les graphes dirigés symétriques obtenus à partir de graphes simples. Dans ce cas là, pour tout graphe simple, $\mathbf{G} = (G, \lambda)$, on peut définir $Dir(\mathbf{G}) = (Dir(G), \lambda')$ de la manière suivante. L'ensemble des sommets de $V(Dir(G))$ est l'ensemble $V(G)$ et pour chaque arête $\{u, v\} \in E(G)$, il existe deux arcs $a_{u,v}$ et $a_{v,u}$ dans $A(Dir(G))$ tels que $s(a_{u,v}) = t(a_{v,u}) = u$, $s(a_{v,u}) = t(a_{u,v}) = v$ et $Sym(a_{u,v}) = a_{v,u}$. Pour tout sommet $u \in V(G)$, $\lambda'(u) = \lambda(u)$ et pour toute arête $\{u, v\}$, $\lambda'(a_{u,v}) = \lambda'(a_{v,u}) = \lambda(\{u, v\})$. Il est facile de voir que dans le cas des graphes simples, cette construction permet d'obtenir le même graphe dirigé symétrique que la précédente.

Un exemple de cette construction est présenté sur la Figure 3.

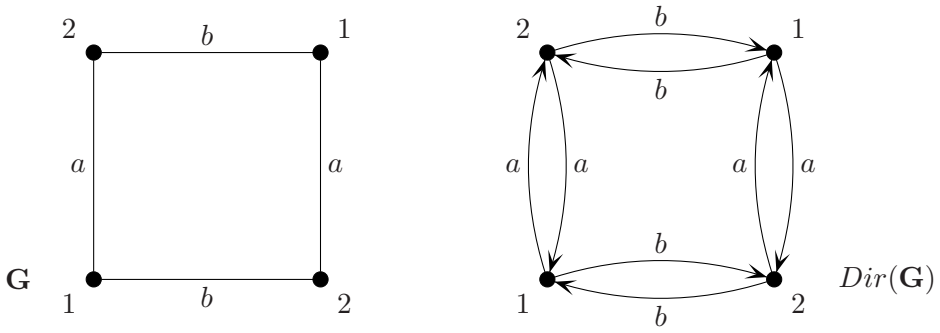


FIG. 3 – Un graphe simple **G** et le graphe dirigé symétrique $Dir(\mathbf{G})$ correspondant.

1.3 Réétiquetages

Dans les Chapitres 2, 3, 4, 5 et 6, on va modéliser des algorithmes distribués par des relations de réétiquetage de graphe. On présente ici les modèles les plus généraux qu'on considère. Les modèles plus restrictifs sont introduits dans les chapitres où ils sont étudiés.

1.3.1 Définitions

Étant donnée une relation binaire $\mathcal{R}$ sur les graphes étiquetés, $\mathcal{R}$ définit une relation de *réécriture* de graphes. On ne considère que des relations closes par isomorphisme, i.e., si $\mathbf{G}\ \mathcal{R}\ \mathbf{H}$ et $\mathbf{G} \simeq \mathbf{G}'$, alors il existe un graphe $\mathbf{H}'$ tel que $\mathbf{G}'\ \mathcal{R}\ \mathbf{H}'$ et $\mathbf{H} \simeq \mathbf{H}'$.

Par ailleurs, on ne considère que des relations de *réétiquetage* de graphes qui ne modifient pas la structure du graphe mais seulement son étiquetage.

Définition 1.30 *Une relation de réécriture de graphes $\mathcal{R}$ est une* relation de réétiquetage *de graphes si pour tout couple de graphes en relations, les graphes sous-jacents sont égaux, i.e.,*

$$(G, \lambda)\ \mathcal{R}\ (G', \lambda') \implies G = G'.$$

On note $\mathcal{R}^*$ la fermeture réflexive et transitive de $\mathcal{R}$. La relation $\mathcal{R}$ est *noetherienne* s'il n'existe pas de chaîne infinie $(G, \lambda_1)\ \mathcal{R}\ (G, \lambda_2)\ \mathcal{R} \ldots \mathcal{R}\ (G, \lambda_i)\ \mathcal{R} \ldots$.

Dans la suite de ce mémoire, on ne considère que des relations de réétiquetage récursives et cela pour considérer des modèles de calcul ayant une puissance de calcul raisonnable.

1.3.2 Relations de Réétiquetage sur les Arêtes

Une relation de réétiquetage est localement engendrée sur les arêtes si sa restriction aux arêtes détermine son comportement sur tout le graphe.

Définition 1.31 *Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les arêtes si la condition suivante est satisfaite. Pour tous graphes (G, λ), (G, λ'), (H, η), (H, η') et toutes arêtes $e \in E(G)$ et $f \in E(H)$ telles que $ext(e) = \{v_1, v_2\}$ et $ext(f) = \{w_1, w_2\}$, si les trois conditions suivantes sont vérifiées :*

1. *$\lambda(v_1) = \eta(w_1)$, $\lambda(v_2) = \eta(w_2)$, $\lambda(e) = \eta(f)$, $\lambda'(v_1) = \eta'(w_1)$, $\lambda'(v_2) = \eta'(w_2)$ et $\lambda'(e) = \eta'(f)$,*

FIG. 4 – Une règle de réétiquetage d'arête.

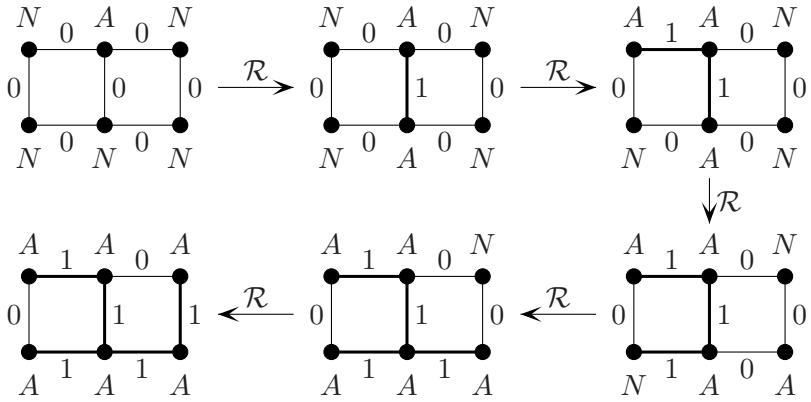


FIG. 5 – Une exécution de l'algorithme décrit par la règle de la Figure 4.

2. *pour tout sommet* $v \in V(G)$ *différent de* v_1 *et de* v_2, $\lambda(v) = \lambda'(v)$ *et pour toute arête* $e' \in E(G)$ *différente de* e, $\lambda(e') = \lambda'(e')$,
3. *pour tout sommet* $w \in V(H)$ *différent de* w_1 *et de* w_2, $\lambda(w) = \lambda'(w)$ *et pour toute arête* $f' \in E(H)$ *différente de* f, $\lambda(f') = \lambda'(f')$,

alors $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ *si et seulement si* $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

Dans les Chapitres 3, 5 et 6, on considère des relations de réétiquetages sur les arêtes (avec parfois des contraintes supplémentaires).

Étant donné une relation de réétiquetage $\mathcal{R}$, un *pas de calcul* sur le graphe $\mathbf{G}$ est la modification de l'étiquetage d'une arête de $\mathbf{G}$ pour obtenir un graphe $\mathbf{G}'$ tel que $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$. Une *exécution* de $\mathcal{R}$ sur $\mathbf{G}$ est alors une suite $\mathbf{G} = \mathbf{G}_0\ \mathcal{R}\ \mathbf{G}_1\ \mathcal{R}\ \ldots\ \mathcal{R}\ \mathbf{G}_i\ \mathcal{R}\ \ldots$ Le graphe étiqueté $\mathbf{G}_i$ est la *configuration* du graphe à l'étape i de l'exécution. Une configuration $\mathbf{G}$ est *finale* s'il n'existe aucun $\mathbf{G}'$ tel que $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$. On remarque que si $\mathcal{R}$ est noethérienne, il n'existe pas d'exécution infinie de $\mathcal{R}$ sur $\mathbf{G}$ et toute exécution atteint donc une configuration finale.

Une relation de réétiquetage localement engendrée sur les arêtes peut être décrite par un ensemble récursif de règles de la forme présentées sur la Figure 4. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage localement engendrée sur les arêtes. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

Exemple 1.32 *On présente sur la Figure 5, une exécution de l'algorithme décrit par la règle de réétiquetage présentée sur la Figure 4. Pour tout graphe connexe* $\mathbf{G}$ *dans lequel il y a un unique sommet étiqueté* A *et où tous les autres sommets sont étiquetés* N, *toute exécution de cet algorithme sur* $\mathbf{G}$ *permet d'atteindre une configuration finale* $\mathbf{G}'$ *où l'ensemble des arêtes étiquetées* 1 *définit un arbre couvrant de* $\mathbf{G}'$.

Remarque 1.33 *On sait que toute exécution de l'algorithme défini par la règle de la Figure 4 termine sur tout graphe* $\mathbf{G}$ *où un seul sommet est étiqueté* A *et où tous les autres sommets sont étiquetés* N. *Cependant, on remarque que dans la configuration finale, aucun*

sommet ne peut déduire à partir de son état que l'exécution est terminée : la terminaison de l'algorithme est dite implicite.

Lorsqu'un sommet peut détecter à partir de son état que le résultat global de l'exécution a été calculé, on parle de terminaison explicite. *On reviendra sur ces différentes notions de terminaison dans la Section 1.5.*

1.3.3 Relations de Réétiquetage sur les Étoiles

Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les étoiles fermées si sa restriction aux étoiles détermine son comportement sur tout le graphe.

Définition 1.34 *Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les étoiles si la condition suivante est satisfaite. Pour tous graphes (G,λ), (G,λ'), (H,η), (H,η'), pour tous sommets $v \in V(G)$ et $w \in V(H)$ tels qu'il existe un isomorphisme $\varphi : B_G(v) \to B_G(w)$, si les conditions suivantes sont vérifiées :*

1. *pour tout $x \in V(B_G(v)) \cup E(B_G(v))$, $\lambda(x) = \eta(\varphi(x))$ et $\lambda'(x) = \eta'(\varphi(x))$,*
2. *pour tout $x \notin V(B_G(v)) \cup E(B_G(v))$, $\lambda'(x) = \lambda(x)$,*
3. *pour tout $x \notin V(B_H(v)) \cup E(B_H(v))$, $\eta'(x) = \eta(x)$,*

alors $(G,\lambda)\ \mathcal{R}\ (G,\lambda')$ si et seulement si $(H,\eta)\ \mathcal{R}\ (H,\eta')$.

Dans les Chapitres 2, 3 et 4, on considère des relations de réétiquetages sur les étoiles (avec parfois des contraintes supplémentaires).

Étant donné une relation de réétiquetage $\mathcal{R}$, un *pas de calcul* sur le graphe $\mathbf{G}$ est la modification de l'étiquetage d'une étoile de $\mathbf{G}$ pour obtenir un graphe $\mathbf{G}'$ tel que $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$. Les notions, d'*exécution*, de *configuration* et de *configuration finale* sont définies de la même manière que pour les relations de réétiquetage localement engendrées sur les arêtes.

Une relation de réétiquetage localement engendrée sur les étoiles peut être décrite par un ensemble récursif de règles de réétiquetage où chaque règle permet de modifier les étiquettes d'une étoile du graphe en fonction seulement des étiquettes apparaissant dans l'étoile. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage localement engendrée sur les étoiles. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

Une relation de réétiquetage localement engendrée sur les étoiles sera généralement décrite par un ensemble de règles «génériques». Une règle générique permet de décrire la règle de réétiquetage d'une étoile, quel que soit le degré du centre de l'étoile. En général, on considère une boule générique $(B(v_0),\lambda)$ de centre v_0 et la règle est décrite par une précondition portant sur les étiquettes présentes dans $(B(v_0),\lambda)$ et un réétiquetage $(B(v_0),\lambda')$. La règle peut être appliquée dans un graphe $\mathbf{G}$ sur une étoile $B_G(u)$ de centre u si la précondition est vérifiée par $B_G(u)$ et les étiquettes des sommets de $B_G(u)$ sont alors modifiées en fonction du réétiquetage λ'. En général, on ne mentionne pas dans le réétiquetage les étiquettes des sommets qui ne sont pas modifiées.

Par exemple, on considère l'algorithme décrit par les deux règles $\mathcal{E}_1$ et $\mathcal{E}_2$ présentées ci-dessous. Cet algorithme est un algorithme d'élection pour les arbres où tous les sommet sont initialement étiquetés A.

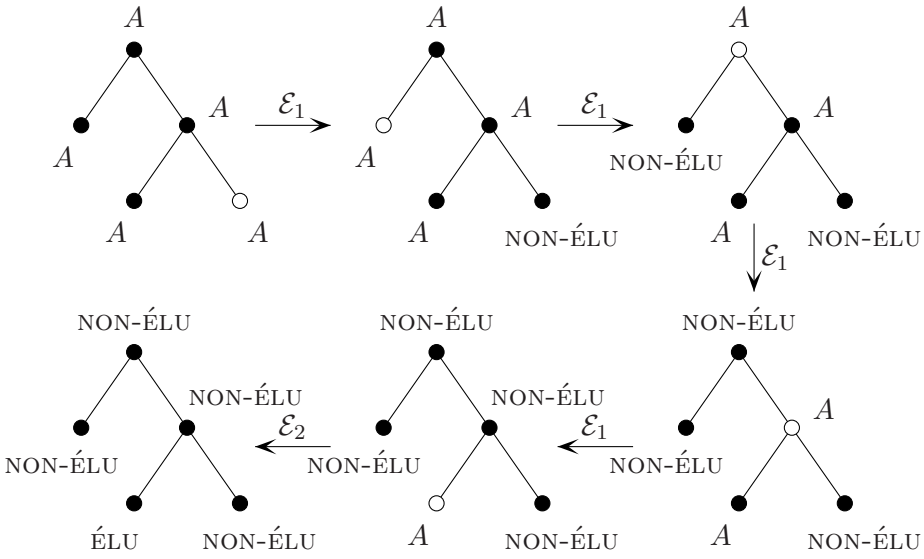


FIG. 6 – Une exécution de l'algorithme décrit par les règles $\mathcal{E}_1$ et $\mathcal{E}_2$. Pour chaque configuration, le centre de l'étoile sur laquelle est appliquée la règle $\mathcal{E}_i$ pour passer à la configuration suivante est le sommet blanc.

$\mathcal{E}_1$: **Règle d'Élagage**

Précondition :
- $\lambda(v_0) = A$,
- $\exists! v \in V(B(v_0)) \setminus \{v_0\}$ tel que $\lambda(v) = A$.

Réétiquetage :
- $\lambda'(v_0) :=$ NON-ÉLU.

$\mathcal{E}_2$: **Règle d'Élection**

Précondition :
- $\lambda(v_0) = A$,
- $\nexists v \in V(B(v_0)) \setminus \{v_0\}$ tel que $\lambda(v) = A$.

Réétiquetage :
- $\lambda'(v_0) :=$ ÉLU.

La règle $\mathcal{E}_1$ peut être appliquée dans un graphe $\mathbf{G}$ sur une étoile $B_G(u)$ de centre u si l'étiquette de u est A et si u a un unique voisin étiqueté A. Lorsque cette règle est appliquée, seule l'étiquette de u est modifiée et devient NON-ÉLU.

La règle $\mathcal{E}_2$ peut être appliquée dans un graphe $\mathbf{G}$ sur une étoile $B_G(u)$ de centre u si l'étiquette de u est A et si u n'a aucun voisin étiqueté A. Lorsque cette règle est appliquée, seule l'étiquette de u est modifiée et devient ÉLU.

Une exécution de cet algorithme est présentée sur la Figure 6. On va montrer que pour tout arbre $\mathbf{T} = (T, \lambda)$ où tous les sommets sont étiquetés A (i.e., $\forall v \in V(T), \lambda(v) = A$), toute exécution de l'algorithme décrit par les règles $\mathcal{E}_1$ et $\mathcal{E}_2$ permet de résoudre l'élection dans $\mathbf{T}$.

On considère un arbre $\mathbf{T}$ et une exécution de l'algorithme décrit précédemment sur $\mathbf{T}$. Pour chaque étape i de l'exécution, on note $\lambda(v)$ l'étiquette du sommet v après le $i$ème pas de réétiquetage.

On observe qu'à chaque application d'une des deux règles, le nombre de sommets qui n'ont pas d'étiquettes finales diminue strictement. On est donc assuré que toute exécution de l'algorithme termine. On montre dans le lemme suivant un invariant qui permet de prouver la correction de l'algorithme.

Lemme 1.35 *Pour toute étape i, le graphe induit par l'ensemble des sommets étiquetés A est un arbre et s'il existe un sommet étiqueté A alors aucun sommet n'a l'étiquette* ÉLU.

Preuve : On montre ce lemme par récurrence sur i. Initialement, tous les sommets sont étiquetés A et puisque $\mathbf{T}$ est un arbre, la propriété est bien vérifiée. On suppose que la propriété est vérifiée à l'étape i.

Si la règle $\mathcal{E}_1$ est appliquée à l'étape $i+1$ sur l'étoile $B_T(v)$ de centre v, alors v est une feuille de l'arbre induit par les sommets étiquetés A à l'étape i. Par conséquent, à l'étape $i+1$, le graphe induit par les sommets étiquetés A est toujours un arbre et aucun sommet n'a l'étiquette ÉLU.

Si la règle $\mathcal{E}_2$ est appliquée à l'étape $i+1$ sur l'étoile $B_T(v)$ de centre v, cela signifie que v n'a aucun voisin étiqueté A et par conséquent, l'arbre induit par les sommets étiquetés A à l'étape i ne contient que le sommet v. À l'étape $i+1$, il n'y a aucun sommet étiqueté A et la propriété est vraie. □

On considère la configuration finale d'une exécution de l'algorithme sur $\mathbf{T}$. S'il existe encore des sommets étiquetés A, alors d'après le Lemme 1.35, le graphe induit par les sommets étiquetés A est un arbre et ou bien, cet arbre est réduit à un sommet v et la règle $\mathcal{E}_2$ peut être appliquée sur l'étoile $B_T(v)$ de centre v, ou bien il existe un sommet v qui est une feuille dans cet arbre et la règle $\mathcal{T}_1$ peut être appliquée sur l'étoile $B_T(v)$ de centre v. Par conséquent, dans la configuration finale, tous les sommets ont l'étiquette ÉLU ou NON-ÉLU. De plus, puisqu'à chaque étape, l'étiquette d'un seul sommet est modifié, le dernier sommet qui a changé d'étiquette a nécessairement pris l'étiquette ÉLU, et d'après le Lemme 1.35, les autres sommets ont l'étiquette NON-ÉLU.

L'algorithme décrit par les règles $\mathcal{E}_1$ et $\mathcal{E}_2$ permet donc de résoudre le problème de l'élection dans la famille des arbres.

1.3.4 Sérialisation

Les notions d'exécution introduites ci-dessus correspondent à des exécutions *séquentielles* des algorithmes. Cependant, il faut noter que si des règles de réétiquetage peuvent être appliquées sur deux arêtes (ou étoiles) disjointes, i.e., qui n'ont aucun sommet en commun, alors ces règles peuvent être appliquées simultanément. Ainsi, on peut définir une exécution distribuée en disant que deux pas de réétiquetages consécutifs appliqués à des arêtes (ou des étoiles) disjointes peuvent être appliqués dans n'importe quel ordre. On dit que de tels pas commutent et peuvent être appliqués de manière concurrente.

Plus généralement, deux suites de réétiquetage partant du même graphe étiqueté et telles qu'on peut obtenir l'une à partir de l'autre par ce type de commutation aboutiront au même résultat, i.e., au même graphe étiqueté final. Ainsi, la notion d'exécution définie par

une suite de réétiquetages peut être vue comme une *sérialisation* [Maz87] d'une exécution distribuée donnée.

Pour l'analyse de nos algorithmes, on considérera des exécutions séquentielles, mais il est important de se rappeler qu'elles peuvent être exécutées de manière distribuée.

1.3.5 ViSiDiA

Au LaBRI, sous la direction de Mohamed Mosbah, un projet logiciel pour la simulation et la visualisation d'algorithmes distribués est en cours depuis quelques années. Ce projet s'appelle ViSiDiA (Visualization and Simulation of Distributed Algorithms) et est disponible sur le site `http://www.labri.fr/projet/visidia`.

Cet outil permet d'implémenter des algorithmes distribués dans différents modèles. Un algorithme est généralement implémenté dans un système asynchrone où les processus communiquent en échangeant des messages. Cependant, il est aussi possible d'implémenter des algorithmes codés par des relations de réétiquetage de graphes ; pour cela, les méthodes de synchronization locale probabilistes présentées dans [MSZ02, MSZ03] sont utilisées. L'algorithme de Mazurkiewicz présenté dans le Chapitre 2 a en particulier été implémenté sous ViSiDiA par Mosbah et Sellami [MS01]. On a aussi implémenté l'algorithme d'énumération qu'on présente dans le Chapitre 7 où les processus communiquent par échange de message.

Pour plus de détails sur l'outil ViSiDiA, on peut se référer au Chapitre 5 de la thèse d'Afif Sellami [Sel04] et au Chapitre 6 de la thèse Bilel Derbel [Der06].

1.4 Élection et Nommage

Le problème de l'*élection* est un problème fondamental en algorithmique distribuée qui a été pour la première fois étudié par Lelann [LeL77]. Un algorithme distribué permet de résoudre le problème de l'élection sur un graphe $\mathbf{G}$ si toute exécution de l'algorithme sur $\mathbf{G}$ termine et si dans la configuration finale, il existe exactement un sommet $v \in V(G)$ qui est dans l'état ÉLU, alors que tous les autres sommets de $\mathbf{G}$ sont dans l'état NON-ÉLU. On suppose par ailleurs, que les états ÉLU et NON-ÉLU sont *finaux*, i.e., une fois qu'un sommet est dans l'un de ces états, alors son état n'est plus modifié jusqu'à la fin de l'exécution de l'algorithme. Le problème de l'élection est important en algorithmique distribuée puisqu'une fois qu'un sommet a été élu, ce sommet peut être utilisé pour centraliser ou diffuser de l'information, pour initialiser l'exécution d'un autre algorithme qui nécessite un sommet distingué (comme par exemple, l'algorithme de calcul d'un arbre couvrant présenté dans la Section 1.3.2), pour prendre une décision de manière centralisée, etc.

Le but d'un algorithme de *nommage* est d'arriver dans une configuration finale où un identifiant unique est associé à chaque sommet. Ce problème aussi est très important en algorithmique distribuée, puisque de nombreux algorithmes distribués fonctionnent correctement sous l'hypothèse que tous les sommets ont des identifiants uniques. Le problème de l'*énumération* est une variante du problème de nommage. Le but d'un algorithme d'énumération pour un graphe $\mathbf{G}$ est d'attribuer à chaque sommet de $\mathbf{G}$ un entier de telle sorte que dans la configuration finale, l'ensemble des numéros des sommets de $\mathbf{G}$ soit exactement $[1, |V(G)|]$. L'algorithme de Mazurkiewicz [Maz97] qu'on va présenter dans le Chapitre 2 est un algorithme d'énumération codé par une relation de réétiquetage localement engendrée sur les étoiles.

Les problèmes du nommage et de l'élection sont équivalents dans certains modèles, comme ceux étudiés aux Chapitre 2, 3, 5 et 7. Il existe cependant des modèles où il existe strictement plus de graphes admettant un algorithme d'élection que de graphes admettant un algorithme de nommage ; c'est le cas pour les modèles étudiés dans les Chapitres 4 et 6.

1.5 Terminaison Implicite et Explicite

On considère un algorithme distribué $\mathcal{A}$ et un graphe $\mathbf{G}$ tel que toute exécution de $\mathcal{A}$ sur $\mathbf{G}$ termine. On va expliciter les différents types de terminaison qu'on considère par la suite.

La terminaison est *implicite* si toute exécution de l'algorithme $\mathcal{A}$ termine, i.e., aucun pas de calcul ne peut plus être effectué sur le graphe $\mathbf{G}$. Dans ce cas là, les sommets du graphe ne peuvent pas forcément détecter à partir de leurs états que l'exécution de l'algorithme est globalement terminée.

Cependant, si on dispose d'un algorithme de nommage $\mathcal{A}$ et qu'on veut utiliser les identités assignées à chaque sommet par $\mathcal{A}$ pour pouvoir exécuter un algorithme $\mathcal{A}'$ qui nécessite que tous les sommets aient des identifiants distincts, il faut que les sommets puissent détecter qu'ils ont obtenus leurs numéros finaux afin de pouvoir les utiliser pour exécuter l'algorithme $\mathcal{A}'$.

En général, lorsqu'un algorithme distribué est exécuté sur un graphe $\mathbf{G}$, chaque sommet $v \in V(G)$ utilise des variables pour stocker des informations qui sont nécessaires à l'exécution de l'algorithme, mais qui ne font pas partie du «résultat» de l'algorithme (ni de ses spécifications). Dans les modèles qu'on considère, l'étiquette de chaque sommet représente son état et les valeurs de ces variables supplémentaires apparaissent dans son étiquette.

Ainsi, si un algorithme $\mathcal{A}$ utilisant un ensemble d'étiquettes L résout un problème $\mathcal{P}$ sur un graphe $\mathbf{G}$, on suppose qu'il existe un ensemble S d'étiquettes et une fonction $\mathbf{res} : L \to S$ tels que pour toute exécution ρ de $\mathcal{A}$ sur $\mathbf{G}$ termine et l'étiquetage final λ_ρ de G est tel que l'étiquetage $\mathbf{res} \circ \lambda_\rho$ est une solution de $\mathcal{P}$ sur $\mathbf{G}$.

Parfois, il n'est pas possible de détecter que les étiquettes de tous les sommets ne vont plus être modifiées, mais un sommet peut détecter que pour tout sommet v, $\mathbf{res}(v)$ ne sera plus modifiée dans la suite de l'exécution. Dans ce cas là, on dit qu'on peut détecter la terminaison de l'algorithme et on parle de terminaison *explicite*.

Dans les chapitres suivants, on va présenter des algorithmes qui permettent de résoudre des problèmes (avec détection de la terminaison) sur des familles de graphes, i.e., l'algorithme permet de résoudre (avec détection de la terminaison) le problème sur tous les graphes de la famille. La définition suivante présente une définition formelle de la notion de *détection de la terminaison* étendue aux familles de graphes.

Définition 1.36 *Un algorithme $\mathcal{A}$ utilisant un ensemble d'étiquette L permet de résoudre un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison s'il existe un ensemble S, une fonction $\mathbf{res} : L \to S$ et un ensemble $L_f \subseteq L$ tels que les propriétés suivantes sont toujours vérifiées pour tout graphe $\mathbf{G} \in \mathcal{F}$.*

- *Toute exécution ρ de $\mathcal{A}$ sur $\mathbf{G}$ termine et l'étiquetage final λ_ρ de G est tel que l'étiquetage $\mathbf{res} \circ \lambda_\rho$ est une solution de $\mathcal{P}$ sur $\mathbf{G}$.*

- *Pour toute exécution ρ de $\mathcal{A}$ sur $\mathbf{G}$, il existe un sommet v et une étape i tels que $\lambda_i(v) \in L_f$ (où $\lambda_i(v)$ est l'étiquette de v après la $i$ème étape de l'exécution ρ).*
- *Pour toute exécution ρ de $\mathcal{A}$ sur $\mathbf{G}$, s'il existe un sommet $v \in V(G)$ et une étape i telle que $\lambda_i(v) \in L_f$, alors pour tout $i' > i$, $\mathbf{res} \circ \lambda_{i'} = \mathbf{res} \circ \lambda_i$.*

Remarque 1.37 *Si un algorithme $\mathcal{A}$ permet de résoudre un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison, alors pour toute exécution de $\mathcal{A}$ sur un graphe $\mathbf{G} \in \mathcal{F}$, un sommet peut détecter que tous les sommets ont calculé leurs valeurs finales, mais il ne peut pas forcément détecter que l'exécution est terminée. On assure toutefois dans la Définition 1.36 que toute exécution de $\mathcal{A}$ sur un graphe $\mathbf{G} \in \mathcal{F}$ termine. Par ailleurs, si on veut exécuter un second algorithme qui utilise le résultat du calcul de $\mathcal{A}$, un sommet peut commencer à exécuter le second algorithme une fois qu'il sait que tous les sommets ont calculé leurs valeurs finales dans l'exécution de $\mathcal{A}$.*

1.6 Connaissances Initiales

Dans les chapitres suivants, on va caractériser les graphes qui admettent des algorithmes d'élection ou de nommage dans différents modèles. On va aussi étudier quelles sont les connaissances sur le graphe dont on doit disposer afin de pouvoir obtenir un algorithme qui permet de résoudre ces problèmes.

Exemple 1.38 *Si on sait qu'un graphe $\mathbf{G}$ est un arbre, alors il existe un algorithme codé par une relation de réétiquetage localement engendrée sur les étoiles qui permet de résoudre l'élection dans $\mathbf{G}$: c'est l'algorithme présenté dans la Section 1.3.3.*

Par la suite, on va distinguer deux types de connaissances initiales : les connaissances *globales* et les connaissances *locales*. Une connaissance est *globale* si c'est une connaissance qui porte sur le graphe $\mathbf{G}$. Par exemple, savoir si le graphe $\mathbf{G}$ sur lequel est exécuté un algorithme est un arbre est une connaissance globale. Un algorithme $\mathcal{A}$ permet de résoudre un problème $\mathcal{P}$ sur une famille avec une connaissance globale $\mathcal{C}$ si pour tout graphe $\mathbf{G}$ tel que $\mathcal{C}$ est une propriété de $\mathbf{G}$ ($\mathbf{G}$ est un arbre, par exemple), l'algorithme $\mathcal{A}$ permet de résoudre le problème $\mathcal{P}$ sur $\mathbf{G}$.

Dans [Sak99], Sakamoto étudie différents types de connaissances initiales globales et étudie quelles sont les informations qui peuvent être déduites à partir d'autres connaissances initiales. Dans les chapitres suivants, on va plus particulièrement étudier les connaissances globales suivantes et étudier ce qu'elles permettent de calculer.

- la topologie du graphe : l'algorithme dépend de la topologie du graphe ;
- la taille du graphe : l'algorithme dépend de la taille du graphe ;
- une borne serrée B sur la taille du graphe : l'algorithme doit fonctionner pour tout graphe $\mathbf{G}$ tel que $|V(G)| \leq B < 2|V(G)|$;
- une borne B sur la taille (ou le diamètre) du graphe : l'algorithme doit fonctionner pour tout graphe $\mathbf{G}$ tels que $|V(G)| \leq B$ (ou $D(G) \leq B$) ;
- aucune connaissance globale, i.e., l'algorithme doit fonctionner sur tous les graphes.

Une connaissance est *locale* si chaque sommet v d'un graphe $\mathbf{G}$ dispose d'une information qui dépend de v. Par exemple, si initialement chaque sommet connaît son degré, alors on parle de connaissance initiale *locale* : un sommet v ne peut pas déduire quelles

sont les connaissances initiales des autres sommets à partir de l'information dont il dispose initialement. Généralement, les connaissances initiales locales sont stockés dans les étiquettes initiales de chaque sommet. Par exemple, si dans un graphe $\mathbf{G} = (G, \lambda)$, les sommets connaissent initialement leurs degrés, on suppose que pour tout sommet $v \in V(G)$, $\lambda(v) = (d(v), \lambda'(v))$ où $d(v)$ est le degré de v dans $\mathbf{G}$. Par la suite, la seule connaissance initiale locale dont on va étudier l'importance est la connaissance du degré.

Étant donné un problème $\mathcal{P}$, on remarque qu'un algorithme $\mathcal{A}$ permet de résoudre $\mathcal{P}$ sans connaissance initiale si et seulement si $\mathcal{A}$ permet de résoudre $\mathcal{P}$ sur la famille de tous les graphes. De même, si $\mathcal{A}$ permet de résoudre $\mathcal{P}$ avec la connaissance initiale de la taille, alors pour tout entier n, il existe un algorithme $\mathcal{A}_n$ qui permet de résoudre le problème $\mathcal{P}$ sur la famille des graphes de taille n. Par ailleurs, si $\mathcal{A}$ permet de résoudre $\mathcal{P}$ avec la connaissance initiale de la topologie, alors pour tout graphe $\mathbf{G}$, il existe un algorithme (qui dépend de $\mathbf{G}$) qui permet de résoudre $\mathcal{P}$ sur $\mathbf{G}$. Par la suite, on va donc dire indistinctement qu'on peut résoudre un problème $\mathcal{P}$ avec une connaissance initiale ou qu'on peut résoudre $\mathcal{P}$ sur la famille des graphes disposant de la même connaissance initiale (les graphes de taille donnée, par exemple).

Un algorithme $\mathcal{A}$ qui permet de résoudre l'élection (ou le nommage) sur tous les graphes de la famille $\mathcal{F}$ est un algorithme *universel* d'élection (ou de nommage) pour $\mathcal{F}$. Cependant, dans les différents modèles qu'on considère dans les chapitres suivants, il existe des graphes qui n'admettent pas d'algorithme d'élection (ou de nommage). Parfois, on souhaite donc pouvoir détecter si un graphe admet un algorithme d'élection (ou de nommage). Ainsi, un algorithme $\mathcal{A}$ est un algorithme *effectif* d'élection (ou de nommage) pour une famille $\mathcal{F}$ si pour toute exécution de $\mathcal{A}$ sur tout graphe $\mathbf{G} \in \mathcal{F}$, $\mathcal{A}$ résout l'élection (ou le nommage) sur $\mathbf{G}$, ou alors $\mathcal{A}$ détecte qu'il n'existe pas d'algorithme d'élection ou de nommage pour $\mathbf{G}$. Les termes employés sont tirés de [BFFS03a] et la définition suivante rappelle les propriétés souhaitées pour ce type d'algorithme.

Définition 1.39 *Un algorithme $\mathcal{A}$ est un algorithme* universel *d'élection (resp. de nommage) pour une famille $\mathcal{F}$ si pour tout graphe $\mathbf{G} \in \mathcal{F}$, toute exécution de $\mathcal{A}$ sur $\mathbf{G}$ permet de résoudre l'élection (resp. le nommage) sur $\mathbf{G}$.*

Un algorithme $\mathcal{A}$ est un algorithme effectif *d'élection (resp. de nommage) pour une famille $\mathcal{F}$ si pour tout graphe $\mathbf{G} \in \mathcal{F}$, toute exécution de $\mathcal{A}$ sur $\mathbf{G}$ permet ou bien de résoudre l'élection (resp. le nommage) sur $\mathbf{G}$, ou bien de détecter qu'il n'existe pas d'algorithme d'élection (resp. de nommage) pour $\mathbf{G}$.*

Remarque 1.40 *Étant donné un algorithme effectif d'élection (ou de nommage) pour une famille de graphes $\mathcal{F}$, si lors d'une exécution de $\mathcal{A}$ sur un graphe $\mathbf{G} \in \mathcal{F}$, un sommet $v \in V(G)$ est dans un état indiquant qu'il n'existe pas d'algorithme d'élection (ou de nommage) pour $\mathbf{G}$, alors v ne modifiera pas son état par la suite, i.e., les états indiquant qu'on ne peut pas résoudre l'élection (ou le nommage) sont finaux.*

Chapitre 2

Calculs Locaux sur les Étoiles Fermées

Sommaire

2.1 Introduction

Dans ce chapitre, on étudie les *calculs locaux sur les étoiles fermées* qui correspondent aux relations de réétiquetage localement engendrées sur les étoiles introduites au Chapitre 1 (Definition 1.34). Dans ce modèle, les graphes considérés sont des graphes simples. En un pas de calcul, un sommet peut observer l'état de ses voisins, l'état des arêtes qui lui sont incidentes et modifier son état ainsi que l'état de ses voisins et des arêtes qui lui sont incidentes. Dans ce chapitre, on ne présente pas de résultats originaux, mais des résultats obtenus par Mazurkiewicz [Maz97] et par Godard et Métivier [GM02].

2.1.1 Caractérisation

Dans le modèle des calculs locaux sur les étoiles fermées, on présente une caractérisation des graphes admettant un algorithme d'élection et de nommage obtenue par Mazurkiewicz [Maz97] (dans ce modèle, élection et nommage peuvent être résolus sur les mêmes graphes). Cette caractérisation est basé sur la notion de *revêtements simples* (qui correspondent aux homomorphismes de graphes simples localement bijectifs) et utilise un résultat d'impossibilité d'Angluin [Ang80].

Un graphe admet un algorithme d'élection et de nommage utilisant des calculs locaux sur les étoiles fermées si et seulement si il est minimal pour les revêtements simples (Théorème 2.21).

On étudie ensuite les connaissances initiales nécessaires afin de pouvoir résoudre l'élection ou le nommage dans ce modèle. En fait, il suffit de connaître une borne sur le diamètre pour pouvoir élire ou nommer dans un graphe minimal pour les revêtements simples (Théorème 2.30). Cette condition suffisante est un cas particulier des résultats plus généraux de Godard et Métivier [GM02] qui caractérisent les familles de graphes admettant un algorithme universel d'élection. Par ailleurs, la connaissance d'une borne serrée sur la taille permet d'obtenir un algorithme effectif d'élection et de nommage (Théorème 2.31).

Dans la conclusion de ce chapitre, on présente les extensions obtenues par Godard, Métivier, Muscholl et Tel [God02b, GM02, GM03, GMM04, MT00] dans lesquelles l'algorithme de Mazurkiewicz est utilisé.

2.1.2 Travaux Liés

Dans [AG81], Angluin et Gardiner étudient des propriétés combinatoires des revêtements simples, et ils montrent que pour tous graphes G, H réguliers de même degré, il existe un graphe K fini qui est un revêtement simple de G et de H. Dans [Lei82], Leighton généralise ce résultat et montre que pour tous graphes G, H qui ont le même «raffinement de degré», il existe un graphe K fini qui est un revêtement simple de G et de H. La construction de Leighton est utilisé dans [GMM04] pour caractériser les familles de graphes reconnaissables par des calculs locaux sur les étoiles sans connaissance structurelle.

Dans [Bod89], Bodlaender étudie les revêtements simples afin de pouvoir «émuler» un réseau sur un réseau plus petit. Pour certaines classes de graphes, il présente une classification des graphes qui sont des revêtements d'autres graphes de la même classe. De plus, Bodlaender étudie la complexité de décider si un graphe G donné est un revêtement d'un graphe H donné. Depuis, de nombreux résultats ont été publiés sur la complexité de décider si un graphe G donné est un revêtement simple d'un graphe H fixé [AFS91, Kra91, KPT94, KPT97, KPT98].

2.2 Revêtements Simples

Dans ce chapitre, les homomorphismes localement bijectifs de graphe simples, i.e., les revêtements simples, sont les homomorphismes qui permettent de donner des conditions nécessaires que doivent vérifier les graphes admettant un algorithme de nommage ou d'élection utilisant des calculs locaux sur les étoiles fermées.

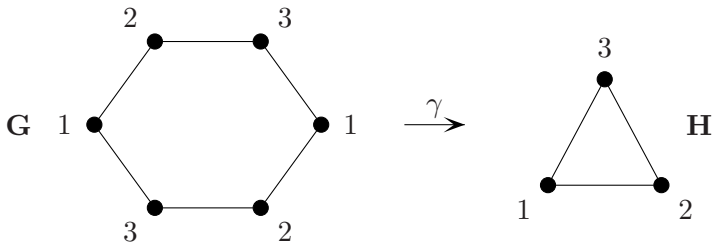


Fig. 7 – Le graphe **G** est un revêtement simple de **H** à travers l'homomorphisme γ qui envoie chaque sommet de **G** étiqueté i sur l'unique sommet de **H** dont l'étiquette est i.

Définition 2.1 *Un graphe simple G est un revêtement d'un graphe simple H à travers un homomorphisme $\gamma\colon G \to H$ si pour tout sommet $v \in V(G)$, γ induit une bijection entre $N_G(u)$ et $N_H(\varphi(u))$, i.e., si les conditions suivantes sont vérifiées :*

- $|N_G(u)| = |N_H(\gamma(u))|$,
- $\gamma(N_G(u)) = N_H(\gamma(u))$.

On dit alors que l'homomorphisme γ est localement bijectif

Un graphe G est un revêtement simple propre *de H si γ n'est pas un isomorphisme et G est* minimal pour les revêtements simples *si G n'est un revêtement propre d'aucun autre graphe.*

Naturellement, un graphe simple étiqueté (G, λ) est un revêtement simple d'un graphe simple étiqueté (H, η) à travers γ si G est un revêtement simple de H à travers γ et si γ conserve l'étiquetage.

Exemple 2.2 *Le graphe* **G** *de la Figure 7 est un revêtement simple de* **H** *à travers l'homomorphisme γ.*

Le graphe **G** *est un revêtement propre de* **H** *et n'est donc pas minimal pour les revêtements simples ; le graphe* **H** *est minimal pour les revêtements simples.*

Dans le lemme suivant, on montre que si **G** est un revêtement simple de **H**, pour tout sommet v de $V(H)$, l'image inverse de l'étoile de **H** centrée en v est une union disjointe d'étoiles de **G**.

Lemme 2.3 *On considère des graphes simples $\mathbf{G}, \mathbf{H}$ tels que* **G** *est un revêtement simple de* **H** *à travers un homomorphisme γ. Soit v un sommet de $V(H)$ et soient u_1, u_2 deux sommets distincts de $\gamma^{-1}(v)$. Pour tous sommets $u_1' \in N_G(u_1) \cup \{u_1\}$ et $u_2' \in N_G(u_2) \cup \{u_2\}$, $u_1' \neq u_2'$.*

Preuve : On considère des graphes simples $\mathbf{G}, \mathbf{H}$ tels que **G** est un revêtement simple de **H** à travers un homomorphisme γ. Soit v un sommet de $V(H)$ et soient u_1, u_2 deux sommets de $\gamma^{-1}(v)$. Puisque **H** est un graphe simple, on sait que $u_2 \notin N_G(u_1)$. S'il existe $u \in N_G(u_1) \cap N_G(u_2)$, alors γ n'est pas localement bijectif en u puisque u a deux voisins qui ont la même image par γ. □

La relation «être revêtement simple» est une relation transitive comme le montre la proposition suivante.

Proposition 2.4 *Pour tous graphes simples $\mathbf{G}, \mathbf{H}, \mathbf{K}$, si* **G** *est un revêtement simple de* **H** *a travers un homomorphisme γ et si* **H** *est un revêtement simple de* **K** *à travers un*

homomorphisme φ, alors **G** *est un revêtement simple de* **K** *à travers l'homomorphisme $\varphi \circ \gamma$.*

Preuve : Il est clair que $\varphi \circ \gamma$ est un homomorphisme de **G** dans **K**. De plus, pour tout sommet $u \in V(G)$, pour tout voisin $w' \in N_K(\varphi(\gamma(u)))$, il existe un unique sommet $v' \in N_H(\gamma(u))$ telle que $\varphi(v') = w'$ et par conséquent, il existe un unique sommet $u' \in N_G(u)$ telle que $\varphi(\gamma(u')) = w'$. Ainsi l'homomorphisme $\varphi \circ \gamma$ est localement bijectif et **G** est un revêtement simple de **K** à travers $\varphi \circ \gamma$. □

Puisqu'on ne considère que des graphes simples connexes, tout homomorphisme localement bijectif est surjectif.

Proposition 2.5 *Si un graphe simple* **G** *est un revêtement simple d'un graphe simple connexe* **H** *à travers γ, alors γ est surjectif.*

Preuve : Soit **G** un revêtement simple d'un graphe simple connexe **H** à travers un homomorphisme γ.

On considère un sommet $v \in \gamma(V(G))$. Il existe donc $u \in V(G)$ tel que $\gamma(u) = v$. Puisque γ induit une bijection entre $N_G(u)$ et $N_H(v)$, pour tout sommet $v' \in N_H(v)$, il existe un sommet $u' \in N_G(u)$ telle que $\gamma(u') = v'$ et donc $v' \in \gamma(V(G))$. Ainsi, puisque **H** est connexe, on sait que γ est un homomorphisme surjectif de **G** dans **H**. □

Si un graphe simple **G** est un revêtement simple d'un graphe simple **H** connexe, alors tous les sommets de **H** ont le même nombre d'antécédents, qui est appelé le *nombre de feuillets* du revêtement.

Proposition 2.6 *Si un graphe simple* **G** *est un revêtement simple d'un graphe simple connexe* **H** *à travers γ, alors il existe une constante q telle que pour tout $v \in V(H)$, $|\gamma^{-1}(v)| = q$.*

Cette constante q est appelée le nombre de feuillets *du revêtement.*

Preuve : On considère un graphe simple **G** qui est un revêtement simple d'un graphe simple connexe **H** à travers un homomorphisme γ. On considère un sommet $v \in V(H)$ et un sommet $v' \in N_H(v)$.

Puisque γ est un homomorphisme localement bijectif, pour tout sommet $u \in \gamma^{-1}(v)$, il existe un unique sommet $u' \in N_G(u)$ tel que $\gamma(u') = v'$. De plus, puisque γ est localement bijectif en u', pour tout sommet $u'' \in N_G(u')$ différent de u, $\gamma(u'') \neq \gamma(u)$. Par conséquent, $|\gamma^{-1}(v)| \leq |\gamma^{-1}(v')|$ et par symétrie, $|\gamma^{-1}(v)| = |\gamma^{-1}(v)|$. Ainsi, puisque le graphe **H** est connexe, pour tout $v, v' \in V(H)$, $|\gamma^{-1}(v)| = |\gamma^{-1}(v')|$. □

Exemple 2.7 *On présente ici quelques exemples de graphes connexes simples qui sont minimaux pour les revêtements simples.*

- *les graphes dont le nombre d'arêtes et le nombre de sommets sont premiers entre eux,*
- *les arbres,*
- *les anneaux de taille première.*

Reidemeister a décrit dans [Rei32] une méthode pour construire tous les revêtements d'un graphe simple H donné. Étant donné un arbre couvrant T de H, Reidemeister a montré que tout revêtement de H à q feuillets peut être obtenu en prenant q copies de T

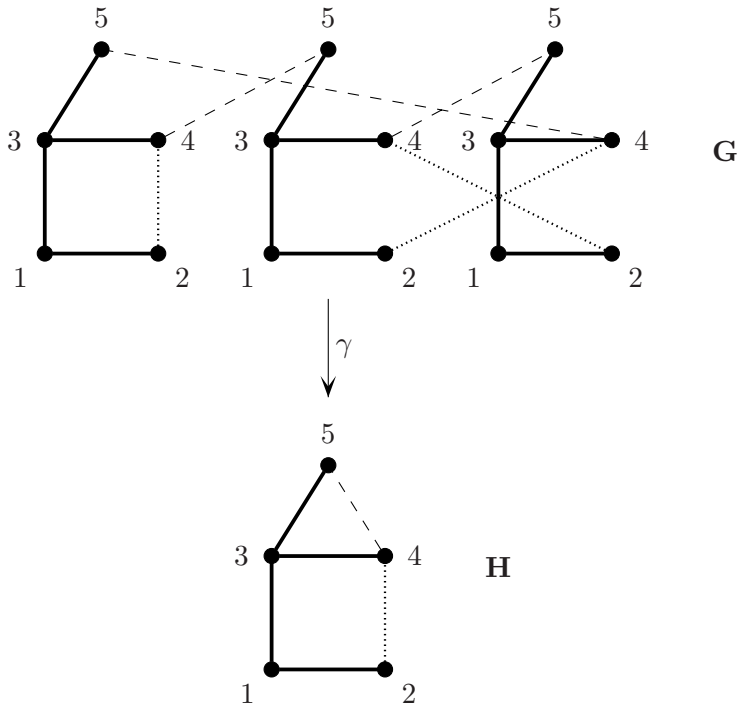


Fig. 8 – Le graphe **G** est obtenu par la construction de Reidemeister d'un revêtement à 3 feuillets de **H**. Les arêtes de l'arbre couvrant de **H** sont représentés en gras et les permutations associées aux arêtes $\{2,4\}$ et $\{4,5\}$ de **H** sont respectivement $\sigma_{(2,4)} = (1)(23)$ et $\sigma_{(4,5)} = (123)$.

et en reliant les copies de T entre elles en fonction de permutations de $[1,q]$ associées à chaque arête de H qui n'est pas dans T.

Théorème 2.8 ([Rei32]) *Soit* $\mathbf{H} = (H, \eta)$ *un graphe simple et* $\mathbf{T}$ *un arbre couvrant de* $\mathbf{H}$. *Un graphe simple* $\mathbf{G}$ *est un revêtement simple de* $\mathbf{H}$ *si et seulement s'il existe un entier* q *et un ensemble* $\Sigma = \{\sigma_{(u,v)} \mid \{u,v\} \in E(H) \setminus E(T)\}$ *de permutations de* $[1,q]$ *(avec la propriété* $\sigma_{(u,v)} = \sigma^{-1}_{(v,u)}$*) tels que* $\mathbf{G}$ *est isomorphe au graphe* $\mathbf{H}_{T,\Sigma} = (H_{T,\Sigma}, \lambda)$ *défini de la manière suivante.*

$$\begin{array}{rcll} V(H_{T,\Sigma}) & = & \{(u,i) \mid v \in V(H) \text{ et } i \in [1,q]\}, & \\ E(H_{T,\Sigma}) & = & \{\{(u,i),(v,i)\} \mid \{u,v\} \in E(T)\} \cup & \text{De plus,} \\ & & \{\{(u,i),(v,\sigma_{(u,v)}(i))\} \mid \{u,v\} \in E(H) \setminus E(T) \text{ et } i \in [1,q]\}. & \end{array}$$

pour tout $(u,i) \in V(H_{T,\Sigma})$, $\lambda((u,i)) = \eta(u)$ *et pour tout* $\{(u,i),(v,j))\} \in E(H_{T,\Sigma})$, $\lambda(\{(u,i),(v,j)\}) = \eta(\{u,v\})$.

Un exemple de construction de revêtement en utilisant le théorème de Reidemeister apparaît sur la Figure 8.

Remarque 2.9 *La construction de Reidemeister permet d'obtenir tous les revêtements à* q *feuillets d'un graphe* $\mathbf{H}$. *Cependant, certains des graphes obtenus ne sont pas connexes et on peut obtenir plusieurs fois certains graphes, puisque certaines permutations produisent des graphes isomorphes.*

On présente maintenant une caratérisation des graphes simples minimaux pour les revêtements simples en terme de coloration. Cette coloration a été introduite par Mazurkiewicz [Maz97] pour caractériser les graphes admettant un algorithme d'énumération utilisant des calculs locaux sur les étoiles fermées. Dans [Maz97], la coloration que nous allons décrire porte le nom de *coloration localement bijective*, mais nous préférerons le terme de *coloration régulière parfaite*. Un étiquetage ℓ d'un graphe simple non-étiqueté est une coloration régulière parfaite si deux sommets voisins ont des couleurs distinctes et si le graphe induit par deux classes de couleurs adjacentes est un couplage parfait.

Définition 2.10 *Une* coloration régulière parfaite *d'un graphe simple* G *est un étiquetage* ℓ *de* G *tel que*

- *pour tout* $i \in \ell(V(G))$, $G[i]$ *est un stable,*
- *pour tout* $i,j \in \ell(V(G))$ *avec* $i \neq j$, $G[i,j]$ *est un stable, ou alors* $G[i,j]$ *est un couplage parfait.*

Dans la proposition suivante, on caractérise les graphes simples qui sont minimaux pour les revêtements simples à l'aide de la notion de coloration régulière parfaite. On rappelle qu'une coloration ℓ est dite propre si $|\ell(V(G))| < |V(G)|$. Ce lien entre les revêtements et les colorations régulières parfaites a été montré par Godard, Métivier et Muscholl dans [GMM04].

Proposition 2.11 ([GMM04]) *Un graphe simple non-étiqueté* G *est minimal pour les revêtements simples si et seulement si* G *n'admet aucune coloration régulière parfaite propre.*

Preuve : On considère un graphe simple non-étiqueté G qui est un revêtement simple propre d'un graphe simple H à travers un homomorphisme γ. On va montrer que γ est une coloration régulière parfaite de G et que l'ensemble des couleurs utilisés est $V(H)$. Puisque H est un graphe simple, pour toute arête $\{u,u'\} \in E(G)$, $\gamma(u) \neq \gamma(u')$. Pour

toute arête $\{v, v'\} \in E(H)$, pour tout sommet $u \in \gamma^{-1}(v)$ (resp. $u' \in \gamma^{-1}(v')$), il existe un unique sommet $u' \in N_G(u) \cap \gamma^{-1}(v')$ (resp. $u \in N_G(u') \cap \gamma^{-1}(v)$) et par conséquent, $G[v, v']$ est un couplage parfait. De plus, puisque γ est un homomorphisme, pour tous sommets $v, v' \in V(H)$ et pour tous sommets $u \in \gamma^{-1}(v)$ et $u' \in \gamma^{-1}(v')$, si $\{v, v'\} \notin E(H)$, alors $\{u, u'\} \notin E(G)$. Ainsi, puisque G est un revêtement simple propre de H, $|V(H)| < |V(G)|$ et γ est une coloration régulière parfaite propre.

On considère maintenant une coloration régulière parfaite propre ℓ d'un graphe simple G. On considère un graphe H défini de la manière suivante : $V(H) = \ell(V(G))$ et pour tous $v, v' \in V(H)$, $\{v, v'\} \in E(H)$ si et seulement si $G[v, v']$ n'est pas un stable. Ainsi, pour toute arête $\{u, u'\} \in E(G)$, $G[\ell(u), \ell(u')]$ n'est pas un stable et $\{\gamma(u), \gamma(u')\} \in E(H) : \ell$ est donc un homomorphisme de G dans H. De plus, pour tout sommet $u \in V(G)$, pour tout sommet $v' \in N_H(\ell(u))$, $G[\ell(u), v']$ est un couplage parfait et il existe donc un unique sommet $u' \in \ell^{-1}(v') \cap N_G(u)$. Ainsi l'homomorphisme ℓ est localement bijectif et puisque $|V(H)| = |\ell(V(G))| < |V(G)|$, G est un revêtement propre de H à travers ℓ. □

2.3 Calculs Locaux sur les Étoiles Fermées

Dans cette partie, on montre que les calculs locaux sur les étoiles fermées correspondent aux relations de réétiquetage localement engendrées sur les étoiles, puis on étudie leurs relations avec les revêtements simples.

2.3.1 Définitions

On rappelle qu'informellement, les calculs locaux sur les étoiles fermées permettent en un pas de calcul à un sommet de modifier son étiquette, les étiquettes de ses voisins et et les étiquettes des arêtes qui lui sont incidentes. L'application d'un tel pas de calcul ne dépend que de son étiquette, des étiquettes de ses voisins et des étiquettes qui lui sont incidentes.

On rappelle la définition des relations de réétiquetage localement engendrée sur les étoiles (Définition 1.34).

Définition 2.12 *Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les étoiles si la condition suivante est satisfaite. Pour tous graphes (G, λ), (G, λ'), (H, η), (H, η'), pour tous sommets $v \in V(G)$ et $w \in V(H)$ tels qu'il existe un isomorphisme $\varphi : B_G(v) \to B_G(w)$, si les conditions suivantes sont vérifiées :*

1. *pour tout $x \in V(B_G(v)) \cup E(B_G(v))$, $\lambda(x) = \eta(\varphi(x))$ et $\lambda'(x) = \eta'(\varphi(x))$,*
2. *pour tout $x \notin V(B_G(v)) \cup E(B_G(v))$, $\lambda'(x) = \lambda(x)$,*
3. *pour tout $x \notin V(B_H(v)) \cup E(B_H(v))$, $\eta'(x) = \eta(x)$,*

alors $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ si et seulement si $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

Par définition, les *calculs locaux sur les étoiles fermées* correspondent aux relations de réétiquetage localement engendrées sur les étoiles.

2.3.2 Revêtements Simples et Calculs Locaux sur les Étoiles Fermées

On présente maintenant le lemme qui met en évidence le lien entre les calculs locaux sur les étoiles fermées et les revêtements simples. Ce lemme a été prouvé par Angluin [Ang80].

Lemme 2.13 (Lemme de relèvement [Ang80]) *On considère un graphe simple* $\mathbf{G}$ *qui est un revêtement simple d'un graphe simple* $\mathbf{H}$ *à travers un homomorphisme* γ *et une relation* $\mathcal{R}$ *de réétiquetage localement engendrée sur les étoiles. Si* $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$*, alors il existe* $\mathbf{G}'$ *tel que* $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$ *et* $\mathbf{G}'$ *est un revêtement simple de* $\mathbf{H}'$ *à travers* γ*.*

Preuve : Il suffit de prouver ce lemme pour un pas de calcul. On considère deux graphes simples (G, λ) et (H, η) tels que (G, λ) est un revêtement simple de (H, η) à travers γ. On considère un pas de réétiquetage qui modifie les étiquettes d'une étoile $B_H(v)$ pour un sommet $v \in V(H)$. On note η' l'étiquetage de H obtenu après l'application de ce pas de réétiquetage.

Puisque γ est un homomorphisme localement bijectif, on sait que pour tout sommet $u \in \gamma^{-1}(v)$, $(B_G(u), \lambda)$ est isomorphe à $(B_H(v), \eta)$. De plus, d'après le Lemme 2.3, on sait que pour tous $u, u' \in \gamma^{-1}(v)$, les étoiles $B_G(u)$ et $B_G(u')$ sont disjointes. On peut donc appliquer le pas de réétiquetage sur chacune des étoiles $B_G(u)$ pour tout $u \in \gamma^{-1}(v)$ de telle sorte que tout sommet $u' \in V(B_G(u))$ soit réétiqueté avec la même étiquette que $\gamma(u')$. On note λ' l'étiquetage de G obtenu après l'application de tous ces réétiquetages. Le graphe (G, λ') ainsi obtenu est un revêtement de (H, η') à travers γ. □

Le diagramme suivant représente la propriété du Lemme 2.13.

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow{\mathcal{R}^*} & \mathbf{G}' \\ \text{revêtement simple}\downarrow & & \downarrow\text{revêtement simple} \\ \mathbf{H} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{H}' \end{array}$$

2.4 Énumération, Nommage et Élection

On s'intéresse maintenant aux problèmes d'élection et de nommage dans le cadre des calculs locaux sur les étoiles fermées. On va montrer que les graphes où on peut résoudre l'élection, le nommage et l'énumération dans ce modèle sont les graphes minimaux pour les revêtements simples. Le fait que les graphes admettant un algorithme d'élection admettent un algorithme de nommage est lié à la Proposition 2.6 : si on arrive à élire un sommet, i.e., à donner à un sommet une étiquette qui n'apparaît qu'une fois, alors on peut donner des étiquettes uniques à tous les sommets.

On donne d'abord le résultat d'impossibilité d'Angluin [Ang80] qui a montré qu'il ne peut pas exister d'algorithme d'élection ou de nommage pour un graphe simple $\mathbf{G}$ qui n'est pas minimal pour les revêtements simples. On présente ensuite l'algorithme d'énumération de Mazurkiewicz [Maz97] qui permet de résoudre l'énumération sur les graphes simples minimaux pour les revêtements simples.

2.4.1 Résultats d'Impossibilité

Proposition 2.14 ([Ang80]) *Soit* **G** *un graphe simple étiqueté qui n'est pas minimal pour les revêtements simples. Il n'existe pas d'algorithme de nommage, d'énumération ou d'élection pour le graphe* **G** *utilisant des calculs locaux sur les étoiles fermées.*

Preuve : On considère un graphe simple étiqueté **H** qui n'est pas isomorphe à **G** et tel que **G** soit un revêtement simple de **H** à travers un un homomorphisme γ. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux sur les étoiles fermées, on considère une exécution de $\mathcal{R}$ sur **H**. Si cette exécution est infinie sur **H**, alors d'après le Lemme 2.13, il existe une exécution infinie de $\mathcal{R}$ sur **G** ; auquel cas, $\mathcal{R}$ n'est ni un algorithme d'énumération, ni de nommage, ni d'élection.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur **H** et on considère la configuration finale $\mathbf{H}'$. D'après le Lemme 2.13, il existe une exécution de $\mathcal{R}$ sur **G** qui permet d'atteindre une configuration $\mathbf{G}'$ telle que $\mathbf{G}'$ est un pseudo-revêtement de $\mathbf{H}'$ à travers γ. Si $\mathbf{G}'$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe un sommet $u \in V(G)$ tel qu'on puisse appliquer une règle de réétiquetage sur l'étoile $B_{G'}(u)$. Dans ce cas là, on peut aussi appliquer cette même règle de réétiquetage sur l'étoile $B_{H'}(\gamma(u))$ et la configuration $\mathbf{H}'$ n'est pas une configuration finale de $\mathcal{R}$. Par conséquent, $\mathbf{G}'$ est une configuration finale de $\mathcal{R}$. Mais puisque $\mathbf{G}'$ n'est pas isomorphe à $\mathbf{H}'$, on sait d'après la Proposition 2.6 que chaque étiquette de $\mathbf{H}'$ apparaît au moins deux fois dans $\mathbf{G}'$ et par conséquent, $\mathcal{R}$ n'est ni un algorithme de nommage, ni un algorithme d'énumération, ni un algorithme d'élection. □

2.4.2 L'Algorithme de Mazurkiewicz

On va maintenant décrire l'algorithme d'énumération $\mathcal{M}$ de Mazurkiewicz [Maz97] qui permet de résoudre le problème de l'énumération sur les graphes simples minimaux pour les revêtements simples.

Durant l'exécution de l'algorithme, chaque sommet v essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. À chaque fois qu'un sommet modifie son numéro, il en informe immédiatement ses voisins. Chaque sommet v peut donc tenir à jour la liste des numéros de ses voisins, qui sera appelée la *vue locale* de v. Lorsqu'un sommet v modifie son numéro ou sa vue locale, il diffuse dans le réseau son numéro accompagné de son étiquette initiale et de sa vue locale.

Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette $\lambda(u)$ et sa vue locale avec l'étiquette $\lambda(v)$ et la vue locale de v : si l'étiquette de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre qu'on expliquera par la suite), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire) et informe ses voisins de ce changement de numéro. Ensuite, les numéros et les vues locales qui ont été modifiées sont diffusés à nouveau dans le graphe. Lorsque l'exécution est terminée, si le graphe **G** est minimal pour les revêtements simples, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda \colon V(G) \cup E(G) \to L$ est un étiquetage initial, qui ne sera pas modifié par l'algorithme. Lors de l'exécution, les étiquettes des arêtes ne vont pas être modifiées et chaque sommet va obtenir une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $N(v) \in \mathcal{P}_{\text{fin}}(L \times \mathbb{N})$[1] est la *vue locale* du sommet v qui contient des informations sur les voisins de v ; c'est un ensemble fini de couples $(\ell, n) \in L \times \mathbb{N}$. À tout moment de l'exécution, pour chaque voisin v' de v tel que $n(v') \neq 0$, la vue locale de v contient le couple $(\lambda(\{v, v'\}), n(v'))$.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(L \times \mathbb{N})$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

Un Ordre sur les Vues Locales

Les bonnes propriétés de l'algorithme de Mazurkiewicz sont basées sur un ordre total sur les vues locales. Cet ordre permet à un sommet u de modifier son numéro s'il découvre qu'il existe un autre sommet avec le même numéro, la même étiquette et une vue locale «plus forte». Afin d'éviter des exécutions infinies, il faut que lorsque la vue locale d'un sommet est modifiée, elle ne puisse pas devenir plus faible, et ce pour éviter qu'un sommet ne modifie son numéro à cause d'un message qu'il ait lui même envoyé lors d'une étape précédente.

On définit donc un ordre total sur les ensembles finis de couples de $L \times N$. Pour cela, on considère que $L \times N$ est muni de l'ordre lexicographique usuel : $(\ell, n) \leq (\ell', n')$ si $\ell <_L \ell'$, ou si $\ell = \ell'$ et $n \leq n'$.

Ensuite, étant données deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(L \times \mathbb{N})$ distincts, on dit que $N_1 \prec N_2$ si le maximum pour l'ordre lexicographique sur $L \times N$ de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2.

On peut aussi voir cet ordre comme l'ordre lexicographique usuel sur les ensembles ordonnés N_1 et N_2. On note $(\ell_1, n_1), (\ell_2, n_2), \ldots, (\ell_k, n_k)$ et $(\ell'_1, n'_1), (\ell'_2, n'_2), \ldots, (\ell'_l, n'_l)$ les éléments respectifs de N_1 et de N_2 dans l'ordre décroissant (pour l'ordre lexicographique sur $L \times \mathbb{N}$) : $(\ell_1, n_1) \geq (\ell_2, n_2) \geq \cdots \geq (\ell_k, n_k)$ et $(\ell'_1, n'_1) \geq (\ell'_2, n'_2) \geq \cdots \geq (\ell'_l, n'_l)$. Alors $N_1 \prec N_2$ si l'une des conditions suivantes est vérifiée :

- $k < l$ et pour tout $i \in [1, k]$, $(\ell_i, n_i) = (\ell'_i, n'_i)$,
- $(\ell_i, n_i) < (\ell'_i, n_i)$ où i est le plus petit indice pour lequel $(\ell_i, n_i) \neq (\ell'_i, n_i)$.

Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre

[1] Pour tout ensemble S, on note $\mathcal{P}_{\text{fin}}(S)$ l'ensemble des parties finies de S.

$\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(L \times \mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

Les Règles de Réétiquetage

On décrit maintenant l'algorithme d'énumération grâce à des règles de réétiquetage. La première règle $\mathcal{M}_0$ est une règle spéciale qui permet à chaque sommet v de modifier son étiquette $\lambda(v)$ pour obtenir l'étiquette $(\lambda(v), 0, \emptyset, \emptyset)$.

Les règles sont décrites pour une étoile $B(v_0)$ de centre v_0. L'étiquette d'un sommet $v \in N_G(v_0) \cup \{v_0\}$ avant l'application de la règle est notée $(\lambda(v), n(v), N(v), M(v))$ et on note $(\lambda(v), n'(v), N'(v), M'(v))$ l'étiquette de v après l'application de la règle de réétiquetage. Par ailleurs, afin de rendre plus lisible les règles, on ne mentionne pas les différents champs des étiquettes qui ne sont pas modifiés.

La première règle permet aux sommets d'une même étoile d'échanger les informations dont ils disposent (i.e., contenues dans leur boîte-aux-lettres) à propos des étiquettes présentes dans le graphe. Cette règle ne peut être appliquée que si un tel échange d'information est nécessaire (i.e., tous les sommets de $B(v_0)$ n'ont pas la même boîte-aux-lettres).

$\mathcal{M}_1$: **Règle de Diffusion**
Précondition :
– $\exists v \in N_G(v_0)$ tel que $M(v) \neq M(v_0)$.
Réétiquetage :
– $\forall v \in V(B(v_0))$, $M'(v) := \bigcup_{w \in V(B(v_0))} M(w)$.

La deuxième règle permet à un sommet v_0 de changer de numéro s'il n'a pas encore de numéro (i.e., $n(v_0) = 0$) ou s'il sait qu'il existe un sommet dans le graphe qui a le même numéro que lui et qui a une étiquette ou une vue locale plus forte que la sienne. Dans ce cas là, v_0 choisit un nouveau numéro et modifie la vue locale de ses voisins. Toutes les boîtes-aux-lettres des sommets de l'étoile $B(v_0)$ sont modifiées : on ajoute à chaque boîte-aux-lettres tous les couples $(n'(v), \lambda(v), N'(v))$ pour tous les sommets v' de l'étoile $B(v_0)$.

$\mathcal{M}_2$: **Règle de Renommage**
Précondition :
– $\forall v \in N_G(v_0)$, $M(v) = M(v_0)$,
– $n(v_0) = 0$ ou
$\exists (n(v_0), \ell, N) \in M(v_0)$ tel que $(\lambda(v_0), N(v_0)) \prec (\ell, N)$
Réétiquetage :
– $n'(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
– $\forall v \in N_G(v_0)$, $N'(v) := N(v) \setminus \{(\lambda(\{v_0, v\}), n(v_0))\} \cup \{(\lambda(\{v_0, v\}), n'(v_0))\}$;
– $\forall v \in V(B(v_0))$, $M'(v) := M(v) \cup \bigcup_{w \in V(B(v_0))} \{(n'(w), \lambda(w), N'(w))\}$.

2.4.3 Correction de l'Algorithme d'Énumération

On considère un graphe simple étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage

de l'algorithme $\mathcal{M}$ décrit ci-dessus. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Propriétés Satisfaites lors de l'Exécution

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur le nombre d'étapes, rappelle quelque propriétés simples qui sont toujours satisfaites par l'étiquetage.

Lemme 2.15 *Pour tout sommet $v \in V(G')$, et pour toute étape i,*

1. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
2. $\exists(\ell_e, n) \in N_i(v) \iff \exists v' \in N_G(v)$ *tel que* $n_i(v') = n > 0$ *et* $\lambda(\{v, v'\}) = \ell_e$,
3. $\forall(\ell_e, n) \in N_i(v), n \neq n_i(v)$ *et* $\exists(n, \ell, N) \in M_i(v)$,
4. $\forall v', v'' \in N_G(v), n_i(v), n_i(v') > 0 \implies n_i(v') \neq n_i(v'')$.

L'algorithme $\mathcal{M}$ a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 2.16 *Pour chaque sommet v et chaque étape i,*
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

De plus, à chaque étape i, il existe un sommet v telle qu'au moins une de ces inégalités (ou inclusions) est stricte pour v.

Preuve : La propriété est trivialement vraie pour les sommets qui ne sont pas réétiquetés lors de la $(i+1)$ème étape. De plus, il est facile de voir que quelque soit la règle appliquée à l'étape $i+1$, on a toujours $M_i(v) \subseteq M_{i+1}(v)$ pour tout sommet $v \in V(G)$.

Pour chaque sommet v tel que $n_i(v) \neq n_{i+1}(v)$, alors la règle $\mathcal{M}_2$ a été appliquée sur l'étoile $B(v)$ et on sait que $n_{i+1}(v) = 1 + \max\{n' \mid \exists(n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 2.15, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Pour chaque sommet v tel que $N_i(v) \neq N_{i+1}(v)$, alors la règle $\mathcal{M}_2$ a été appliquée sur une étoile $B(v_0)$ telle que $v_0 \in N_G(v)$. On sait que $N_{i+1}(v) = N_i(v) \backslash \{(\lambda(\{v_0, v\}), n_i(v_0))\} \cup \{(\lambda(\{v_0, v\}), n_{i+1}(v_0))\}$ et que pour tout $(n, \ell, N) \in M_i(v)$, $n_{i+1}(v_0) > n$. Ainsi, on a $\max N_{i+1}(v) \triangle N_i(v) = (\lambda(\{v_0, v\}), n_{i+1}(v_0)) \in N_{i+1}(v)$. Par conséquent, $N_i(v) \prec N_{i+1}(v)$.

Puisque chaque application d'une règle modifie l'étiquette d'au moins un sommet v, on sait que l'une de ces inégalités est stricte pour v. □

Les informations dont dispose chaque sommet v dans sa boîte-aux-lettres permettent d'obtenir des informations vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet w tel que $n_i(w) = m$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet w tel que $n_i(w) = m$.

Lemme 2.17 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid \forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, il existe exactement un élément $(u, i_0) \in U'$ puisqu'à chaque étape, le numéro d'au plus un sommet peut être modifié. Le numéro $n_{i_0}(u) = m$ a donc été modifié à l'étape $i_0 + 1$, mais par maximalité de $(\lambda(u), N_{i_0}(u))$, la règle $\mathcal{M}_2$ n'a pas pu être appliquée à u à l'étape i_0. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 2.18 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. La propriété est trivialement vraie à l'étape $i + 1$ pour tout sommet $w \in V(G)$ dont l'étiquette n'est pas modifiée à l'étape $i + 1$. Soit v un sommet dont l'étiquette est modifiée à l'étape $i + 1$.

Si la règle $\mathcal{M}_1$ a été appliquée à l'étape $i + 1$, alors pour tout $(m, \ell, N) \in M_{i+1}(v)$, il existe v' tel que $(m, \ell, N) \in M_i(v')$ et $M_i(v') \subseteq M_{i+1}(v)$. Par conséquent, par hypothèse de récurrence, pour tout $m' < m$, il existe $(m', \ell', N') \in M_i(v') \subseteq M_{i+1}(v)$. On suppose maintenant que la règle $\mathcal{M}_2$ a été appliquée à l'étape $i+1$ sur une étoile $B(v_0)$ qui contient v. Soit $(m, \ell, N) \in M_{i+1}(v) \backslash M_i(v)$. Si $m = n_{i+1}(v_0) = 1 + \max\{m' \mid \exists (m', \ell', N') \in M_i(v)\}$, pour tout $m' < m$, il existe $(m', \ell', N') \in M_i(v) \subseteq M_{i+1}(v)$. Si $m \neq n_{i+1}(v_0)$, alors il existe $v' \in N_G(v_0)$ tel que $m = n_i(v') = n_{i+1}(v')$ et d'après le Lemme 2.15, on sait qu'il existe $(m, \ell', N') \in M_i(v') = M_i(v)$. Ainsi la propriété est vérifiée à l'étape $i + 1$. □

On veut maintenant montrer que toute exécution de l'algorithme $\mathcal{M}$ termine sur $\mathbf{G}$. D'après les Lemmes 2.17 et 2.18, on voit qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1, k]$ ou un ensemble $[0, k]$ avec $k \leq |V(G)|$. Par conséquent, d'après le Lemme 2.16, on sait qu'il existe une étape i_0 telle que pour tout sommet v et toute étape $i \geq i_0$, $n_{i+1}(v) = n_i(v)$.

De plus, pour chaque sommet v et chaque étape i, si $N_i(v)$ contient un couple (ℓ_e, n'), alors il existe $v' \in N_G(v)$ tel que $n_i(v') = n'$ et $\lambda(\{v, v'\}) = \ell_e$. Par conséquent $N(v)$ ne peut prendre qu'un nombre fini de valeurs et il en est de même pour $M(v)$. Ainsi le nombre de valeurs différentes que peut prendre l'étiquette de chaque sommet est fini (mais dépend de la taille du graphe). Par ailleurs, les étiquettes consécutives de chaque sommet v forment une suite croissante et puisqu'à chaque étape i, l'étiquette d'au moins un sommet est modifiée, toute exécution de l'algorithme termine : la relation $\mathcal{M}$ est noethérienne.

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final.

Lemme 2.19 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. *$M_\rho(v) = M_\rho(v')$,*
3. *$(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,*
4. *si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et $N_\rho(v) = N_\rho(v')$,*
5. *$\forall w, w' \in N_G(v), n_\rho(w) \neq n_\rho(w')$,*
6. *$(\ell_e, n) \in N_\rho(v)$ si et seulement s'il existe $w \in N_G(v)$ tel que $n_\rho(w) = n$ et $\lambda(\{v, w\}) = \ell_e$; auquel cas, $(\ell_e, n_\rho(v)) \in N_\rho(w)$.*

Preuve :

1. D'après les Lemmes 2.17 et 2.18 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
2. Dans le cas contraire, la règle $\mathcal{M}_1$ peut être appliquée.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 2.15.
4. Dans le cas contraire, la règle $\mathcal{M}_2$ peut être appliquée à v ou à v'.
5. D'après le Lemme 2.15 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
6. D'après le Lemme 2.15.

□

Grâce au Lemme 2.19, on peut prouver que l'étiquetage final permet de construire un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est un revêtement de $\mathbf{H}$.

Proposition 2.20 *Étant donné un graphe simple $\mathbf{G}$, on peut construire, à partir de l'étiquetage final obtenu après une exécution ρ de $\mathcal{M}$, un graphe simple $\mathbf{H}$ tel qu'il existe un homomorphisme localement bijectif de $\mathbf{G}$ dans $\mathbf{H}$.*

Preuve : On utilise les notations du Lemme 2.19.

On considère le graphe H défini par $V(H) = \{m \in \mathbb{N} \mid \exists v \in V(G), n_\rho(v) = m\}$ et $E(H) = \{\{m, m'\} \mid \exists v, v' \in V(G); n_\rho(v) = m, n_\rho(v') = m' \text{ et } \{v, v'\} \in E(G)\}$. D'après le Lemme 2.15, on sait qu'il n'existe pas d'arête $\{v, v'\} \in E(G)$ telle que $n_\rho(v) = n_\rho(v')$: le graphe H ne contient donc pas de boucle. De plus, de par la définition de $E(H)$, on sait que H ne contient pas d'arêtes multiples. Ainsi le graphe H est bien un graphe simple. On définit un étiquetage η de H en posant $\eta(n_\rho(v)) = \lambda(v)$ et pour toute arête $\{v, v'\} \in E(H)$, $\eta(\{n_\rho(v), n_\rho(v')\}) = \lambda(\{v, v'\})$. D'après le Lemme 2.19, si deux sommets $v, v' \in V(G)$ ont le même numéro, alors $\lambda(v) = \lambda(v')$. De plus, pour toutes arêtes $\{v, v'\}, \{w, w'\} \in E(G)$ telles que $n_\rho(v) = n_\rho(w) = n$ et $n_\rho(v') = n_\rho(w') = n'$, on sait d'après le Lemme 2.19 que $(\lambda(\{v, v'\}), n') \in N_\rho(v)$ et que $(\lambda(\{w, w'\}), n') \in N_\rho(w)$. Puisque $n_\rho(v) = n_\rho(w)$, on a $N_\rho(w) = N_\rho(v)$. Ainsi, puisqu'il ne peut exister deux couples distincts $(\ell_1, n'), (\ell_2, n')$

dans $N_\rho(v) = N_\rho(w)$, $\lambda(\{w, w'\}) = \lambda(\{v, v'\})$. Par conséquent, l'étiquetage η de H est bien défini.

On considère maintenant la fonction $n_\rho : V(G) \to V(H)$. Par définition de H, on sait que si $\{v, v'\} \in E(G)$, alors $\{n_\rho(v), n_\rho(v')\} \in E(G)$ et $\eta(\{n_\rho(v), n_\rho(v')\}) = \lambda(\{v, v'\})$. De plus, pour tout $v \in V(G)$, $\eta(n_\rho(v)) = \lambda(v)$ et par conséquent, n_ρ est un homomorphisme de $\mathbf{G}$ dans $\mathbf{H}$. D'après le Lemme 2.19, pour tout sommet v, si $m \in N_H(n_\rho(v))$, alors il existe $w \in N_G(v)$ tel que $n_\rho(w) = m$ et par conséquent, $n_\rho(N_G(v)) = N_H(n_\rho(v))$. De plus, pour tous $w, w' \in N_G(v)$, $n_\rho(w) \neq n_\rho(w')$ et on a donc $|N_G(v)| = |N_H(n_\rho(v))|$. Ainsi, l'homomorphisme n_ρ est localement bijectif et $\mathbf{G}$ est un revêtement de $\mathbf{H}$. □

On considère maintenant un graphe simple $\mathbf{G}$ qui est minimal pour les revêtements simples. Pour chaque exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, le graphe obtenu à partir de l'étiquetage final est isomorphe à $\mathbf{G}$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique. L'algorithme $\mathcal{M}$ permet de résoudre le nommage et l'énumération sur la famille des graphes minimaux pour les revêtements simples, mais si aucune information à propos de $\mathbf{G}$ n'est disponible, les sommets ne peuvent pas détecter la terminaison.

Cependant, il est possible de détecter la terminaison pour un graphe simple $\mathbf{G}$ donné (l'algorithme dépend alors de $\mathbf{G}$). En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 2.17 et 2.18, il sait que tous les sommets de $\mathbf{G}$ ont un numéro unique qui ne va plus être modifié. Dans ce cas là, on peut aussi résoudre le problème de l'élection, puisque ce sommet peut prendre l'étiquette ÉLU et diffuser ensuite l'information qu'un sommet a été élu.

Par ailleurs, d'après la Proposition 2.14, on sait que pour tout graphe simple $\mathbf{G}$ qui n'est pas minimal pour les revêtements simples, il n'existe aucun algorithme utilisant des calculs locaux sur les étoiles fermées qui permettent de résoudre les problèmes du nommage ou de l'élection sur $\mathbf{G}$. On a donc prouvé le théorème suivant.

Théorème 2.21 ([Maz97]) *Pour tout graphe simple étiqueté $\mathbf{G}$, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour $\mathbf{G}$ utilisant des calculs locaux sur les étoiles fermées,*
2. *il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $\mathbf{G}$ utilisant des calculs locaux sur les étoiles fermées,*
3. *$\mathbf{G}$ est minimal pour les revêtements simples.*

Remarque 2.22 *Étant donné un graphe simple $\mathbf{G}$ minimal pour les revêtements simples, pour détecter que l'algorithme $\mathcal{M}$ a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de $\mathbf{G}$. Ainsi, l'algorithme $\mathcal{M}$ permet de résoudre l'élection ainsi que le nommage avec détection de la terminaison sur les graphes minimaux pour les revêtements simples de taille donnée.*

2.4.4 Complexité

On s'intéresse à la complexité de l'algorithme de Mazurkiewicz. Dans le cadre des calculs locaux, on s'intéresse au nombre de pas de réétiquetages effectuées lors d'une

exécution. La proposition suivante, qui a été montrée par Godard [God02a], donne une borne supérieure sur le nombre de pas de réétiquetages de toute exécution de l'algorithme $\mathcal{M}$ sur un graphe à n sommets.

Proposition 2.23 ([God02a]) *Pour tout graphe* $\mathbf{G}$ *à* n *sommets, durant toute exécution de l'algorithme de Mazurkiewicz,* $O(n^3)$ *règles sont appliquées.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et une exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$. D'après les Lemmes 2.17 et 2.18, on sait que la règle $\mathcal{M}_2$ ne peut pas être appliquée plus de $\frac{n(n-1)}{2}$ fois durant l'exécution ρ.

Si la règle $\mathcal{M}_2$ est appliquée à l'étape $i+1$ sur un sommet v, on note $M'(i,v)$ l'ensemble de triplets (n,ℓ,N) ajoutés à $M(v)$ à l'étape $i+1$, i.e., $M_{i+1}(v) = M_i(v) \cup M'(i,v)$. On remarque qu'il existe au plus n étapes j lors desquelle la règle $\mathcal{M}_1$ est appliquée et modifie l'étiquette d'un sommet w tel que $M'(i,v) \nsubseteq M_{j-1}(w)$ et $M'(i,v) \subseteq M_{j-1}(w)$. Par conséquent, puisque la règle est appliquée $O(n^2)$ fois lors de l'exécution durant l'exécution ρ, la règle $\mathcal{M}_1$ est appliquée $O(n^3)$ fois. □

Remarque 2.24 *Dans sa thèse [God02a], Godard a montré qu'il existait des graphes admettant des exécutions de* $\mathcal{M}$ *nécessitant* $\Omega(n^3)$ *pas de réétiquetages.*

On peut aussi s'intéresser à l'espace nécessaire à chaque sommet pour stocker son étiquette. On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets et à toutes les arêtes de G).

Proposition 2.25 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, l'algorithme de Mazurkiewicz nécessite* $O(\Delta n \log n)$ *bits de mémoire par sommet.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets dont le degré maximal est Δ. 0n sait que la vue locale de chaque sommet contient au plus Δ couples (ℓ_e, n_0) qui peuvent être représentés avec $O(\log n)$ bits. Ainsi, pour chaque sommet v, $N(v)$ peut être représenté avec $O(\Delta \log n)$ bits.

On remarque que durant l'exécution de l'algorithme, la boîte-aux-lettres de chaque sommet v contient de plus en plus d'informations. Cependant, elle contient des informations inutiles qui peuvent être éliminées à chaque étape. En effet, pour tout $(n_0,\ell,N) \in M(v)$, s'il existe $(n_0,\ell',N') \in M(v)$ tel que $(\ell,N) \prec (\ell',N')$, on peut supprimer le couple (n_0,ℓ,N) de $M(v)$ sans perturber l'exécution de l'algorithme. Ainsi, à chaque étape, chaque sommet v peut remplacer sa boîte aux lettres $M(v)$ par $\{(n_0,\ell,N) \in M(v) \mid \forall (n_0,\ell',N') \in M(v), (\ell',N') \preceq (\ell,N)\}$.

Par conséquent, dans la boîte-aux-lettres de chaque sommet, il existe au plus n triplets (n_0,ℓ,N) dont la taille est en $O(\Delta \log n)$ bits. Ainsi, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta n \log n)$ bits. □

2.5 Importance de la Connaissance Initiale

Dans la partie précédente, on a montré que l'algorithme de Mazurkiewicz permettait de résoudre le problème de l'élection et du nommage avec détection de la terminaison sur la famille des graphes minimaux pour les revêtements simples de taille donnée. Cependant,

étant donné un graphe **G** minimal pour les revêtements simples, l'algorithme de Mazurkiewicz à lui seul ne permet pas de résoudre l'élection sur **G** si on ne connaît qu'une borne sur la taille de **G**. Par ailleurs, étant donné un graphe **G** qui n'est pas minimal pour les revêtements, l'algorithme de Mazurkiewicz ne permet pas de détecter qu'on ne peut pas élire dans **G** si on ne connait que la taille de **G**.

Cependant, il n'est pas forcément nécessaire de connaître la taille pour pouvoir élire ; d'autres types de connaissance initiales permettent de résoudre le problème de l'élection, ou du nommage avec détection de la terminaison. Par exemple, si on sait que le graphe **G** est un arbre, on peut toujours élire un sommet, sans connaître d'autre information sur **G**. En effet, l'algorithme d'élection pour la famille des arbres présenté dans le Chapitre 1 peut être implémenté avec des calculs locaux sur les étoiles fermées et on a donc le théorème suivant.

Théorème 2.26 *Il existe un algorithme d'élection pour la famille des arbres utilisant des calculs locaux sur les étoiles fermées.*

On va maintenant présenter une extension de l'algorithme de détection de la terminaison de Szymansky, Shi et Prywes [SSP85] qui permet d'obtenir un algorithme d'élection pour des classes de graphes minimaux pour les revêtements simples qui n'ont pas tous la même taille. Cette extension a été introduite dans [MT00, GM03]. Dans [GM02], cet outil a assi été utilisé pour caractériser les classes de graphes admettant un algorithme universel d'élection dans ce modèle. On va présenter ici une version plus faible de ce résultat.

2.5.1 L'Algorithme de Szymansky, Shi et Prywes

Cet algorithme a été initialement décrit pour détecter la terminaison d'un algorithme dans un modèle où les processus où les sommets communiquent en échangeant des messages. Étant donné un algorithme où chaque processus sait qu'il a calculé sa valeur finale, l'algorithme de Szymansky, Shi et Prywes [SSP85] (noté SSP) permet aux processus de détecter, sous certaines conditions, un instant où tous les sommets ont calculé leurs valeurs finales : le calcul est globalement terminé.

Étant donné un graphe G, on associe à chaque sommet $v \in V(G)$ un prédicat $P(v)$ et un entier $a(v)$, son *niveau de confiance*. Initialement $P(v)$ est Faux et $a(v)$ est égal à -1. Lorsqu'un processus a calculé sa valeur finale, il modifie la valeur de $P(v)$ qui devient Vrai (et qui ne changera plus). À chaque fois qu'un processus modifie la valeur de $a(v)$, il en informe ses voisins.

La modification de la valeur de $a(v_0)$ d'un sommet v_0 dépend seulement de la valeur de $P(v_0)$ et des informations dont v_0 dispose à propos des valeurs $\{a(v_1), \ldots, a(v_d)\}$ de ses voisins :

- Si $P(v_0)$ =Faux, alors $a(v_0) = -1$;
- si $P(v_0)$ =Vrai, alors $a(v_0) = 1 + \min\{a(v_k) \mid 0 \leq k \leq d\}$.

Le principe de l'algorithme de SSP généralisé (noté GSSP) introduite et utilisée dans [MT00, GM03] est le suivant. Au lieu de considérer que la fonction P est un prédicat qui ne peut changer qu'une fois de valeur pour passer de Faux à Vrai, la fonction P peut prendre une valeur quelconque et peut être modifiée un nombre arbitraire de fois. Cependant, on suppose que pour tout sommet v, si $P(v)$ a une valeur α qui est modifiée par la suite, alors $P(v)$ ne pourra plus jamais être égal à α. Autrement dit, entre deux

moments où un sommet v est tel que $P(v) = \alpha$, la fonction $P(v)$ est constante. On dit alors que la fonction P est *connexe par valeurs*.

2.5.2 Une Adaptation de l'Algorithme de Mazurkiewicz

On va utiliser l'algorithme GSSP pour vérifier que tous les sommets du graphe ont la même boîte-aux-lettres et qu'aucun d'eux ne peut appliquer la règle $\mathcal{M}_2$ de l'algorithme de Mazurkiewicz, i.e., changer de numéro. Si on peut s'assurer que ces deux propriétés sont vraies, alors on sait que l'exécution de l'algorithme est terminée et on sait que les propriétés du Lemme 2.19 et de la Proposition 2.20 sont vérifiées. D'après le Lemme 2.16, on sait que les boîtes aux lettres des sommets ne peuvent qu'augmenter pour l'inclusion et $M : v \mapsto M(v)$ est donc une fonction connexe par valeurs.

L'étiquette des sommets de $\mathbf{G} = (G, \lambda)$ sont alors de la forme $(\lambda(v), n(v), N(v), M(v), a(v))$ où la seule différence avec l'algorithme de Mazurkiewicz est le champ $a(v)$ qui est le *rayon de confiance* du sommet v. Initialement, tous les sommets ont l'étiquette $(\lambda(v), 0, \emptyset, \emptyset, -1)$.

Les deux premières règles de l'algorithme de Mazurkiewicz sont adaptées de telle sorte qu'à chaque fois que la boîte-aux-lettres d'un sommet est modifiée, alors son *rayon de confiance* est réinitialisé à -1 puisque le sommet a modifié sa boîte-aux-lettres.

$\mathcal{M}'_1$: **Règle de Diffusion**

Précondition :
- $\exists v \in N_G(v_0)$ tel que $M(v) \neq M(v_0)$.

Réétiquetage :
- $\forall v \in V(B(v_0))$, $M'(v) := \bigcup_{w \in V(B(v_0))} M(w)$;
- $\forall v \in V(B(v_0))$, $a'(v) := -1$.

$\mathcal{M}'_2$: **Règle de Renommage**

Précondition :
- $\forall v \in N_G(v_0)$, $M(v) = M(v_0)$,
- $n(v_0) = 0$ ou
  $\exists (n(v_0), \ell, N) \in M(v_0)$ tel que $(\lambda(v_0), N(v_0)) \prec (\ell, N)$

Réétiquetage :
- $n'(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
- $\forall v \in N_G(v_0)$, $N'(v) := N(v) \setminus \{(\lambda(\{v_0, v\}), n(v_0))\} \cup \{(\lambda(\{v_0, v\}), n'(v_0))\}$;
- $\forall v \in V(B(v_0))$, $M'(v) := M(v) \cup \bigcup_{w \in V(B(v_0))} (n'(w), \lambda(w), N'(w))$;
- $\forall v \in V(B(v_0))$, $a'(v) := -1$.

Une troisième règle est ajouté qui permet à un sommet v qui ne peut pas modifier son numéro d'augmenter son rayon de confiance si tous ses voisins ont la même boîte-aux-lettres que v et si leurs rayons de confiances sont tous supérieurs ou égaux à celui de v.

$\mathcal{M}'_3$: **Règle GSSP**

Précondition :

– $\forall v \in N_G(v_0)$, $M(v) = M(v_0)$,
– $n(v_0) > 0$ et $\forall (n(v_0), \ell, N) \in M(v_0), (\ell, N) \preceq (\lambda(v_0), N(v_0))$,
– $\forall v \in N_G(v_0)$, $a(v) \geq a(v_0)$
Réétiquetage :
– $a'(v_0) := 1 + \min\{a(v) \mid v \in N_G(v_0)\}$.

2.5.3 Propriétés Satisfaites par l'Algorithme

Comme précedemment, on considère une exécution de l'algorithme $\mathcal{M}'$ sur un graphe simple étiqueté **G**. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v), a_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage.

Le lemme suivant qui peut être facilement prouvé par induction sur le nombre d'étapes montre que le rayon de confiance d'un sommet ne peut qu'augmenter tant que sa boîte-aux-lettres n'est pas modifiée.

Lemme 2.27 *Pour tout sommet v et toute étape i, si $M_i(v) = M_{i+1}(v)$, alors $a_{i+1}(v) \geq a_i(v)$. De plus, si $a_i(v) \geq 0$, alors la règle $\mathcal{M}'_2$ ne peut être appliquée sur l'étoile de centre v.*

Dans le lemme suivant, on montre que le rayon de confiance d'un sommet permet d'obtenir des informations sur le contenu des boîtes-aux-lettres d'autres sommets du graphe lors d'étapes précédentes de l'exécution.

Lemme 2.28 *Pour tout sommet $v \in V(G)$ et toute étape i, pour tout sommet $w \in V(G)$ tel que $\text{dist}_G(v, w) \leq a_i(v)$, il existe une étape $j \geq i$ telle que $a_j(w) \geq a_i(v) - \text{dist}_G(v, w)$ et $M_j(v) = M_i(v)$.*

Preuve : On fait une démonstration par récurrence sur la distance k entre v et w. Si $k = 0$, la propriété est trivialement vraie. On suppose maintenant que la propriété est vraie pour tous sommets v, w tels que $\text{dist}_G(v, w) \leq k$.

On considère deux sommets v, w et une étape i tels que $a_i(v) \geq k + 1$ et $\text{dist}_G(v, w) = k + 1$. Il existe un sommet $u \in N_G(v)$ tels que $\text{dist}_G(u, w) = k$. On considère la dernière étape j' où la règle $\mathcal{M}'_3$ a été appliquée au sommet v. On sait que $a_{j'}(u) \geq a_{j'}(v) - 1 = a_i(v) - 1$ et $M_{j'}(u) = M_{j'}(v) = M_i(v)$. Par hypothèse de récurrence, on sait qu'il existe une étape $j < j'$ telle que $a_j(w) \geq a_{j'}(u) - k \geq a_i(v) - (k+1)$ et $M_j(w) = M_{j'}(u) = M_i(v)$. Ainsi, la propriété est vérifiée pour tous sommets v, w à distance $k + 1$. □

Dans le lemme suivant, on montre que si à un moment donné, un sommet a un rayon de confiance supérieur au diamètre du graphe, alors tous les sommets du graphe ont la même boîte-aux-lettres et ont tous un rayon de confiance supérieur à 0.

Lemme 2.29 *S'il existe un sommet v et une étape i telle que $a_i(v) \geq D(G)$, alors pour tout $w \in V(G)$, $M_i(w) = M_i(v)$ et $a_i(v) \geq 0$.*

Preuve : Puisque $a_i(v) > D(G)$, on sait d'après le Lemme 2.28 que pour tout sommet $w \in V(G)$, il existe une étape $i_w < i$ telle que $a_{i_w} \geq 0$ et $M_{i_w}(w) = M_i(v)$.

Supposons qu'il existe un sommet w tel que $M_i(w) \neq M_{i_w}(w)$. Soit j l'étape de l'exécution lors de laquelle pour la première fois, la boîte-aux-lettres d'un sommet w qui valait $M_i(v)$ a été modifiée. Autrement dit, pour tout sommet $w \in V(G)$, il existe une étape $j' \geq j - 1$ telle que $M_{j'}(w) = M_i(v)$ et il existe un sommet w tel que

$M_{j-1}(w) = M_i(v) \subsetneq M_j(w)$. Cela signifie que la règle $\mathcal{M}'_1$ ou $\mathcal{M}'_2$ a été appliquée lors de l'étape j ; on note v_0 le centre de l'étoile sur laquelle cette règle a été appliquée.

On sait que pour tout sommet $w \in V(B_G(V_0))$, $M_{j-1}(w) = M_i(v) \subsetneq M_j(w)$ et qu'il existe une étape $i_w \leq j-1$ telle que $M_{i_w}(w) = M_i(v)$ et $a_{i_w}(w) \geq 0$. D'après le Lemme 2.27, on sait que pour tout sommet $w \in V(B(V_0))$, $M_{j-1}(w) = M_i(v)$ et $a_{j-1}(w) \geq 0$. Par conséquent, puisque tous les sommets de $B_G(v_0)$ ont la même boîte-aux-lettres à l'étape $j-1$, la règle $\mathcal{M}'_1$ ne peut pas être appliquée sur l'étoile $B_G(v_0)$ lors de l'étape j. De plus, $a_j(v_0) \geq 0$ et d'après le Lemme 2.27, on sait que la règle $\mathcal{M}'_2$ ne peut pas être appliquée sur l'étoile de centre v_0. Par conséquent, pour tout sommet $w \in V(G)$, $M_i(w) = M_i(v)$. □

Ainsi, si on connaît une borne B sur le diamètre de $\mathbf{G}$, l'algorithme $\mathcal{M}'$ permet de détecter que l'exécution de l'algorithme de Mazurkiewicz sous-jacent est terminé. En effet, une fois qu'un sommet a un rayon de confiance supérieur ou égal à B, il sait que tous les sommets de $\mathbf{G}$ ont la même boîte-aux-lettres et qu'ils ont leurs numéros et vues locales finaux.

Si on sait que le graphe $\mathbf{G}$ est minimal pour les revêtements simples, on sait alors d'après la Proposition 2.20 que tous les sommets ont un identifiant unique et dans ce cas là, le sommet dont le numéro est 1 peut prendre l'étiquette ÉLU et diffuser l'information. On a par conséquent montré le théorème suivant, qui est un cas particulier de la caractérisation des familles de graphes admettant un algorithme d'élection présenté dans [GM02].

Théorème 2.30 ([GM02]) *Pour tout entier B, il existe un algorithme d'élection et un algorithme de nommage avec détection de la terminaison utilisant des calculs locaux sur les étoiles fermées pour la famille des graphes minimaux pour les revêtements simples dont le diamètre est bornée par B.*

Ainsi, il n'est pas nécessaire de connaître la taille pour pouvoir élire dans un graphe $\mathbf{G}$ que l'on sait minimal pour les revêtements simples : une borne sur la taille ou le diamètre est suffisante.

De plus, Angluin [Ang80] a montré qu'on ne pouvait pas résoudre l'élection si les sommets ne disposaient d'aucune information à propos de $\mathbf{G}$, i.e., il n'existe pas d'algorithme d'élection universel pour la famille des graphes minimaux pour les revêtements simples. Godard et Métivier [GM02] ont généralisé ce résultat en montrant qu'il n'existait pas d'algorithme universel d'élection pour toute famille de graphes minimaux contenant strictement la famille des arbres.

Cependant, on ne sait pas nécessairement si le graphe $\mathbf{G}$ est minimal pour les revêtements simples. On peut donc souhaiter avoir un algorithme qui permet ou bien de résoudre l'élection sur $\mathbf{G}$, ou bien de détecter que le graphe $\mathbf{G}$ n'est pas minimal pour les revêtements simples. On va montrer que l'algoritme $\mathcal{M}'$ permet de résoudre ce problème si on a une borne serrée sur la taille de $\mathbf{G}$.

Théorème 2.31 *Pour tout entier B, il existe un algorithme effectif d'élection et de nommage avec détection de la terminaison utilisant des calculs locaux sur les étoiles fermées pour la classe des graphes $\mathbf{G}$ tels que $V(G) \leq B < 2|V(G)|$.*

Preuve : On sait d'après la Proposition 2.20 et le Lemme 2.29 qu'avec la connaissance d'une borne serrée B sur la taille d'un graphe $\mathbf{G}$, toute exécution de $\mathcal{M}'$ sur $\mathbf{G}$ permet de construire un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est un revêtement simple de $\mathbf{H}$. Si $\mathbf{G}$ est un revêtement

simple propre de **H**, on sait d'après la Proposition 2.6 que $2|V(H)| \leq |V(G)|$. Puisque $|V(G)| \leq B < 2|V(G)|$, on sait que si **G** est un revêtement simple propre de **H**, $2|V(H)| \leq B$. Au contraire, si **G** est isomorphe à **H**, alors $B < 2|V(G)| = 2|V(H)|$. Par conséquent, il suffit de tester si la taille du graphe **H** construit à partir de l'étiquetage final obtenu après l'exécution de $\mathcal{M}'$ sur **G** vérifie l'inégalité $2|V(H)| \leq B$. Si l'inégalité est vérifiée, alors le graphe **G** n'est pas minimal pour les revêtements simples, et dans le cas contraire, le graphe **H** est isomorphe à **G** et on peut donc résoudre l'élection et le nommage avec détection de la terminaison sur **G**. □

On remarque que si on ne connait qu'une borne sur la taille de **G** et que cette borne n'est pas serrée, on ne peut pas trouver d'algorithme utilisant des calculs locaux sur les étoiles fermées qui permet de résoudre le problème de l'élection sur **G** ou de détecter que **G** n'est pas minimal pour les revêtements simples. En effet, il suffit de considérer deux graphes **G** et **H** de telle sorte que **G** soit un revêtement simple à deux feuillets de **H** (ce qui est toujours possible d'après le Théorème de Reidemeister), comme par exemple les graphes de la Figure 7. Si la borne fournie à l'algorithme est $|V(G)|$ et qu'elle n'est pas serrée, l'algorithme doit résoudre le problème de l'élection sur **H** et par conséquent, d'après le Lemme 2.13, il existe une exécution de l'algorithme sur **G** telle que dans la configuration finale, l'étiquette élu apparaît deux fois dans **G**.

2.6 Conclusion

Dans ce chapitre, on a présenté une caractérisation des graphes admettant un algorithme d'élection (ou de nommage) utilisant des calculs locaux sur les étoiles fermées. Cette caractérisation s'exprime de manière élégante à l'aide des revêtements simples. La condition nécessaire est un résultat d'Angluin [Ang80] et la condition suffisante est obtenue grâce à l'algorithme de Mazurkiewicz [Maz97].

On remarque que permettre à un algorithme de pouvoir modifier ou non les étiquettes des arêtes des étoiles réétiquetées ne permet pas de résoudre élection ou nommage dans des graphes différents. En effet, le résultat d'impossibilité de la Proposition 2.14 est vrai lorsque les arêtes peuvent être réétiquetées alors que le résultat de possibilité (l'algorithme de Mazurkiewicz) n'utilise pas cette possibilité. On verra que les modèles étudiés dans les chapitres suivants n'ont pas cette propriété.

Dans [God02b], Godard a présenté une version auto-stabilisante de l'algorithme d'énumération de Mazurkiewicz. Dans [GMM04], Godard, Métivier et Muscholl caractérisent les classes de graphes reconnaissables dans ce modèle sans connaissance initiale ; ils considèrent des systèmes de reconnaissance à terminaison implicite. Ils montrent qu'une classe de graphes est reconnaissable dans ce cadre si cette classe est close pour la fermeture symétrique, réflexive et transitive de la relation «être revêtement». Dans [GM03] Godard et Métivier caractérisent les classes de graphes qui peuvent être reconnues dans ce modèle en fonction de la connaissance initiale disponible.

Dans la Section 2.5, on a étudié l'importance des connaissances initiales pour pouvoir élire dans un graphe minimal pour les revêtements. Les résultats présentés sont des cas particuliers des résultats obtenus par Godard et Métivier dans [GM02], où une caractérisation des classes de graphes admettant un algorithme universel d'élection est

présentée. Pour obtenir des conditions nécessaires, les travaux présentés dans [GM02] reposent sur le lemme de relèvement d'Angluin [Ang80] et sur la notion de quasi-revêtements introduite dans [MMW97]. Les conditions suffisantes sont obtenues à partir d'une étude fine des propriétés de l'algorithme de Mazurkiewicz [Maz97] et de l'algorithme de Szymansky, Shi et Prywes [SSP85].

Dans la Section 2.5, on a aussi présenté un algorithme effectif d'élection nécessitant la connaissance d'une borne serrée sur la taille du graphe. Ce résultat est lié aux travaux de Métivier et Tel [MT00] où ils étudient sous quelles conditions un algorithme à terminaison implicite utilisant des calculs locaux sur les étoiles fermées peut être transformé en un algorithme à terminaison explicite.

Chapitre 3

Calculs Locaux sur les Arêtes Étiquetées

Sommaire

FIG. 9 – Forme générique d'un règle de calcul pour les calculs locaux sur les arêtes étiquetées.

FIG. 10 – Forme générique d'un règle de calcul pour les calculs locaux cellulaires sur les arêtes étiquetées.

3.1 Introduction

Dans ce chapitre, on étudie les *calculs locaux sur les arêtes étiquetées* qui correspondent aux relations de réétiquetage localement engendrées sur les arêtes introduites au Chapitre 1 (Definition 1.31). Dans ce modèle, on considère des graphes qui peuvent avoir des arêtes multiples mais qui n'ont pas de boucle. Un pas de calcul est décrit par une règle de réétiquetage de la forme présentée sur la Figure 9. Si dans un graphe G, il existe une arête e étiquetée α dont les extrémités sont étiquetées X et Y, alors lors de l'application de cette règle sur e, on remplace α par α', X par X' et Y par Y'. Les étiquettes de tous les autres sommets et arêtes de G ne sont pas prises en compte lors de l'application de la règle et ne sont pas modifiées. Les sommets et l'arête de G dont les étiquettes sont modifiées sont dits *actifs* et les autres sommets et arêtes de G sont dits *inactifs*. Les calculs réalisés en utilisant uniquement ce type de règles de réétiquetage sont appelés *calculs locaux sur les arêtes étiquetées*. Il faut noter que si une règle de calcul est appliquée sur une arête dont les deux extrémités ont la même étiquette, les nouvelles étiquettes des deux sommets peuvent être différentes. Autrement dit, les règles ne sont pas symétriques.

On considère aussi une restriction de ce modèle où en un pas de calcul, l'étiquette d'au plus un sommet peut être modifiée. Les règles sont donc de la forme présentée sur la Figure 10. Si dans un graphe G il existe une arête e étiquetée α dont les extrémités sont étiquetées X et Y, alors lors de l'application de cette règle sur e, on remplace α par α' et X par X'. Les étiquettes de tous les autres sommets et arêtes de G ne sont pas prises en compte pour l'application de la règle de réétiquetage et restent inchangées. Le sommet de G dont l'étiquette est modifiée est dit *actif* (et est représenté en noir sur les figures), le voisin du sommet actif qui est impliqué dans l'application de la règle est dit *passif* (et est représenté en blanc sur les figures). Tous les autres sommets de G qui ne participent pas à l'application de la règle sont dits *inactifs*. L'arête dont l'étiquette est modifiée est dite *active* et toutes les autres arêtes de G sont dites *inactives*. Les calculs réalisés en utilisant uniquement ce type de réétiquetage sont appelés *calculs locaux cellulaires sur les arêtes étiquetées*.

On considère enfin un modèle où en un pas de calcul, un sommet peut observer l'état de ses voisins, l'état des arêtes qui lui sont incidentes et modifier son état ainsi que l'état des arêtes qui lui sont incidentes. La différence avec les calculs locaux sur les étoiles fermées étudiés au Chapitre 2 est que le sommet qui exécute ce pas de calcul ne peut pas modifier les étiquettes de ses voisins. Les calculs réalisés en utilisant uniquement ce type de réétiquetage sont appelés *calculs locaux sur les étoiles ouvertes*.

3.1.1 Résultats

Dans ce chapitre, on montre d'abord que les calculs locaux cellulaires sur les arêtes étiquetées ont la même puissance de calcul que les calculs locaux sur les arêtes étiquetées (Proposition 3.20).

Dans ces deux modèles, on caractérise les graphes admettant un algorithme d'élection et de nommage (dans ce modèle, élection et nommage peuvent être résolus sur les mêmes graphes). Cette caractérisation est basée sur la notion de *revêtements* (qui correspondent aux homomorphismes de graphes localement bijectifs) et utilise un résultat d'impossibilité d'Angluin [Ang80].

On montre qu'un graphe admet un algorithme d'élection ou de nommage utilisant des calculs locaux (cellulaire) sur les arêtes étiquetées si et seulement s'il est minimal pour les revêtements (Théorème 3.29). L'algorithme d'énumération qu'on présente dans la Section 3.5.2 est une adpatation de l'algorithme de Mazurkiewicz. Cependant, dans le modèle considéré ici, un sommet ne peut pas consulter l'état de tous ses voisins pour mettre à jour son étiquette. Afin de permettre à chaque sommet de distinguer ses voisins, un identifiant va être attribué à chaque arête du graphe de telle sorte que deux arêtes incidentes à un même voisin aient des identifiants différents.

On étudie ensuite l'influence de la connaissance initiale du degré et on montre que si initialement, chaque sommet connaît son degré, alors les calculs locaux (cellulaires) sur les arêtes étiquetées avec connaissance initiale du degré ont la même puissance de calcul que les calculs locaux sur les étoiles ouvertes (Proposition 3.42). On montre ensuite que la connaissance initiale du degré permet d'obtenir un modèle de calcul strictement plus puissant, bien que les graphes dans lesquels on peut élire en utilisant des calculs locaux sur les étoiles ouvertes sont les mêmes que ceux qui admettent un algorithme d'élection utilisant des calculs locaux (cellulaires) sans connaissance initiale du degré.

On montre par ailleurs que si chaque sommet connaît initialement son degré, il suffit de connaître une borne sur le diamètre pour pouvoir nommer ou élire dans un graphe minimal pour les revêtements (Théorème 3.49) et que la connaissance d'une borne serrée sur la taille permet d'obtenir un algorithme effectif d'élection et de nommage (Théorème 3.50).

Les résultats présentés dans ce chapitre ont été obtenus en collaboration avec Yves Métiver et une partie de ces résultats a été publiée dans [CM04].

3.1.2 Travaux Liés

Dans [Ang80], Angluin a pour la première fois donné un résultat d'impossibilité pour le problème de l'élection dans les réseaux anonymes. Ce résultat est basé sur la notion de revêtement et le résultat d'impossibilité qu'on présente dans ce Chapitre (Proposition 3.22) est très proche du résultat d'Angluin. Le modèle considéré par Angluin est proche des modèles étudiés dans ce chapitre et on montre dans la Section 3.6.1 que le modèle d'Angluin a une puissance de calcul équivalent aux calculs locaux sur les arêtes étiquetés avec connaissance initiale du degré. On obtient de cette manière la première caractérisation des graphes dans lesquels on peut résoudre le problème de l'élection dans le modèle d'Angluin (Théorème 3.35).

Dans un réseau globalement asynchrone, le mode de communication synchrone a été étudié par Charron-Bost et al. [CBMT96]. Dans ce modèle, lorsqu'une communication a lieu entre deux processeurs, l'émetteur et le destinataire du message se synchronisent

lors de la transmission. On montre dans la Section 3.6.1 que ce modèle est équivalent aux modèles considérés dans ce chapitre lorsque chaque sommet connaît initialement son degré et on en déduit donc une caractérisation des réseaux admettant un algorithme d'élection dans ce modèle (Théorème 3.36). Le langage CSP (Communicating Sequential Processes) introduit par Hoare [Hoa78] et le π-calcul introduit par Milner [Mil99] sont des exemples de langages où les processus communiquent de manière synchrone. Bougé a étudié dans [Bou88] le problème de l'élection symétrique dans le cadre de CSP ; les résultats qu'il a obtenu reposent sur la notion d'automorphisme et sont différents des résultats présentés dans ce chapitre puisque dans le modèle qu'il considère, chaque sommet connaît initialement le graphe et sa position dans ce graphe. Palamidessi a étudié dans [Pal03] le même problème dans le cadre du π-calcul afin de montrer qu'il existe une hiérarchie stricte entre différents modèles du π-calcul.

3.2 Revêtements

Dans ce chapitre, les homomorphismes localement bijectifs, i.e., les revêtements, sont les homomorphismes qui permettent de donner des conditions nécessaires que doivent vérifier les graphes admettant un algorithme de nommage ou d'élection. Il faut noter que contrairement au Chapitre 2, les graphes considérés dans ce chapitre peuvent avoir des arêtes multiples (mais pas de boucle).

Définition 3.1 *Un graphe G est un revêtement d'un graphe H à travers un homomorphisme $\gamma\colon G \to H$ si pour tout sommet $v \in V(G)$, γ induit une bijection entre $I_G(u)$ et $I_H(\varphi(u))$, i.e., si les conditions suivantes sont vérifiées :*

- $|I_G(u)| = |I_H(\gamma(u))|$,
- $\gamma(I_G(u)) = I_H(\gamma(u))$.

On dit alors que l'homomorphisme γ est localement bijectif.

Un graphe G est un revêtement propre *de H si γ n'est pas un isomorphisme et G est* minimal pour les revêtements *si G n'est un revêtement propre d'aucun autre graphe.*

Naturellement, un graphe étiqueté (G, λ) est un revêtement d'un graphe étiqueté (H, η) à travers γ si G est un revêtement de H à travers γ et si γ conserve l'étiquetage.

Remarque 3.2 *Si les deux graphes* **G** *et* **H** *sont simples, on remarque que la notion de revêtement correspond à la notion de revêtement simple.*

Exemple 3.3 *Le graphe* **G** *de la Figure 11 est un revêtement de* **H** *à travers l'homomorphisme γ.*

Le graphe **G** *est un revêtement propre de* **H** *et n'est donc pas minimal pour les revêtements ; le graphe* **H** *est minimal pour les revêtements.*

Puisqu'on ne considère que des graphes connexes, tout homomorphisme localement bijectif est surjectif.

Proposition 3.4 *Si un graphe* **G** *est un revêtement d'un graphe connexe* **H** *à travers γ, alors γ est surjectif.*

Preuve : Soit **G** un revêtement d'un graphe **H** à travers un homomorphisme γ.

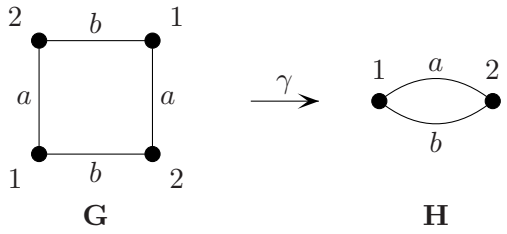


Fig. 11 – Le graphe **G** est un revêtement de **H** à travers l'homomorphisme γ qui envoie chaque sommet (resp. chaque arête) de **G** étiqueté i sur l'unique sommet (resp. arête) de **H** dont l'étiquette est i.

On considère un sommet $v \in \gamma(V(G))$. Il existe donc $u \in V(G)$ tel que $\gamma(u) = v$. Puisque γ induit une bijection entre $I_G(u)$ et $I_H(v)$, pour toute arête $f \in I_H(v)$, il existe une arête $e \in I_G(u)$ telle que $\gamma(e) = f$ et donc $f \in \gamma(E(G))$.

De même pour toute arête $f \in \gamma(E(G))$, il existe $e \in E(G)$ telle que $\gamma(e) = f$. Puisque γ est un homomorphisme, $ext(f) = \gamma(ext(e))$ et ainsi pour tout $v \in ext(f)$, $v \in \gamma(V(G))$.

Ainsi, puisque **H** est connexe et que $\gamma(V(G)$ est non-vide, on sait que γ est un homomorphisme surjectif de **G** dans **H**. □

Comme pour les revêtements simples, si un graphe **G** est un revêtement d'un graphe **H** connexe, alors tous les sommets et les arêtes de **H** ont le même nombre d'antécédents, qui est appelé le nombre de feuillets du revêtement.

Proposition 3.5 *Si un graphe* **G** *est un revêtement d'un graphe connexe* **H** *à travers* γ*, alors il existe une constante* q *telle que pour tout* $x \in V(H) \cup E(H)$, $|\gamma^{-1}(x)| = q$.

Cette constante q *est appelée le* nombre de feuillets*du revêtement.*

Preuve : On considère un graphe **G** qui est un revêtement d'un graphe connexe **H** à travers un homomorphisme γ. On considère un sommet $v \in V(H)$ et une arête $f \in I_H(v)$.

Puisque γ est localement bijectif, pour tout sommet $u \in \gamma^{-1}(v)$, il existe une unique arête $e \in I_G(u)$ telle que $\gamma(e) = f$. De plus, puisque **H** ne contient pas de boucle, on sait que $|\gamma(ext(e))| = |ext(f)| = 2$ et par conséquent, e a une seule extrémité dont l'image est v. Ainsi $|\gamma^{-1}(f)| \geq |\gamma^{-1}(v)|$.

Réciproquement, pour toute arête $e \in \gamma^{-1}(f)$, il existe un unique sommet $u \in ext(e)$ tel que $\gamma(u) = v$. Puisque γ est localement bijectif, on sait que pour toute arête $e' \in I_G(u)$ différente de e, $\gamma(e') \neq f$. Par conséquent, $|\gamma^{-1}(f)| \leq |\gamma^{-1}(u)|$.

Ainsi, puisque le graphe **H** est connexe, pour tout $x, x' \in V(H) \cup E(H)$, $|\gamma^{-1}(x)| = |\gamma^{-1}(x')|$. □

Exemple 3.6 *On présente ici quelques exemples de graphes connexes simples qui sont minimaux pour les revêtements.*

- *les graphes dont le nombre d'arêtes et le nombre de sommets sont premiers entre eux,*
- *les arbres,*
- *les anneaux de taille première.*

La notion de coloration régulière permet de caractériser les graphes simples non-étiquetés minimaux pour les revêtements en termes de colorations de graphes. Un étiquetage ℓ d'un graphe simple non-étiqueté est une coloration régulière si deux sommets voisins ont des couleurs distinctes et si le graphe induit par deux classes de couleurs adjacentes est un graphe biparti régulier.

Définition 3.7 *Une* coloration régulière *d'un graphe simple G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un stable,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ est un graphe biparti régulier.*

Dans la proposition suivante, on caractérise les graphes qui sont minimaux pour les revêtements à l'aide de la notion de coloration régulière. On rappelle qu'une coloration ℓ est propre si $|\ell(V(G))| < |V(G)|$.

Proposition 3.8 *Un graphe simple non-étiqueté G est minimal pour les revêtements si et seulement si G n'admet aucune coloration régulière propre.*

Preuve : On considère un graphe simple non-étiqueté G qui est un revêtement propre d'un graphe H à travers un homomorphisme γ.

On va montrer que γ est une coloration régulière de G et que l'ensemble des couleurs utilisés est $V(H)$. Puisque H ne contient pas de boucle, pour toute arête $\{u, u'\} \in E(G)$, $\gamma(u) \neq \gamma(u')$. Pour tous $v, v' \in V(H)$, on note $d_{\{v,v'\}}$ le nombre d'arêtes $f \in E(H)$ telles que $ext(f) = \{v, v'\}$. Pour chaque arête $f \in E(H)$ telles que $ext(f) = \{v, v'\}$, tout sommet de $\gamma^{-1}(v) \cup \gamma^{-1}(v')$ est incident à exactement une arête de $\gamma^{-1}(f)$. Par conséquent, tout sommet de $\gamma^{-1}(v)$ (resp. $\gamma^{-1}(v')$) est adjacent à exactement $d_{\{v,v'\}}$ sommets de $\gamma^{-1}(v')$ (resp. $\gamma^{-1}(v)$). Le graphe $G[v, v']$ est donc un graphe biparti $d_{\{v,v'\}}$-régulier. Finalement, puisque $|V(H)| < |V(G)|$, γ est une coloration régulière propre de G.

Réciproquement, étant donnée une coloration régulière propre ℓ d'un graphe G, on va définir un graphe H tel que G soit un revêtement de H. On pose $V(H) = \ell(V(G))$. Pour tous $v, v' \in V(H)$, il existe un entier $d_{\{v,v'\}}$ tel que $G[v, v']$ est un graphe biparti $d_{\{v,v'\}}$-régulier et alors, on définit $E(H)$ de la manière suivante, pour tous sommets $v, v' \in E(H)$, il existe $d_{\{v,v'\}}$ arêtes $f_{1,\{v,v'\}}, \ldots, f_{d_{\{v,v'\}},\{v,v'\}} \in E(H)$ dont les extrémités sont v et v'. D'après le théorème des mariages de Hall [Hal35], on sait qu'il existe un couplage parfait dans $G[v, v']$ et on peut donc partitionner les arêtes de $G[v, v']$ en $d_{\{v,v'\}}$ couplages parfaits $M_{1,\{v,v'\}}, \ldots, M_{d_{\{v,v'\}},\{v,v'\}}$. Ainsi, on peut définir γ de la manière suivante. pour tout sommet $u \in V(G)$, $\gamma(u) = \ell(u)$ et pour toute arête $e \in E(G)$, $\gamma(e) = f_{i,\{v,v'\}}$ si $e \in M_{i,\{v,v'\}}$. Puisque chaque $M_{i,\{v,v'\}}$ est un couplage parfait de $G[v, v']$, tout sommet $u \in \gamma^{-1}(v)$ est incident à exactement une arête $e \in \gamma^{-1}(f_{i,\{v,v'\}})$ et γ est donc un homomorphisme localement bijectif de G dans H. Puisque $|V(H)| = |\ell(V(G))| < |V(G)|$, G et H ne sont pas isomorphes et G n'est donc pas minimal pour les revêtements. □

3.3 Calculs Locaux (Cellulaires) sur les Arêtes Étiquetées

Dans cette partie, on présente les définitions formelles des calculs locaux sur les arêtes étiquetées et des calculs locaux cellulaires sur les arêtes étiquetées puis on étudie leurs

relations avec les revêtements.

3.3.1 Définitions

On rappelle qu'informellement, les calculs locaux sur les arêtes étiquetées sont les calculs réalisés en utilisant uniquement des règles de la forme présentée sur la Figure 9 : à chaque pas de calcul, l'étiquette d'une arête et de ses extrémités sont modifiées par l'application d'une règle qui ne dépend que des étiquettes de l'arête et de ses extrémités. On présente maintenant un définition plus formelle du modèle.

On rappelle la définition des relations de réétiquetage localement engendrée sur les arêtes (Definition 1.31).

Définition 3.9 *Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les arêtes si la condition suivante est satisfaite. Pour tous graphes (G,λ), (G,λ'), (H,η), (H,η') et toutes arêtes $e \in E(G)$ et $f \in E(H)$ telles que $ext(e) = \{v_1, v_2\}$ et $ext(f) = \{w_1, w_2\}$, si les trois conditions suivantes sont vérifiées :*

1. *$\lambda(v_1) = \eta(w_1)$, $\lambda(v_2) = \eta(w_2)$, $\lambda(e) = \eta(f)$, $\lambda'(v_1) = \eta'(w_1)$, $\lambda'(v_2) = \eta'(w_2)$ et $\lambda'(e) = \eta'(f)$*
2. *pour tout sommet $v \in V(G)$ différent de v_1 et de v_2, $\lambda(v) = \lambda'(v)$ et pour toute arête $e' \in E(G)$ différente de e, $\lambda(e') = \lambda'(e')$,*
3. *pour tout sommet $w \in V(H)$ différent de w_1 et de w_2, $\lambda(w) = \lambda'(w)$ et pour toute arête $f' \in E(H)$ différente de f, $\lambda(f') = \lambda'(f')$,*

alors $(G,\lambda)\ \mathcal{R}\ (G,\lambda')$ si et seulement si $(H,\eta)\ \mathcal{R}\ (H,\eta')$.

Une relation de réétiquetage $\mathcal{R}$ localement engendrée sur les arêtes est cellulaire si lors de l'application d'une règle de réétiquetage, l'étiquette d'un seul sommet est modifiée.

Définition 3.10 *Une relation de réétiquetage $\mathcal{R}$ localement engendrée sur les arêtes est* cellulaire *si lorsque $(G,\lambda)\ \mathcal{R}\ (G,\lambda')$ alors il existe un sommet $v \in V(G)$ tel que $\forall w \in V(G), w \neq v \implies \lambda(w) = \lambda'(w)$.*

Par définition, les *calculs locaux sur les arêtes étiquetées* correspondent aux relations de réétiquetage localement engendrées sur les arêtes et les *calculs locaux cellulaires sur les arêtes étiquetées* correspondent aux relations de réétiquetage cellulaires localement engendrées sur les arêtes.

Comme indiqué dans le Chapitre 1, une relation de réétiquetage localement engendrée sur les arêtes peut être décrite par un ensemble récursif de règles de la forme présentées sur la Figure 9. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage localement engendrée sur les arêtes. De même, une relation de réétiquetage cellulaire localement engendrée sur les arêtes peut être décrite par un ensemble récursif de règles de la forme présentées sur la Figure 10 et un tel ensemble de règles induit une relation de réétiquetage cellulaire localement engendrée sur les arêtes. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

3.3.2 Revêtements et Calculs Locaux sur les Arêtes Étiquetées

On présente maintenant le lemme qui met en évidence le lien entre les calculs locaux sur les arêtes étiquetées et les revêtements. Ce lemme de relèvement est le lemme de

relèvement d'Angluin [Ang80] adapté aux revêtements de graphes qui peuvent avoir des arêtes multiples.

Lemme 3.11 (Lemme de relèvement [Ang80]) *On considère un graphe* $\mathbf{G}$ *qui est un revêtement d'un graphe* $\mathbf{H}$ *à travers un homomorphisme* γ *et une relation* $\mathcal{R}$ *de réétiquetage localement engendrée sur les arêtes. Si* $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$*, alors il existe* $\mathbf{G}'$ *tel que* $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$ *et* $\mathbf{G}'$ *est un revêtement de* $\mathbf{H}'$ *à travers* γ*.*

Preuve : Il suffit de prouver ce lemme pour un pas de calcul. On considère deux graphes (G, λ) et (H, η) tels que (G, λ) est un revêtement de (H, ν) à travers γ. On considère un pas de réétiquetage sur H sur une arête f dont les extrémités sont w et w' et on note η' le nouvel étiquetage de H obtenu après ce pas de réétiquetage.

Puisque γ est un homomorphisme qui préserve l'étiquetage, pour chaque arête $e \in \gamma^{-1}(f)$, $\lambda(e) = \eta(f)$ et il existe $v, v' \in ext(e)$ tels que $\gamma(v) = w$, $\gamma(v') = w'$, $\lambda(v) = \eta(w)$ et $\lambda(v') = \eta(w')$. Il est donc possible d'appliquer un pas de réétiquetage sur e de telle sorte que l'étiquette de e (resp. v, v') devienne $\eta(f)$ (resp. $\eta(w)$, $\eta(w')$). De plus, puisque γ est localement bijectif, on sait que pour toutes arêtes $e, e' \in \gamma^{-1}(f)$, il n'existe pas de sommet $v \in ext(e) \cap ext(e')$. En effet, dans le cas contraire, γ ne serait pas localement bijectif en v. Par conséquent, on peut appliquer la règle de réétiquetage sur chaque arête $e \in \gamma^{-1}(f)$ et on note λ' l'étiquetage de G obtenu. Ainsi, puisque γ est localement bijectif, pour tout sommet $v \in \gamma^{-1}(w)$ (resp $v' \in \gamma^{-1}(w')$), il existe une arête $e \in \gamma^{-1}(f)$ incidente à v (resp. v') et on a donc $\lambda'(v) = \eta'(w)$ (resp. $\lambda'(v') = \eta'(w')$). Par conséquent, le graphe $\mathbf{G}' = (G, \lambda')$ est un revêtement de $\mathbf{H}' = (H, \lambda')$ à travers γ.

On note que chaque pas de réétiquetage dans $\mathbf{H}$ est simulé par plusieurs pas de réétiquetage dans $\mathbf{G}$ où la même règle de réétiquetage est appliquée. □

Le diagramme suivant représente la propriété du Lemme 3.11.

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow{\mathcal{R}^*} & \mathbf{G}' \\ {\scriptstyle \text{revêtement}}\downarrow & & \downarrow{\scriptstyle \text{revêtement}} \\ \mathbf{H} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{H}' \end{array}$$

Puisqu'une relation de réétiquetage cellulaire sur les arêtes étiquetées est aussi une relation de réétiquetage sur les arêtes étiquetées, le Lemme 3.11 est aussi vrai si la relation $\mathcal{R}$ est une relation de réétiquetage cellulaire sur les arêtes étiquetées.

3.4 Équivalence entre les Calculs Locaux Cellulaires sur les Arêtes Étiquetées et les Calculs Locaux sur les Arêtes Étiquetées

Dans cette partie, on montre que les calculs locaux cellulaires sur les arêtes étiquetées ont la même puissance de calcul que les calculs locaux sur les arêtes étiquetées.

On s'intéresse seulement aux algorithmes qui terminent et on définit maintenant la notion de simulation qu'on utilise par la suite.

Définition 3.12 *On considère deux algorithmes $\mathcal{R}_1$ et $\mathcal{R}_2$ utilisant respectivement des ensembles d'étiquettes L_1 et L_2 et une fonction $\pi : L_1 \to L_2$.*

On considère un graphe $\mathbf{G} = (G, \lambda)$ sur lequel toute exécution de $\mathcal{R}_1$ termine. Si toute exécution de $\mathcal{R}_2$ sur $\mathbf{G}$ termine et si de plus, pour tout configuration finale (G, λ'_2) atteinte par une exécution de $\mathcal{R}_2$ sur $\mathbf{G}$, il existe une configuration finale (G, λ'_1) atteinte par une exécution de $\mathcal{R}_1$ sur $\mathbf{G}$ telle que $\pi \circ \lambda'_2 = \lambda'_1$, alors on dit que $\mathcal{R}_2$ simule $\mathcal{R}_1$ sur $\mathbf{G}$ à travers π.

Si pour tout graphe $\mathbf{G}$ sur lequel toute exécution de $\mathcal{R}_1$ termine, $\mathcal{R}_2$ simule $\mathcal{R}_1$ à travers π, alors on dit que $\mathcal{R}_2$ simule $\mathcal{R}_1$.

Dans cette partie, on montre que tout algorithme utilisant des calculs locaux sur les arêtes étiquetées peut être simulé par un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées et réciproquement. Dans la définition précédente, on ne considère pas le problème de détection de la terminaison, i.e., les propriétés qui sont conservées par la simulation ne portent que sur l'étiquetage final. Cependant, on va aussi montrer qu'il existe un algorithme utilisant des calculs locaux sur les arêtes étiquetées permettant de résoudre un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison si et seulement s'il existe un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées permettant de résoudre $\mathcal{P}$ sur $\mathcal{F}$ avec détection de la terminaison.

Il est évident qu'un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées est un algorithme utilisant des calculs locaux sur les arêtes étiquetées. On s'intéresse donc à montrer qu'on peut obtenir un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées à partir d'un algorithme utilisant des calculs locaux sur les arêtes étiquetées.

3.4.1 Principe de l' Algorithme

Le principe de l'algorithme de simulation est le suivant. Chaque sommet et chaque arête a un statut qui peut être **free**, **ask**, **acc**, **ack** qui correspondent aux différentes étapes de la simulation d'une étape de réétiquetage de $\mathcal{R}$. Initialement, tous les sommets et toutes les arêtes ont le statut **free**.

Si les deux extrémités v et v' d'une arête e sont libres (leurs statuts sont **free**) et qu'une règle de l'algorithme $\mathcal{R}$ peut être appliquée sur e, alors le sommet v prend le statut **ask** et il modifie aussi le statut de e qui devient **ask** pour signifier que v veut appliquer une règle sur l'arête e, et non sur une autre arête incidente. De plus, on stocke dans l'étiquette de e le pas de calcul de $\mathcal{R}$ qui peut être simulé sur e.

Si un sommet v est libre et qu'une arête e incidente à v a le statut **ask**, alors si v n'a pas modifié son étiquette depuis que e a le statut **ask**, v prend le statut **acc** pour signifier qu'il accepte de simuler un pas de calcul de $\mathcal{R}$ sur e. Dans ce cas là, le statut de e devient **acc** et les étiquettes $\lambda(v)$ et $\lambda(e)$ sont modifiées selon la règle simulée.

Un sommet v dont le statut est **ask** et qui est incident à une arête dont le statut est **acc** modifie son étiquette $\lambda(v)$ selon la règle de $\mathcal{R}$ qui est simulé et il redevient libre. Le statut de e devient alors **ack**. Un sommet v dont le statut est **acc** et qui est incident à une arête dont le statut est **ack** peut prendre le statut **free** seulement si l'autre extrémité de e a le statut **free**.

Afin d'éviter tout inter-blocage lors de l'exécution, on permet à un sommet v dont le statut est **ask** de redevenir libre si le voisin v' avec lequel v voulait simuler une règle de

$\mathcal{R}$ a modifié son étiquette $\lambda(v')$ et si v' est libre.

Remarque 3.13 *L'intérêt de ne pas permettre à un sommet de redevenir libre dès qu'il accepte de simuler un pas de calcul est de s'assurer que si un sommet a le statut* **free***, alors pour toutes les règles qu'il a simulées, les étiquettes $\lambda(x)$ de tous les sommets et arêtes qui ont été impliquées dans l'application de ces règles ont bien été modifiées.*

3.4.2 Étiquettes

On décrit les étiquettes utilisées pour obtenir un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées à partir d'un algorithme $\mathcal{R}$ utilisant des calculs locaux sur les arêtes. On suppose que l'étiquette $\perp$ n'apparaît pas dans les étiquettes utilisées par $\mathcal{R}$.

Chaque sommet v a une étiquette $(\lambda(v), \mathbf{status}(v))$ dont les différents champs ont les significations suivantes :

- $\lambda(v)$ est l'étiquette de v dans l'algorithme $\mathcal{R}$ simulé,
- $\mathbf{status}(v) \in \{\mathbf{free}, \mathbf{ask}, \mathbf{acc}\}$ est le statut de v et correspond aux différentes étapes de la simulation,

L'étiquette initiale de chaque sommet v est de la forme $(\lambda(v), \mathbf{free})$: cela signifie que le sommet v peut simuler une règle avec n'importe quel voisin.

Chaque arête e a une étiquette $(\lambda(e), \mathbf{status}(e), r(e))$ dont les champs ont la signification suivante :

- $\lambda(e)$ est l'étiquette de e dans l'algorithme $\mathcal{R}$ simulé,
- $\mathbf{status}(e) \in \{\mathbf{free}, \mathbf{ask}, \mathbf{acc}, \mathbf{ack}\}$ est le statut de e et correspond aux différentes étapes de la simulation,
- $r(e) = (\ell_1, \ell_e, \ell_2, \ell'_1, \ell'_e, \ell'_2)$ est le pas de réétiquetage qui est en cours de simulation sur e ; si le statut de e est **free**, alors $r(e) = r_\perp = (\perp, \perp, \perp, \perp, \perp, \perp)$ puisqu'aucune règle n'est simulée sur e.

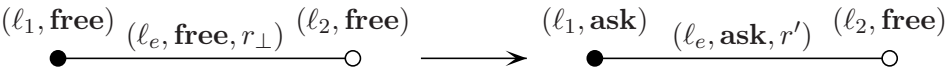
Initialement, chaque arête e à l'étiquette $(\lambda(e), \mathbf{free}, r_\perp)$ puisqu'aucune règle n'est simulée sur e.

3.4.3 Règles de Réétiquetage

On présente ici les règles de réétiquetage qui permettent de simuler les calculs locaux sur les arêtes étiquetées à l'aide de calculs locaux cellulaires sur les arêtes étiquetées. Les règles suivantes correspondent aux différentes étapes évoquées dans la description informelle de l'algorithme.

Si les deux extrémités d'une arête ont le statut **free** et qu'une règle de $\mathcal{R}$ peut être appliquée sur cette arête, la première règle permet à une des extrémités de modifier son statut et celui de l'arête afin d'essayer de simuler la règle de $\mathcal{R}$. Dans ce cas là les statuts de l'arête et du sommet passent à **ask** et on stocke dans l'étiquette de l'arête le pas de réétiquetage qui peut être simulé.

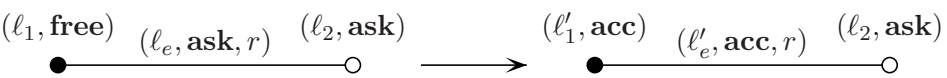
$\mathcal{S}_1(\mathcal{R})$:



Cette règle est applicable si on peut appliquer une règle de $\mathcal{R}$ sur une arête étiquetée (ℓ_1, ℓ_e, ℓ_2) pour obtenir l'étiquetage $(\ell'_1, \ell'_e, \ell'_2)$, auquel cas $r' = (\ell_1, \ell_e, \ell_2, \ell'_1, \ell'_e, \ell'_2)$.

Si un sommet v a un voisin v' qui a appliqué la règle $\mathcal{S}_1(\mathcal{R})$, si le statut de v est **free** et que l'étiquette $\lambda(v)$ n'a pas été modifiée entre-temps, v peut appliquer la deuxième règle le long de l'arête e reliant v à v'. Les statuts de v et de e deviennent alors **acc** et les étiquettes $\lambda(v)$ et $\lambda(e)$ sont modifiées selon la règle simulée.

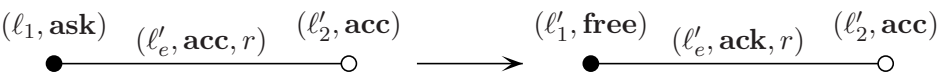
$\mathcal{S}_2(\mathcal{R})$:



Cette règle est applicable si $r = (\ell_2, \ell_e, \ell_1, \ell'_2, \ell'_e, \ell'_1)$.

La troisième règle peut être appliquée par un sommet v qui avait appliqué la règle $\mathcal{S}_1(\mathcal{R})$ en fonction d'un voisin v' le long d'une arête e si v' a accepté de simuler la règle le long de e. Dans ce cas là, le sommet v modifie son étiquette $\lambda(v)$ et retrouve le statut **free**. Le statut de l'arête devient **ack** pour signifier au sommet v' que la simulation a bien été effectuée.

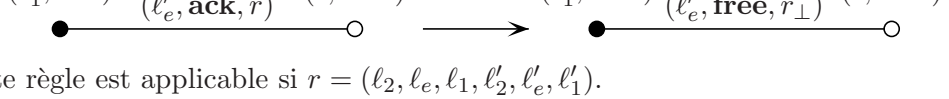
$\mathcal{S}_3(\mathcal{R})$:



Cette règle est applicable si $r = (\ell_1, \ell_e, \ell_2, \ell'_1, \ell'_e, \ell'_2)$.

La quatrième règle permet à un sommet v qui a accepté de simuler une règle de retrouver le statut **free** s'il existe une arête e incidente à v dont le statut est **ack** et dont l'autre extrémité v' a le statut **free**. Dans ce cas là, l'arête e retrouve aussi son statut **free**.

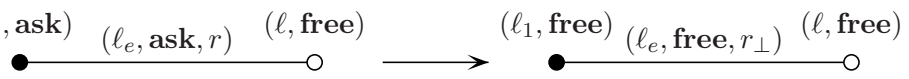
$\mathcal{S}_4(\mathcal{R})$:



Cette règle est applicable si $r = (\ell_2, \ell_e, \ell_1, \ell'_2, \ell'_e, \ell'_1)$.

La cinquième règle a pour rôle d'éviter tout inter-blocage lors de l'exécution de l'algorithme de simulation. Si un sommet v avait exécuté la règle $\mathcal{S}_1(\mathcal{R})$ avec un voisin v' le long d'une arête e et qu'entre temps, l'étiquette $\lambda(v')$ a été modifiée, parce que v' a réussi a simuler une règle avec un autre de ses voisins, alors la cinquième règle permet au sommet v de retrouver le statut **free**.

$\mathcal{S}_5(\mathcal{R})$:



Cette règle est applicable si $r = (\ell_1, \ell_e, \ell_2, \ell'_1, \ell'_e, \ell'_2)$ et $\ell \neq \ell_2$.

3.4.4 Correction de l'Algorithme de Simulation

On considère un graphe étiqueté $\mathbf{G}$ et une exécution de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$. On note $(\lambda(v), \mathbf{status}_i(v))$ l'étiquette du sommet v et $(\lambda(e), \mathbf{status}_i(e), r_i)$ l'étiquette de l'arête e après la $i$ème étape de l'exécution. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Afin de simplifier les preuves, on va associer un graphe simple orienté D_i à chaque étape de l'exécution de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$. À tout moment, l'ensemble des sommets de D est l'ensemble des arêtes et des sommets de G, i.e., $V(D_i) = \{u_e \mid e \in E(G)\} \cup \{u_v \mid v \in V(G)\}$. Initialement, D ne contient aucun arc, i.e., $A(D_0) = \emptyset$.

Pour construire le graphe D_{i+1} à partir de D_i, on considère le pas de réétiquetage effectué à l'étape i sur le graphe $\mathbf{G}$. Si la règle $\mathcal{S}_1(\mathcal{R})$ ou $\mathcal{S}_2(\mathcal{R})$ est appliquée sur une arête e et modifie l'état d'un sommet v, on ajoute un arc orienté de u_v à u_e étiqueté v et un arc orienté de u'_e à u_v pour toute arête $e' \neq e$ incidente à v dans G, i.e., $E(D_{i+1}) = E(D_i) \cup \{(u_v, u_e)\} \cup \{(u_{e'}, u_v) \mid e' \in I_G(v) \text{ et } e' \neq e\}$.

Si la règle $\mathcal{S}_3(\mathcal{R})$, $\mathcal{S}_4(\mathcal{R})$ ou $\mathcal{S}_5(\mathcal{R})$ est appliquée lors de l'étape i sur une arête e et modifie l'état d'un sommet v, on supprime toutes les arêtes incidentes à u_v, $E(D_{i+1}) = E(D_i) \setminus (\{(u_v, u_e) \in E(D)\} \cup \{(u_{e'}, u_v) \in E(D)\})$.

On remarque que le sommet $u_e \in V(D_i)$ correspondant à une arête $e \in E(G)$ ne peut être voisin que des sommets $u_v, u_{v'}$ correspondant aux extrémités de e dans G.

Dans le lemme suivant, on montre comment le graphe D_i est lié à l'état des sommets et des arêtes de $\mathbf{G}$ à l'étape i.

Lemme 3.14 *Pour toute étape i, toute arête $e \in E(G)$ et tout sommet $v \in V(G)$, les propriétés sont satisfaites.*

(1) Le graphe D_i est un graphe simple orienté qui ne contient pas de circuit.

(2) Si $\mathbf{status}_i(e) = \mathbf{free}$, alors il n'existe aucun arc $(u_v, u_e) \in A(D_i)$.

(3) Si $\mathbf{status}_i(e) = \mathbf{ask}$, alors il existe exactement un arc $(u_v, u_e) \in A(D_i)$ et de plus, $v \in ext(e)$, $\mathbf{status}_i(v) = \mathbf{ask}$ et $r_i(e) = (\lambda_i(v), \lambda_i(e), \ell_2, \ell'_1, \ell'_e, \ell'_2)$.

(4) Si $\mathbf{status}_i(e) = \mathbf{acc}$, alors il existe $v, v' \in ext(e)$ tels que $(u_v, u_e), (u_{v'}, u_e) \in A(D_i)$, $\mathbf{status}_i(v) = \mathbf{acc}$ et $\mathbf{status}_i(v') = \mathbf{ask}$. De plus, $r_i(e) = (\lambda_i(v'), \ell_e, \ell_2, \ell'_1, \lambda_i(e), \lambda_i(v))$.

(5) Si $\mathbf{status}_i(e) = \mathbf{ack}$, alors il existe exactement un arc $(u_v, u_e) \in A(D_i)$ et de plus, $v \in ext(e)$, $\mathbf{status}_i(v) = \mathbf{acc}$ et $r_i(e) = (\ell_1, \ell_e, \ell_2, \ell'_1, \lambda_i(e), \lambda_i(v))$.

(6) Si $\mathbf{status}_i(v) = \mathbf{free}$, alors u_v n'est incident à aucun arc dans $\mathbf{D}_i$.

(7) Si $\mathbf{status}_i(v) = \mathbf{ask}$, alors il existe une arête $e \in I_G(v)$ telle que $(u_v, u_e) \in A(D_i)$ et $\mathbf{status}_i(e) \in \{\mathbf{ask}, \mathbf{acc}\}$. De plus, pour tout $e' \in I_G(v)$, $(u_{e'}, u_v) \in A(D_i)$.

(8) si $\mathbf{status}_i(v) = \mathbf{acc}$, alors il existe une arête $e \in I_G(v)$ telle que $(u_v, u_e) \in A(D_i)$ et $\mathbf{status}_i(e) \in \{\mathbf{acc}, \mathbf{ack}\}$. De plus, pour tout $e' \in I_G(v)$, $(u_{e'}, u_v) \in A(D_i)$.

Preuve : On va montrer le lemme par récurrence sur le nombre d'étapes i. Initialement, $E(D_0) = \emptyset$ et pour tout $x \in V(G) \cup E(G)$, $\mathbf{status}(x) = \mathbf{free}$: les propriétés sont donc vérifiées. On suppose maintenant que les propriétés sont vraies à l'étape i. On suppose qu'à l'étape $i+1$ une règle de réétiquetage est appliquée sur une arête e dont les extrémités sont v et v' et que l'étiquette de v est modifiée.

Si la règle $\mathcal{S}_1(\mathcal{R})$ ou $\mathcal{S}_2(\mathcal{R})$ est appliquée à l'étape $i+1$, alors $\mathbf{status}_i(v) = \mathbf{free}$ et puisque tous les arcs ajoutés à $A(D_i)$ pour construire D_{i+1} sont incidents à u_v, on sait que le graphe D_i reste simple. On sait que les seuls voisins que u_e peut avoir dans D_i sont u_v et $u_{v'}$. Si la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée, alors puisque $\mathbf{status}_i(v) = \mathbf{status}_i(v') = \mathbf{free}$, u_e n'a pas de voisins dans D_i et D_{i+1} ne contient donc pas de circuit. Si la règle $\mathcal{S}_2(\mathcal{R})$ est appliquée alors on sait que $(u_{v'}, u_e) \in A(D_i)$ et par conséquent, il n'existe aucun arc $(u_e, u_{v''}) \in A(D_{i+1})$. Ainsi la propriété (1) est vérifiée à l'étape $i+1$. Si la règle $\mathcal{S}_3(\mathcal{R})$, $\mathcal{S}_4(\mathcal{R})$ ou $\mathcal{S}_5(\mathcal{R})$ est appliquée à l'étape $i+1$, on supprime des arcs de D_i pour construire D_{i+1} et la propriété (1) est donc conservée.

Quelle que soit la règle appliquée à l'étape $i+1$, pour tout sommet $v'' \notin \{v, v'\}$, aucun arc incident à v'' n'est ajouté ou supprimé et les étiquettes des arêtes e' incidentes à v'' ne sont pas modifiées. Ainsi, les propriétés (6,7,8) sont conservées pour tous les sommets $v'' \notin \{v, v''\}$.

Si la règle $\mathcal{S}_1(\mathcal{R})$ ou $\mathcal{S}_2(\mathcal{R})$ est appliquée lors de l'étape $i+1$, alors pour toute arête $e' \neq e$, aucun arc $(u_{v''}, u_{e'})$ n'est ajouté à l'ensemble $A(D_i)$ pour construire D_{i+1}. Par conséquent, puisqu'aucun arc n'est supprimé, les propriétés (2,3,4,5) sont conservées pour toutes les arêtes $e' \neq e$.

Si la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée lors de l'étape $i+1$, alors $\mathbf{status}_i(v') = \mathbf{status}_{i+1}(v') = \mathbf{free}$ et puisqu'aucun arc incident à v' n'est ajouté, la propriété (6) est conservée pour v'. Par ailleurs, on sait que u_v est un sommet isolé dans D_i, qu'il n'existe aucun arc $(u_{v''}, u_e) \in A(D_i)$, et que $r_{i+1}(e) = (\lambda_{i+1}(v), \lambda_{i+1}(e), \lambda_{i+1}(v'), \ell'_1, \ell'_e, \ell'_2)$. Ainsi, de par la construction de $A(D_{i+1})$, la propriété (3) est vraie pour e et la propriété (7) est vraie pour v.

Si la règle $\mathcal{S}_2(\mathcal{R})$ est appliquée lors de l'étape $i+1$, alors $\mathbf{status}_i(e) = \mathbf{status}_i(v') = \mathbf{status}_{i+1}(v') = \mathbf{ask}$, $\mathbf{status}_i(v) = \mathbf{free}$ et $\mathbf{status}_{i+1}(v) = \mathbf{status}_{i+1}(e) = \mathbf{acc}$. Puisqu'aucun arc de D_i n'est incident à v, on sait que $(u_{v'}, u_e) \in A(D_i)$. Par conséquent, la propriété (7) est conservée à l'étape $i+1$ pour v'. Par construction et puisqu'on peut appliquer la règle $\mathcal{S}_2(\mathcal{R})$, on sait aussi qu'à l'étape $i+1$, la propriété (4) est vraie pour e et la propriété (8) est vraie pour v.

Si la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée lors de l'étape $i+1$, alors $\mathbf{status}_i(v) = \mathbf{ask}$, $\mathbf{status}_i(e) = \mathbf{status}_i(v') = \mathbf{status}_{i+1}(v') = \mathbf{acc}$, $\mathbf{status}_{i+1}(v) = \mathbf{free}$ et $\mathbf{status}_{i+1}(e) = \mathbf{ack}$. Par hypothèse de récurrence, on sait que $(u_v, u_e), (u_{v'}, u_e) \in A(D_i)$ et ainsi pour toute arête $e' \neq e$, aucun arc $(u_{v''}, u_{e'})$ n'est supprimé de l'ensemble $A(D_i)$ pour construire D_{i+1}. Par conséquent, puisqu'aucun arc n'est ajouté, les propriétés (2,3,4,5) sont conservées pour toutes les arêtes $e' \neq e$. De plus, puisqu'on supprime l'arc (u_v, u_e) et que la règle $\mathcal{S}_3(\mathcal{R})$ peut être appliquée, la propriété (5) est vraie pour e et la propriété (8) est vraie pour v'. Puisque tous les arcs incidents à v sont supprimés de $A(D_i)$ pour construire D_{i+1}, la propriété (6) est vraie pour v.

Si la règle $\mathcal{S}_4(\mathcal{R})$ ou la règle $\mathcal{S}_5(\mathcal{R})$ est appliquée, on a $\mathbf{status}_i(v') = \mathbf{status}_{i+1}(v') = \mathbf{status}_{i+1}(v) = \mathbf{free}$ et donc $(u_v, u_e) \in A(D_i)$. Ainsi, pour toute arête $e' \neq e$, on ne supprime aucun arc $(u_{v''}, u_{e'})$ et puisque l'arc (u_v, u_e) est supprimé, les propriétés (2,3,4,5)

sont conservées. Puisqu'on supprime tous les arcs incidents à v dans D_i, la propriété (6) est vraie pour v et v'.

Par conséquent, pour toute étape i, les propriétés (1,2,3,4,5,6,7,8) sont toujours vraies. □

On peut donc montrer facilement que dans toute configuration finale de l'algorithme, tous les sommets et toutes les arêtes ont le statut **free**, i.e., l'algorithme de simulation ne crée pas d'inter-blocage.

Lemme 3.15 *Dans toute configuration finale de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$, pour tout $x \in V(G) \cup E(G)$,* $\mathbf{status}(x) = \mathbf{free}$

Preuve : On considère une configuration finale de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$ et on note D le graphe orienté associé. On suppose qu'il existe un sommet ou une arête dont le statut est différent de **free**. D'après le Lemme 3.14, puisque D ne contient pas de circuit, on sait qu'il existe une arête $e \in E(G)$ telle que $\mathbf{status}(e) \neq \mathbf{free}$ et telle qu'il n'existe pas d'arc $(u_e, u_v) \in A(D)$.

Si $\mathbf{status}(e) = \mathbf{ask}$, alors d'après le Lemme 3.14, il existe $v \in ext(e)$ tel que $(u_v, u_e) \in A(D)$, $\mathbf{status}(v) = \mathbf{ask}$ et $r_i(e) = (\lambda(v), \lambda(e), \ell_2, \ell'_1, \ell'_e, \ell'_2)$. On note v' l'autre extrémité de e. Si $\mathbf{status}(v') \neq \mathbf{free}$, alors il existe un arc $(u_{v'}, u_{e'}) \in A(D)$ et $e' \neq e$ puisque $(u_v, u_e) \in A(D)$ et qu'il existe exactement un arc $(u_{v''}, u_e) \in A(D)$. Mais d'après le Lemme 3.14, si $(u_{v'}, u_{e'}) \in A(D)$, alors pour tout $e'' \in I_G(v')$, si $e'' \neq e'$, alors $(u_{e''}, u_{v'}) \in A(D)$. Cela implique alors que $(u_e, u_{v'}) \in A(D)$, ce qui est impossible de par notre choix de e. Par conséquent, si $\lambda(v) = \ell_2$, on peut appliquer la règle $\mathcal{S}_2(\mathcal{R})$ sur e et sinon, on peut appliquer la règle $\mathcal{S}_5(\mathcal{R})$ sur e. Dans les deux cas, l'exécution de $\mathcal{S}(\mathcal{R})$ n'est pas terminée.

Si $\mathbf{status}(e) = \mathbf{acc}$, alors d'après le Lemme 3.14, il existe $v, v' \in ext(e)$ tels que $\mathbf{status}(v) = \mathbf{acc}$, $\mathbf{status}_i(v') = \mathbf{ask}$ et $r(e) = (\lambda(v'), \ell_e, \ell_2, \ell'_1, \lambda(e), \lambda(v))$. Par conséquent, la règle $\mathcal{S}_3(\mathcal{R})$ peut être appliquée sur e et l'exécution de $\mathcal{S}(\mathcal{R})$ n'est pas terminée.

Si $\mathbf{status}(e) = \mathbf{ack}$, alors d'après le Lemme 3.14, il existe $v \in ext(e)$ tel que $(u_v, u_e) \in A(D)$, $\mathbf{status}(v) = \mathbf{acc}$, et $r(e) = (\ell_1, \ell_e, \ell_2, \ell'_1, \lambda(e), \lambda(v))$. Si on note v' l'autre extrémité de e, on peut montrer comme précédemment que de par notre choix de v, $\mathbf{status}(v') = \mathbf{free}$. Par conséquent, la règle $\mathcal{S}_4(\mathcal{R})$ peut être appliquée sur e et l'exécution de $\mathcal{S}(\mathcal{R})$ n'est pas terminée.

Ainsi, pour toute exécution de $\mathcal{S}(\mathcal{R})$ qui termine, tous les sommets et toutes les arêtes ont le statut **free** dans la configuration finale. □

Dans les trois lemmes suivants, on montre que l'algorithme de simulation ne permet de simuler que des règles de $\mathcal{R}$. Dans le lemme suivant, on montre que si un sommet v a le statut **ask** et qu'on peut appliquer la règle $\mathcal{S}_2(\mathcal{R})$ ou $\mathcal{S}_5(\mathcal{R})$ sur une arête e incidente à v, alors précédemment la règle $\mathcal{S}_1(\mathcal{R})$ a été appliquée sur e et entre-temps l'étiquette de v n'a pas été modifiée.

Lemme 3.16 *On considère une arête e dont les extrémités sont v et v'. On considère une étape i_1+1 lors de laquelle, ou bien la règle $\mathcal{S}_2(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v', ou bien la règle $\mathcal{S}_5(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v. Soit $i_2 \leq i_1$ la dernière étape lors de laquelle la règle $\mathcal{S}_1(\mathcal{R})$ a été appliquée sur e. Alors lors de l'étape i_2, l'étiquette de v a été modifiée et pour tout $i \in [i_2, i_1]$, $\mathbf{status}_i(v) = \mathbf{ask}$ et $\lambda_i(v) = \lambda_{i_2}(v)$.*

Preuve : On considère une arête e dont les extrémités sont v_1 et v'_1. On considère une étape i_1+1 lors de laquelle, ou bien la règle $\mathcal{S}_2(\mathcal{R})$ est appliquée sur e et modifie l'étiquette

de v'_1, ou bien la règle $\mathcal{S}_5(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v_1. On sait que $\mathbf{status}_{i_1}(e) = \mathbf{ask}$ et on considère la dernière étape $i_2 \leq i_1$ lors de laquelle la règle $\mathcal{S}_1(\mathcal{R})$ a été appliquée sur e. On note v_2 le sommet dont l'étiquette a été modifiée lors de l'étape i_2 et on va montrer que pour tout $i \in [i_1, i_2]$, $\mathbf{status}_i(v_2) = \mathbf{ask}$ et $\lambda_i(v_2) = \lambda_{i_2}(v_2)$ et que $v_2 = v_1$.

On sait que $(u_{v_2}, u_e) \in A(D_{i_2})$ et que pour toute étape $i \in [i_2, i_1]$, $\mathbf{status}_i(e) = \mathbf{ask}$ et donc il existe exactement un arc $(u_v, u_e) \in A(D_i)$. De plus, il n'existe aucune étape i lors de laquelle, on ajoute ou on supprime un arc $(u_{v''}, u_e)$ de $A(D_i)$ pour construire D_{i+1}. Par conséquent, aucune règle qui modifie l'étiquette de v_2 n'a pu être appliquée à une étape $i \in [i_2, i_1]$. Ainsi, pour toute étape $i \in [i_2, i_1]$, $(u_e, u_{v_2}) \in A(D_i)$, $\mathbf{status}_i(v_2) = \mathbf{ask}$ et $\lambda_i(v_2) = \lambda_{i_2}(v_2)$. En particulier, on a $(u_e, u_{v_2}) \in A(D_{i_1})$. Mais $\mathbf{status}_{i_1}(v'_1) = \mathbf{free}$ et donc $(u_e, u_{v'_1}) \notin A(D_{i_1})$. Puisque les seuls voisins de u_e dans D_{i_1} ne peuvent être que u_{v_1} et $u_{v'_1}$, cela signifie que $v_1 = v_2$. □

Dans le lemme suivant, on montre que si on applique la règle $\mathcal{S}_3(\mathcal{R})$ sur une arête e, alors précédemment la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée sur e et entre-temps, les étiquettes des extrémités de e n'ont pas été modifiées.

Lemme 3.17 *On considère une arête e dont les extrémités sont v et v'. On considère une étape $i_1 + 1$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v'. Soit $i_2 \leq i_1$ la dernière étape lors de laquelle la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée sur e. Alors lors de l'étape i_2, l'étiquette de v a été modifiée et pour tout $i \in [i_2, i_1]$, $\mathbf{status}_i(v) = \mathbf{ask}$, $\mathbf{status}_i(v') = \mathbf{acc}$, $\lambda_i(v) = \lambda_{i_2}(v)$ et $\lambda_i(v') = \lambda_{i_2}(v')$.*

Preuve : On considère une arête e dont les extrémités sont v_1 et v'_1. On considère une étape $i_1 + 1$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v'_1.

On sait que $\mathbf{status}_{i_1}(e) = \mathbf{acc}$ et on considère la dernière étape $i_2 \leq i_1$ lors de laquelle la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée sur e. On note v_2 le sommet dont l'étiquette a été modifiée lors de l'étape i_2 et v'_2 l'autre extrémité de e. On va montrer qu'aucune règle n'a pu modifier les étiquettes de v_2 et de v'_2 et que $v_1 = v_2$ et $v'_1 = v'_2$.

On sait que $(u_{v_2}, u_e), (u_{v'_2}, u_e) \in A(D_{i_2})$, que $\mathbf{status}_{i_2}(v) = \mathbf{ask}$, que $\mathbf{status}_{i_2}(v') = \mathbf{acc}$ et que pour toute étape $i \in [i_2, i_1]$, $\mathbf{status}_i(e) = \mathbf{acc}$. Ainsi, d'après le Lemme 3.14, pour tout $i \in [i_2, i_1]$, on sait que $(u_{v_2}, u_e), (u_{v'_2}, u_e) \in A(D_i)$. Si une règle est appliquée et modifie l'étiquette de $v \in \{v_2, v'_2\}$ lors d'une étape $i \in [i_2, i_1]$, alors $\mathbf{status}_i(v) = \mathbf{free}$ et $(u_e, u_v) \notin A(D_i)$, ce qui est impossible. Par conséquent, pour tout $i \in [i_1, i_2]$, $\mathbf{status}_i(v_2) = \mathbf{ask}$, $\mathbf{status}_i(v'_2) = \mathbf{acc}$, $\lambda_i(v_2) = \lambda_{i_2}(v_2)$ et $\lambda_i(v'_2) = \lambda_{i_2}(v'_2)$ et on a donc $v_1 = v_2$ et $v'_1 = v'_2$. □

Dans le lemme suivant, on montre que si la règle $\mathcal{S}_4(\mathcal{R})$ est appliquée sur une arête e dont une extrémité v a le statut **acc**, alors précédemment la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée sur e et entre-temps, l'étiquette de v n'a pas été modifiée.

Lemme 3.18 *On considère une arête e dont les extrémités sont v et v'. On considère une étape $i_1 + 1$ lors de laquelle la règle $\mathcal{S}_4(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v. Soit $i_2 \leq i_1$ la dernière étape lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée sur e. Alors lors de l'étape i_2, l'étiquette de v' a été modifiée et pour tout $i \in [i_2, i_1]$, $\mathbf{status}_i(v) = \mathbf{acc}$, et $\lambda_i(v) = \lambda_{i_2}(v)$.*

Preuve : On considère une arête e dont les extrémités sont v_1 et v'_1. On considère

une étape $i_1 + 1$ lors de laquelle la règle $\mathcal{S}_4(\mathcal{R})$ est appliquée sur e et modifie l'étiquette de v_1. On sait que $\mathbf{status}_{i_1}(e) = \mathbf{ack}$, que $\mathbf{status}_{i_1}(v_1) = \mathbf{acc}$ et que $\mathbf{status}_{i_1}(v_1) = \mathbf{status}_{i_1+1}(v_1') = \mathbf{status}_{i_1+1}(v_1) = \mathbf{free}$. On considère la dernière étape $i_2 \leq i_1$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée sur e. On note v_2' le sommet dont l'étiquette a été modifiée lors de l'étape i_2 et v_2 l'autre extrémité de e. On sait que $\mathbf{status}_{i_2}(v_2) = \mathbf{acc}$ et $\mathbf{status}_{i_2}(v_2') = \mathbf{free}$. On va montrer qu'aucune règle n'a pu modifier l'étiquette de v_2 et que $v_1 = v_2$.

On sait que $(u_e, u_{v_2}) \in A(D_{i_2})$ et que pour toute étape $i \in [i_2, i_1]$, $\mathbf{status}_i(e) = \mathbf{ack}$ et donc il existe exactement un arc $(u_v, u_e) \in A(D_i)$. Si une règle est appliquée et modifie l'étiquette de v_2 lors d'une étape $i \in [i_2, i_1]$, alors $\mathbf{status}_i(v_2) = \mathbf{free}$ et $(u_{v_2}, u_e) \notin A(D_i)$. Mais puisque $(u_{v_2}, u_e) \in A(D_{i-1})$, cela signifie qu'il n'existe pas d'arc $(u_v, u_e) \in A(D_i)$, ce qui est impossible. Par conséquent, pour tout $i \in [i_1, i_2]$, $\mathbf{status}_i(v_2) = \mathbf{acc}$, $\lambda_i(v_2) = \lambda_{i_2}(v_2)$ et on a donc $v_1 = v_2$. □

On montre dans le lemme suivant que la règle $\mathcal{S}_1(\mathcal{R})$ ne peut pas être appliquée infiniment souvent sans que les règles $\mathcal{S}_2(\mathcal{R}), \mathcal{S}_3(\mathcal{R})$ soient appliquées.

Lemme 3.19 *Pour toute exécution de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$, si la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée $|V(G)| + 1$ fois entre deux étapes i_1 et i_2, alors il existe une étape $i \in [i_1, i_2]$ lors de laquelle la règle $\mathcal{S}_2(\mathcal{R})$ (resp. $\mathcal{S}_3(\mathcal{R})$) a été appliquée.*

Preuve : L'étiquette $\lambda(v)$ d'un sommet $v \in V(G)$ ne peut être modifiée que si l'une des règles $\mathcal{S}_2(\mathcal{R}), \mathcal{S}_3(\mathcal{R})$ est appliquée sur v.

Puisque la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée $|V(G)| + 1$ fois entre deux étapes i_1 et i_2, alors il existe un sommet v et deux étapes $j_1 < j_2$ entre i_1 et i_2 tels que la règle $\mathcal{S}_1(\mathcal{R})$ modifie deux fois l'étiquette de v. On note e l'arête sur laquelle la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée lors de l'étape j_1 et v' l'extrémité de e différente de v. Si la règle $\mathcal{S}_1(\mathcal{R})$ peut être appliquée lors de l'étape j_2, cela signifie que la règle $\mathcal{S}_3(\mathcal{R})$ ou $\mathcal{S}_5(\mathcal{R})$ a été appliquée entre les étapes j_1 et j_2 et l'application de cette règle a modifiée l'étiquette de v.

Si la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée lors d'une étape $j \in [j_1, j_2]$ et que l'étiquette de v est modifiée lors de cette étape j, on sait d'après les Lemmes 3.16 et 3.17 que la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée lors d'une étape $j' \in [j_1, j]$ et que l'étiquette de v' a été modifiée lors de cette étape.

Si la règle $\mathcal{S}_5(\mathcal{R})$ est appliquée lors d'une étape $j \in [j_1, j_2]$ et que l'étiquette de v est modifiée lors de cette étape j, on sait d'après le Lemme 3.16 que l'étiquette de v' a été modifiée entre-temps et donc, une des règles $\mathcal{S}_2(\mathcal{R}), \mathcal{S}_3(\mathcal{R})$ a été appliquée lors d'une étape $j' \in [j_1, j]$ et a modifiée l'étiquette de v'.

Si la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée lors de l'étape j' et a modifié l'étiquette de v', puisque $\mathbf{status}_j(v') = \mathbf{free}$, il existe une étape $j'' \in [j', j]$ lors de laquelle la règle $\mathcal{S}_4(\mathcal{R})$ a été appliquée et a modifié l'étiquette de v'. D'après les Lemmes 3.17 et 3.18, la règle $\mathcal{S}_3(\mathcal{R})$ a donc été appliquée à une étape $j''' \in [j', j'']$.

Si la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée lors de l'étape j' et a modifié l'étiquette de v', alors d'après les Lemmes 3.16 et3.17, on sait que la règle $\mathcal{S}_1(\mathcal{R})$ a été appliquée sur une arête e' lors d'une étape $j'' \in [j_1, j']$ et que l'étiquette de v' a été modifiée lors de cette étape. De plus, on sait aussi que la règle $\mathcal{S}_2(\mathcal{R})$ a du être appliquée lors d'une étape $j''' \in [j'', j']$ sur l'arête e'.

Ainsi, si la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée $|V(G)| + 1$ fois entre deux étapes i_1 et i_2, alors

les deux règles $\mathcal{S}_2(\mathcal{R})$ et $\mathcal{S}_3(\mathcal{R})$ sont appliquées entre les étapes i_1 et i_2. □

Dans la proposition suivante, on montre que l'algorithme $\mathcal{S}(\mathcal{R})$ simule l'algorithme $\mathcal{R}$, ce qui prouve l'équivalence entre les calculs locaux cellulaires sur les arêtes étiquetées et les calculs locaux sur les arêtes étiquetées.

Proposition 3.20 *Pour tout algorithme $\mathcal{R}$ utilisant des calculs locaux sur les arêtes étiquetées, l'algorithme $\mathcal{S}(\mathcal{R})$ utilise des calculs locaux cellulaires sur les arêtes étiquetées et simule l'algorithme $\mathcal{R}$.*

Preuve : On considère la fonction π telle que $\pi((\ell, \mathbf{status})) = \ell$ et $\pi((\ell, \mathbf{status}, r)) = \ell$. On veut montrer que pour tout graphe $\mathbf{G}$ telle que toute exécution de $\mathcal{R}$ sur $\mathbf{G}$ termine, $\mathcal{S}(\mathcal{R})$ simule $\mathcal{R}$ sur $\mathbf{G}$ à travers π.

Étant donnée une exécution ρ de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$, on va construire une exécution valide ρ' de $\mathcal{R}$ sur $\mathbf{G}$. Par la suite, pour tout sommet $v \in V(G)$ (resp. toute arête $e \in E(G)$), $(\lambda_i(v), \mathbf{status}_i(v))$ (resp. $(\lambda_i(e), \mathbf{status}_i(e), r_i(e))$) est l'étiquette de v après la $i$ème étape de l'exécution ρ de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$ et $\lambda'_j(v)$ (resp. $\lambda'_j(e)$) est l'étiquette de v après la $j$ème étape de l'exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$.

On note $i_1, i_2, \ldots, i_p, \ldots$ les étapes de ρ où la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée. L'exécution ρ' de $\mathcal{R}$ est l'exécution construite de la manière suivante. On considère l'arête e sur laquelle la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée lors de l'étape i_j de ρ. On note v et v' les extrémités de e, on suppose que l'étiquette de v' a été modifiée lors de l'étape i_j et que $r_{i_j}(e) = (\ell_1, \ell_e, \ell_2, \ell'_1, \ell'_e, \ell'_2)$. Le $j$ème pas de calcul de l'exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$ est alors le réétiquetage de l'arête de la manière suivante : $\lambda'_j(v) = \ell'_1$, $\lambda'_j(v') = \ell'_2$ et $\lambda'_j(e) = \ell'_e$.

Si on considère la première étape $i > i_j$ (resp. $i' > i_j$) de ρ où v (resp. v') a le statut **free**, on sait d'après les Lemmes 3.16, 3.17 et 3.18 que $\lambda_i(v) = \ell_1$ et $\lambda_i(e) = \ell'_e$ (resp. $\lambda_{i'}(v') = \ell_2$ et $\lambda_{i'}(e) = \ell'_e$). Ainsi, l'exécution ρ' de $\mathcal{R}$ ainsi construite est une exécution valide. Par ailleurs, on note que pour toute étape $i \in [i_j, i_{j+1} - 1]$ où pour tout $x \in V(G) \cup E(G)$, $\mathbf{status}_{i_j}(x) = \mathbf{free}$, alors pour tout $x \in V(G) \cup E(G)$, $\lambda_i(x) = \lambda'_j(x)$,

De plus, si l'étiquette $\lambda(v)$ d'un sommet v est modifiée, une des règles $\mathcal{S}_2(\mathcal{R}), \mathcal{S}_3(\mathcal{R})$ est appliquée sur une arête incidente à v. On sait d'après les Lemmes 3.16, 3.17 et 3.18 que si la règle $\mathcal{S}_2(\mathcal{R})$ est appliquée sur une arête e dont les extrémités sont v et v', les étiquettes $(\lambda(v), \mathbf{status}(v))$ et $(\lambda(v'), \mathbf{status}(v'))$ ne peuvent être modifiées que si la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée. Ainsi, si l'étiquette de v est modifiée par une application de la règle $\mathcal{S}_2(\mathcal{R})$, ou bien la règle de $\mathcal{R}$ simulée sur l'arête e apparaîtra dans ρ', ou alors l'exécution ρ' est infinie.

Par ailleurs, si l'exécution ρ est infinie, alors la règle $\mathcal{S}_1(\mathcal{R})$ est appliquée infiniment souvent, et d'après le Lemme 3.19, on sait que la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée infiniment souvent, et l'exécution ρ' est donc infinie, ce qui est impossible puisque toute exécution de $\mathcal{R}$ termine sur $\mathbf{G}$.

De plus, dans la configuration finale, on sait d'après le Lemme 3.15 que pour tout sommet $v \in V(G)$ (resp. toute arête $e \in E(G)$), $\mathbf{status}(v) = \mathbf{free}$ (resp. $\mathbf{status}(e) = \mathbf{free}$). Ainsi, si aucune règle de $\mathcal{S}(\mathcal{R})$ ne peut être appliquée dans la configuration finale de ρ, alors aucune règle de $\mathcal{R}$ ne peut être appliquée dans la configuration finale de ρ'. Par conséquent, $\mathcal{S}(\mathcal{R})$ simule $\mathcal{R}$ sur $\mathbf{G}$ à travers la projection π. □

On montre dans la proposition suivante que si l'algorithme $\mathcal{R}$ résout un problème avec détection de la terminaison, alors l'algorithme $\mathcal{S}(\mathcal{R})$ permet aussi de résoudre ce problème

avec détection de la terminaison.

Proposition 3.21 *Pour tout algorithme $\mathcal{R}$ utilisant des calculs locaux sur les arêtes étiquetées, si l'algorithme $\mathcal{R}$ résout un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison, alors l'algorithme $\mathcal{S}(\mathcal{R})$ utilise des calculs locaux cellulaires sur les arêtes étiquetées et résout le problème $\mathcal{P}$ sur la famille $\mathcal{F}$ avec détection de la terminaison.*

Preuve : On suppose que l'algorithme $\mathcal{R}$ permet de résoudre un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison. Il existe donc une fonction $\mathbf{res}'$ et un ensemble L'_f tels que les propriétés suivantes sont toujours vérifiées pour tout graphe $\mathbf{G} \in \mathcal{F}$.

- Toute exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$ termine et l'étiquetage final $\lambda'_{\rho'}$ de G est tel que l'étiquetage $\mathbf{res}' \circ \lambda'_{\rho'}$ est une solution de $\mathcal{P}$ sur $\mathbf{G}$.
- Pour toute exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$, il existe un sommet v et une étape j tels que $\lambda'_j(v) \in L'_f$.
- Pour toute exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$, s'il existe un sommet $v \in V(G)$ et une étape j telle que $\lambda'_j(v) \in L'_f$, alors pour tout $j' > j$, $\mathbf{res}' \circ \lambda'_{j'} = \mathbf{res}' \circ \lambda'_j$.

On veut montrer que l'algorithme $\mathcal{S}(\mathcal{R})$ permet aussi de résoudre le problème $\mathcal{P}$ sur $\mathcal{F}$ avec détection de la terminaison. Pour cela, il faut définir une fonction $\mathbf{res}$ et un ensemble L_f. La fonction $\mathbf{res}$ qu'on va considérer est $\mathbf{res}' \circ \pi$, où π est telle que $\pi((\ell, \mathbf{status})) = \ell$ et $\pi((\ell, \mathbf{status}, r)) = \ell$. L'ensemble L_f est l'ensemble $\{(\ell, \mathbf{free}) \mid \ell \in L'_f\}$.

D'après la Proposition 3.20, on sait que pour tout graphe $\mathbf{G} \in \mathcal{F}$, toute exécution de $\mathcal{S}(\mathcal{R})$ termine sur $\mathbf{G}$ puisque toute exécution de $\mathcal{R}$ termine sur $\mathbf{G}$.

On considère un graphe $\mathbf{G} \in \mathcal{F}$. Pour toute exécution ρ de $\mathcal{S}(\mathcal{R})$ sur $\mathbf{G}$, on considère la même exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$ que dans la preuve de la Proposition 3.20. On sait qu'il existe une étape j de ρ' lors de laquelle $\lambda'_j(v) \in L'_f$ et on suppose que $\lambda'_j(v) \neq \lambda'_{j-1}(v)$. On note $e \in E(G)$ l'arête sur laquelle un pas de réétiquetage est effectué lors de l'étape j de l'exécution ρ'. On sait que $v \in ext(e)$ et d'après les Lemmes 3.16, 3.17 et 3.18, il existe une étape $i > i_j$ telle que $\lambda_i(v) = \lambda'_j(v) \in L'_f$ et $\mathbf{status}_i(v) = \mathbf{free}$. Par conséquent, il existe une étape i de l'exécution ρ et un sommet v tels que $(\lambda_i(v), \mathbf{status}_i(v)) \in L_f$.

On considère maintenant un sommet $v \in V(G)$ et une étape i de ρ telle que $\lambda_i(v) \in L'_f$. On suppose que $(\lambda_i(v), \mathbf{status}_i(v)) \neq (\lambda_{i-1}(v), \mathbf{status}_{i-1}(v))$. On considère l'arête $e \in E(G)$ sur laquelle un pas de réétiquetage est effectué lors de l'étape i de l'exécution ρ. On sait que $v \in ext(e)$ et qu'une des deux règles $\mathcal{S}_3(\mathcal{R}), \mathcal{S}_4(\mathcal{R})$ a été appliquée lors de l'étape i. D'après les Lemmes 3.17 et 3.18, il existe une étape $i' \leq i$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ a été appliquée sur e. Il existe donc une étape $j \in \rho'$ telle que $i' = i_j$ et d'après les Lemmes 3.17 et 3.18, on sait que $\lambda_i(v) = \lambda'_j(v)$. Si on note v' l'autre extrémité de e, on sait aussi que $\lambda_i(v') = \lambda'_j(v')$ d'après les Lemmes 3.17 et 3.18. On sait que pour toute étape $j' > j$, et pour tout $x \in V(G) \cup E(G)$, $\mathbf{res}'(\lambda'_{j'}(x)) = \mathbf{res}'(\lambda'_j(x))$.

Pour tout $e \in E(G)$, si $\lambda_{i_j}(e) \neq \lambda'_j(e)$, cela signifie que la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée sur e avant l'étape i_j et que la règle $\mathcal{S}_3(\mathcal{R})$ n'a pas encore été appliquée sur e. Cependant, l'exécution ρ est finie et d'après les Lemmes 3.15 et 3.17, il existe donc une étape $i' > i_j$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée sur e. Par conséquent, il existe une étape $j' > j$ telle que $i' = i_{j'}$. Ainsi $\mathbf{res}(\lambda_{i'}(e)) = \mathbf{res}'(\lambda'_{j'}(e)) = \mathbf{res}'(\lambda'_j(e)) = \mathbf{res}(\lambda_{i_j}(e))$.

Pour tout sommet $v \in V(G)$, si $\lambda_{i_j}(v) \neq \lambda'_j(v)$, cela signifie que la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée sur une arête e incidente à v avant l'étape i_j et que la règle $\mathcal{S}_3(\mathcal{R})$ n'a

pas encore été appliquée sur e. Cependant, comme précédemment, on peut montrer que $\mathbf{res}(\lambda_{i'}(v)) = \mathbf{res}(\lambda_{i_j}(v))$.

Pour toute étape $i' \geq i$ et toute arête $e \in E(G)$ tel que $\lambda_{i'+1}(e) \neq \lambda_{i'}(e)$, cela signifie que la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée à l'étape i' sur e. Pour toute étape $i' \geq i$ et tout sommet $v \in V(G)$ tel que $\lambda_{i'+1}(v) \neq \lambda_{i'}(v)$, cela signifie qu'une des règles $\mathcal{S}_2(\mathcal{R}), \mathcal{S}_3(\mathcal{R})$ a été appliquée à l'étape i' sur une arête e incidente à v. Si la règle $\mathcal{S}_2(\mathcal{R})$ a été appliquée à l'étape i' sur une arête e, on sait qu'il existe une étape $i'' > i'$ lors de laquelle la règle $\mathcal{S}_3(\mathcal{R})$ est appliquée sur l'arête e. Par conséquent, dans les deux cas, il existe une étape $j' > j$ de ρ' lors de laquelle la règle simulée sur l'arête e à l'étape i' de ρ est appliquée sur e. Par conséquent, pour tout $x \in \{e\} \cup ext(e)$, $\mathbf{res}(\lambda_{i'}(x)) = \mathbf{res}'(\lambda'_{j'}(x)) = \mathbf{res}'(\lambda'_j(x)) = \mathbf{res}(\lambda_{i_j}(x))$.

Ainsi, si l'algorithme $\mathcal{R}$ permet de résoudre $\mathcal{P}$ sur la famille $\mathcal{F}$, l'algorithme $\mathcal{S}(\mathcal{R})$ résout aussi $\mathcal{P}$ sur la famille $\mathcal{F}$. □

On a donc montré qu'on pouvait transformer tout algorithme utilisant des calculs locaux sur les arêtes étiquetées en un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées, au sens où l'étiquetage final d'un graphe obtenu par toute exécution du nouvel algorithme sur ce graphe est un étiquetage final du graphe qui peut être obtenu par une exécution de l'algorithme initial. De plus, on a montré que la transformation permettait de conserver la propriété de détection de la terminaison.

3.5 Énumération, Nommage et Élection

On s'intéresse aux problèmes d'élection, d'énumération et du nommage dans le cadre des calculs locaux sur les arêtes étiquetées. On montre d'abord que le résultat d'impossibilité d'Angluin [Ang80] reste valide dans ces modèles : il est impossible de trouver un algorithme d'élection, d'énumération ou de nommage pour un graphe $\mathbf{G}$ qui n'est pas minimal pour les revêtements. On donne ensuite un algorithme d'énumération pour les graphes minimaux pour les revêtements qui est inspiré de l'algorithme de Mazurkiewicz [Maz97].

3.5.1 Résultats d'Impossibilité pour l'Énumération et le Nommage

Proposition 3.22 ([Ang80]) *Soit* $\mathbf{G}$ *un graphe simple étiqueté qui n'est pas minimal pour les revêtements. Il n'existe pas d'algorithme d'élection, d'énumération ou de nommage pour le graphe* $\mathbf{G}$ *utilisant des calculs locaux sur les arêtes étiquetées.*

Preuve : Soit $\mathbf{H}$ un graphe étiqueté non-isomorphe à $\mathbf{G}$ et tel qu'il existe un homomorphisme localement bijectif γ de $\mathbf{G}$ dans $\mathbf{H}$. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux cellulaires sur les arêtes étiquetées, on considère une exécution de $\mathcal{R}$ sur $\mathbf{H}$. Si cette exécution est infinie sur $\mathbf{H}$, alors d'après le Lemme 3.11, il existe une exécution infinie de $\mathcal{R}$ sur $\mathbf{G}$; auquel cas, $\mathcal{R}$ n'est ni un algorithme d'énumération, ni de nommage, ni d'élection.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur $\mathbf{H}$ et on considère la configuration finale $\mathbf{H}'$. D'après le Lemme 3.11, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui permet d'atteindre une configuration $\mathbf{G}'$ telle que $\mathbf{G}'$ est un revêtement de $\mathbf{H}'$ à travers γ. Si $\mathbf{G}'$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe une arête $e \in E(G)$ telle qu'on puisse appliquer une règle de réétiquetage sur l'arête e. Dans ce cas là, on peut appliquer

la même règle dans $\mathbf{H}'$ sur l'arête $\gamma(e)$, ce qui est impossible. Par conséquent, $\mathbf{G}'$ est une configuration finale de $\mathcal{R}$. Mais puisque $\mathbf{G}'$ n'est pas isomorphe à $\mathbf{H}'$, cela implique d'après la Proposition 3.5 que chaque étiquette apparaissant sur un sommet de $\mathbf{H}'$ apparaît au moins deux fois dans $\mathbf{G}'$. Par conséquent, $\mathcal{R}$ ne permet ni d'attribuer de noms distincts à tous les sommets de G, ni de distinguer un sommet. Ainsi $\mathcal{R}$ ne résout ni le nommage, ni l'énumération, ni l'élection sur $\mathbf{G}$. □

3.5.2 Un Algorithme d'Énumération

On va maintenant décrire un algorithme d'énumération $\mathcal{M}$ pour les graphes minimaux pour les revêtements. Cet algorithme s'inspire de l'algorithme de Mazurkiewicz.

Durant l'exécution de l'algorithme, chaque sommet v essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. Chaque sommet va ensuite échanger son numéro avec ses voisins. Un numéro p va être associé à chaque arête de telle sorte que deux arêtes incidentes à un même sommet aient des numéros différents. Grâce à ces échanges de numéros, chaque sommet peut construire sa *vue locale*, telle que pour chaque voisin v' de v, la vue locale de v contient le triplet $(p, \lambda(e), n)$ où p est le numéro donné à l'arête e reliant v à v', $\lambda(e)$ est l'étiquette initiale de e et n est le numéro qu'avait v' lors de la dernière synchronisation entre v et v'. Ensuite, chaque sommet va diffuser dans tout le graphe son numéro et sa vue locale. Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette initiale $\lambda(u)$ et sa vue locale avec l'étiquette initiale $\lambda(v)$ et la vue locale de v : si l'étiquette de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre proche de l'ordre utilisé dans l'algorithme de Mazurkiewicz), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire) et le diffuse à nouveau avec sa vue locale. Lorsque l'exécution est terminée, si le graphe $\mathbf{G}$ est minimal pour les revêtements, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda\colon V(G) \cup E(G) \to L$ est un étiquetage initial, qui ne sera pas modifié par l'algorithme. Lors de l'exécution, chaque arête e va obtenir une étiquette de la forme $(\lambda(e), p(e))$ qui représente les informations suivantes :

- la première composante $\lambda(e)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $p(e) \in \mathbb{N}$ est un entier associé à chaque arête de telle sorte que deux arêtes e et e' incidentes à un même sommet vont avoir des numéros non-nuls distincts. Cette valeur sera fixée par une règle de réétiquetage sur e et ne sera plus modifiée par la suite.

Initialement, chaque arête e est étiqueté $(\lambda(e), 0)$ puisqu'aucun numéro n'a encore été attribué à e.

Lors de l'exécution, chaque sommet va obtenir une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.

- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N})$ est la *vue locale* du sommet v. Informellement, la vue locale contient l'information la plus récente que v a de ses voisins. Si le sommet v est incident à une arête e dont l'autre extrémité est un sommet v', un pas de réétiquetage sur e va permettre d'ajouter le triplet $(p(e), \lambda(e), n(v'))$ à la vue locale de v. Ainsi $N(v)$ est toujours un ensemble fini de triplets de $\mathbb{N} \times L \times \mathbb{N}$.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N})$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

La différence principale entre les étiquettes utilisées ici et les étiquettes utilisés dans l'algorithme de Mazurkiewicz est la présence de numéros sur les arêtes. Dans l'algorithme de Mazurkiewicz, les voisins d'un sommet qui ont des numéros non-nuls ont toujours des numéros différents, ce qui permet à tout sommet de distinguer ses voisins. Les numéros associés aux arêtes vont permettre ici à chaque sommet de distinguer ses voisins.

Un Ordre sur les Vues Locales

Comme pour l'algorithme de Mazurkiewicz [Maz97], les bonnes propriétés de l'algorithme reposent sur un ordre sur les vues locales, i.e., sur les ensembles finis de triplets de $\mathbb{N} \times L \times \mathbb{N}$. Pour cela, on considère que $\mathbb{N} \times L \times \mathbb{N}$ est muni de l'ordre lexicographique usuel : $(p, \ell, n) < (p', \ell', n')$ si $p < p'$, ou si $p = p'$ et $\ell <_L \ell'$, ou si $p = p'$, $\ell = \ell'$ et $n < n'$.

Ensuite, on utilise le même ordre sur les ensembles que Mazurkiewicz : étant donnés deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N})$ distincts, on dit que $N_1 \prec N_2$ si le maximum pour l'ordre lexicographique sur $\mathbb{N} \times L \times \mathbb{N}$ de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2.

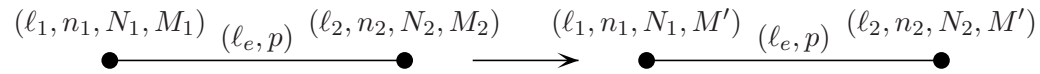
Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre $\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

Les Règles de Réétiquetage

On décrit maintenant l'algorithme d'énumération grâce à des règles de réétiquetage. Il y a deux règles spéciales $\mathcal{M}_0$ et $\mathcal{M}'_0$ qui permettent d'initialiser l'étiquetage. La règle $\mathcal{M}_0$ permet à chaque sommet de modifier son étiquette initiale $\lambda(v)$ pour obtenir l'étiquette $(\lambda(v), 0, \emptyset, \emptyset)$. La règle $\mathcal{M}'_0$ permet de modifier l'étiquette $\lambda(e)$ d'une arête e qui devient l'étiquette $(\lambda(e), 0)$. Cette règle est applicable quelles que soient les étiquettes des extrémités de e.

Les deux premières règles $\mathcal{M}_1$ et $\mathcal{M}_2$ sont proches des deux règles de l'algorithme de Mazurkiewicz. La première règle permet à deux sommets voisins v et v' de partager les informations contenues dans leurs boîtes-aux-lettres à propos des étiquettes apparaissant dans le graphe.

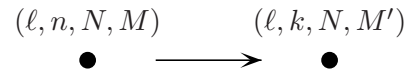
$\mathcal{M}_1$:



Si $M_1 \neq M_2$
alors $M' := M'_1 \cup M'_2$.

Les règles $\mathcal{M}_2$ et $\mathcal{M}_3$ permettent à un sommet v de modifier son numéro s'il existe un autre sommet dans le graphe qui a le même numéro. La deuxième règle ne dépend que de l'étiquette d'un seul sommet v. Elle permet à v de modifier son numéro si v n'a pas encore choisi de numéro (son numéro courant est 0), ou si la boîte-aux-lettres de v contient un message indiquant qu'il existe un autre sommet dans le graphe ayant le même numéro que v et qui a une étiquette supérieure à celle de v ou qui a une vue locale plus forte que celle de v.

$\mathcal{M}_2$:



Si $n = 0$ ou $\exists (n, \ell', N') \in M$ tel que $(\ell, N) \prec (\ell', N')$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$;
$M' := M \cup \{(k, \ell, N)\}$.

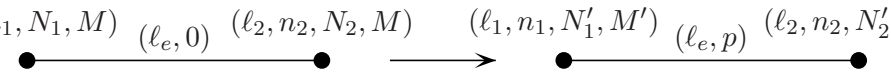
La troisième règle permet à un sommet v de modifier son numéro s'il a un voisin v' qui est exactement dans le même état, i.e., $(\lambda(v), n(v), N(v), M(v)) = (\lambda(v'), n(v'), N(v'), M(v'))$. Cette règle ne peut être appliquée que si on ne peut pas appliquer la règle $\mathcal{M}_2$ à v ou à v'.

$\mathcal{M}_3$:

Si $n > 0$ et
$\forall (n, \ell', N') \in M$, $(\ell', N') \preceq (\ell, N)$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$;
$M' := M \cup \{(k, \ell, N)\}$.

La quatrième règle peut être appliquée sur une arête e dont les extrémités sont v et v' si e n'a pas encore de numéro, i.e., $p(e) = 0$. Un numéro p est alors généré de telle sorte qu'il n'existe aucune arête e' incidente à v ou v' dont le numéro est p. Les vues locales et les boîtes-aux-lettres de v et v' sont alors mises à jour. Cette règle ne peut être appliquée sur l'arête e seulement si les règles $\mathcal{M}_1, \mathcal{M}_2$ et $\mathcal{M}_3$ ne peuvent pas être appliquée sur e.

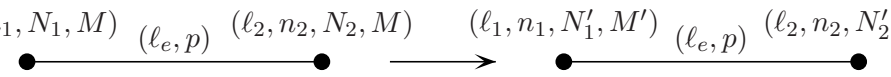
$\mathcal{M}_4$:



Si $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell_1', N_1') \in M, (\ell_1', N_1') \preceq (\ell_1, N_1)$ et
$\forall (n_2, \ell_2', N_2') \in M, (\ell_2', N_2') \preceq (\ell_2, N_2)$
alors $p := 1 + \max\{p' \mid \exists (p', \ell', n') \in N_1 \cup N_2\}$;
$N_1' := N_1 \cup \{(p, \ell_e, n_2)\}$;
$N_2' := N_2 \cup \{(p, \ell_e, n_1)\}$;
$M' = M \cup \{(n_1, N_1'), (n_2, N_2')\}$.

La cinquième règle peut être appliquée sur une arête e dont les extrémités sont v et v' si une mise à jour de la vue locale de v ou de v' est nécessaire, i.e., $(p(e), \lambda(e), n(v')) \notin N(v)$ ou $(p(e), \lambda(e), n(v)) \notin N(v')$. Cette règle ne peut pas être appliquée si une des quatre règles précédentes peut être appliquées sur e. Dans ce cas là les vues locales et les boîtes-aux-lettres de v et v' sont mises à jour.

$\mathcal{M}_5$:



Si $p > 0$, $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell_1', N_1') \in M, (\ell_1', N_1') \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell_2', N_2') \in M, (\ell_2', N_2') \preceq (\ell_2, N_2)$ et
$(p, \ell_e, n_2) \notin N_1$
alors $N_1' := N_1 \setminus \{(p', \ell_e', n_2') \mid p' = p\}) \cup \{(p, \ell_e, n_2\}$;
$N_2' := N_2 \setminus \{(p', \ell_e', n_1') \mid p' = p\}) \cup \{(p, \ell_e, n_1\}$;
$M' := M \cup \{(n_1, \ell_1, N_1')\} \cup \{(n_2, \ell_2, N_2')\}$.

3.5.3 Correction de l'Algorithme d'Énumération

On considère un graphe simple étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$ et toute arête $e \in E(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ l'étiquette du sommet v et $(\lambda(e), p_i(e))$ l'étiquette de l'arête e après la $i$ème étape de réétiquetage de l'algorithme $\mathcal{M}$ décrit ci-dessus. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Propriétés Satisfaites lors de l'Exécution

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur le nombre d'étapes, rappelle quelque propriétés simples qui sont toujours satisfaites par l'étiquetage.

Lemme 3.23 *Pour toutes arêtes $e, e' \in E(G)$, pour tout sommet $v \in V(G')$, et pour toute étape i,*

1. $p_i(e) \neq 0 \implies p_{i+1}(e) = p_i(e)$,
2. $\exists (p, \ell_e, n) \in N_i(v) \iff \exists e \in I_G(v)$ *telle que* $p_i(e) = p > 0$ *et* $\lambda(e) = \ell_e$,

3. $p_i(e) \neq 0, p_i(e') \neq 0, v \in ext(e) \cap ext(e') \implies p_i(e) \neq p_i(e')$,
4. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
5. $\forall (p, \ell_e, n) \in N_i(v), n \neq 0, p \neq 0$ *et* $\exists (n, \ell', N') \in M_i(v)$,
6. $\nexists (p, \ell_e, n_i(v)) \in N_i(v)$,
7. $\forall (p, \ell_e, n), (p', \ell'_e, n') \in N_i(v), p \neq p'$.

L'algorithme $\mathcal{M}$ a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 3.24 *Pour chaque sommet v et chaque étape i,*
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

De plus, à chaque étape i, il existe un sommet v telle qu'au moins une de ces inégalités (ou inclusions) est stricte pour v.

Preuve : La propriété est trivialement vraie pour les sommets qui ne sont pas réétiquetés lors de la $(i+1)$ème étape. De plus, il est facile de voir que quelque soit la règle appliquée à l'étape $i+1$, on a toujours $M_i(v) \subseteq M_{i+1}(v)$ pour tout sommet $v \in V(G)$.

Pour chaque sommet v tel que $n_i(v) \neq n_{i+1}(v)$, alors la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ a été appliquée à v et $n_{i+1}(v) = 1 + \max\{n' \mid \exists (n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 3.23, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Pour chaque sommet v tel que $N_i(v) \neq N_{i+1}(v)$, la règle $\mathcal{M}_4$ ou $\mathcal{M}_5$ a été appliquée sur une arête e incidente à v et a un sommet v'. Si la règle $\mathcal{M}_4$ a été appliquée, $N_{i+1}(v) = N_i(v) \cup \{(p_i(e), \lambda(e), n_i(v'))\}$ et $(p_i(e), \lambda(e), n_i(v')) \notin N_i(v)$; par conséquent $N_i(v) \prec N_{i+1}(v)$. Si la règle $\mathcal{M}_5$ est appliquée, alors d'après le Lemme 3.23, il existe $(p_i(e), \lambda(e), n) \in N_i(v)$. De plus, on sait que cela signifie que la règle $\mathcal{M}_4$ ou $\mathcal{M}_5$ a été appliquée à une étape $i' < i$ sur l'arête e et $n = n_{i'}(v)$. Ainsi, puisque $n_{i'}(v) \leq n_i(v)$, ou bien $N_{i+1}(v) = N_i(v)$ ou $\max N_{i+1}(v) \triangle N_i(v) = (p_i(e), \lambda(e), n_i(v)) \in N_{i+1}(v)$ et donc $N_i(v) \preceq N_{i+1}(v)$.

Puisque chaque application d'une règle modifie l'étiquette d'au moins un sommet v, on sait que l'une de ces inégalités est stricte pour v. □

Les informations dont dispose chaque sommet v dans sa boîte-aux-lettres permettent d'obtenir des informations vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet v' tel que $n_i(v') = m'$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet v' tel que $n_i(v') = m'$.

Lemme 3.25 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid$

$\forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, il existe exactement un élément $(u, i_0) \in U'$ puisqu'à chaque étape, le numéro d'au plus un sommet peut être modifié. Le numéro $n_{i_0}(u) = m$ a donc été modifié à l'étape $i_0 + 1$, mais puisque à cette étape, le sommet u n'avait aucun voisin avec le même numéro m, la règle $\mathcal{M}_3$ n'a pas pu être appliquée, et par maximalité de $(\lambda(u), N_{i_0}(u))$, la règle $\mathcal{M}_2$ n'a pas non plus pu être appliquée à u à l'étape i_0. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 3.26 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. La propriété est trivialement vraie à l'étape $i + 1$ pour tout sommet $w \in V(G)$ dont l'étiquette n'est pas modifiée à l'étape $i + 1$. Soit v un sommet dont l'étiquette est modifiée à l'étape $i + 1$.

Si la règle $\mathcal{M}_1$ est appliquée à l'étape $i+1$ à v et à un de ses voisins v', alors $M_{i+1}(v) = M_i(v) \cup M_i(v')$ et la propriété est vérifiée à l'étape $i + 1$ puisqu'elle était vraie pour v et v' à l'étape i.

Si la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ est appliquée à v à l'étape $i + 1$, alors $M_{i+1}(v) = M_i(v) \cup \{1 + \max\{m \mid (m, \ell, N) \in M_i(v)\}, \lambda(v), N_i(v))\}$, et par conséquent pour chaque $m \in M_{i+1}(v)$, la propriété reste vraie.

Si la règle $\mathcal{M}_4$ ou $\mathcal{M}_5$ est appliquée à v à l'étape $i + 1$, pour tout $(m, \ell, N) \in M_{i+1}(v)$, il existe $(m, \ell, N') \in M_i(v)$ et la propriété reste donc vraie. □

On veut maintenant montrer que toute exécution de l'algorithme $\mathcal{M}$ termine sur $\mathbf{G}$. D'après les Lemmes 3.25 et 3.26, on voit qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1, k]$ ou un ensemble $[0, k]$ avec $k \leq |V(G)|$. Par conséquent, d'après le Lemme 3.24, on sait qu'il existe une étape i_0 telle que pour tout sommet v et toute étape $i \geq i_0$, $n_{i+1}(v) = n_i(v)$. De plus, on sait que la règle $\mathcal{M}_4$ ne peut être appliquée qu'une seule fois sur chaque arête et par conséquent, pour tout sommet v, $N_i(v)$ et $M_i(v)$ ne peuvent prendre qu'un nombre fini de valeurs. Ainsi, d'après le Lemme 3.24, on sait que toute exécution de $\mathcal{M}$ sur $\mathbf{G}$ termine.

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final.

Lemme 3.27 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ des sommets et (λ, p_ρ) des arêtes vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. $M_\rho(v) = M_\rho(v')$,
3. $(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,
4. *si* $n_\rho(v) = n_\rho(v')$, *alors* $\lambda(v) = \lambda(v')$ *et* $N_\rho(v) = N_\rho(v')$,
5. *pour toutes arêtes* $e, e' \in I_G(v)$, $p_\rho(e) > 0$, $p_\rho(e') > 0$ *et* $p_\rho(e) \neq p_\rho(e')$.
6. $(p, \ell, n) \in N_\rho(v)$ *si et seulement s'il existe une arête* e *incidente à* v *telle que* $p_\rho(e) = p$ *et* $\lambda(e) = \ell$. *De plus, si* $ext(e) = \{v, w\}$, *alors* $n_\rho(w) = n$ *et* $(p, \ell, n_\rho(v)) \in N_\rho(w)$.

Preuve :

1. D'après les Lemmes 3.25 et 3.26 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
2. Dans le cas contraire, la règle $\mathcal{M}_1$ peut être appliquée.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 3.23.
4. Dans le cas contraire, la règle $\mathcal{M}_2$ peut être appliquée à v ou à v'.
5. D'après le Lemme 3.23 et puisque la règle $\mathcal{M}_4$ ne peut plus être appliquée.
6. D'après le Lemme 3.23 et puisque les règles $\mathcal{M}_3$, $\mathcal{M}_4$ et $\mathcal{M}_5$ ne peuvent plus être appliquées.

□

Grâce au Lemme 3.27, on peut prouver que l'étiquetage final permet de construire un graphe **H** tel que **G** est un revêtements de **H**.

Proposition 3.28 *Étant donné un graphe* **G**, *on peut construire, à partir de l'étiquetage final obtenu après une exécution* ρ *de* $\mathcal{M}$, *un graphe* **H** *tel qu'il existe un homomorphisme localement bijectif de* **G** *dans* **H**.

Preuve : On utilise les notations du Lemme 3.27.

On considère le graphe H défini par $V(H) = \{m \in \mathbb{N} \mid \exists v \in V(G), n_\rho(v) = m\}$, $E(H) = \{f_{p_\rho(e),\lambda(e),\{n_\rho(v),n_\rho(v')\}} \mid \exists e \in E(G)$ telle que $ext(e) = \{v, v'\}\}$ et pour toute arête $f_{p,\ell,\{m,m'\}} \in E(H)$, $ext(f_{p,\ell,\{m,m'\}}) = \{m, m'\}$. On définit un étiquetage η de H en posant $\eta(n_\rho(v)) = \lambda(v)$ et $\eta(f_{p,\ell,\{m,m'\}}) = \ell$. D'après le Lemme 3.27, si deux sommets $v, v' \in V(G)$ ont le même numéro, alors $\lambda(v) = \lambda(v')$ et par conséquent, cet étiquetage est bien défini.

D'après le Lemme 3.27, on sait qu'il n'existe pas d'arête $e \in E(G)$ telle que $ext(e) = \{v, v'\}$ et $n_\rho(v) = n_\rho(v')$. Par conséquent, le graphe **H** est un graphe étiqueté qui ne contient pas de boucles.

On définit maintenant un homomorphisme γ de **G** dans **H** de la manière suivante : pour tout sommet $v \in V(G)$, $\gamma(v) = n_\rho(v)$ et pour toute arête $e \in E(G)$ dont les extrémités sont v et v' , $\gamma(e) = f_{p_\rho(e),\lambda(e),\{n_\rho(v),n_\rho(v')\}}$. De par la définition de **H**, il est clair que γ est un homomorphisme de **G** dans **H**.

Pour tout sommet $v \in V(G)$, pour toutes arêtes $e, e' \in I_G(v)$, on sait d'après le Lemme 3.27 que $p_\rho(e) \neq p_\rho(e')$ et ainsi, $\gamma(e) \neq \gamma(e')$. Par conséquent, pour tout sommet $v \in V(G)$, $|\gamma(I_G(v))| = |I_G(v)|$. De plus, pour tout sommet $n \in V(H)$, si $f_{p,\ell,\{n,n'\}} \in I_H(n)$, on sait d'après le Lemme 3.27 qu'il existe un sommet $v \in \gamma^{-1}(n)$ tel que $(p, \ell, n') \in \mathbb{N}_\rho(v)$. Puisque, pour tout $v' \in \gamma^{-1}(n)$, $N_\rho(v) = N_\rho(v')$, il existe une arête $e' \in I_G(v')$ telle que $\gamma(e') = f_{p,\ell,\{n,n'\}}$. Ainsi, pour tout $v \in V(G)$, $\gamma(I_G(v)) = I_H(\gamma(v))$ et par conséquent, γ est un homomorphisme localement bijectif de **G** dans **H**.

Le graphe **G** est donc un revêtement du graphe **H** à travers γ. □

On considère maintenant un graphe $\mathbf{G}$ qui est minimal pour les revêtements. Pour chaque exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, le graphe obtenu à partir de l'étiquetage final est isomorphe à $\mathbf{G}$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique. L'algorithme $\mathcal{M}$ permet de résoudre le nommage sur la famille des graphes minimaux pour les revêtements, mais si aucune information à propos de $\mathbf{G}$ n'est disponible, les sommets ne peuvent pas détecter la terminaison.

Cependant, il est possible de détecter la terminaison pour un graphe $\mathbf{G}$ donné (l'algorithme dépend alors de $\mathbf{G}$). En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 3.25 et 3.26, il sait que tous les sommets de $\mathbf{G}$ ont un numéro unique qui ne va plus être modifié. Dans ce cas là, on peut aussi résoudre le problème de l'élection, puisque ce sommet peut prendre l'étiquette élu et diffuser ensuite l'information qu'un sommet a été élu.

Par ailleurs, d'après la Proposition 3.20, on sait qu'il existe un algorithme $\mathcal{M}'$ utilisant des calculs locaux cellulaires sur les arêtes étiquetées qui simule $\mathcal{M}$. Il existe donc un algorithme de nommage (sans détection de la terminaison) utilisant des calculs locaux cellulaires pour la famille des graphes minimaux pour les revêtements. De plus, d'après la Proposition 3.21, on sait aussi que pour tout graphe $\mathbf{G}$ minimal pour les revêtements, il existe un algorithme d'élection pour $\mathbf{G}$ et un algorithme d'énumération avec détection de la terminaison pour $\mathbf{G}$ qui utilisent des calculs locaux cellulaires sur les arêtes étiquetées.

Par ailleurs, d'après la Proposition 3.22, on sait que pour tout graphe $\mathbf{G}$ qui n'est pas minimal pour les revêtements, il n'existe aucun algorithme utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées qui permette de résoudre les problèmes du nommage ou de l'élection sur $\mathbf{G}$. On a donc prouvé le théorème suivant.

Théorème 3.29 *Pour tout graphe étiqueté $\mathbf{G}$, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour $\mathbf{G}$ utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées,*
2. *il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $\mathbf{G}$ utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées,*
3. *$\mathbf{G}$ est minimal pour les revêtements.*

Remarque 3.30 *Étant donné un graphe $\mathbf{G}$ minimal pour les revêtements, pour détecter que l'algorithme $\mathcal{M}$ a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de $\mathbf{G}$. Ainsi, l'algorithme $\mathcal{M}$ permet de résoudre l'élection ainsi que le nommage avec détection de la terminaison sur les graphes minimaux pour les revêtements de taille donnée.*

3.5.4 Complexité

On s'intéresse à la complexité de l'algorithme $\mathcal{M}$ présenté ci-dessus. Dans le cadre des calculs locaux, on s'intéresse au nombre de pas de réétiquetages effectuées lors d'une exécution. La proposition suivante donne une borne supérieure sur le nombre de pas de réétiquetages de toute exécution de l'algorithme $\mathcal{M}$ sur un graphe à n sommets de degré maximal Δ.

Proposition 3.31 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, durant toute exécution de l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes étiquetées,* $O(\Delta n^3)$ *règles sont appliquées.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et une exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$. D'après les Lemmes 3.25 et 3.26, on sait que les règles $\mathcal{M}_2$ et $\mathcal{M}_3$ ne peuvent pas être appliquées plus de $\frac{n(n-1)}{2}$ fois durant l'exécution ρ.

Entre deux étapes où une des règles $\mathcal{M}_2, \mathcal{M}_3$ est appliquée, les règles $\mathcal{M}_4$ et $\mathcal{M}_5$ sont appliquées au plus Δ fois (une fois pour chaque voisin du sommet dont le numéro a été modifié). Les règles $\mathcal{M}_4$ et $\mathcal{M}_5$ sont donc appliquées $O(\Delta n^2)$ fois.

À chaque fois qu'un sommet v modifie son numéro ou sa vue locale, un couple (n_0, ℓ, N) est ajouté à $M(v)$. Pour chacun de ces couples, la règle $\mathcal{M}_1$ est appliquée au plus n fois.

Ainsi, durant toute exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, $O(\Delta n^3)$ règles sont appliquées. □

Le facteur Δ qui diffère entre la borne de la Proposition 2.23 et celle de la Proposition 3.31 est du au fait que dans le modèle de calculs locaux sur les étoiles fermées, si un sommet modifie son numéro, il en informe immédiatement ses voisins, ce qui n'est pas possible dans le cadre des calculs locaux sur les arêtes étiquetées.

On s'intéresse à la mémoire nécessaire à chaque sommet pour stocker son étiquette. On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets et à toutes les arêtes de G).

Proposition 3.32 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes étiquetées nécessite* $O(\Delta n \log n)$ *bits de mémoire par sommet.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et m arêtes dont le degré maximal est Δ. L'étiquette de chaque arête est modifiée exactement une fois par toute exécution de l'algorithme, et chaque arête e a un numéro $p(e)$ compris entre 1 et m. 0n sait que la vue locale de chaque sommet contient au plus Δ couples (p, ℓ_e, n_0) qui peuvent être représentés avec $O(\log n)$ bits. Ainsi, pour chaque sommet v, $N(v)$ peut être représenté avec $O(\Delta \log n)$ bits.

Chaque sommet peut ne conserver dans sa boîte-aux-lettres que l'information utile, i.e., l'ensemble $\{(n_0, \ell, N) \in M(v) \mid \forall (n_0, \ell', N') \in M(v), (\ell', N') \preceq (\ell, N)\}$. Ainsi, dans la boîte-aux-lettres de chaque sommet, il existe au plus n triplets (n_0, ℓ, N) qu'on peut représenter avec $O(\Delta \log n)$ bits. Par conséquent, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta n \log n)$ bits. □

3.6 Importance de la Connaissance du Degré

Dans cette partie, on suppose qu'initialement, chaque sommet v d'un graphe $\mathbf{G} = (G, \lambda)$ connaît son degré, i.e., son degré fait partie de son étiquette initiale $\lambda(v)$. On montre d'abord le lien entre le modèle des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré, le modèle d'Angluin et le modèle de communication synchrone. On définit ensuite formellement les calculs locaux sur les étoiles ouvertes et on montre que tout algorithme utilisant des calculs locaux sur les étoiles ouvertes peut être simulé sur par un

algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré. On montre enfin que le modèle de calcul ainsi obtenu est strictement plus puissant que les calculs locaux sans connaissance initiale du degré. Pour cela, on va considérer le problème de l'élection sur les familles de graphes minimaux pour les revêtements dont le diamètre est bornée.

3.6.1 Relations avec le Modèle D'Angluin et le Modèle de Communication Synchrone

Le Modèle d'Angluin

Dans [Ang80], Angluin considère des graphes simples disposant d'un étiquetage des ports. Un pas de calcul permet à deux sommets voisins d'échanger leurs états, puis chaque sommet modifie son état en fonction de son propre état et de l'état de ses voisins. Afin de briser certaines symétries, Angluin suppose que lors d'un tel pas de calcul, deux sommets voisins qui sont dans le même état peuvent obtenir des états différents (les sommets peuvent par exemple briser cette symétrie par un tirage au sort). Angluin cherche a caractériser les graphes $\mathbf{G}$ qui admettent un algorithme d'élection quel que soit l'étiquetage des ports. Elle montre en particulier que si un graphe n'est pas minimal pour les revêtements simples, alors il n'existe pas d'algorithme d'élection pour ce graphe.

Puisque dans le modèle d'Angluin, chaque sommet v connaît les numéros des ports incident à v, chaque sommet connaît son degré. Par ailleurs, les sommets peuvent distinguer les arêtes qui leurs sont incidentes grâce aux numéros de ports. Ainsi, il est facile de simuler dans ce modèle un algorithme utilisant des calculs locaux sur les arêtes étiquetés avec connaissance initiale du degré : on stocke l'étiquette d'une arête e dans les étiquettes des extrémités de e. Ainsi, le modèle d'Angluin permet de simuler tout algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré.

Réciproquement, on va montrer que le résultat d'impossibilité présenté dans la Proposition 3.22 reste vrai dans le modèle d'Angluin. Étant donné un graphe simple $\mathbf{G}$ avec un étiquetage des ports δ, on va montrer que si $\mathbf{G}$ est un revêtement d'un graphe $\mathbf{H}$, alors il existe un étiquetage des ports δ' de $\mathbf{H}$ de telle sorte que pour tout algorithme $\mathcal{A}$, il existe une exécution de $\mathcal{A}$ sur $(\mathbf{G}, \delta)$ qui est relevée d'une exécution de $\mathcal{A}$ sur $(\mathbf{H}, \delta)$.

Puisqu'on doit étendre la notion d'étiquetage des ports à des graphes qui peuvent avoir des arêtes multiples, on utilise la définition suivante.

Définition 3.33 *Étant donné un graphe $\mathbf{G}$, un étiquetage des ports δ est un ensemble de fonctions $\{\delta_u \mid u \in V(G)\}$ tel que pour tout sommet u, δ_u est une bijection entre $I_G(u)$ et $[1, \deg_G(u)]$.*

Ainsi, un étiquetage des ports permet à chaque sommet d'attribuer un numéro à chacune des arêtes qui lui sont incidentes et non à chacun de ses voisins. On remarque que dans le cas des graphes simples, les deux définitions sont équivalentes.

On dit qu'un graphe $\mathbf{G}$ avec un étiquetage des ports δ est un revêtement d'un graphe $\mathbf{H}$ avec un étiquetage des ports δ' à travers γ si $\mathbf{G}$ est un revêtement de $\mathbf{H}$ à travers γ et si pour toute arête $e \in E(G)$ et pour tout sommet $u \in ext(e)$, $\delta_u(e) = \delta_{\gamma(u)}(\gamma(e))$.

Proposition 3.34 *Pour tous graphes $\mathbf{G}$, $\mathbf{H}$ tels que $\mathbf{G}$ est un revêtement de $\mathbf{H}$, pour tout étiquetage des ports δ', il existe un étiquetage des ports δ tel que $(\mathbf{G}, \delta)$ est un revêtement de $(\mathbf{H}, \delta')$*

Preuve : On considère un graphe $\mathbf{G}$ qui est un revêtement d'un graphe $\mathbf{H}$ à travers γ et un étiquetage des ports δ' de $\mathbf{H}$. Pour tout sommet $u \in V(G)$, pour toute arête $e \in I_G(u)$, on sait que $\gamma(e) \in I_H(\gamma(e))$ et on définit alors $\delta_u(e) = \delta'_{\gamma(u)}(\gamma(e))$. Il suffit de montrer que la fonction δ ainsi obtenu est bien un étiquetage des ports de $\mathbf{G}$. On considère un sommet $u \in V(G)$. Puisque $\gamma_{|I_G(u)}$ induit une bijection entre $I_G(u)$ et $I_H(\gamma(u))$ et que $\delta'_{\gamma(u)}$ induit une bijection entre $I_H(\gamma(u))$ et $[1, \deg_H(\gamma(u))]$, $\delta_u = \delta'_{\gamma(u)} \circ \gamma_{|I_G(u)}$ induit une bijection entre $I_G(u)$ et $[1, \deg_H(\gamma(u)) = \deg_G(u)]$. Ainsi, δ est bien un étiquetage des ports de $\mathbf{G}$. □

Avec la même preuve que pour la Proposition 3.22, il est facile de voir que si un graphe $\mathbf{G}$ avec un étiquetage des ports δ est un revêtement d'un graphe $\mathbf{H}$ avec un étiquetage des ports δ', alors dans le modèle d'Angluin, pour tout algorithme $\mathcal{A}$, il existe une exécution de $\mathcal{A}$ sur $(\mathbf{G}, \delta)$ qui est relevée d'une exécution sur $(\mathbf{H}, \delta')$. Par conséquent, pour tout graphe $\mathbf{G}$ qui n'est pas minimal pour les revêtements, il existe un étiquetage des ports de $\mathbf{G}$ tel qu'on ne peut pas résoudre le nommage ou l'élection sur $(\mathbf{G}, \delta)$ dans le modèle d'Angluin. Réciproquement, puisqu'on peut simuler l'Algorithme $\mathcal{M}$ dans le modèle d'Angluin, on obtient le théorème suivant. On rappelle qu'un algorithme d'élection pour un graphe $\mathbf{G}$ permet de résoudre l'élection sur $\mathbf{G}$ quel que soit l'étiquetage des ports.

Théorème 3.35 *Pour tout graphe simple étiqueté* $\mathbf{G}$*, il existe un algorithme de nommage et un algorithme d'élection pour* $\mathbf{G}$ *dans le modèle d'Angluin si et seulement si* $\mathbf{G}$ *est minimal pour les revêtements.*

Le Modèle de Communication Synchrone

Dans un système où les processus communiquent par échange de messages, on peut considérer le modèle où une étape de transmission de message nécessite une synchronisation entre l'expéditeur et le destinataire du message. Autrement dit, en un pas de calcul, deux sommets voisins v et v' se synchronisent, le sommet v envoie un message à v' (qui peut être son état), v' modifie son état en fonction de son propre état et du message reçu, et v modifie son état indépendemment de l'état de v'. Dans ce modèle, on suppose que le réseau dispose d'un étiquetage des ports pour permettre à chaque sommet de distinguer ses voisins.

Il faut noter qu'on parle de communication synchrone, puisque toute transmission de message nécessite une synchronisation, mais que le comportement global du système distribué est asynchrone. En général, pour éviter les inter-blocages, on suppose que les processus peuvent faire des choix non-déterministes sur les actions qu'ils effectuent (par exemple, un processus peut choisir de manière non-déterministe s'il doit d'abord envoyer un message ou recevoir un message). Pour des définitions formelles, on peut se référer au livre de Tel [Tel00] (pp. 47-49). Charron-Bost et al. [CBMT96] ont étudié les différences entre les modes de communication synchrone et asynchrone (le modèle asynchrone est étudié au Chapitre 7 de ce mémoire).

Il est clair que le modèle d'Angluin est plus puissant que le modèle de communication synchrone puisque dans le modèle d'Angluin, les deux sommets ont accès à l'état de leur voisin. Par ailleurs, comme indiqué pour le modèle d'Angluin, on peut stocker l'étiquette de chaque arête dans les étiquettes de ses extrémités. Ainsi, il est facile de voir qu'on peut simuler tout algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées

dans le mode de communication synchrone. D'après les Théorèmes 3.29 et 3.35, on a donc le théorème suivant.

Théorème 3.36 *Pour tout graphe simple étiqueté* $\mathbf{G}$*, il existe un algorithme de nommage et un algorithme d'élection pour* $\mathbf{G}$ *dans le modèle de communication synchrone si et seulement si* $\mathbf{G}$ *est minimal pour les revêtements.*

Remarque 3.37 *Les autres résultats présentés dans cette section sont basés sur l'hypothèse supplémentaire que chaque sommet connaît initialement son degré. Puisque dans les deux modèles considérés ici, cette propriété est assurée, tous les résultats présentés dans la suite de cette section restent vrais dans ces modèles.*

3.6.2 Calculs Locaux sur les Étoiles Ouvertes

On rappelle qu'informellement, les calculs locaux sur les étoiles ouvertes permettent en un pas de calcul à un sommet de modifier son étiquette et les étiquettes des arêtes qui lui sont incidentes. L'application d'un tel pas de calcul ne dépend que de son étiquette, des étiquettes de ses voisins et des étiquettes des arêtes qui lui sont incidentes.

On définit maintenant les relations de réétiquetage localement engendrées sur les étoiles ouvertes.

Définition 3.38 *Une relation de réétiquetage* $\mathcal{R}$ *est localement engendrée sur les étoiles ouvertes si la condition suivante est satisfaite. Pour tous graphes simples* (G,λ)*,* (G,λ')*,* (H,η)*,* (H,η')*, pour tout sommets* $v \in V(G)$ *et* $w \in V(H)$ *tels qu'il existe un isomorphisme* $\varphi : B_G(v) \to B_G(w)$*, si les conditions suivantes sont vérifiées :*

1. $\lambda(v) = \eta(\varphi(v))$ *et* $\lambda'(v) = \eta'(\varphi(v))$*,*
2. *pour toute arête* $e \in I_G(v)$*,* $\lambda(e) = \eta(\varphi(e))$ *et* $\lambda'(e) = \eta'(\varphi(e))$*,*
3. *pour tout* $v' \in N_G(v)$*,* $\lambda(v) = \eta(\varphi(v))$*,*
4. *pour tout* $x \notin \{v\} \cup I_G(v)$*,* $\lambda'(x) = \lambda(x)$*,*
5. *pour tout* $x \notin \{w\} \cup I_H(w)$*,* $\eta'(x) = \eta(x)$*,*

alors $(G,\lambda)\ \mathcal{R}\ (G,\lambda')$ *si et seulement si* $(H,\eta)\ \mathcal{R}\ (H,\eta')$*.*

On considère seulement des relations de réétiquetages récursives et cela pour considérer des modèles de calcul ayant une puissance de calcul raisonnable. Par définition, les *calculs locaux sur les étoiles ouvertes* correspondent aux relations de réétiquetage localement engendrées sur les étoiles ouvertes.

Une relation de réétiquetage localement engendrée sur les étoiles ouvertes peut être décrite par un ensemble récursif de règles de réétiquetage où chaque règle permet de modifier les étiquettes d'un sommet et des arêtes incidentes à ce sommet en fonction des étiquettes apparaissant dans l'étoile centrée en ce sommet. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage localement engendrée sur les étoiles ouvertes. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

3.6.3 Équivalence entre les Calculs Locaux sur les Arêtes Étiquéetées avec Connaissance Initiale du Degré et les Calculs Locaux sur les Étoiles Ouvertes

On dit qu'un algorithme $\mathcal{R}$ résout un problème $\mathcal{P}$ sur une famille de graphe $\mathcal{F}$ en utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré si $\mathcal{R}$ utilise des calculs locaux sur les arêtes étiquetées et que la propriété suivante est vérifiée. Pour tout graphe $(G, \lambda) \in \mathcal{F}$, toute exécution de $\mathcal{R}$ sur le graphe $(G, (\lambda, d))$ résout le problème $\mathcal{P}$ où pour tout $v \in V(G)$, $d(v)$ est le degré de v dans le graphe G. Ainsi un algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré est un algorithme qui peut utiliser le fait que chaque sommet connaît son degré.

On montre maintenant que les calculs locaux sur les étoiles ouvertes ont la même puissance de calcul que les calculs locaux sur les arêtes étiquetés avec connaissance initiale du degré.

Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux sur les étoiles ouvertes, à chaque pas de réétiquetage, le sommet qui modifie son étiquette peut calculer son degré. Puisqu'il est évident qu'on peut simuler un algorithme utilisant des calculs locaux cellulaires sur les arêtes étiquetées en utilisant des calculs locaux sur les étoiles ouvertes, on sait d'après la Proposition 3.20 que tout algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré peut être simulé par un algorithme utilisant des calculs locaux sur les étoiles ouvertes.

On considère maintenant un algorithme $\mathcal{R}$ utilisant des calculs locaux sur les étoiles ouvertes et on va montrer qu'il existe un algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré qui simule $\mathcal{R}$.

Un Processus d'Initialisation

Pour cela, on va commencer par décrire un processus d'initialisation qui va permettre à chaque sommet d'obtenir de l'information à propos de ses voisins et d'établir un ordre entre les extrémités de chaque arête.

L'étiquette de chaque arête e est de la forme $(\lambda(e), p(e))$ où $\lambda(e)$ est l'étiquette initiale de e, qui ne sera pas modifiée par le processus d'initialisation et $p(e)$ est un numéro attribué à chaque arête par l'algorithme de telle sorte que deux arêtes incidentes à un même sommet ont des numéros différents. Initialement, chaque arête est étiqueté $(\lambda(e), 0)$.

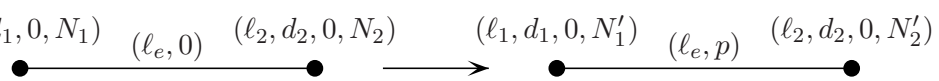
L'étiquette de chaque sommet v est de la forme $(\lambda(v), d(v), n(v), N_(v))$ dont la signification est la suivante :

- $\lambda(v)$ est l'étiquette initiale de v et n'est pas modifiée par le processus d'initialisation,
- $d(v) \in \mathbb{N}$ est le degré de v dans le graphe et n'est pas modifié par le processus d'initialisation,
- $n(v) \in \mathbb{N}$ est un numéro associé à chaque sommet qui est modifié de telle sorte que dans la configuration finale, deux sommets adjacents ont deux numéros différents,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N}^3)$ est un ensemble de triplets d'entiers qui contient les informations dont v dispose à propos de ses voisins. Si un triplet $(p, m, n) \in N(v)$, cela signifie que lors du dernier pas de réétiquetage effectuée sur l'arête e incidente à v dont le numéro est p, le numéro de v était n et le numéro de l'autre extrémité de e était m.

Initialement, chaque sommet est étiqueté $(\lambda(v), d(v), 0, \emptyset)$.

La première règle de réétiquetage $\mathcal{I}_1$ permet à deux sommets voisins v et v' de donner un numéro à l'arête e les reliant si celle-ci n'en a pas encore. Dans ce cas là, le couple $(p(e), 0, 0)$ est ajouté à $N(v)$ et $N(v')$. Ainsi, à tout moment de l'exécution, si un couple $(p, m, n) \in N(v)$, cela signifie qu'il existe une arête incidente à v dont le numéro est p. De cette manière, en raison du choix du numéro attribué à chaque arête lors de l'application de la règle $\mathcal{I}_1$, on sait que deux arêtes incidentes à un même sommet qui ont des numéros non-nuls ont toujours des numéros différents.

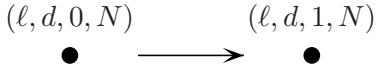
$\mathcal{I}_1$:



où $p := 1 + \max\{p' \mid \exists (p', 0, 0) \in N_1 \cup N_2\}$,
$N_1' := N_1 \cup \{(p, 0, 0)\}$,
$N_2' := N_2 \cup \{(p, 0, 0)\}$.

La deuxième règle permet à un sommet v qui sait qu'il ne peut plus appliquer la règle $\mathcal{I}_1$ de prendre le numéro 1. Si $N(v)$ contient $d(v)$ triplets, cela signifie que le sommet v a appliqué $d(v)$ fois la règle $\mathcal{I}_1$ et par conséquent, puisque $d(v)$ est le degré de v, on sait que toutes les arêtes incidentes à v ont des numéros non-nuls qui sont tous différents. Ainsi, si un sommet v a un numéro non-nul, cela signifie que toutes les arêtes incidentes à v ont des numéros différents et v connaît ces numéros (i.e., il peut les retrouver à partir de $N(v)$).

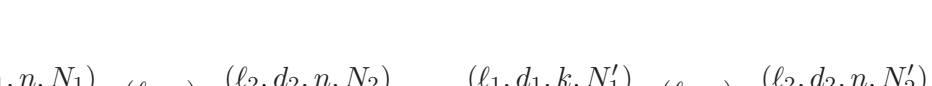
$\mathcal{I}_2$:



Cette règle est applicable si $|N| = d$.

La règle $\mathcal{I}_3$ permet à deux sommets voisins v et v' reliées par une arête e qui ont le même numéro n de briser cette symétrie, i.e., le sommet v prend un nouveau numéro k lorsque cette règle est appliquée. Cette règle ne peut être appliquée que si les deux sommets ont des numéros non-nuls, i.e., toutes les arêtes incidentes aux deux sommets ont des numéros non-nuls. De plus, le triplet $(p(e), n, k)$ est ajouté à $N(v)$ et le triplet $(p(e), k, n)$ est ajouté à $N(v')$, puisque le but de $N(v)$ est de stocker le numéro de chaque voisin v' de v ainsi que le numéro qu'avait v lors du dernier échange de numéros entre v et v'. On remarque que le numéro k généré est plus grand que tous les numéros que v connait dans son voisinage. Ainsi, si l'information que v a à propos de son voisinage est correcte, v a un numéro différent de tous ses voisins.

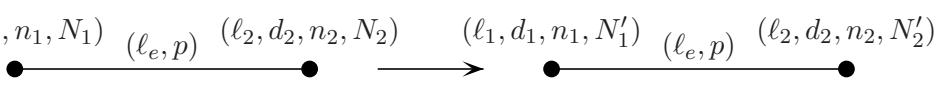
$\mathcal{I}_3$:



Si $p > 0$ et $n > 0$
alors $k := 1 + \max\{n' \mid \exists (p', n', m') \in N_1\}$,
$N_1' := N_1 \setminus \{(p', n', m') \mid p = p'\} \cup \{(p, n, k)\}$,
$N_2' := N_2 \setminus \{(p', n', m') \mid p = p'\} \cup \{(p, k, n)\}$.

La quatrième règle sert à mettre à jour l'information dont disposent les sommets à propos de leurs voisins. Si un sommet v a un voisin v' relié à v par une arête e et si $(p(e), n(v'), n(v)) \notin N(v)$, cela signifie que v et v' n'ont pas échangé leurs numéros depuis que v ou v' a modifié son numéro.

$\mathcal{I}_4$:



Si $p, n_1, n_2 > 0$, $n_1 \neq n_2$ et
$(p, n_2, n_1) \notin N_1$ ou $(p, n_1, n_2) \notin N_2$
alors $N_1' := N_1 \setminus \{(p', n', m') \mid p = p'\} \cup \{(p, n_2, n_1)\}$,
$N_2' := N_2 \setminus \{(p', n', m') \mid p = p'\} \cup \{(p, n_1, n_2)\}$.

On considère un graphe $\mathbf{G}$ et une exécution de l'algorithme d'initialisation $\mathcal{I}$ sur $\mathbf{G}$. Pour chaque étape i de l'exécution, on note $(\lambda(v), d(v), n_i(v), N_i(v))$ l'étiquette du sommet v et $(\lambda(e), p_i(e))$ l'étiquette de l'arête e après le $i$ème pas de réétiquetage. On peut facilement prouver le lemme suivant par récurrence sur le nombre d'étapes.

Lemme 3.39 *Pour tous sommets $v, v' \in V(G)$ et toute arête $e \in E(G)$ dont les extrémités sont v et v', pour toute étape i, les propriétés suivantes sont satisfaites :*

- $p_i(e) > 0 \implies p_{i+1}(e) = p_i(e)$,
- $p_i(e) \neq 0 \implies \exists (p_i(e), n, m) \in N_i(v)$,
- $n_i(v) > 0 \implies p_i(e) > 0$,
- $n_i(v) > 0 \implies \forall (p, m, n) \in N_i(v), m \neq n_i(v)$,
- $(p_i(e), m, n) \in N_i(v) \iff (p_i(e), n, m) \in N_i(v')$.

On montre maintenant qu'un sommet peut détecter que son numéro ne sera plus modifié lors de la suite de l'exécution.

Lemme 3.40 *Pour tout sommet $v \in V(G)$ et pour toute étape i, si $n_i(v) > 0$ et si pour tout $(p, n, m) \in N_i(v)$, $m = n_i(v)$, alors pour toute étape $i' \geq i$, $n_{i'}(v) = n_i(v)$.*

Preuve : Pour tout sommet $v \in V(G)$ et pour toute étape i, si $n_i(v) > 0$ et si pour tout $(p, n, m) \in N_i(v)$, $m = n_i(v)$, on sait d'après le Lemme 3.39, que pour tout $v' \in N_G(v)$, il existe $(p', n_i(v), m') \in N_i(v')$. Par conséquent, d'après le Lemme 3.39, on sait que pour tout $v' \in N_G(v)$, $n_i(v') \neq n_i(v)$. De plus, la seule règle qui permet à un sommet de changer de numéro est la règle $\mathcal{I}_3$ et de par la définition de cette règle, on sait que pour tout $v' \in N_G(v)$, si v' modifie son numéro à une étape $i' > i$, $n_{i'(v')} > n_i(v)$. Par conséquent, pour tout $i' \geq i$, pour tout $v' \in N_G(v)$, $n_{i'}(v') \neq n_i(v)$. Ainsi, le numéro de v ne peut pas être modifiée à une étape $i' \geq i$ puisque v n'a pas de voisin avec le même numéro que lui. □

Par conséquent, chaque sommet v sait quand le numéro qui lui est attribué est son numéro définitif. De plus, si tous les voisins de v ont aussi leurs numéros définitifs, la règle $\mathcal{I}_4$ ne peut plus être appliquée qu'une seule fois entre le sommet v et chacun de ses voisins. Une fois que v a son numéro définitif, il lui suffit de s'assurer que chaque sommet $v' \in N_G(v)$ (qu'il distingue grâce aux numéros des arêtes) a son numéro définitif et que l'information dont dispose v a propos de v' est correcte. Ainsi, il est possible pour chaque sommet de détecter qu'il a calculé son numéro définitif, de s'assurer que ses voisins ont

aussi leurs numéros définitifs et que la valeur de $N(v)$ est définitive. Le sommet v peut ensuite observer et stocker dans $N(v)$ les étiquettes initiales de toutes les arêtes qui lui sont incidentes et de ses voisins correspondants.

L'Algorithme de Simulation

Une fois qu'un sommet v s'est assuré que les valeurs $n(v)$ et $N(v)$ ne seront plus modifiées, il va utiliser ces valeurs pour essayer de simuler des règles de l'algorithme $\mathcal{R}$ utilisant des calculs locaux sur les arêtes ouvertes.

Le principe de la simulation est le suivant : on va utiliser un mécanisme classique de synchronisation présenté par Rosenstiehl et al. [RFH72]. La simulation va être divisée en phases de telle sorte que l'écart entre les phases de deux voisins est d'au plus 1. La méthode est la suivante, chaque sommet qui a terminé la période d'initialisation (i.e., $n(v)$ a sa valeur définitive $N(v)$ contient les étiquettes initiales des arêtes incidentes à v et des voisins de v) passe à la phase 1. Pour passer à la phase $i+1$, chaque sommet consulte d'abord ses voisins (qu'il distingue grâce aux numéros attribués aux arêtes) et s'assure qu'ils sont tous à une phase supérieure ou égale à i. Il est montré dans [RFH72] qu'au lieu d'utiliser un compteur non-borné, on peut se contenter d'un compteur à trois valeurs et on additionne alors modulo 3, mais pour simplifier l'explication, on ne considère pas cette version de l'algorithme de synchronisation ici.

On utilise maintenant le mécanisme de synchronisation pour simuler $\mathcal{R}$. Le principe est qu'à chaque phase, chaque sommet v vérifie s'il peut simuler une règle de $\mathcal{R}$ sur l'étoile centrée en v. Ensuite, on utilise les numéros des sommets pour gérer les conflits entre les sommets voisins afin que seulement un ensemble indépendant de sommets simule une règle de $\mathcal{R}$.

Une étape de simulation se déroule sur deux phases de synchronisation. Dans les phases impaires, à partir de l'information (qui est à jour) dont dispose un sommet v à propos des étiquettes des arêtes qui lui sont incidentes et de ses voisins, le sommet v observe si une règle de $\mathcal{R}$ peut être appliquée sur l'étoile de centre v. Si une telle règle peut être appliquée, il prend un statut **ask** et sinon, il prend un statut **idle**. Ensuite, chaque sommet v consulte l'état de chacun de ses voisins. Si v a le statut **ask** et s'il a un voisin v' dont le statut est **ask**, alors on compare les numéros $n(v)$ et $n(v')$ qu'on sait différents. Si $n(v) < n(v')$, alors v prend le statut **idle** et sinon v garde le statut **ask** (v' prend alors le statut **idle**). Si on considère l'ensemble $A_{2i+1} \subseteq V(G)$ des sommets qui avaient le statut **ask** au début de la phase $2i+1$, on sait que chaque sommet de $\{v \in A_{2i+1} \mid \forall v' \in A_{2i+1}, n(v') \leq n(v)\}$ a conservé le statut **ask** à la fin de la phase $2i+1$. Un sommet qui a conservé le statut **ask** à la fin de cette phase (on sait qu'il en existe au moins un) est autorisé à simuler une règle de réétiquetage de $\mathcal{R}$.

Dans les phases paires de la synchronisation, chaque sommet v observe l'état de chacun de ses voisins. Lorsque deux sommets voisins v et v' reliées par une arête e se synchronisent et que le statut de v est **ask** (cela signifie que v a été autorisé à simuler une règle de $\mathcal{R}$), l'étiquette $\lambda(e)$ de l'arête e est mise à jour si nécessaire (si l'application de la règle simulée modifie l'étiquette de e) et $N(v')$ est modifiée de façon à ce que v' connaisse l'état de v et de l'arête e pour la phase suivante. Une fois qu'un sommet dont le statut est **idle** a vu tous ses voisins durant cette phase, il peut passer à la phase suivante. Une fois qu'un sommet dont le statut est **ask** a vu tous ses voisins, il met à jour son étiquette $\lambda(v)$ ainsi

que $N(v)$ (en fonction de la règle qu'il a simulé) et il reprend la statut **idle** avant de passer à la phase de synchronisation suivante.

L'algorithme ainsi décrit assure que dans chaque étape de simulation, constituée de deux phases de synchronisation, une règle de $\mathcal{R}$ est effectivement simulée si l'exécution de $\mathcal{R}$ n'est pas terminée.

Pour assurer la terminaison de l'algorithme décrit précédemment, on le modifie de la manière suivante. Un sommet v est autorisé à passer à la phase $2i+1$ seulement si une règle de $\mathcal{R}$ peut être simulé sur l'étoile de centre v, ou si v a un voisin v' qui est à l'étape $2i+1$.

Propriétés de l'Algorithme

On considère maintenant un graphe $\mathbf{G} = (G, \lambda)$ sur lequel toute exécution de $\mathcal{R}$ termine et une exécution ρ de l'algorithme de simulation sur $\mathbf{G}$. Pour toute $x \in V(G) \cup E(G)$, pour toute étape i, on note $\lambda_i(x)$ la partie de l'étiquette de x après le $i$ème pas de réétiquetage de ρ qui correspond à l'étiquette de x dans l'algorithme simulé. On va maintenant construire une exécution valide de $\mathcal{R}$ sur $\mathbf{G}$ qui correspond à l'exécution ρ. Soient $i_1, i_2, \ldots, i_j, \ldots$ les étapes de ρ lors desquelles un sommet v modifie son étiquette $\lambda(v)$. Cela signifie qu'une règle de $\mathcal{R}$ a été simulée sur l'étoile de centre v. On considère l'exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$ où à l'étape j, on applique sur un sommet v la règle qui a fini d'être simulée à l'étape i_j sur le sommet v dans l'exécution ρ. En raison de la procédure de synchronisation, il est clair que l'exécution ρ' est une exécution valide de $\mathcal{R}$ sur $\mathbf{G}$. Par la suite, pour tout $x \in V(G) \cup E(G)$ et pour toute étape j de ρ', on note $\lambda'_j(x)$ l'étiquette de x après le $j$ème pas de réétiquetage de ρ'.

Ainsi, si l'exécution ρ ne termine pas, l'exécution ρ' ne termine pas, ce qui est impossible. Si on note λ_f l'étiquetage de G obtenu à partir de la configuration finale de ρ et λ'_f l'étiquetage de la configuration finale de ρ', il est clair que pour tout $x \in V(G) \cup E(G)$, $\lambda_f(x) = \lambda'_f(x)$ et que dans la configuration finale de l'algorithme de simulation, tous les sommets de G ont le statut **idle** et sont rendus à la même phase de simulation. Par conséquent, on a bien réussi à trouver un algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré qui permet de simuler $\mathcal{R}$.

On suppose maintenant que $\mathcal{R}$ permet de résoudre un problème $\mathcal{P}$ avec détection de la terminaison. On note L l'ensemble d'étiquettes utilisés par $\mathcal{R}$ et on sait qu'il existe un ensemble S, une fonction $\mathbf{res} : L \to S$ et un ensemble $L_f \subseteq L$ tels que les propriétés suivantes sont toujours vérifiées pour tout exécution ρ' de $\mathcal{R}$ sur $\mathbf{G}$.

- L'étiquetage $\mathbf{res} \circ \lambda'_f$ est une solution de $\mathcal{P}$ sur $\mathbf{G}$.
- Il existe un sommet v et une étape j tels que $\lambda_j(v)' \in L_f$.
- S'il existe un sommet $v \in V(G)$ et une étape j telle que $\lambda'_j(v) \in L_f$, alors pour tout $j' > j$, $\mathbf{res} \circ \lambda'_{j'} = \mathbf{res} \circ \lambda'_j$.

Pour montrer que l'algorithme de simulation permet de détecter la terminaison, on considère le même ensemble L_f et la même fonction **res**, modulo la projection pour obtenir $\lambda(v)$ à partir de l'étiquette de v dans l'algorithme de simulation.

Puisqu'on sait que l'algorithme de simulation simule $\mathcal{R}$, on sait que dans la configuration finale λ_f de ρ, $\mathbf{res} \circ \lambda_f = \mathbf{res} \circ \lambda'_f$ est une solution de $\mathcal{P}$ sur $\mathbf{G}$. De plus, puisque ρ' est une exécution valide de $\mathcal{R}$ sur $\mathbf{G}$, il existe un sommet $v \in V(G)$ et une étape j telle que $\lambda'_j(v) \in L_f$ et par conséquent, il existe un sommet v et une étape i_j lors de laquelle

$\lambda_{i_j}(v) = \lambda'_j(v) \in L_f$.

On prouve dans le lemme suivant la dernière propriété qui permet d'assurer que l'algorithme de simulation permet de détecter la terminaison.

Lemme 3.41 *S'il existe un sommet v et une étape i telle que $\lambda(i) \in L_f$, alors pour toute étape $i' \geq i$ et pour tout $x \in V(G) \cup E(G)$, $\mathbf{res}(\lambda_{i'}(x)) = \mathbf{res}(\lambda_i(x))$.*

Preuve : On considère un sommet v et une étape i lors de laquelle $\lambda_i(v) \in L_f$. On considère la dernière étape où l'étiquette de v a été modifiée. De part la définition de ρ', on sait que cela a eu lieu lors d'une étape i_j et que $\lambda'_j(v) = \lambda_{i_j}(v) \in L_f$. D'après la définition de ρ', pour tout sommet $v' \in V(G)$, $\lambda_{i_j}(v') = \lambda'_j(v')$ et pour toute étape $i' \geq i_j$, si $\lambda_{i'+1}(v') \neq \lambda_{i'}(v')$, il existe une étape $j' > j$ telle que $\lambda_{i'+1}(v') = \lambda'_{j'}(v)$ et par conséquent, pour tout $i' \geq i_j$, $\mathbf{res}(\lambda_{i'}(v)) = \mathbf{res}(\lambda'_{j'}(v)) = \mathbf{res}(\lambda'_j(v)) = \mathbf{res}(\lambda_{i_j}(v))$.

Pour toute arête $e \in E(G)$, si $\lambda'_j(e) \neq \lambda_{i_j}(e)$, cela signifie que e est incidente à un sommet v' dont le statut est **ask** et qui est en train d'exécuter une phase paire de l'algorithme de simulation. Puisque toute exécution de l'algorithme de simulation termine, il existe une étape $i_{j'} > i_j$ de ρ lors de laquelle v' terminera cette phase. Ainsi, il existe une étape $j' > j$ lors de laquelle $\lambda'_{j'}(e) = \lambda_{i_{j'}}(e) = \lambda_{i_j}(e)$ et par conséquent, $\mathbf{res}(\lambda_{i_j}(e)) = \mathbf{res}(\lambda'_{j'}(e)) = \mathbf{res}(\lambda'_j(e))$. Pour les mêmes raisons que précédemment, pour toute étape $i' \geq i_j$ et toute arête $e \in E(G)$ telle que $\lambda_{i'+1}(e) \neq \lambda_{i'}(e)$, il existe une étape $j' > j$ telle que $\lambda_{i'+1}(e) = \lambda'_{j'}(e)$ et par conséquent, pour tout $i' \geq i_j$, $\mathbf{res}(\lambda_{i'}(e)) = \mathbf{res}(\lambda'_{j'}(e)) = \mathbf{res}(\lambda'_j(e)) = \mathbf{res}(\lambda_{i_j}(e))$. □

On a donc montré la proposition suivante.

Proposition 3.42 *Pour tout algorithme $\mathcal{R}$ utilisant des calculs locaux sur les étoiles ouvertes, il existe un algorithme utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées avec connaissance initiale du degré qui simule $\mathcal{R}$.*

De plus, si $\mathcal{R}$ permet de résoudre un problème $\mathcal{P}$ sur une famille de graphes $\mathcal{F}$ avec détection de la terminaison, alors il existe un algorithme utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées avec connaissance initiale du degré qui permet de résoudre le problème $\mathcal{P}$ sur la famille $\mathcal{F}$ avec détection de la terminaison.

3.6.4 Un Modèle Strictement plus Puissant

On va montrer ici que la connaissance initiale du degré permet d'obtenir un modèle strictement plus puissant lorsqu'on considère les calculs locaux sur les arêtes étiquetées.

Il faut toutefois noter qu'il n'existe pas de graphe $\mathbf{G}$ qui n'est pas minimal pour les revêtements et qui admette un algorithme d'élection utilisant des calculs locaux sur les arêtes étiquetées avec connaissance du degré. Cela est du au fait que si un graphe $\mathbf{G}$ est un revêtement d'un graphe $\mathbf{H}$ à travers un homomorphisme γ, alors γ préserve le degré. Autrement dit, même si les sommets connaissent initialement leur degré, le lemme de relèvement est toujours valide ainsi que le résultat d'impossibilité de la Proposition 3.22. On a donc d'après le Théorème 3.29 et la Proposition 3.42 le théorème suivant.

Théorème 3.43 *Pour tout graphe simple étiqueté $\mathbf{G}$, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour $\mathbf{G}$ utilisant des calculs locaux sur les étoiles ouvertes,*

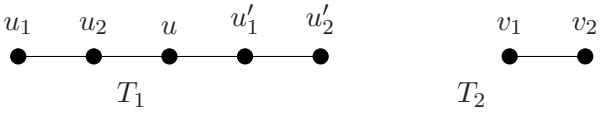


FIG. 12 – Les graphes T_1 et T_2 sont tous les deux minimaux pour les revêtements, mai sil n'existe pas d'algorithme utilisant les calculs locaux sur les arêtes étiquetées permettant de résoudre l'élection dans ces deux graphes.

2. *il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour* $\mathbf{G}$ *utilisant des calculs locaux sur les étoiles ouvertes,*
3. $\mathbf{G}$ *est minimal pour les revêtements.*

On va cependant montrer que si on considère le problème de l'élection sur des familles de graphes, on peut résoudre avec la connaissance initiale du degré le problème de l'élection dans des familles de graphes n'admettant pas d'algorithme universel d'élection utilisant des calculs locaux sur les arêtes étiquetées sans connaissance initiale du degré.

On considère les deux arbres non-étiquetés T_1 et T_2 de la Figure 12. On sait que ces deux arbres sont tous les deux minimaux pour les revêtements. On montre dans la proposition suivante qu'il n'existe pas d'algorithme utilisant des calculs locaux sur les arêtes étiquetées qui permette de résoudre le problème de l'élection sur la famille constituée de ces deux graphes.

Proposition 3.44 *Il n'existe pas d'algorithme* $\mathcal{R}$ *utilisant des calculs locaux sur les arêtes étiquetées (sans connaissance initiale du degré) qui permette de résoudre le problème de l'élection sur* T_1 *et sur* T_2*.*

Preuve : Soit $\mathcal{R}$ un algorithme utilisant des calculs locaux sur les arêtes étiquetées qui permet de résoudre le problème de l'élection sur l'arbre T_2. On considère une exécution ρ de $\mathcal{R}$ sur T_2 et on suppose, sans perte de généralité que le sommet v_1 a l'étiquette ÉLU et le sommet v_2 l'étiquette NON-ÉLU. On considère maintenant le sous graphe T (resp. T') de T_1 induit par les sommets u_1 et u_2 (resp. u'_1 et u'_2). Puisque T (resp. T') est isomorphe à T_2, il existe une exécution de $\mathcal{R}$ sur T (resp. T') tel que u_1 (resp. u'_1) obtienne l'étiquette ÉLU et u_2 (resp. u'_2) l'étiquette NON-ÉLU. On peut donc construire une exécution de $\mathcal{R}$ sur T_1 de telle sorte que cette exécution atteigne une configuration où les deux sommets u_1 et u'_1 ont l'étiquette ÉLU. Puisque l'étiquette ÉLU est une étiquette finale, cela signifie que $\mathcal{R}$ n'est pas un algorithme d'élection pour T_1. □

Cependant, l'algorithme d'élection pour la famille des arbres présenté dans le Chapitre 1 peut être implémenté avec des calculs locaux sur les étoiles ouvertes et d'après la Proposition 3.42, on a donc le théorème suivant.

Théorème 3.45 *Il existe un algorithme d'élection pour la famille des arbres utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré.*

Ainsi, les calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré définissent un modèle de calcul strictement plus puissant que les calculs locaux sur les arêtes étiquetées sans connaissance initiale du degré.

3.6.5 Élection dans les Familles de Diamètre Borné

On montre dans cette partie que pour pouvoir élire dans un graphe $\mathbf{G}$ minimal pour les revêtements, il n'est pas nécessaire de connaître la taille de $\mathbf{G}$, mais qu'une borne sur le diamètre suffit si chaque sommet connaît initialement son degré. Pour cela, on explique comment implémenter l'algorithme GSSP décrit dans le Chapitre 2 utilisant des calculs locaux sur les arêtes étiquetées avec connaissance du degré.

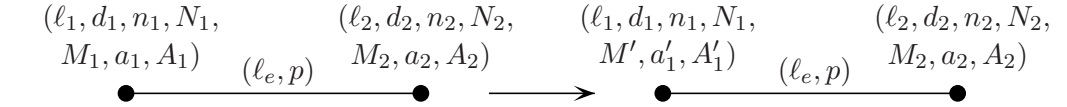
Comme dans le Chapitre 2, l'algorithme qu'on va décrire maintenant va permettre de vérifier si tous les sommets du graphe sont d'accord sur la même boîte-aux-lettres. Cet algorithme utilise des calculs locaux sur les arêtes étiquetées et il va donc être nécessaire que chaque sommet stocke l'information qu'il a à propos du rayon de confiance de ses voisins. C'est la méthode utilisé dans l'algorithme SSP original [SSP85]. Il faut cependant faire attention à réinitialiser toutes ces informations à chaque fois qu'un sommet v modifie sa boîte-aux-lettres.

On modifie l'algorithme $\mathcal{M}$ présenté précédemment de telle sorte que si l'algorithme est exécuté sur un graphe $\mathbf{G} = (G, \lambda)$, l'étiquette de chaque arête est la même que dans $\mathcal{M}$ et l'étiquette de chaque sommet $v \in V(G)$ est $(\lambda(v), d(v), n(v), N(v), M(v), a(v), A(v))$ où les champs $\lambda(v), n(v), N(v)$ et $M(v)$ ont le même rôle que dans l'algorithme $\mathcal{M}$. L'entier $d(v)$ est le degré du sommet v dans le graphe $\mathbf{G}$ et sa valeur ne sera pas modifiée lors de l'exécution lors de l'exécution de l'algorithme. L'entier $a(v)$ est le *rayon de confiance* du sommet v et va jouer le même rôle que dans l'algorithme GSSP. L'ensemble $A(v)$ est un ensemble de couples d'entiers de la forme (p, a) et il va contenir l'information dont dispose v a propos du rayon de confiance de ses voisins : si le numéro associé à l'arête $\{v, v'\}$ est p et si $(p, a) \in A(v)$ cela signifie que la dernière fois que v et v' ont échangé leurs rayons de confiance, la valeur de $a(v')$ était a.

Initialement, l'étiquette de chaque arête $e \in E(G)$ est $(\lambda(e), 0)$ comme dans l'algorithme $\mathcal{M}$ et l'étiquette de chaque sommet $v \in V(G)$ est $(\lambda(v), d(v), 0, \emptyset, \emptyset, -1, \emptyset)$.

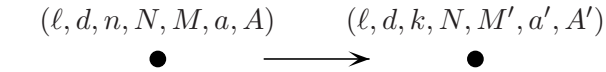
Les règles $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ restent les mêmes que dans l'algorithme $\mathcal{M}$ a la seule différence que si la boîte-aux-lettres d'un sommet v est modifiée par l'application d'une de ces règles, alors $a(v)$ est réinitialisé à -1 et la nouvelle valeur de $A(v)$ est $\{(p, -1) \mid \exists (p, a) \in A(v)\}$. En effet, puisque le rôle de $A(v)$ est de stocker le rayon de confiance des voisins de v par rapport à la valeur de $M(v)$, il faut réinitialiser ces informations devenues obsolètes à chaque fois que la boîte aux lettres de v est modifiée. On note $\mathcal{M}'_1$, $\mathcal{M}'_2$, $\mathcal{M}'_3$ les règles ainsi obtenues.

$\mathcal{M}'_1$:



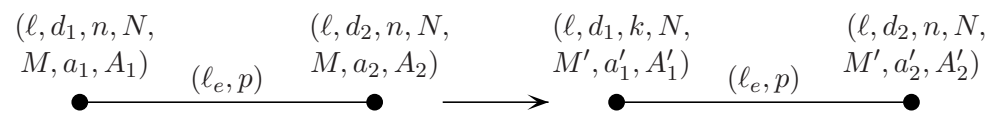
Si $M_2 \setminus M_1 \neq \emptyset$
alors $M' := M'_1 \cup M'_2$;
$a'_1 := -1$;
$A'_1 := \{(p', -1) \mid \exists (p', a) \in A_1\}$.

$\mathcal{M}'_2$:



Si $n = 0$ ou $\exists (n, \ell', N') \in M$ tel que $(\ell, N) \prec (\ell', N')$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$;
$M' := M \cup \{(k, \ell, N)\}$;
$a' := -1$;
$A' := \{(p', -1) \mid \exists (p', a_1) \in A\}$.

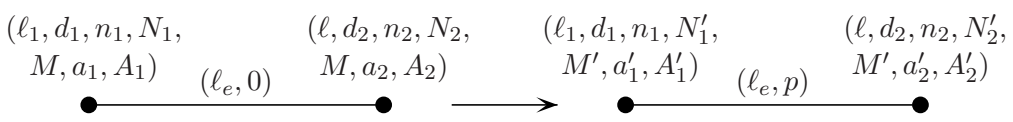
$\mathcal{M}'_3$:



Si $n > 0$ et
$\forall (n, \ell', N') \in M$, $(\ell', N') \preceq (\ell, N)$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$;
$M' := M \cup \{(k, \ell, N)\}$;
$a'_1 := -1$;
$A'_1 := \{(p', -1) \mid \exists (p, a) \in A_1\}$;
$a'_2 := -1$;
$A'_2 := \{(p', -1) \mid \exists (p, a) \in A_2\}$;

La règle $\mathcal{M}_4$ est modifiée de telle sorte que si deux sommets voisins v et v' appliquent la règle $\mathcal{M}_4$, le couple $(p, -1)$ est ajouté à $A(v)$ et $A(v')$. Comme précédemment, les valeurs de $a(v)$ et $a(v')$ sont réinitialisées ainsi que les informations contenues dans $A(v)$ et $A(v')$. On note $\mathcal{M}'_4$ la règle ainsi obtenue.

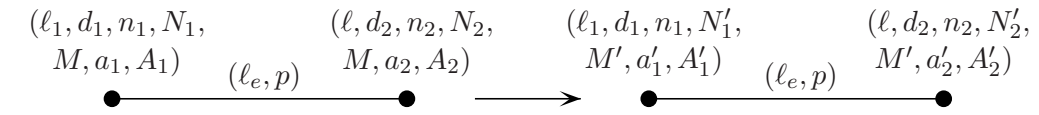
$\mathcal{M}'_4$:



Si $n_1 \neq n_2$, $n_1 > 0$, $n_2 > 0$,
$\forall (n_1, \ell'_1, N'_1) \in M, (\ell'_1, N'_1) \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell'_2, N'_2) \in M$, $(\ell'_2, N'_2) \preceq (\ell_2, N_2)$ et
alors $p := 1 + \max\{p' \mid \exists (p', \ell', n') \in N_1 \cup N_2\}$;
$N'_1 := N_1 \cup \{(p, \ell_e, n_2)\}$;
$N'_2 := N_2 \cup \{(p, \ell_e, n_1)\}$;
$M' = M \cup \{(n_1, N'_1), (n_2, N'_2)\}$;
$a'_1 := -1$;
$A'_1 := \{(p', -1) \mid \exists (p', a) \in A_1\} \cup \{(p, -1)\}$;
$a'_2 := -1$;
$A'_2 := \{(p', -1) \mid \exists (p', a) \in A_2\} \cup \{(p, -1)\}$;

La règle $\mathcal{M}_5$ est modifiée de la même manière que les règles $\mathcal{M}_1, \mathcal{M}_2$ et $\mathcal{M}_3$.

$\mathcal{M}'_5$:

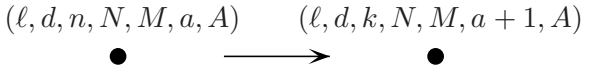


Si $p > 0$, $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell'_1, N'_1) \in M, (\ell'_1, N'_1) \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell'_2, N'_2) \in M$, $(\ell'_2, N'_2) \preceq (\ell_2, N_2)$ et
$(p, \ell_e, n_2) \notin N_1$

alors $N'_1 := N_1 \setminus \{(p, \ell_e, n'_2) \in N_1\}) \cup \{(p, \ell_e, n_2\}$;
$N'_2 := N_2 \setminus \{(p, \ell_e, n'_1) \in N_2\}) \cup \{(p, \ell_e, n_1\}$;
$M' := M \cup \{(n_1, \ell_1, N'_1)\} \cup \{(n_2, \ell_2, N'_2)\}$;
$a'_1 := -1$;
$A'_1 := \{(p', -1) \mid \exists (p', a) \in A_1\}$;
$a'_2 := -1$;
$A'_2 := \{(p', -1) \mid \exists (p', a) \in A_2\}$;

Comme dans l'algorithme GSSP présenté dans le chapitre précédent, il faut une règle qui permette à un sommet d'augmenter son rayon de confiance. Un sommet v peut augmenter son rayon de confiance s'il ne peut pas appliquer la règle $\mathcal{M}'_2$, s'il connaît exactement $d(v)$ voisins (i.e., il existe exactement $d(v)$ couples (p, a) dans $A(v)$) et si chacun de ses voisins a un rayon de confiance supérieur ou égal au sien.

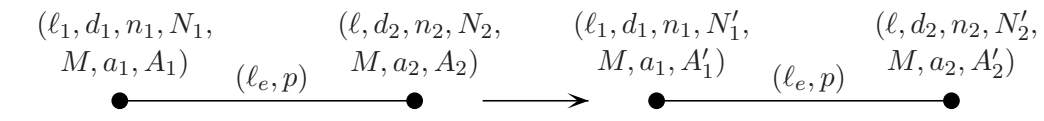
$\mathcal{M}'_6$:



Cette règle est applicable si $n > 0$,
$\forall (n, \ell', N') \in M, (\ell', N') \preceq (\ell, N)$
$|A| = d$ et
$\forall (p', a') \in A, a' \geq a$.

Il faut enfin une dernière règle qui permette à deux sommets d'échanger leurs rayons de confiance une fois que ceux ci ont été modifiés par la règle précédente. Cette règle ne peut être appliquée sur une arête e que si aucune des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4, \mathcal{M}'_5$ ne peut être appliquée sur e.

$\mathcal{M}'_7$:



Si $p > 0$, $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell'_1, N'_1) \in M, (\ell'_1, N'_1) \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell'_2, N'_2) \in M$, $(\ell'_2, N'_2) \preceq (\ell_2, N_2)$,
$(p, \ell_e, n_2) \in N_1$, $(p, \ell_e, n_1) \in N_2$ et
$(p, a_1) \notin A_2$ ou $(p, a_2) \notin A_1$
alors $A'_1 := A_1 \setminus \{(p, a)\} \cup \{(p, a_2)\}$;
$A'_2 := A_2 \setminus \{(p, a)\} \cup \{(p, a_1)\}$;

On s'intéresse maintenant aux propriétés satisfaites par toute exécution de l'algorithme $\mathcal{M}'$. Comme précédemment, on considère une exécution de l'algorithme $\mathcal{M}'$ sur un graphe étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), d(v), n_i(v), N_i(v), M_i(v), a_i(v), A_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage.

Le lemme suivant qui peut être facilement prouvé par récurrence sur le nombre d'étapes montre que le rayon de confiance d'un sommet ne peut qu'augmenter tant que sa boîte-aux-lettres n'est pas modifiée.

Lemme 3.46 *Pour tout sommet v et toute étape i, si $M_i(v) = M_{i+1}(v)$, alors $a_{i+1}(v) \geq a_i(v)$ et pour tout $(p, a) \in A_i(v)$, il existe $(p, a') \in A_{i+1}(v)$ et $a' \geq a \geq a_{i+1}(v) - 1$. De plus, si $a_i(v) \geq 0$, alors la règle $\mathcal{M}'_2$ ne peut être appliquée sur le sommet v.*

Dans le lemme suivant, on montre que le rayon de confiance d'un sommet permet d'obtenir des informations sur le contenu des boîtes-aux-lettres d'autres sommets du graphe lors d'étapes précédentes de l'exécution. Cette propriété est la même que la propriété de l'algorithme du Chapitre 2.

Lemme 3.47 *Pour tout sommet $v \in V(G)$ et toute étape i, pour tout sommet $w \in V(G)$ tel que $\text{dist}_G(v, w) \leq a_i(v)$, il existe une étape $j \geq i$ telle que $a_j(w) \geq a_i(v) - \text{dist}_G(v, w)$ et $M_j(v) = M_i(v)$.*

Preuve : On fait une démonstration par récurrence sur la distance k entre v et w dans G. Si $k = 0$, la propriété est trivialement vraie. On suppose maintenant que la propriété est vraie pour tous sommets v, w tels que $\text{dist}_G(v, w) \leq k$.

On considère deux sommets v, w et une étape i tels que $a_i(v) \geq k + 1 \geq 1$ et $\text{dist}_G(v, w) = k + 1$. Il existe un sommet $u \in N_G(v)$ tels que $\text{dist}_G(u, w) = k$. On note e l'arête reliant u à v. On sait que $p_i(e) > 0$ et qu'il existe $(p_i(e), a) \in A_i(v)$ tel que $a \geq a_i(v) - 1 \geq 0$.

On considère la dernière étape j' où la règle $\mathcal{M}'_7$ a été appliquée sur l'arête e. Lors de cette étape $M_{j'}(u) = M_{j'}(v) = M_i(v)$ et $a_{j'}(u) = a \geq a_i(v) - 1$. Par hypothèse de récurrence, on sait qu'il existe une étape $j < j'$ telle que $a_j(w) \geq a_{j'}(u) - k \geq a_i(v) - (k+1)$ et $M_j(w) = M_{j'}(u) = M_i(v)$. Ainsi, la propriété est vérifiée pour tous sommets v, w à distance $k + 1$ dans G. □

Comme dans l'algorithme GSSP du Chapitre 2, on montre que si à un moment donné, un sommet a un rayon de confiance supérieur au diamètre du graphe, alors tous les sommets

du graphe ont la même boîte-aux-lettres et ont tous un rayon de confiance supérieur à 0.

Lemme 3.48 *S'il existe un sommet v et une étape i telle que $a_i(v) \geq D(G)+1$, alors pour tout $w \in V(G)$, $M_i(w) = M_i(v)$ et $a_i(v) \geq 0$.*

Preuve : Puisque $a_i(v) > D(G)$, on sait d'après le Lemme 3.47 que pour tout sommet $w \in V(G)$, il existe une étape $i_w < i$ telle que $a_{i_w} \geq 1$ et $M_{i_w}(w) = M_i(v)$.

Supposons qu'il existe un sommet w tel que $M_i(w) \neq M_{i_w}(w)$. Soit j l'étape de l'exécution lors de laquelle pour la première fois, la boîte-aux-lettres d'un sommet w qui valait $M_i(v)$ a été modifiée. Autrement dit, pour tout sommet $w \in V(G)$, il existe une étape $j' \geq j-1$ telle que $M_{j'}(w) = M_i(v)$ et il existe un sommet w tel que $M_{j-1}(w) = M_i(v) \subsetneq M_j(w)$. Cela signifie qu'une des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4, \mathcal{M}'_5$ a été appliquée lors de l'étape j. Soit w un sommet tel que $M_{j-1}(w) \subsetneq M_j(w)$; on sait que $M_{j-1}(w) = M_i(v)$ et $a_{j-1}(w) \geq 0$. Par conséquent, d'après le Lemme 3.46, la règle $\mathcal{M}'_2$ n'a pas pu être appliquée sur le sommet w.

On note e l'arête sur laquelle une règle a été appliquée lors de l'étape j et on note w et w' ses extrémités. On suppose que l'étiquette de w a été modifiée lors de l'étape j. En raison du choix de j, on sait que la règle $\mathcal{M}'_1$ n'a pas pu être appliquée sur e à l'étape j. Par ailleurs, on sait que $a_{j-1}(w) \geq 1$ et par conséquent, il existe $(p_{j-1}(e), a) \in A_{j-1}(e)$ tel que $a \geq a_{j-1}(v) - 1 \geq 0$. Par conséquent, cela signifie qu'il existe une étape $j' < j$ lors de laquelle la règle $\mathcal{M}'_7$ a été appliquée sur l'arête e et de plus, $p_{j'}(e) = p_{j-1}(e)$, $n_{j'}(w) = n_{j-1}(w)$, $n_{j'}(w') = n_{j-1}(w')$, $N_{j'}(w) = N_{j-1}(w)$, $N_{j'}(w') = N_{j-1}(w')$ et $M_{j'}(w) = M_{j'}(w') = M_{j-1}(v)$. Ainsi, on sait qu'aucune des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4, \mathcal{M}'_5$ ne peut être appliquée sur l'arête e à l'étape j. Par conséquent, pour tout sommet $w \in V(G)$, $M_i(w) = M_i(v)$. □

Ainsi, si on connaît une borne B sur le diamètre de $\mathbf{G}$, l'algorithme $\mathcal{M}'$ permet de détecter que l'exécution de l'algorithme $\mathcal{M}$ sous-jacent est terminé. En effet, une fois qu'un sommet a un rayon de confiance supérieur ou égal à $B+1$, il sait que tous les sommets de $\mathbf{G}$ ont la même boîte-aux-lettres et qu'ils ont leurs numéros et vues locales finaux.

Si on sait que le graphe $\mathbf{G}$ est minimal pour les revêtements, on sait alors d'après la Proposition 3.28 que tous les sommets ont un identifiant unique et dans ce cas là, le sommet dont le numéro est 1 peut prendre l'étiquette élu et diffuser l'information. On a par conséquent montré le théorème suivant.

Théorème 3.49 *Pour tout entier B, il existe un algorithme d'élection et un algorithme de nommage avec détection de la terminaison utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré pour la famille des graphes minimaux pour les revêtements dont le diamètre est bornée par B.*

Ainsi, comme dans le modèle considéré dans le Chapitre 2, il n'est pas nécessaire de connaître la taille pour pouvoir élire dans un graphe $\mathbf{G}$ que l'on sait minimal pour les revêtements : une borne sur la taille ou le diamètre est suffisante.

Algorithme Effectif d'Élection

On montre ici qu'en connaissant une borne serrée sur la taille d'un graphe $\mathbf{G}$, l'algorithme $\mathcal{M}'$ permet ou bien de résoudre le problème de l'élection sur $\mathbf{G}$, ou bien de détecter que $\mathbf{G}$ n'est pas minimal pour les revêtements.

FIG. 13 – Le graphe C_6 qui n'est pas minimal pour les revêtements est un sous-graphe du graphe C'_6 qui est minimal pour les revêtements.

Théorème 3.50 *Pour tout entier B, il existe un algorithme effectif d'élection et de nommage avec détection de la terminaison utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré pour la classe des graphes* $\mathbf{G}$ *tels que* $V(G) \leq B < 2|V(G)|$.

Preuve : On sait d'après la Proposition 3.28 et le Lemme 3.48 qu'avec la connaissance d'une borne serrée B sur la taille d'un graphe $\mathbf{G}$, toute exécution de $\mathcal{M}'$ sur $\mathbf{G}$ permet de construire un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est un revêtement de $\mathbf{H}$. Si $\mathbf{G}$ est un revêtement propre de $\mathbf{H}$, on sait d'après la Proposition 3.5 que $2|V(H)| \leq |V(G)|$. Puisque $|V(G)| \leq B < 2|V(G)$, on sait que si $\mathbf{G}$ est un revêtement propre de $\mathbf{H}$, $2|V(H)| \leq B$ et que si $\mathbf{G}$ est isomorphe à $\mathbf{H}$, $B < 2|V(G)| = 2|V(H)|$. Par conséquent, il suffit de tester si la taille du graphe $\mathbf{H}$ construit à partir de l'étiquetage final obtenu après l'exécution de $\mathcal{M}'$ sur $\mathbf{G}$ vérifie l'inégalité $2|V(H)| \leq B$. Si l'inégalité est vérifiée, alors le graphe $\mathbf{G}$ n'est pas minimal pour les revêtements, et dans le cas contraire, le graphe $\mathbf{H}$ est isomorphe à $\mathbf{G}$ et on peut donc résoudre l'élection et le nommage avec détection de la terminaison sur $\mathbf{G}$. □

Comme dans le cas des calculs locaux sur les étoiles fermées, on remarque que si la borne sur la taille de $\mathbf{G}$ n'est pas serrée, il n'existe pas d'algorithme utilisant des calculs locaux sur les arêtes étiquetées avec connaissance initiale du degré qui permet de résoudre le problème de l'élection sur $\mathbf{G}$ ou de détecter que $\mathbf{G}$ n'est pas minimal pour les revêtements.

On montre maintenant que si initialement les sommets ne connaissent pas leurs degrés, même en connaissant la taille n du graphe $\mathbf{G}$, si celle-ci n'est pas un nombre premier (sinon, d'après la Remarque 3.30, on peut élire dans la famille des graphes de taille n puisque ceux-ci sont tous minimaux pour les revêtements), il n'existe pas d'algorithme utilisant des calculs locaux sur les arêtes étiquetées qui permet de résoudre le problème de l'élection sur $\mathbf{G}$ ou de détecter que $\mathbf{G}$ n'est pas minimal pour les revêtements.

Pour tout entier n qui n'est pas premier, soit C_n le cycle de taille n et C'_n le graphe obtenu en choisissant un sommet quelconque du cycle de taille n et en le reliant à tous les autres sommets. Sur la Figure 13, on a représenté les graphes C_6 et C'_6.

On suppose qu'il existe un algorithme effectif d'élection $\mathcal{R}$ pour la classe des graphes de taille n utilisant des calculs locaux sur les arêtes étiquetées sans connaissance initiale du degré. Puisque C_n n'est pas minimal pour les revêtements, il existe une exécution ρ de $\mathcal{R}$ sur C_n telle que dans la configuration finale de ρ, chaque sommet de C_n a un étiquette finale indiquant que C_n n'est pas minimal pour les revêtements. Cependant, C_n est un sous-graphe de C'_n et par conséquent on peut trouver une exécution ρ' sur C'_n dont le préfixe est ρ. Puisqu'une fois que ρ est exécutée sur C'_n, tous les sommets de C'_n ont une étiquette finale, ces étiquettes ne sont pas modifiées par la suite et par conséquent, il existe une exécution de $\mathcal{R}$ sur C'_n telle que dans la configuration finale, tous les sommets de C'_n ont conclu que le graphe C'_n n'est pas minimal pour les revêtements, ce qui est faux.

Dans le Théorème 3.50, on suppose que chaque sommet connaît initialement son degré ainsi qu'une borne serrée sur la taille du graphe. On a donc montré qu'on ne pouvait pas affaiblir le théorème en ne conservant qu'une seule de ces hypothèses.

3.7 Conclusion et Perspectives

Dans ce chapitre, on a montré l'équivalence entre les calculs locaux sur les arêtes étiquetées et les calculs locaux cellulaires sur les arêtes étiquetées (Proposition 3.20). Notre algorithme de simulation utilise le fait que les arêtes du graphe peuvent être étiquetées et réétiquetées. On va voir dans les chapitres suivants qu'il n'existe pas de théorème similaire lorsqu'on ne peut pas modifier les étiquettes des arêtes.

On a ensuite présenté une caractérisation des graphes admettant un algorithme d'élection (ou de nommage) utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées (Théorème 3.29). Cette caractérisation s'exprime à l'aide des revêtements de graphes pouvant avoir des arêtes multiples.

Ce résultat nous permet de penser qu'il est possible d'utiliser les techniques présentées dans [GM03, GMM04] afin de caractériser les classes de graphes qui peuvent être reconnues par des calculs locaux (cellulaires) sur les arêtes étiquetées avec ou sans connaissance initiale (la terminaison est alors implicite). Il semble que lorsqu'aucune connaissance initiale n'est disponible, les classes de graphes simples reconnaissables dans ce modèle doivent vérifier les mêmes propriétés que celles présentées dans [GMM04]. En revanche, lorsqu'on dispose d'une connaissance initiale, on peut reconnaître strictement moins de classes de graphes en utilisant des calculs locaux (cellulaires) sur les arêtes étiquetées qu'avec des calculs locaux sur les étoiles fermées.

Dans la Section 3.6, on a étudié l'importance de la connaissance initiale du degré. On a montré que le modèle des calculs locaux (cellulaires) avec connaissance initiale du degré avait la même puissance de calcul que les calculs locaux sur les étoiles ouvertes (Proposition 3.42), que le modèle d'Angluin et que le modèle où la communication entre deux processus est synchrone (Section 3.6.1). On a aussi montré qu'avec la connaissance initiale du degré, la connaissance initiale d'une borne sur le diamètre du graphe permettait de résoudre élection et nommage dans un graphe minimal pour les revêtements (Théorème 3.49) et que la connaissance d'une borne serrée sur la taille du graphe permettait d'obtenir un algorithme effectif de nommage et d'élection (Théorème 3.50).

Ces résultats nous permettent de penser que les techniques utilisées dans [GM02, MT00] peuvent être étendus aux calculs locaux (cellulaires) sur les arêtes étiquetées avec

connaissance initiale du degré afin de caractériser les classes de graphes qui admettent un algorithme universel d'élection dans ce modèle et plus généralement de caractériser les fonctions qui peuvent être calculées avec détection de la terminaison. En revanche, il ne semble pas possible d'obtenir des caractérisations aussi élégantes que celles présentées dans [GM02, MT00] si les sommets ne connaissent pas initialement leur degré.

Par ailleurs, même si chaque sommet connaît initialement son degré, l'algorithme d'énumération auto-stabilisant de Godard présenté dans [God02b] ne semble pas pouvoir être étendu aux modèles considérés dans ce chapitre. Contrairement au modèle étudié dans le Chapitre 2, dans le modèle des calculs locaux sur les arêtes étiquetées, un sommet ne peut pas forcément se rendre compte que l'information dont il dispose à propos de ses voisins (sa vue locale) est erronnée.

Chapitre 4

Calculs Locaux Cellulaires sur les Étoiles

Sommaire

4.1 Introduction

Dans ce chapitre, on étudie les *calculs locaux cellulaires sur les étoiles*. Contrairement aux modèles étudiés dans les Chapitres 2 et 3, dans les graphes considérés dans ce chapitre, seuls les sommets peuvent être étiquetés. Dans ce modèle, en un pas de calcul, un sommet peut modifier son état en fonction de son propre état et de l'état de ses voisins. Les calculs réalisés en utilisant uniquement ce type de réétiquetage sont appelés *calculs locaux cellulaires sur les étoiles*. La différence avec les calculs locaux sur les étoiles ouvertes est que les arêtes du graphe ne peuvent pas être étiquetées (et réétiquetées) et on va voir

que le modèle obtenu a une puissance de calcul strictement plus faible que le modèle des calculs locaux sur les étoiles ouvertes.

Dans ce chapitre, on présente principalement des résultats obtenus par Boldi et al. [BCG+96]. Il faut cependant noter que les méthodes utilisées dans ce chapitre sont différentes de celles de Boldi et al. Les méthodes développées ici sont inspirées des résultats de Mazurkiewicz [Maz97] ainsi que des résultats de Godard et Métivier [GM02]. L'intérêt de cette étude est de présenter dans ce mémoire une présentation complète des résultats existants pour les différents modèles de calculs locaux. Il faut aussi noter que l'algorithme d'énumération qu'on présente dans la Section 4.4 nécessite que chaque sommet ait une mémoire de taille polynomiale en la taille du graphe, alors que l'algorithme de Boldi et al. nécessite une mémoire de taille exponentielle. Par ailleurs, les résultats portant sur les familles de graphes dans la Section 4.5 utilisent le fait que le graphe sur lequel est exécuté l'algorithme est un graphe simple, alors que dans [BCG+96], Boldi et al. ne font pas cette hypothèse : les méthodes et résultats présentés sont donc légèrement différents.

4.1.1 Caractérisations

Dans le modèle des calculs locaux cellulaires sur les arêtes non-étiquetées, on présente la caractérisation de Boldi et al. [BCG+96] des graphes admettant un algorithme de nommage et des graphes admettant un algorithme d'élection. Contrairement aux modèles étudiés dans les Chapitres 2 et 3, les deux problèmes ne sont pas équivalents dans le modèle considéré dans ce chapitre. Ces caractérisations sont basées sur la notion de *fibrations* dont les propriétés ont été étudiées par Boldi et Vigna dans [BV02a].

Dans le modèle étudié dans ce chapitre, un graphe admet un algorithme de nommage si et seulement s'il est minimal pour les fibrations discrètes (Théorème 4.22). L'algorithme d'énumération qu'on présente dans la Section 4.4 est adapté de l'algorithme de Mazurkiewicz et est très différent de l'algorithme de Boldi et al. utilisé dans [BCG+96]. Dans ce modèle, un graphe admet un algorithme d'élection si et seulement s'il est minimal pour les fibrations discrètes non-triviales (Théorème 4.33).

On étudie ensuite l'importance des connaissances initiales dans ce modèle. On montre qu'il suffit de connaître une borne sur le diamètre pour pouvoir nommer dans un graphe minimal pour les fibrations discrètes (Théorème 4.28). De même, la connaissance d'une borne sur le diamètre est suffisante pour pouvoir élire dans un graphe minimal pour les fibrations discrètes non-triviales (Théorème 4.34). Par ailleurs, la connaissance d'une borne serrée sur la taille permet d'obtenir des algorithmes effectifs de nommage et d'élection (Théorèmes 4.36 et 4.37).

4.1.2 Travaux Liés

Les résultats présentés dans ce chapitre correspondent aux résultats obtenus par Boldi et al [BCG+96] et par Boldi et Vigna [BV99, BV01] dans le modèle *entrelacé*, i.e., deux sommets voisins ne peuvent pas modifier leurs états simultanément. Ces auteurs ont aussi étudié une variante synchrone, où lors d'une étape de calcul, tous les sommets du graphe modifient leurs états en fonction de l'état de tous leurs voisins.

Dans [RFH72], Rosenstiehl et al. ont introduit ce type de modèle synchrone sous le nom de *graphes intelligents*. Leur modèle est assez proche du modèle des automates cellulaires,

puisqu'il est synchrone et que chaque processus est une machine à état fini ; cependant, le système de communication est modélisé par un graphe fini et non par une structure régulière infinie. Le problème de l'élection a été étudié dans ce cadre pour certaines classes de graphes planaires par Nichitiu et al. [NMR01, NPR04].

4.2 Fibrations

Dans ce chapitre, on a besoin de considérer des graphes dirigés sans boucle afin de pouvoir exprimer des conditions nécessaires et suffisantes pour qu'un graphe admette un algorithme de nommage ou d'élection utilisant des calculs locaux cellulaires sur les étoiles non-étiquetées.

Les fibrations sont les homomorphismes de graphes dirigés qui préservent le voisinage sortant de chaque sommet. Les fibrations ont été introduites par Boldi et al. [BCG$^+$96] pour étudier le problème de l'élection dans plusieurs modèles, dont le modèle considéré dans ce chapitre. Les définitions et propriétés présentées ici sont dues à Boldi et Vigna [BV02a].

Une fibration est un homomorphisme de graphe étiqueté qui induit une bijection entre les arcs entrants d'un sommet et les arcs entrants de son image.

Définition 4.1 *Un graphe dirigé D est* fibré *sur un graphe dirigé D' à travers un homomorphisme $\gamma\colon D \to D'$ si pour tout arc $a' \in A(D')$ et pour tout sommet $v \in \gamma^{-1}(t(a'))$, il existe un unique arc $a \in A(D)$ tel que $t(a) = v$ et $\varphi(a) = a'$.*

On dit alors que l'homomorphisme γ est une fibration *de D dans D'. Un graphe dirigé D est* fibré proprement *sur D' si γ n'est pas un isomorphisme.*

Naturellement, un graphe dirigé étiqueté (D, λ) est fibré sur un graphe dirigé étiqueté (D', λ') à travers γ si D est fibré sur D' à travers γ et si γ conserve l'étiquetage.

Puisqu'on ne considère que des graphes dirigés fortement connexes, toute fibration est un homomorphisme surjectif.

Proposition 4.2 *Si un graphe dirigé $\mathbf{D}$ est fibré sur un graphe dirigé $\mathbf{D}'$ fortement connexe à travers un homomorphisme γ, alors γ est surjectif.*

Preuve : On considère deux graphes dirigés fortement connexes $\mathbf{D}$ et $\mathbf{D}'$ tels que $\mathbf{D}$ soit fibré sur $\mathbf{D}'$ à travers une fibration γ.

On considère un arc $a' \in A(D')$ et un arc $a \in A(D)$ tels que $\gamma(a) = a'$. Puisque γ est un homomorphisme, on sait que $\gamma(s(a)) = s(a')$ et $\gamma(t(a)) = t(a')$. Ainsi, pour tout arc $a' \in \gamma(A(D))$, $s(a')$ et $t(a')$ appartiennent à $\gamma(V(D))$.

On considère maintenant un sommet $v' \in \gamma(V(D))$ et un arc $a' \in A(D')$ tel que $t(a') = v'$. On considère un sommet $v \in \gamma^{-1}(v')$. Puisque γ est une fibration, il existe un arc $a \in A(D)$ tel que $t(a) = v$ et $\varphi(a) = a'$. Par conséquent, pour tout arc $a' \in A(D)$, si $t(a') \in \gamma(V(D))$, alors $a' \in \gamma(A(D))$.

Puisque le graphe dirigé $\mathbf{D}'$ est fortement connexe, l'homomorphisme γ est donc surjectif. □

On ne considère dans ce chapitre que des graphes dirigés sans boucle fortement connexes. En particulier si un graphe dirigé $\mathbf{D}$ est fibré sur un graphe dirigé sans boucle $\mathbf{D}'$,

pour tout arc $a \in A(D)$, on sait que $\gamma(s(a)) \neq \gamma(t(a))$ et cela implique que $\mathbf{D}$ ne contient pas de boucle. On définit maintenant les graphes minimaux pour les fibrations discrètes.

Définition 4.3 *Un graphe dirigé fortement connexe sans boucle* $\mathbf{D}$ *est* minimal pour les fibrations discrètes *si* $\mathbf{D}$ *n'est fibré proprement sur aucun autre graphe dirigé sans boucle fortement connexe.*

Un graphe simple $\mathbf{G}$ *est* minimal pour les fibrations discrètes *si* $Dir(G)$ *est minimal pour les fibrations discrètes.*

On définit maintenant les fibrations non-triviales qui sont les fibrations telles que l'image inverse de chaque sommet n'est pas réduite à un singleton.

Définition 4.4 *Si un graphe dirigé* $\mathbf{D}$ *est fibré sur un graphe dirigé fortement connexe* $\mathbf{D}'$ *à travers une fibration* γ*, pour tout sommet* $v \in V(D')$*, l'ensemble* $\gamma^{-1}(v)$ *est* la fibre *de* v*. La fibre d'un sommet* v *est* triviale *si elle est réduite à un singleton, i.e.,* $|\gamma^{-1}(v)| = 1$ *et* non-triviale *sinon.*

La fibration γ *est* non-triviale *si toutes ses fibres sont non-triviales. On dit alors que* $\mathbf{D}$ *est* non-trivialement *fibré sur* $\mathbf{D}'$*.*

Les graphes dirigés minimaux pour les fibrations discrètes non-triviales sont les graphes qui ne sont pas fibrés non-trivialement sur un autre graphe dirigé.

Définition 4.5 *Un graphe dirigé fortement connexe sans boucle* $\mathbf{D}$ *est* minimal pour les fibrations discrètes non-triviales *si* $\mathbf{D}$ *n'est fibré non-trivialement sur aucun autre graphe dirigé sans boucle fortement connexe.*

Un graphe simple $\mathbf{G}$ *est* minimal pour les fibrations discrètes non-triviales *si* $Dir(G)$ *est minimal pour les fibrations discrètes non-triviales.*

Dans la proposition suivante, on montre le lien entre les revêtements et les fibrations. On remarque que si $\mathbf{G}$ et $\mathbf{H}$ sont deux graphes non-dirigés, ils ne contiennent pas de boucle et par conséquent, ni $Dir(\mathbf{G})$, ni $Dir(\mathbf{H})$ ne contient de boucle.

Proposition 4.6 *Pour tous graphes étiquetés* $\mathbf{G}, \mathbf{H}$*, si* $\mathbf{G}$ *est un revêtement de* $\mathbf{H}$*, alors* $Dir(\mathbf{G})$ *est fibré sur* $Dir(\mathbf{H})$*.*

Preuve : On considère deux graphes $\mathbf{G} = (G, \lambda), \mathbf{H} = (H, \eta)$. On sait que $Dir(\mathbf{G})$ et $Dir(\mathbf{H})$ sont deux graphes dirigés fortement connexes symétriques. De plus, pour toute arête $e \in E(G)$ dont les extrémités sont u et u', il existe deux arcs $a_{e,u,u'}, a_{e,u',u} \in A(Dir(\mathbf{G}))$ tels que $s(a_{e,u,u'}) = t(a_{e,u',u}) = u$ et $Sym(a_{e,u,u'}) = a_{e,u',u}$. De même, pour toute arête $f \in E(H)$ dont les extrémités sont v et v', il existe deux arcs $a_{f,v,v'}, a_{f,v',v} \in A(Dir(\mathbf{H}))$ tels que $s(a_{f,v,v'}) = t(a_{f,v',v}) = v$ et $Sym(a_{f,v,v'}) = a_{f,v',v}$.

Si $\mathbf{G}$ est un revêtement de $\mathbf{H}$ à travers un homomorphisme γ, on considère l'homomorphisme φ de $Dir(\mathbf{G})$ dans $Dir(\mathbf{H})$ tel que pour tout $u \in V(Dir(\mathbf{G})) = V(G)$, $\varphi(u) = \gamma(u)$, et pour tout arc $a_{e,u,u'} \in A(G)$, $\varphi(a_{e,u,u'}) = a_{\gamma(e),\gamma(u),\gamma(u')}$. Pour tout arc $a_{f,v,v'} \in A(Dir(\mathbf{H}))$, on sait qu'il existe une arête $f \in E(H)$ incidente à v'. Par conséquent, pour tout sommet $u' \in \varphi^{-1}(t(a_{f,v,v'})) = \gamma^{-1}(v')$, on sait qu'il existe une unique arête e incidente à u' telle que $\gamma(e) = f$. Si on note u l'extrémité de e différente de u', on sait que $\gamma(u) = v$ puisque γ est un homomorphisme. Ainsi, $\varphi(a_{e,u,u'}) = a_{f,v,v'}$ et puisqu'il existe une unique arête e incidente à u' telle que $\gamma(e) = f$, on sait qu'il existe un unique arc $a \in A(Dir(\mathbf{G}))$ tel que $t(a) = u'$ et $\varphi(a) = a_{f,v,v'}$. Le graphe dirigé $Dir(\mathbf{G})$ est donc fibré sur $Dir(\mathbf{H})$ à travers l'homomorphisme φ. □

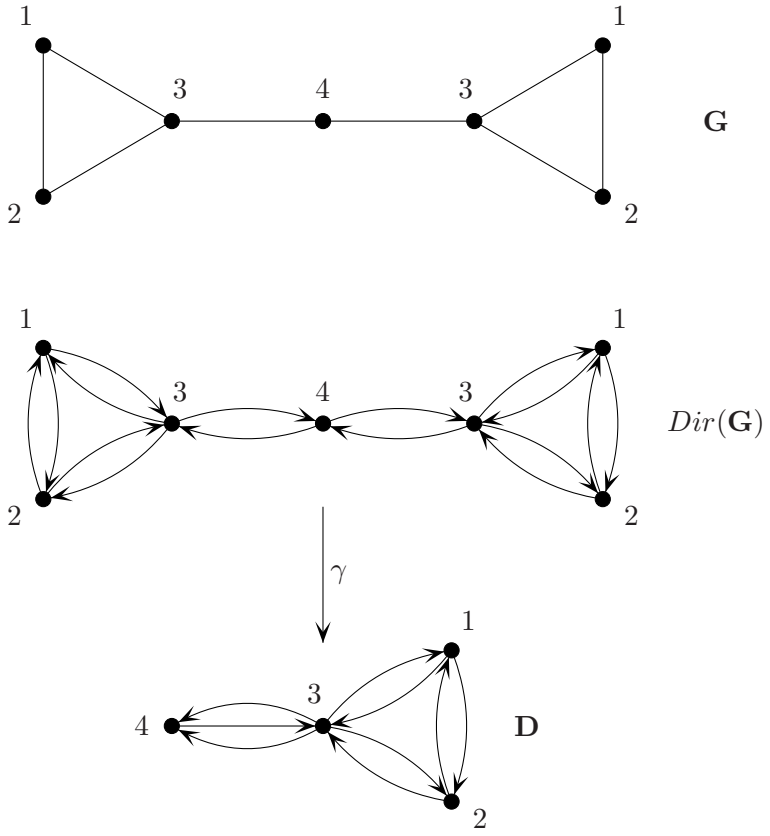


Fig. 14 – Le graphe $Dir(\mathbf{G})$ est fibré proprement sur le graphe $\mathbf{D}$ à travers un homomorphisme γ qui envoie chaque sommet étiqueté i de $Dir(\mathbf{G})$ sur l'unique sommet étiqueté i de $\mathbf{D}$.

Par conséquent, pour tout graphe $\mathbf{G}$, si $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes, alors $\mathbf{G}$ est minimal pour les revêtements. Il faut cependant noter qu'il existe des graphes minimaux pour les revêtements qui ne sont pas minimaux pour les fibrations discrètes ou pour les fibrations discrètes non-triviales. De tels exemples sont présentés sur les Figures 14 et 15.

Exemple 4.7 *Le graphe* $\mathbf{G}$ *de la Figure 14 et le graphe* $\mathbf{H}$ *de la Figure 15 sont minimaux pour les revêtements puisqu'ils ont un nombre premier de sommets.*

Le graphe $Dir(\mathbf{G})$ *est fibré proprement sur le graphe dirigé sans boucle* $\mathbf{D}$ *et n'est donc pas minimal pour les fibrations discrètes. Cependant, on note que le graphe* $Dir(\mathbf{G})$ *est minimal pour les fibrations discrètes non-triviales.*

Le graphe $Dir(\mathbf{H})$ *est fibré non-trivialement sur le graphe dirigé sans boucle* $\mathbf{D}'$ *et n'est donc pas minimal pour les fibrations discrètes non-triviales.*

Le graphe $\mathbf{D}$ *de la Figure 14 et le graphe* $\mathbf{D}'$ *de la Figure 15 sont minimaux pour les fibrations discrètes.*

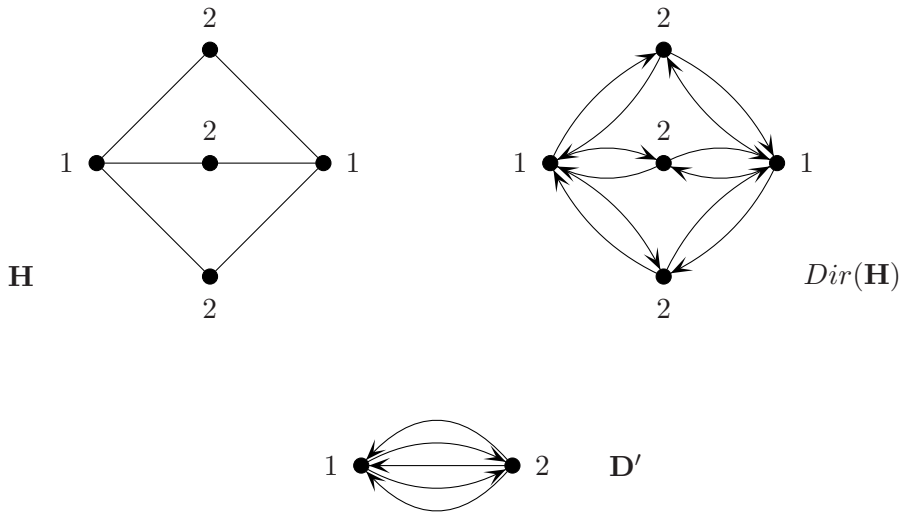


FIG. 15 – Le graphe $Dir(\mathbf{H})$ est fibré non-trivialement sur le graphe $\mathbf{D}'$ à travers un homomorphisme γ qui envoie chaque sommet étiqueté i de $Dir(\mathbf{H})$ sur l'unique sommet étiqueté i de $\mathbf{D}'$.

La notion de coloration semi-régulière permet de caractériser les graphes simples non-étiquetés qui ne sont pas minimaux pour les fibrations discrètes en termes de colorations de graphes. Un étiquetage ℓ d'un graphe simple non-étiqueté est une coloration régulière si deux sommets voisins ont des couleurs distinctes et si deux sommets qui ont la même couleur ont les mêmes couleurs dans leur voisinage avec la même multiplicité.

Définition 4.8 *Une* coloration semi-régulière *d'un graphe simple G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un stable,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ est un graphe biparti semi-régulier.*

Dans la proposition suivante, on caractérise les graphes simples non-étiquetés qui sont minimaux pour les fibrations discrètes à l'aide de la notion de coloration semi-régulière. On rappelle qu'une coloration ℓ est dite propre si $|\ell(V(G))| < |V(G)|$.

Proposition 4.9 *Un graphe simple non-étiqueté G est minimal pour les fibrations discrètes si et seulement si il n'admet pas de coloration semi-régulière propre.*

Un graphe simple non-étiqueté G est minimal pour les fibrations discrètes non-triviales si et seulement si pour toute coloration semi-régulière ℓ de G, il existe un sommet v de G tel que $|\ell^{-1}(\ell(v))|$.

Preuve : On considère un graphe dirigé fortement connexe sans boucle D tel que $Dir(G)$ est fibré sur D à travers une fibration γ. On va montrer que γ est une coloration semi-régulière de G et que l'ensemble des couleurs utilisés est $V(D)$. Puisque D ne contient pas de boucle, pour toute arête $\{u, u'\} \in E(G)$, $\gamma(u) \neq \gamma(u')$. Pour tous $v, v' \in V(D')$, on note $d_{(v',v)}$ le nombre d'arcs $a \in A(D')$ tels que $s(a) = v'$ et $t(a) = v$. Pour tout sommet $u \in \gamma^{-1}(v)$, il existe exactement $d_{(v',v)}$ voisins u' de u tels que $\gamma(u') = v'$. Par conséquent, le graphe $G[v, v']$ est un graphe $((d_{(v',v)}, d_{(v,v')})$-régulier.

Ainsi, si G n'est pas minimal pour les fibrations discrètes, il existe un graphe dirigé fortement connexe sans boucle tel que G est fibré proprement sur D à travers une fibration γ et donc $|\gamma(V(G))| < |V(G)|$. Ainsi, γ est une coloration semi-régulière propre de G.

Si G n'est pas minimal pour les fibrations discrètes non-triviales, il existe un grape dirigé fortement connexe sans boucle tel que G est fibré non-trivialement sur D à travers une fibration γ. Ainsi, γ est une coloration semi-régulière de G telle que pour tout $u \in V(G)$, $|\gamma^{-1}(\gamma(u))| > 1$.

Réciproquement, étant donnée une coloration semi-régulière ℓ d'un graphe G, on va définir un graphe D' tel que $Dir(G)$ soit fibré sur D'. On pose $V(D') = \ell(V(G))$. Pour tous $v, v' \in V(D')$, le graphe $G[v, v']$ est semi-régulier et il existe donc un entier $d_{(v',v)}$ tel que chaque sommet $u \in \ell^{-1}(v)$ ait exactement $d_{(v',v)}$ voisins dans $\ell^{-1}(v')$. L'ensemble $A(D')$ est construit de telle sorte que pour tous $v, v' \in V(D')$, $A(D')$ contient alors exactement $d_{(v',v)}$ arcs $a'_{1,(v',v)}, \ldots, a'_{d_{(v',v)},(v',v)}$ tels que $s(a'_{i,(v',v)}) = v'$ et $t(a'_{i,(v',v)}) = v$ pour tout $i \in [1, d_{(v',v)}]$. On définit γ de la manière suivante. Pour tout $u \in V(Dir(G))$, $\gamma(u) = \ell(u)$ et pour tout $v' \in V(D')$, considère l'ensemble $A_{u,v'} \in A(Dir(G))$ des arcs a tels que $t(a) = u$ et $\ell(s(a)) = v$. On sait que $A_{u,v'}$ contient exactement $d_{(v',\ell(u))}$ arcs $a_{1,(v',u)}, \ldots, a_{d_{(v',\ell(u))},(v',u)}$ et on définit $\gamma(a_{i,(v',u)}) = a'_{i,(v',\ell(u))}$ pour tout $i \in [1, d_{(v',\ell(u))}]$. Il est clair que pour tout $v \in V(D')$, pour tout arc $a'_{i,(v,v')} \in A(D')$, pour tout $u \in \gamma^1(v)$, il existe un unique arc $a_{i,(u,v')} \in A(Dir(G))$ tel que $t(a_{i,(u,v')}) = u$. Par conséquent, puisqu'il n'existe pas deux sommets adjacents de G qui ont la même image par ℓ, G est fibré sur γ à travers D.

Si ℓ est une coloration semi-régulière propre de G, alors $|V(D)| = |\ell(V(G))| < |V(G)|$ et G n'est donc pas isomorphe à D. Ainsi, G est fibré proprement sur D à travers γ.

Si pour tout $u \in V(G)$, il existe $u' \neq u$ tel que $\ell(u) \neq \ell(u')$, alors pour tout $v \in V(D) = \ell(V(G))$, $|\gamma^{-1}(v)| > 2$ et G est donc fibré non-trivialement sur D à travers γ. □

4.3 Calculs Locaux Cellulaires sur les Étoiles

Dans cette partie, on présente les définitions formelles des calculs locaux cellulaires sur les étoiles, puis on étudie leurs relations avec les fibrations. Le modèle des calculs locaux cellulaires sur les étoiles a été introduit ét étudié par Boldi et al. [BCG+96].

4.3.1 Définitions

On définit d'abord les calculs locaux cellulaires sur les étoiles pour les graphes simples, puis on décrit le modèle équivalent pour les graphes dirigés. On rappelle qu'on ne considère que des graphes simples et des graphes dirigés où seuls les sommets peuvent être étiquetés.

Sur les graphes simples où les arêtes ne sont pas étiquetées, la notion de calculs locaux cellulaires sur les étoiles correspond aux calculs locaux sur les étoiles ouvertes.

Définition 4.10 *Une relation de réétiquetage est cellulaire et localement engendrée sur les étoiles si la condition suivante est satisfaite. Pour tous graphes simples (G, λ), (G, λ'), (H, η), (H, η'), pour tout sommets $v \in V(G)$ et $w \in V(H)$ tels qu'il existe un isomorphisme $\varphi : B_G(v) \to B_G(w)$, si les conditions suivantes sont vérifiées :*

1. *$\lambda(v) = \eta(\varphi(v))$ et $\lambda'(v) = \eta'(\varphi(v))$,*

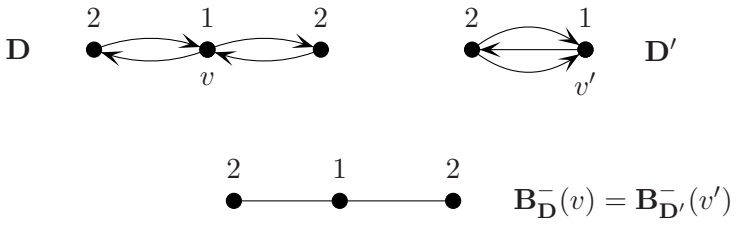


FIG. 16 – Les étoiles entrantes des sommets $v \in V(D)$ et $v' \in V(D')$ sont isomorphes.

2. pour tout $v' \in N_G(v)$, $\lambda(v) = \eta(\varphi(v))$,

3. pour tout $v' \neq v$, $\lambda'(v') = \lambda(v')$,

4. pour tout $w' \neq w$, $\eta'(w') = \eta(w')$,

alors $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ *si et seulement si* $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

Afin de pouvoir obtenir un lemme de relèvement similaire à celui d'Angluin [Ang80] dans le cadre des calculs locaux cellulaires sur les étoiles, on a aussi besoin de définir les calculs locaux cellulaires sur les étoiles sur les graphes dirigés. Pour cela, on définit l'*étoile entrante* d'un sommet v dans un graphe dirigé $\mathbf{D}$.

Informellement, l'étoile entrante d'un sommet v dans un graphe $\mathbf{D}$ est une étoile dont le centre est étiqueté comme v et pour chaque arc a dont la cible v, il existe une feuille v_a de l'étoile étiquetée comme la source de a.

Définition 4.11 *Pour tout graphe dirigé* $\mathbf{D} = (D, \lambda)$ *et pour tout sommet* $v \in V(D)$, *l'*étoile entrante *de* v *dans* $\mathbf{D}$, *noté* $B^-_{\mathbf{D}}(v) = (B^-_D(v), \eta)$ *est le graphe simple défini de la manière suivante.*

- $V(B^-_D(v)) = \{v\} \cup \{v_a \mid \exists a \in A(G), t(a) = v\}$,
- $E(B^-_D(v)) = \{\{v, v_a\} \mid \exists v_a \in V(B^-_D(v))\}$,
- $\eta(v) = \lambda(v)$ *et pour tout* $v_a \in V(B^-_D(v))$, $\eta(v_a) = \lambda(s(a))$.

Des exemples d'étoiles entrantes est présenté sur la Figure 16. On remarque sur cet exemple que deux sommets peuvent avoir des vues entrantes isomorphes sans que les graphes induits par chaque sommet et son voisinage ne soient isomorphes. Il est facile de voir que pour tout graphe simple $\mathbf{G}$ et pour tout sommet $v \in V(G)$, $\mathbf{B}^-_{Dir(\mathbf{G})}(v)$ est isomorphe à $\mathbf{B}_{\mathbf{G}}(v)$.

Dans la proposition suivante, on montre que les fibrations préservent les étoiles entrantes.

Proposition 4.12 *Si un graphe dirigé* $\mathbf{D}_1$ *est fibré sur un graphe dirigé* $\mathbf{D}_2$ *à travers une fibration* γ, *alors pour tout* $v \in V(D_1)$, *les étoiles entrantes* $\mathbf{B}^-_{\mathbf{D}_1}(v)$ *et* $\mathbf{B}^-_{\mathbf{D}_2}(\gamma(v))$ *sont isomorphes.*

Preuve : On considère un graphe dirigé $\mathbf{D}_1$ qui est fibré sur un graphe $\mathbf{D}_2$ à travers une fibration γ et un sommet $v \in V(D_1)$. On définit un homomorphisme φ de $\mathbf{B}^-_{\mathbf{D}_1}(v) = (\mathbf{B}^-_{D_1}(v), \eta_1)$ dans $\mathbf{B}^-_{\mathbf{D}_2}(\gamma(v)) = (\mathbf{B}^-_{D_2}(\gamma(v)), \eta_2)$ de la manière suivante : $\varphi(v) = \gamma(v)$ et pour tout $v_a \in V(B^-_{D_1}(v))$, $\varphi(v_a) = v_{\gamma(a)}$. Puisque γ est une fibration, on sait que pour tout $a' \in A(D_2)$ tel que $t(a') = \gamma(v)$, il existe un unique $a \in A(D_1)$ tel que $\gamma(a) = a'$ et $t(a') = v$. Par conséquent, pour tout $v_{a'} \in V(B^-_{D_2}(v)) \setminus \{v'\}$, il existe un unique $v_a \in V(B^-_{D_1}(v))$ tel

que $\varphi(v_a) = v_{a'}$. De plus, $\eta_1(v) = \lambda_1(v) = \lambda_2(\gamma(v)) = \eta_2(\gamma(v))$ et pour tout $v_a \in B(v_1)$, $\eta_1(v_a) = \lambda_1(s(a)) = \lambda_2(\gamma(s(a))) = \lambda_2(s(\gamma(a))) = \eta_2(v_{\gamma(a)}) = \eta_2(\varphi(v_a))$. Ainsi, φ est un isomorphisme de $\mathbf{B}^-_{\mathbf{D}_1}(v)$ dans $\mathbf{B}^-_{\mathbf{D}_2}(\gamma(v))$. □

On utilise maintenant la notion d'étoile entrante pour définir les relations de réétiquetages cellulaires et localement engendrées sur les étoiles entrantes.

Définition 4.13 *Une relation de réétiquetage est cellulaire et localement engendrée sur les étoiles entrantes si la condition suivante est satisfaite. Pour tous graphes dirigés (D_1, λ_1), (D'_1, λ'_1), (D_2, λ_2), (D_2, λ'_2), pour tout sommets $v \in V(D_1)$ et $w \in V(D_2)$ tels qu'il existe un isomorphisme $\varphi : B^-_{D_1}(v) \to B^-_{D_2}(w)$, si les conditions suivantes sont vérifiées :*

1. *$\lambda_1(v) = \lambda_2(\varphi(v))$ et $\lambda'_1(v) = \lambda'_2(\varphi(v))$,*
2. *pour tout $v' \in V(B^-_{D_1}(v))$, $\lambda_1(v) = \lambda_2(\varphi(v))$,*
3. *pour tout $v' \neq v$, $\lambda'_1(v') = \lambda_1(v')$,*
4. *pour tout $w' \neq w$, $\lambda'_2(w') = \lambda_2(w')$,*

alors $(D_1, \lambda_1)\ \mathcal{R}\ (D_1, \lambda'_1)$ si et seulement si $(D_2, \lambda_2)\ \mathcal{R}\ (D_2, \lambda'_2)$.

On remarque que toute relation de réétiquetage $\mathcal{R}$ cellulaire localement engendrée sur les étoiles entrantes peut être vue comme une relation de réétiquetage cellulaire localement engendrée sur les étoiles : $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$ si et seulement si $Dir(\mathbf{G})\ \mathcal{R}\ Dir(\mathbf{G}')$.

Réciproquement, on peut transformer une relation de réétiquetage $\mathcal{R}$ celullaire localement engendrée sur les étoiles en une relation de réétiquetage $\mathcal{R}'$ cellulaire localement engendrée sur les étoiles entrantes de la manière suivante. Pour tout graphes simples $(G, \lambda), (G, \lambda')$, pour tous graphes dirigés $(D, \eta), (D, \eta')$ et pour tous sommets $v \in V(G)$ et $w \in V(D)$ tels qu'il existe un isomorphisme $\varphi : B^-_G(v) \to B^-_D(w)$ et tels que les conditions suivantes sont vérifiées

- $\lambda(v) = \eta(\varphi(v))$ et $\lambda'(v) = \eta'(\varphi(v))$,
- pour tout $v' \in N_G(v)$, $\lambda(v) = \eta(\varphi(v))$,
- pour tout $v' \neq v$, $\lambda'(v') = \lambda(v')$,
- pour tout $w' \neq w$, $\eta'(w') = \eta(w')$,

on définit $(D, \eta)\ \mathcal{R}'\ (D, \eta')$ si et seulement si $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$. Il est facile de voir que $\mathcal{R}'$ est bien définie et que $\mathcal{R}'$ est une relation de réétiquetage cellulaire localement engendrée sur les étoiles entrantes. Ainsi, $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$ si et seulement si $Dir(\mathbf{G})\ \mathcal{R}'\ Dir(\mathbf{G}')$ et on peut donc considérer seulement des relations de réétiquetage cellulaire localement engendrée sur les étoiles.

Les *calculs locaux cellulaires sur les étoiles* correspondent indistinctement aux relations de réétiquetages cellulaires localement engendrées sur les étoiles ou aux relations de réétiquetages cellulaires localement engendrées sur les étoiles entrantes.

Une relation de réétiquetage cellulaire localement engendrée sur les étoiles peut être décrite par un ensemble récursif de règles de réétiquetage où chaque règle permet de modifier l'étiquette d'un sommet en fonction des étiquettes des voisins de ce sommet. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage cellulaire localement engendrée sur les étoiles. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

4.3.2 Fibrations et Calculs Locaux Cellulaires sur les Étoiles

On présente maintenant le lemme qui met en évidence le lien entre les fibrations et les calculs locaux cellulaires sur les étoiles. C'est l'équivalent du lemme d'Angluin [Ang80] pour les fibrations. Ce lemme a été prouvé par Boldi et al. [BCG+96]

Lemme 4.14 (Lemme de relèvement [BCG+96]) *On considère un graphe dirigé* $\mathbf{D}_1$ *qui est fibré sur un graphe dirigé sans boucle* $\mathbf{D}_2$ *à travers une fibration* γ *et une relation* $\mathcal{R}$ *de réétiquetage cellulaire localement engendrée sur les étoiles. Si* $\mathbf{D}_2\ \mathcal{R}^*\ \mathbf{D}'_2$*, alors il existe* $\mathbf{D}'_1$ *tel que* $\mathbf{D}_1\ \mathcal{R}^*\ \mathbf{D}'_1$ *et* $\mathbf{D}'_1$ *est fibré sur* $\mathbf{D}'_2$ *à travers* γ*.*

Preuve : Il suffit de prouver ce lemme pour un pas de calcul. On considère deux graphes dirigés $\mathbf{D}_1 = (D_1, \lambda_1)$ et $\mathbf{D}_2 = (D_2, \lambda_2)$ tels que (D_1, λ_1) est fibré sur (D_2, λ_2) à travers γ. On considère un pas de réétiquetage qui modifie l'étiquette d'un sommet $v \in V(D_2)$ et on note λ'_2 l'étiquetage de D_2 obtenu après l'application de ce pas de réétiquetage.

D'après la Proposition 4.12, on sait que pour tout $u \in \gamma^{-1}(v)$, $\mathbf{B}^-_{\mathbf{D}_1}(u)$ est isomorphe à $\mathbf{B}^-_{\mathbf{D}_2}(v)$. De plus, puisque $\mathbf{D}_2$ ne contient pas de boucle, pour tous $u, u' \in \gamma^{-1}(v)$, il n'existe pas d'arc a tel que $s(a) = u$ et $t(a) = u'$. On peut donc appliquer le pas de réétiquetage sur chaque sommet $u \in \gamma^{-1}(v)$ et on note λ'_1 l'étiquetage de D_1 ainsi obtenu. On sait que pour tout $u \in \gamma^{-1}(v)$, $\lambda'_1(u) = \lambda'_2(v)$ et que les étiquettes des autres sommets n'ont pas été modifiées. Le graphe (D_1, λ'_1) est donc fibré sur (D_2, λ'_2) à travers γ. □

Le diagramme suivant représente la propriété du Lemme 4.14.

$$\begin{array}{ccc} \mathbf{D}_1 & \xrightarrow{\mathcal{R}^*} & \mathbf{D}'_1 \\ {\scriptstyle \text{fibration}}\downarrow & & \downarrow{\scriptstyle \text{fibration}} \\ \mathbf{D}_2 & \xrightarrow[\mathcal{R}^*]{} & \mathbf{D}'_2 \end{array}$$

4.4 Nommage et Énumération

On s'intéresse aux problèmes d'énumération et du nommage dans le cadre des calculs locaux cellulaires sur les étoiles. On montre d'abord que dans ce modèle, il n'existe pas d'algorithme d'énumération ou de nommage pour un graphe simple $\mathbf{G}$ qui n'est pas minimal pour les fibrations discrètes ; ce qui a été montré par Boldi et al. [BCG+96]. On donne ensuite un algorithme d'énumération pour les graphes minimaux pour les fibrations discretes qui est inspiré de l'algorithme de Mazurkiewicz [Maz97] ; cet algorithme est nouveau et on va voir qu'il nécessite beaucoup moins de mémoire que l'algorithme de Boldi et al. [BCG+96].

4.4.1 Résultats d'Impossibilité pour l'Énumération et le Nommage

Proposition 4.15 ([BCG+96]) *Soit* $\mathbf{G}$ *un graphe simple étiqueté qui n'est pas minimal pour les fibrations discrètes. Il n'existe pas d'algorithme d'énumération ou de nommage pour le graphe* $\mathbf{G}$ *utilisant des calculs locaux cellulaires sur les étoiles.*

Preuve : Soit $\mathbf{D}$ un graphe dirigé étiqueté sans boucle tel que $Dir(\mathbf{G})$ est fibré proprement sur $\mathbf{D}$ à travers une fibration γ. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux

cellulaires sur les étoiles, on considère une exécution de $\mathcal{R}$ sur $\mathbf{D}$. Si cette exécution est infinie sur $\mathbf{D}$, alors d'après le Lemme 4.14, il existe une exécution infinie de $\mathcal{R}$ sur $Dir(\mathbf{G})$ et donc sur $\mathbf{G}$; auquel cas, $\mathcal{R}$ n'est ni un algorithme d'énumération, ni de nommage.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur $\mathbf{D}$ et on considère la configuration finale $\mathbf{D}'$. D'après le Lemme 4.14, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui permet d'atteindre une configuration $\mathbf{G}'$ telle que $Dir(\mathbf{G}')$ est fibré sur $\mathbf{D}'$ à travers γ. Si $\mathbf{G}'$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe un sommet $u \in V(G)$ tel qu'on peut appliquer une règle de $\mathcal{R}$ sur $\mathbf{B}_{\mathbf{G}'}(u)$. Par conséquent, cette règle peut être appliquée sur $\mathbf{B}^-_{Dir(\mathbf{G}')}(u)$ et sur $\mathbf{B}^-_{\mathbf{D}'}(\gamma(u))$ d'après la Proposition 4.12, mais cela signifie que $\mathbf{D}'$ n'est pas une configuration finale de $\mathcal{R}$, ce qui est impossible. Par conséquent, $\mathbf{G}'$ est une configuration finale de $\mathcal{R}$. Mais puisque $Dir(\mathbf{G}')$ n'est pas isomorphe à $\mathbf{D}'$, cela implique qu'il existe deux sommets de G qui ont la même étiquette dans $Dir(\mathbf{G}')$ et donc dans $\mathbf{G}'$. Par conséquent, $\mathcal{R}$ ne permet pas d'attribuer de noms distincts à tous les sommets de G et donc $\mathcal{R}$ n'est pas un algorithme de nommage ou d'énumération pour le graphe $\mathbf{G}$. □

4.4.2 Un Algorithme d'Énumération

On va maintenant décrire un algorithme d'énumération $\mathcal{M}$ très proche de l'algorithme de Mazurkiewicz qui permet de résoudre le problème de l'énumération sur tout graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes.

Durant l'exécution de l'algorithme, chaque sommet v essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. Chaque sommet va ensuite observer les numéros choisis par ses voisins pour construire sa *vue locale*, i.e., les numéros de ses voisins. Ensuite, chaque sommet diffuse dans le réseau son numéro accompagné de son étiquette initiale et de sa vue locale.

Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette $\lambda(u)$ et sa vue locale avec l'étiquette $\lambda(v)$ et la vue locale de v : si l'étiquette de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre similaire à l'ordre utilisé dans l'algorithme de Mazurkiewicz), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire). Ensuite, le sommet u diffuse à nouveau son numéro et sa vue locale dans le réseau. Si un sommet se rend compte que sa vue locale ne correspond pas aux numéros de ses voisins, alors il met à jour sa vue locale et diffuse à nouveau son numéro et sa vue locale. Lorsque l'exécution est terminée, si le graphe $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda\colon V(G) \to L$ est un étiquetage initial, qui ne sera pas modifié par l'algorithme. Lors de l'exécution, chaque sommet va obtenir une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,

– $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ est la *vue locale* du sommet v qui contient des informations sur les voisins de v. À chaque fois que v met à jour sa vue locale, pour chaque numéro n apparaissant dans son voisinage, le couple (n, p) apparaît dans $N(v)$ où p est le nombre de voisins de v ayant l'étiquette n.
– $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

Un Ordre sur les Vues Locales

Comme pour l'algorithme de Mazurkiewicz, on définit un ordre sur les vues locales qui va permettre d'assurer que certaines propriétés seront toujours vérifiées au cours de l'exécution. L'ordre considéré est exactement le même que celui de l'algorithme de Mazurkiewicz, à la différence qu'on définit un ordre sur les ensembles finis de couples d'entiers.

On définit donc un ordre total sur les ensembles finis de couples d'entiers. Pour cela, on considère que $\mathbb{N}^2$ est muni de l'ordre lexicographique usuel : $(n, p) < (n', p')$ si $n < n'$, ou si $n = n'$ et $p < p'$.

Ensuite, étant données deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ distincts, on dit que $N_1 \prec N_2$ si le maximum pour l'ordre lexicographique sur $L \times N$ de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2.

Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre $\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(L \times \mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

Les Règles de Réétiquetage

On décrit maintenant l'algorithme d'énumération grâce à des règles de réétiquetage. La première règle $\mathcal{M}_0$ est une règle spéciale qui permet à chaque sommet v de modifier son étiquette $\lambda(v)$ pour obtenir l'étiquette $(\lambda(v), 0, \emptyset, \emptyset)$.

Les règles sont décrites pour une étoile $B(v_0)$ de centre v_0. L'étiquette d'un sommet $v \in N_G(v_0) \cup \{v_0\}$ avant l'application de la règle est notée $(\lambda(v), n(v), N(v), M(v))$ et on note $(\lambda(v_0), n'(v_0), N'(v_0), M'(v_0))$ l'étiquette de v_0 (la seule étiquette qui peut être modifiée) après l'application de la règle de réétiquetage. Par ailleurs, afin de rendre plus lisible les règles, on ne mentionne pas les différents champs des étiquettes qui ne sont pas modifiés. Les deux règles de l'algorithme sont très proches des règles de l'algorithme de Mazurkiewicz.

La première règle permet non-seulement à un sommet de mettre à jour sa boîte-aux-lettres comme dans l'algorithme de Mazurkiewicz, mais aussi de mettre à jour sa vue locale si celle ci n'est pas correcte. Pour tout sommet $v_0 \in V(G)$, on note $\text{code}(N_G(v_0))$ l'ensemble de couples d'entiers défini de la manière suivante : un couple (n, p) appartient à $\text{code}(N_G(v_0))$ si et seulement si $n > 0$ et il existe exactement p sommets $v_1, \ldots, v_p \in N_G(v_0)$ tels que pour tout $i \in [1, p]$, $n(v_i) = n$.

$\mathcal{M}_1$: **Règle de Diffusion**

Précondition :
- $\exists v \in N_G(v_0)$ tel que $M(v) \setminus M(v_0) \neq \emptyset$ ou $N(v_0) \neq \text{code}(N_G(v_0)$

Réétiquetage :
- $N'(v_0) := \text{code}(N_G(v_0))$;
- $M'(v_0) := (n(v_0), N'(v_0)) \cup \bigcup_{w \in V(B(v_0))} M(w)$.

La seconde règle de l'algorithme permet à un sommet v_0 qui ne peut pas appliquer la première règle de modifier son numéro s'il n'a pas encore choisi son numéro (i.e. $n(v) = 0$) ou s'il sait qu'il existe un sommet dans le graphe qui a le même numéro que lui et qui a une étiquette ou une vue locale plus forte que la sienne. Dans ce cas là, v_0 choisit un nouveau numéro et ajoute à sa boîte-aux-lettres le couple $(n'(v_0), \lambda(v_0), N(v_0))$.

$\mathcal{M}_2$: **Règle de Renommage**

Précondition :
- $\forall v \in N_G(v_0)$, $M(v) = M(v_0)$,
- $N(v_0) = \text{code}(N_G(v_0))$,
- $n(v_0) = 0$ ou $\exists (n(v_0), \ell, N) \in M(v_0)$ tel que $(\lambda(v_0), N(v_0)) \prec (\ell, N)$

Réétiquetage :
- $n'(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
- $M'(v_0) := M(v_0) \cup \{(n'(v_0), \lambda(v_0), N(v_0))\}$.

4.4.3 Correction de l'Algorithme d'Énumération

On considère un graphe simple étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage de l'algorithme $\mathcal{M}$ décrit ci-dessus. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Propriétés Satisfaites lors de l'Exécution

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur le nombre d'étapes, rappelle quelque propriétés simples qui sont toujours satisfaites par l'étiquetage.

Lemme 4.16 *Pour tout sommet $v \in V(G')$, et pour toute étape i,*

1. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
2. $\forall (n, p) \in N_i(v), n > 0, n \neq n_i(v)$ *et* $\exists (n, \ell, N) \in M_i(v)$,
3. $\forall v \in V(G), \forall v \in N_G(v), n_i(v) > 0 \implies n_i(v) \neq n_i(v')$.

Comme l'algorithme de Mazurkiewicz, l'algorithme $\mathcal{M}$ décrit ici a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 4.17 *Pour chaque sommet v et chaque étape i,*
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

De plus, à chaque étape i, il existe un sommet v telle qu'au moins une de ces inégalités (ou inclusions) est stricte pour v.

Preuve : On note v le sommet dont l'étiquette est modifiée lors de la $(i+1)$ème étape de réétiquetage. Pour tout sommet $w \in V(G)$ différent de v, la propriété est trivialement vraie. De plus, quelle que soit la règle appliquée, on a toujours $M_i(v) \subsetneq M_{i+1}(v)$.

Si $n_i(v) \neq n_{i+1}(v)$, alors la règle $\mathcal{M}_2$ a été appliquée sur l'étoile $B(v)$ et on sait que $n_{i+1}(v) = 1 + \max\{n' \mid \exists(n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 4.16, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Pour chaque sommet v tel que $N_i(v) \neq N_{i+1}(v)$, alors la règle $\mathcal{M}_1$ a été appliquée sur l'étoile $B(v)$. Si $N_i(v) = \emptyset$, alors puisque l'ensemble vide est minimal pour l'ordre $\prec$, $N_i(v) \prec N_{i+1}(v)$. Dans le cas contraire, on considère la dernière étape $j < i+1$ lors de laquelle la règle $\mathcal{M}_1$ a été appliquée sur l'étoile $B(v)$: on sait que $N_j(v) = N_i(v)$. Puisqu'on sait que pour chaque sommet $v' \in N_G(v)$, $n_j(v) \leq n_i(v)$, on a $N_j(v) \prec N_{i+1}(v)$.

Enfin, puisqu'une règle ne peut être appliquée sur l'étoile de centre v que si elle modifie l'étiquette de v, on sait que l'une des inégalités est stricte pour v. □

Comme pour l'algorithme de Mazurkiewicz, les informations dont dispose chaque sommet v dans sa boîte-aux-lettres permettent d'obtenir des informations vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet v' tel que $n_i(v') = m'$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet v' tel que $n_i(v) = m'$.

Lemme 4.18 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid \forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, il existe exactement un élément $(u, i_0) \in U'$ puisqu'à chaque étape, le numéro d'au plus un sommet peut être modifié. Le numéro $n_{i_0}(u) = m$ a donc été modifié à l'étape $i_0 + 1$, mais par maximalité de $(\lambda(u), N_{i_0}(u))$, la règle $\mathcal{M}_2$ n'a pas pu être appliquée à u à l'étape i_0. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 4.19 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. On note v le sommet dont l'étiquette est modifiée lors de la $(i+1)$ème étape de réétiquetage. La propriété est trivialement vraie à l'étape $i+1$ pour tout sommet $w \in V(G)$ différent de v.

Si la règle $\mathcal{M}_1$ a été appliquée à l'étape $i+1$, on considère un élément $(m, \ell, N) \in M_{i+1}(v)$. Si $m \neq n_{i+1}(v)$, alors il existe $v' \in N_G(v)$ tel que $(m, \ell, N) \in M_i(v')$ et $M_i(v') \subseteq M_{i+1}(v)$. Par conséquent, par hypothèse de récurrence, pour tout $m' < m$, il existe $(m', \ell', N') \in M_i(v') \subseteq M_{i+1}(v)$. Si $m = n_{i+1}(v)$, alors ou bien $m = 0$ et la propriété est trivialement vraie, ou bien on sait d'après le Lemme 4.16 qu'il existe $(m, \ell', N') \in M_i(v)$ et par hypothèse de récurrence, la propriété est vérifiée.

Si la règle $\mathcal{M}_2$ est appliquée à l'étape $i+1$, alors $M_{i+1}(v) = M_i(v) \cup \{(n_{i+1}(v) = 1 + \max\{m \mid (m, \ell, N) \in M_i(v)\}, \lambda(v), N_i(v))\}$, et par conséquent pour chaque $m \in M_{i+1}(v)$, la propriété reste vraie. □

On peut maintenant montrer que toute exécution de l'algorithme $\mathcal{M}$ termine sur $\mathbf{G}$. D'après les Lemmes 4.18 et 4.19, on sait qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1, k]$ ou un ensemble $[0, k]$ avec $k \leq |V(G)|$. Par conséquent, d'après le Lemme 4.17, on sait qu'il existe une étape i_0 telle que pour tout sommet v et toute étape $i \geq i_0$, $n_{i+1}(v) = n_i(v)$.

De plus, pour chaque sommet v et chaque étape i lors de laquelle la vue locale de v est modifiée, $N_i(v)$ contient un couple (n, p) si et seulement s'il existe exactement p sommets $v_1, \ldots, v_p \in N_G(v)$ tel que $\forall k \in [1, p], n_i(v_k) = n$. Par conséquent $N(v)$ ne peut prendre qu'un nombre fini de valeurs et il en est de même pour $M(v)$. Ainsi le nombre de valeurs différentes que peut prendre l'étiquette de chaque sommet est fini (mais dépend de la taille du graphe). Par ailleurs, on sait que les étiquettes consécutives de chaque sommet v forment une suite croissante et puisqu'on sait qu'à chaque étape i, l'étiquette d'au moins un sommet est modifiée, on sait que toute exécution de l'algorithme termine : la relation $\mathcal{M}$ est noethérienne.

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final.

Lemme 4.20 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. *$M_\rho(v) = M_\rho(v')$,*
3. *$(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,*
4. *si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et $N_\rho(v) = N_\rho(v')$,*
5. *$(n, p) \in N_\rho(v)$ si et seulement s'il existe $p \geq 1$ sommets distincts $v_1, v_2, \ldots v_p \in N_G(v)$ tels que $n_\rho(w) = n$; auquel cas il existe q tel que $\forall k \in [1, p], (n_\rho(v), q) \in N_\rho(v_k)$.*

Preuve :

1. D'après les Lemmes 4.18 et 4.19 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
2. Dans le cas contraire, la règle $\mathcal{M}_1$ peut être appliquée.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 4.16.

4. Dans le cas contraire, la règle $\mathcal{M}_2$ peut être appliquée à v ou à v'.
5. D'après la propriété précédente et puisque la règle $\mathcal{M}_1$ ne peut pas être appliquée.

□

Grâce au Lemme 4.20, on peut prouver que l'étiquetage final permet de construire un graphe dirigé sans boucle fortement connexe $\mathbf{D}$ tel que $Dir(\mathbf{G})$ est fibré proprement sur $\mathbf{D}$.

Proposition 4.21 *Étant donné un graphe simple* $\mathbf{G}$*, on peut construire, à partir de l'étiquetage final obtenu après une exécution* ρ *de* $\mathcal{M}$*, un graphe dirigé sans boucle fortement connexe* $\mathbf{D}$ *tel que* $Dir(\mathbf{G})$ *est fibré proprement sur* $\mathbf{D}$*.*

Preuve : On utilise les notations du Lemme 4.20.

On construit un graphe dirigé $\mathbf{D}$ de la manière suivante. L'ensemble $V(D)$ des sommets est l'ensemble des numéros apparaissant dans le graphe dans la configuration finale, i.e., $V(D) = \{m \mid \exists v \in V(G), n_\rho(v) = m\}$. Pour tous sommets $v, v' \in V(G)$ tels que $n_\rho(v) = n_\rho(v') = m$, on sait d'après le Lemme 4.20 que pour tout entier $m' \in n_\rho(V(G))$, il existe $(m', p) \in N_\rho(v)$ si et seulement si $(m', p) \in N_\rho(v')$ et que $(m', p) \in N_\rho(v)$ (resp. $(m', p) \in N_\rho(v')$) si et seulement si v (resp. v') a p voisins distincts dont le numéro est m'. Ainsi, on peut définir les arcs de D de la manière suivante, pour tout sommets $m, m' \in V(D)$, on choisit un sommet $v \in n_\rho^{-1}(m)$ et si v a p voisins dont le numéro est m', alors $A(D)$ contient p arcs $a_{m',m,1}, \ldots, a_{m',m,p}$ tels que $\forall k \in [1, p], s(a_{m',m,k}) = m'$ et $t(a_{m',m,k}) = m$. D'après le Lemme 4.16, on sait que pour tous sommets voisins $v, v' \in V(G)$, $n_\rho(v) \neq n_\rho(v')$ et par conséquent, le graphe D ne contient pas de boucle.

Puisque pour tout couple $(m, p) \in N_\rho(v)$, $p > 1$, il existe $v' \in N_G(v)$ tel que $n_\rho(v') = m'$ et tel qu'il existe $(n_\rho(v), q) \in N_\rho(v')$. Ainsi, pour tout $m, m' \in n_\rho(V(G))$, s'il existe un arc $a \in A(D)$ tel que $s(a) = m'$ et $t(a) = m$, alors il existe un arc $a' \in A(D)$ tel que $s(a) = m$ et $t(a) = m'$. On sait aussi que le graphe D est connexe puisque $\mathbf{G}$ est connexe et par conséquent, D est un graphe dirigé fortement connexe sans boucle.

Pour tout sommet $v, v' \in V(G)$, si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et on peut donc définir un étiquetage η de D de la manière suivante, pour tout $v \in V(G)$, $\eta(n_\rho(v)) = \lambda(v)$.

On rappelle que $V(Dir(\mathbf{G})) = V(G)$ et que pour tout arête $\{v, v'\} \in E(G)$, il existe deux arcs $a_{v',v}, a_{v,v'} \in A(Dir(G))$ tels que $s(a_{v,v'}) = t(a_{v',v}) = v$ et $t(a_{v,v'}) = s(a_{v',v}) = v'$. De plus, pour tout sommet $v \in V(G)$, l'étiquette de v dans $Dir(\mathbf{G})$ est la même que dans $\mathbf{G}$.

On définit maintenant un homomorphisme γ de $Dir(\mathbf{G})$ dans $\mathbf{D}$ de la manière suivante. Pour tout sommet $v \in V(G)$, $\gamma(v) = n_\rho(v)$. Pour tout sommet v étiqueté n et pour chaque $m \in n_\rho(N_\rho(v))$, on considère un ordre quelconque sur les sommets $v_1, v_2, \ldots, v_p \in N_G(v)$ dont le numéro est m. Ainsi, pour tout arc $a_{v_i,v}$, on pose $\gamma(a_{v_i,v}) = a_{m,n,i}$. De par la définition des arcs de $A(D)$, on sait que si un sommet v étiqueté n a p voisins dont le numéro est m, alors il existe exactement p arcs dont la source est m et la cible est n dans D. Ainsi, γ est une fibration de $Dir(G)$ dans D et puisqu'il est clair que γ préserve l'étiquetage, $Dir(\mathbf{G})$ est fibré sur $\mathbf{D}$. □

On considère maintenant un graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes. Pour chaque exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, le graphe obtenu à partir de

l'étiquetage final est isomorphe à $\mathbf{G}$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique. L'algorithme $\mathcal{M}$ permet de résoudre le nommage sur la famille des graphes minimaux pour les fibrations discrètes, mais si aucune information à propos de $\mathbf{G}$ n'est disponible, les sommmets ne peuvent pas détecter la terminaison.

Cependant, la détection de la terminaison est possible pour un graphe $\mathbf{G}$ donné. En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 4.18 et 4.19, il sait que chaque sommet de $\mathbf{G}$ a un numéro unique qui ne va plus être modifié : il peut donc détecter que le nommage est effectué.

Par ailleurs, d'après la Proposition 4.15, on sait que pour tout graphe $\mathbf{G}$ tel que $Dir(\mathbf{G})$ n'est pas minimal pour les fibrations discrètes, il n'existe aucun algorithme utilisant des calculs locaux cellulaires sur les étoiles non-étiquetées qui permette de résoudre les problèmes du nommage ou de l'énumération sur $\mathbf{G}$. On a ainsi obtenu une nouvelle preuve du théorème suivant qui a été prouvé pour la première fois par Boldi et al.[BCG$^+$96].

Théorème 4.22 ([BCG$^+$96]) *Pour tout graphe simple étiqueté $\mathbf{G}$, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour $\mathbf{G}$ utilisant des calculs locaux cellulaires sur les étoiles non-étiquetées,*
2. *il existe un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $\mathbf{G}$ utilisant des calculs locaux cellulaires sur les étoiles non-étiquetées,*
3. *$Dir(\mathbf{G})$ est minimal pour les fibrations discrètes.*

Remarque 4.23 *Étant donné un graphe $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes, pour détecter que l'algorithme $\mathcal{M}$ a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de $\mathbf{G}$. Ainsi, l'algorithme $\mathcal{M}$ permet de résoudre l'énumération et le nommage avec détection de la terminaison sur les graphes minimaux pour les fibrations discrètes de taille donnée.*

4.4.4 Complexité

On s'intéresse à la complexité de l'algorithme $\mathcal{M}$ présenté ci-dessus. Dans le cadre des calculs locaux, on s'intéresse au nombre de pas de réétiquetages effectuées lors d'une exécution. La proposition suivante donne une borne supérieure sur le nombre de pas de réétiquetages de toute exécution de l'algorithme $\mathcal{M}$ sur un graphe à n sommets de degré maximal Δ.

Proposition 4.24 *Pour tout graphe $\mathbf{G}$ à n sommets de degré maximal Δ, durant toute exécution de l'algorithme $\mathcal{M}$ utilisant des calculs locaux cellullaires sur les étoiles, $O(\Delta n^3)$ règles sont appliquées.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et une exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$. D'après les Lemmes 4.18 et 4.19, on sait que la règle $\mathcal{M}_2$ ne peut pas être appliquée plus de $\frac{n(n-1)}{2}$ fois durant l'exécution ρ.

Entre deux étapes où la règle $\mathcal{M}_2$ est appliquée, la vue locale d'un sommet est modifiée au plus Δ fois (une fois pour chaque voisin du sommet dont le numéro a été modifié). Il existe donc $O(n^2)$ étapes où la règle $\mathcal{M}_1$ est appliquée et modifie la vue locale d'un sommet.

À chaque fois qu'un sommet v modifie son numéro ou sa vue locale, un couple (n_0, ℓ, N) est ajouté $M(v)$. Pour chacun de ces couples, la règle $\mathcal{M}_1$ est appliquée au plus n fois.

Ainsi, durant toute exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, $O(\Delta n^3)$ règles sont appliquées. □

Pour la même raison que dans le Chapitre 3, on remarque que la borne de la Proposition 2.23 et celle de la Proposition 4.24 diffèrent d'un facteur Δ.

On s'intéresse à la mémoire nécessaire à chaque sommet pour stocker son étiquette. On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets de G).

Proposition 4.25 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux cellullaires sur les étoiles nécessite* $O(\Delta n \log n)$ *bits de mémoire par sommet.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets dont le degré maximal est Δ. 0n sait que la vue locale de chaque sommet contient au plus Δ couples (n_0, p_0) qui peuvent être représentés avec $O(\log n)$ bits. Ainsi, pour chaque sommet v, $N(v)$ peut être représenté avec $O(\Delta \log n)$ bits.

Chaque sommet peut ne conserver dans sa boîte-aux-lettres que l'information utile, i.e., l'ensemble $\{(n_0, \ell, N) \in M(v) \mid \forall (n_0, \ell', N') \in M(v), (\ell', N') \preceq (\ell, N)\}$. Ainsi, dans la boîte-aux-lettres de chaque sommet, il existe au plus n triplets (n_0, ℓ, N) dont la taille est en $O(\Delta \log n)$ bits. Par conséquent, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta n \log n)$ bits. □

Remarque 4.26 *L'algorithme décrit par Boldi et al. [BCG+96] nécessite l'application de* $O(n^2)$ *pas de réétiquetages, mais la mémoire nécessaire à chaque sommet est de* $2^{O(n)}$ *bits. Ainsi, on a obtenu un algorithme dont les exécutions nécessitent un nombre d'applications de règles plus important (tout en restant polynomial en la taille du graphe), mais qui nécessite une mémoire polynomiale en la taille du graphe (et non exponentielle) en chaque sommet.*

Cette amélioration est intéressante puisque si on veut implémenter cet algorithme dans un système asynchrone où les processus communiquent en échangeant des messages en utilisant des méthodes de synchronisation locales probabilistes présentées dans [MSZ02] (comme cela est possible dans ViSiDiA), alors il faut que régulièrement les processus échangent leurs états. Ainsi, on peut implémenter notre algorithme à l'aide de méthodes de synchronisation locales probabilistes dans un système asynchrone où les processus communiquent en échangeant des messages de taille polynomiale et non exponentielle.

4.4.5 Importance des Connaissances Initiales

Dans la partie précédente, on a montré que l'algorithme $\mathcal{M}$ permettait de résoudre le problème du nommage avec détection de la terminaison sur la famille des graphes minimaux pour les fibrations discrètes de taille donnée.

Étant donné un graphe $\mathbf{G}$ minimal pour les fibrations discrètes, la connaissance d'une borne sur le diamètre de $\mathbf{G}$ permet de résoudre le problème du nommage sur $\mathbf{G}$. En effet, il est facile de voir qu'on peut modifier l'algorithme $\mathcal{M}$ présenté ci-dessus de la même manière que dans le modèle étudié dans le Chapitre 2 pour obtenir la proposition suivante.

Proposition 4.27 *Pour tout entier B, on peut modifier l'algorithme $\mathcal{M}$ pour obtenir un algorithme $\mathcal{M}'$ utilisant des calculs locaux cellulaires sur les étoiles tel que pour tout graphe simple $\mathbf{G}$ dont le diamètre est borné par B, $\mathcal{M}'$ permet à chaque sommet de $\mathbf{G}$ de détecter que l'algorithme $\mathcal{M}$ sous-jacent est terminé, i.e., l'étiquette $(n(v), N(v), M(v))$ de chaque sommet ne sera plus modifié.*

On en déduit donc le théorème suivant de la même manière que dans le Chapitre 2.

Théorème 4.28 *Pour tout entier B, il existe un algorithme de nommage avec détection de la terminaison utilisant des calcusl locaux cellulaires sur les étoiles pour la famille des graphes simples minimaux pour les fibrations discrètes dont le diamètre est bornée par B.*

4.5 Élection

On s'intéresse maintenant au problème de l'élection dans le cadre des calculs locaux cellulaires sur les étoiles. On présente d'abord un résultat d'impossibilité du à Boldi et al. [BCG$^+$96] qui repose sur la notion de fibrations discrètes, puis on explique comment utiliser l'algorithme $\mathcal{M}$ présenté ci-dessus pour résoudre le problème de l'élection.

4.5.1 Résultats d'Impossibilité

Proposition 4.29 ([BCG$^+$96]) *Soit $\mathbf{G}$ un graphe simple étiqueté tel que $Dir(\mathbf{G})$ est non-trivialement fibré sur un graphe dirigé fortement connexe sans boucle $\mathbf{D}$. Il n'existe pas d'algorithme d'élection pour le graphe $\mathbf{G}$ utilisant des calculs locaux cellulaires sur les étoiles.*

Preuve : On considère un graphe simple $\mathbf{G} = (G, \lambda)$ et un graphe dirigé fortement connexe sans boucle $\mathbf{D} = (D, \eta)$ tel que $Dir(\mathbf{G})$ est non-trivialement fibré sur $\mathbf{D}$ à travers une fibration γ. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux cellulaires sur les étoiles, on considère une exécution de $\mathcal{R}$ sur $\mathbf{D}$. Si cette exécution est infinie sur $\mathbf{D}$, alors d'après le Lemme 4.14, il existe une exécution infinie de $\mathcal{R}$ sur $Dir(\mathbf{G})$ et donc sur $\mathbf{G}$; auquel cas, $\mathcal{R}$ n'est pas un algorithme d'élection.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur $\mathbf{D}$ et on considère la configuration finale $\mathbf{D}'$. D'après le Lemme 4.14, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui permet d'atteindre une configuration $\mathbf{G}'$ telle que $Dir(\mathbf{G}')$ est non-trivialement fibré sur $\mathbf{D}'$ à travers γ. Si $\mathbf{G}'$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe un sommet $u \in V(G)$ tel qu'on peut appliquer une règle de $\mathcal{R}$ sur $\mathbf{B}_{\mathbf{G}'}(u)$. Par conséquent, cette règle peut être appliquée sur $\mathbf{B}^-_{Dir(\mathbf{G}')}(u)$ et sur $\mathbf{B}^-_{\mathbf{D}'}(\gamma(u))$ d'après la Proposition 4.12, mais cela signifie que $\mathbf{D}'$ n'est pas une configuration finale de $\mathcal{R}$, ce qui est impossible. Par conséquent, $\mathbf{G}'$ est une configuration finale de $\mathcal{R}$. Mais puisque $Dir(\mathbf{G}')$ est fibré non-trivialement sur $\mathbf{D}'$, cela implique que pour tout sommet $u \in V(G)$, il existe au moins deux sommets dans $\gamma^{-1}(\gamma(u))$ qui ont la même étiquette dans $\mathbf{G}'$. Par conséquent, il n'existe pas de sommet $u \in V(G)$ qui a une étiquette unique. L'algorithme $\mathcal{R}$ ne permet donc pas de résoudre le problème de l'élection sur $\mathbf{G}$. □

4.5.2 Un Algorithme d'Élection

On va maintenant montrer comment utiliser l'algorithme de la Proposition 4.27 pour résoudre le problème de l'élection sur les graphes simples minimaux pour les fibrations discrètes non-triviales. Pour cela, on utilise le fait que le graphe $\mathbf{G}$ est un graphe simple afin de déterminer le nombre d'antécédents de chaque sommet du graphe dirigé reconstruit à partir de l'étiquetage final de $\mathbf{G}$ lorsque l'exécution de l'algorithem $\mathcal{M}$ sous-jacent est terminée.

Dans la proposition suivante, on montre que si un graphe $Dir(\mathbf{G})$ est fibré sur un graphe dirigé sans boucle $\mathbf{D}$ à travers un homomorphisme γ, alors si on connaît $\mathbf{D}$ et la taille de la fibre d'un sommet $v \in V(D)$, on peut calculer la taille de la fibre de tout sommet $v' \in V(D)$.

Proposition 4.30 *Étant donné un graphe dirigé fortement connexe sans boucle* $\mathbf{D}$*, pour tous sommets* $v, v' \in V(D)$*, il existe deux entiers* $p(v,v'), p(v',v)$ *premiers entre eux tels que pour tout graphe simple* $\mathbf{G}$ *que* $Dir(\mathbf{G})$ *est fibré sur* $\mathbf{D}$ *à travers une fibration* γ*,* $p(v,v')|\gamma^{-1}(v)| = p(v',v)|\gamma^{-1}(v')|$.

Preuve : On considère un graphe dirigé fortement connexe sans boucle $\mathbf{D}$, et deux sommets v, v' tels qu'il existe un arc $a'_1 \in A(D)$ tel que $s(a'_1) = v$ et $t(a'_1) = v'$. On suppose qu'il existe un graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est fibré sur $\mathbf{D}$ à travers une fibration γ. Puisque $\mathbf{D}$ est fortement connexe, γ est un homomorphisme surjectif et il existe $u, u' \in V(Dir(\mathbf{G}))$ et un arc $a_1 \in A(Dir(\mathbf{G}))$ tels que $s(a_1) = u$ et $t(a_1) = u'$, $\gamma(u) = v$, $\gamma(u') = v'$ et $\gamma(a_1) = a'_1$. Par conséquent, il existe un arc $a_2 = Sym(a_1) \in A(Dir(\mathbf{G}))$ tel que $s(\gamma(a_2)) = \gamma(s(a_2)) = v'$ et $t(\gamma(a_2)) = v$. Ainsi, pour tout arc $a'_1 \in A(D)$, il existe un arc $a'_2 \in A(D)$ tel que $s(a'_2) = t(a'_1)$ et $s(a'_1) = t(a'_2)$.

On fait une démonstration par récurrence sur la distance k entre v et v' que pour tous sommets $v, v' \in V(D)$, il existe deux entiers premiers entre eux $p(v,v')$ et $p(v',v)$ telles que pour tout graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est fibré sur $\mathbf{D}$ à travers un homomorphisme γ, $p(v,v')|\gamma^{-1}(v)| = p(v',v)|\gamma^{-1}(v')|$. Si $v = v'$, on définit $p(v,v) = 1$ et la propriété est vérifiée.

Pour tout sommets voisins $v, v' \in V(D)$, on note $d_{(v,v')}$ (resp. $d_{(v',v)}$) le nombre d'arcs a tels que $s(a) = v$ et $t(a) = v'$ (resp. $s(a) = v'$ et $t(a) = v$). On sait que chaque sommet $u \in \gamma^{-1}(v))$ est la cible d'exactement $d_{(v',v)}$ arcs dont la source est envoyé par γ sur v' et puisque $\mathbf{G}$ est un graphe simple, le sommet u a exactement $d_{(v',v)}$ voisins u' tels que $\gamma(u') = v'$. Par conséquent, il existe exactement $d_{(v',v)}|\gamma^{-1}(v)|$ arêtes $\{u, u'\}$ dans $E(G)$ telles que $\gamma(u) = v$ et $\gamma(u') = v'$. De même, on peut montrer qu'il existe exactement $d_{(v,v')}|\gamma^{-1}(v')|$ arêtes $\{u, u'\}$ dans $E(G)$ telles que $\gamma(u) = v$ et $\gamma(u') = v'$ et on a donc $d_{(v',v)}|\gamma^{-1}(v)| = d_{(v,v')}|\gamma^{-1}(v')|$. Ainsi, la propriété est vérifiée pour $\text{dist}_{\mathbf{D}}(v,v') = 1$ en divisant $d_{(v',v)}$ et $d_{(v,v')}$ par leur plus grand dénominateur commun.

On suppose que la propriété est vraie pour tous sommets v, v' à distance inférieure à k dans $\mathbf{D}$. On considère deux sommets $v, v' \in V(D)$ à distance $k+1$ dans $\mathbf{D}$. Il existe alors un voisin v'' de v dans $\mathbf{D}$ qui est à distance k de v' dans $\mathbf{D}$. Par hypothèse de récurrence, il existe donc deux entiers $p = p(v,v''), p'' = p(v'',v)$ et deux entiers $q' = p(v',v''), q'' = p(v'',v')$ tels que pour tout graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est fibré sur $\mathbf{D}$ à travers γ, $p|\gamma^{-1}(v)| = p''|\gamma^{-1}(v'')|$ et $q'|\gamma^{-1}(v')| = q''|\gamma^{-1}(v'')|$. Mais alors, on a $pq''|\gamma^{-1}(v)| = p''q'|\gamma^{-1}(v')|$ et la propriété est bien vérifiée pour v, v' en divisant pq'' et $p''q'$ par leur plus grand dénominateur commun. □

La proposition précédente va permettre de montrer qu'il est possible de résoudre le problème de l'élection dans les graphes simples minimaux pour les fibrations discrètes non-triviales. On définit d'abord les candidats d'un graphe dirigé sans boucle sur lequel est fibré le graphe dirigé obtenu à partir d'un graphe simple.

Définition 4.31 *Étant donné un graphe dirigé fortement connexe sans boucle* $\mathbf{D}$ *tel qu'il existe un graphe simple* $\mathbf{G}$ *tel que* $Dir(\mathbf{G})$ *est fibré sur* $\mathbf{D}$*, un sommet* v *est un* candidat *de* $\mathbf{D}$ *si pour tout* $v' \in V(D)$*,* $p(v', v) = 1$*.*

On note $C_{\mathbf{D}}$ *l'ensemble des candidats d'un graphe dirigé fortement connexe sans boucle* $\mathbf{D}$*.*

Dans le lemme suivant, on montre que si $Dir(\mathbf{G})$ est fibré sur un graphe $\mathbf{D}$ à travers un homomorphisme γ, la fibre d'un sommet de $\mathbf{D}$ qui n'est pas un candidat n'est pas triviale.

Lemme 4.32 *Pour tout graphe* $\mathbf{G}$ *tel que* $Dir(\mathbf{G})$ *est fibré sur un graphe dirigé fortement connexe* $\mathbf{D}$ *à travers un homomorphisme* γ*, pour tout sommet* $v \in V(D) \backslash C_{\mathbf{D}}$*,* $|\gamma^{-1}(v)| > 1$*.*

De plus, pour tous $v, v' \in C_{\mathbf{D}}$*,* $|\gamma^{-1}(v)| = |\gamma^{-1}(v')|$*.*

Preuve : On considère un graphe simple $\mathbf{G}$ tel que $Dir(\mathbf{G})$ est fibré sur un graphe dirigé fortement connexe sans boucle $\mathbf{D}$ à travers un homomorphisme γ. Soit v un sommet de D tel que $|\gamma^{-1}(v)| = 1$. Pour tout $v' \in V(D)$, il existe exactement $p(v, v')/p(v', v)$ sommets dans $\gamma^{-1}(v')$, mais puisque $p(v, v')$ et $p(v', v)$ sont premier entre eux, cela implique que $p(v', v) = 1$. Par conséquent, si $|\gamma^{-1}(v)| = 1$, alors $v \in C_{\mathbf{D}}$.

De plus, pour tous $v, v' \in C_{\mathbf{D}}$, $p(v, v') = p(v', v) = 1$ et alors $|\gamma^{-1}(v)| = |\gamma^{-1}(v')|$. □

On sait d'après la Proposition 4.27, qu'il existe une extension $\mathcal{M}'$ de l'algorithme $\mathcal{M}$ tel que pour chaque graphe simple $\mathbf{G}$ de taille borné par B, chaque sommet $u \in V(G)$ peut détecter que l'algorithme $\mathcal{M}$ sous-jacent est terminé.

Une fois que chaque sommet u a détecté que pour chaque sommet $u' \in V(G)$, les valeurs de $n(u')$, $N(u')$ et $M(u')$ ne seront plus modifiées, il peut reconstruire un graphe dirigé sans boucle $\mathbf{D}$ à partir de sa boîte-aux-lettres comme indiqué dans la Proposition 4.21. De plus, le sommet u sait que tous les sommets ont la même boîte-aux-lettres et qu'ils reconstruiront donc le même graphe dirigé $\mathbf{D}$. Par ailleurs, on sait que $Dir(\mathbf{G})$ est fibré sur $\mathbf{D}$.

Si $\mathbf{G}$ est minimal pour les fibrations discrètes non-triviales, on sait qu'il existe un sommet $v \in V(D)$ dont la fibre est triviale. D'après le Lemme 4.32, le sommet v est un candidat de $\mathbf{D}$ et tous les candidats de $\mathbf{D}$ ont une fibre triviale. Puisque $V(D) = n(V(G))$, on peut choisir d'élire le sommet de $\mathbf{G}$ dont l'image est le candidat de D avec le plus petit numéro, i.e., un sommet u prend l'étiquette ÉLU si $n(u) \in C_{\mathbf{D}}$ et si pour tout $n \in C_{\mathbf{D}}$, $n(u) \leq n$. Les autres sommets, i.e., les sommets v tels que $n(v) \notin C_{\mathbf{D}}$ ou tels qu'il existe $n \leq n(v)$ dans l'ensemble $C_{\mathbf{D}}$ prennent l'étiquette NON-ÉLU. Ainsi, puisqu'il existe un unique sommet dans $n^{-1}(n(u))$, un seul sommet de $\mathbf{G}$ sera élu.

D'après la Proposition 4.29, si $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes non-triviales, il n'existe pas d'algorithme d'élection utilisant des calculs locaux cellulaires sur les étoiles pour $\mathbf{G}$. On a donc montré les deux théorèmes suivants. Le premier a été montré par Boldi et al. [BCG+96]. Le second a été montré par Boldi et Vigna [BV01] dans le cas particulier où les sommets disposent de la connaissance d'une borne sur la taille, et non sur le diamètre.

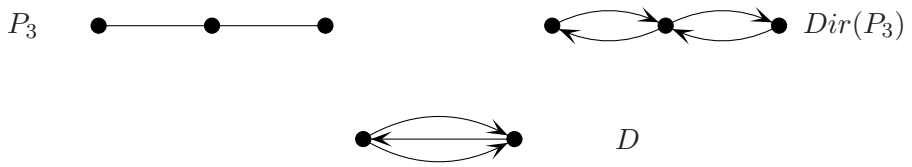


FIG. 17 – Le graphe $Dir(P_3)$ est fibré proprement sur le graphe D et P_3 n'est donc pas minimal pour les fibrations discrètes. Cependant P_3 est minimal pour les fibrations discrètes non-triviales

Théorème 4.33 ([BCG$^+$96]) *Pour tout graphe simple étiqueté* $\mathbf{G}$*, il existe un algorithme d'élection pour* $\mathbf{G}$ *utilisant des calculs locaux cellulaires sur les étoiles si et seulement si* $Dir(\mathbf{G})$ *est minimal pour les fibrations discrètes non-triviales.*

Théorème 4.34 *Pour tout entier* B*, il existe un algorithme d'élection utilisant des calculs locaux cellulaires sur les étoiles pour la famille des graphes minimaux pour les fibrations discrètes non-triviales dont le diamètre est borné par* B*.*

Il existe des graphes dans lesquels on ne peut pas nommer en utilisant des calculs locaux cellulaires sur les étoiles, mais dans lesquels on peut élire. Par exemple, certains arbres ne sont pas minimaux pour les fibrations discrètes, comme par exemple celui de la Figure 17. Ainsi, il n'existe pas d'algorithme universel de nommage utilisant des calculs locaux cellulaires sur les étoiles pour la famille des arbres.

Cependant, l'algorithme d'élection pour la famille des arbres présenté dans le Chapitre 1 utilise des calculs locaux cellulaires sur les étoiles et on a donc le théorème suivant.

Théorème 4.35 *Il existe un algorithme d'élection pour la famille des arbres utilisant des calculs locaux cellulaires sur les étoiles.*

4.5.3 Des Algorithmes Effectifs de Nommage et d'Élection

On suppose maintenant qu'on connaît seulement une borne serrée sur la taille du graphe et on va montrer qu'il existe des algorithmes effectifs d'élection de nommage avec détection de la terminaison utilisant des calculs locaux cellulaires sur les arêtes.

Théorème 4.36 *Pour tout entier* B*, il existe un algorithme effectif de nommage avec détection de la terminaison utilisant des calculs locaux cellulaires sur les étoiles pour la classe des graphes* $\mathbf{G}$ *tels que* $V(G) \leq B < 2|V(G)|$*.*

Preuve : Étant donné un graphe simple $\mathbf{G}$, si on connaît une borne serrée B sur la taille de $\mathbf{G}$, on sait que $|V(G)| \leq B < 2|V(G)|$. Soit $\mathbf{D}$ le graphe reconstruit par chaque sommet u à la fin de l'exécution de l'algorithme $\mathcal{M}'$. Si $C_{\mathbf{D}}$ est différent de $V(D)$, on sait que $Dir(\mathbf{G})$ est fibré proprement sur $\mathbf{D}$ et par conséquent, $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes.

On suppose maintenant que $C_{\mathbf{D}} = V(D)$. Par conséquent, pour tous sommets $v, v' \in V(D)$, $p(v, v') = p(v', v) = 1$. Ainsi, pour tous $v, v' \in V(D)$, $|\gamma^{-1}(v)| = |\gamma^{-1}(v')|$. Par conséquent, si $Dir(\mathbf{G})$ n'est pas isomorphe à $\mathbf{D}$, il existe une constante $q \geq 2$ telle que

pour tout $v \in V(D)$, $|\gamma^{-1}(v)| = q$. Par conséquent, $2|V(D)| \leq |V(G)| \leq B$. Au contraire, si $Dir(\mathbf{G})$ est isomorphe à $\mathbf{D}$, $2|V(D)| = 2|V(G)| > B$.

Puisque tous les sommets de $\mathbf{G}$ connaissent $\mathbf{D}$ à la fin de l'exécution de $\mathcal{M}'$, ils peuvent vérifier si $C_{\mathbf{D}} = V(D)$ et si $2|V(D)| > B$. Si ces deux conditions sont satisfaites, il savent que l'algorithme $\mathcal{M}'$ a attribué des noms uniques à tous les sommets de $\mathbf{G}$. Dans le cas contraire, ils savent que $Dir(\mathbf{G})$ est fibré proprement sur $\mathbf{D}$ et ils peuvent donc conclure que $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes. □

Théorème 4.37 *Pour tout entier B, il existe un algorithme effectif d'élection utilisant des calculs locaux cellulaires sur les étoiles pour la classe des graphes $\mathbf{G}$ tels que $V(G) \leq B < 2|V(G)|$.*

Preuve : Étant donné un graphe simple $\mathbf{G}$, si on connaît une borne serrée B sur la taille de $\mathbf{G}$, on sait que $|V(G)| \leq B < 2|V(G)|$. Soit $\mathbf{D}$ le graphe reconstruit par chaque sommet u à la fin de l'exécution de l'algorithme $\mathcal{M}'$; on rappelle que $V(D) = n(V(G))$. Si $C_{\mathbf{D}}$ ne contient aucun sommet de $V(D)$, on sait que $Dir(\mathbf{G})$ est fibré non-trivialement sur $\mathbf{D}$ et par conséquent, $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes non-triviales.

De plus, pour tous sommet $v \in C_{\mathbf{D}}$ et $v' \in V(D)$, $|\gamma^{-1}(v')| = p(v, v')|\gamma^{-1}(v)|$. Par conséquent, puisque $p(v, v) = 1$, on sait que $|V(G)| = |\gamma^{-1}(v)|T$, où $T = \sum_{v' \in V(G)} p(v, v')$. Si $|\gamma^{-1}(v)| \geq 2$, on sait que $Dir(\mathbf{G})$ est fibré non-trivialement sur $\mathbf{D}$ et que $2T \leq |V(G)| \leq B$. Au contraire si $|\gamma^{-1}(v)| = 1$, $|V(G)| = T$ et donc $2T = 2|V(G)| > B$.

Par conséquent, puisque tous les sommets de $\mathbf{G}$ connaissent $\mathbf{D}$ à la fin de l'exécution de $\mathcal{M}'$, ils peuvent vérifier si $C_{\mathbf{D}} \neq \emptyset$ et si $2T > B$. Si ces deux conditions sont satisfaites, les sommets savent que tous les sommets $u \in V(G)$ tels que $n(u) \in C_{\mathbf{D}}$ ont des étiquettes uniques. Le sommet $u \in V(G)$ tel que $n(u) = \min\{n \mid n \in C_{\mathbf{D}}\}$ prend l'étiquette ÉLU et les autres sommets prennent l'étiquette NON-ÉLU. Dans le cas où l'une des deux conditions n'est pas satisfaite, les sommets de $\mathbf{G}$ savent que $Dir(\mathbf{G})$ est fibré non-trivialement sur $\mathbf{D}$ et ils peuvent donc conclure que $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes non-triviales. □

Comme dans le cas des calculs locaux sur les étoiles fermées, on remarque que si la borne sur la taille de $\mathbf{G}$ n'est pas serrée, il n'existe pas d'algorithme utilisant des calculs locaux cellulaires sur les étoiles qui permet de résoudre le problème du nommage avec détection de la terminaison (resp. le problème de l'élection) sur $\mathbf{G}$ ou de détecter que $\mathbf{G}$ n'est pas minimal pour les fibrations discrètes (resp. les fibrations discrètes non-triviales).

4.6 Conclusions et Perspectives

Dans ce chapitre, on a présenté une caractérisation des graphes admettant un algorithme de nommage utilisant des calculs locaux sur les étoiles ouvertes (Théorème 4.22). On a aussi présenté une caractérisation des graphes admettant un algorithme d'élection (Théorème 4.33). Les caractérisations présentées dans ce chapitre sont basées sur la notion de fibrations.

On a par ailleurs étudié les connaissances initiales nécessaires pour résoudre ces problèmes. Ainsi, la connaissance initiale d'une borne sur le diamètre permet de résoudre

élection (resp. nommage) sur un graphe minimal pour les fibrations non-triviales (resp. minimal pour les fibrations) ; ces résultats sont présentés dans les Théorèmes 4.34 et 4.28. Par ailleurs, la connaissance initiale d'une borne serrée sur la taille permet d'obtenir des algorithmes effectifs de nommage et d'élection (Théorèmes 4.36 et 4.37).

Les résultats obtenus dans ce modèle nous permettent de penser que les techniques présentées dans [GM03, GMM04, God02b, MT00, GM02] peuvent être adaptées dans le modèle étudié dans ce chapitre. Il semble en particulier que lorsqu'aucune connaissance initiale n'est disponible, les classes de graphes simples reconnaissables dans ce modèle doivent vérifier les mêmes propriétés que celles présentées dans [GMM04].

Chapitre 5

Calculs Locaux sur les Arêtes Non-Étiquetées

Sommaire

5.1 Introduction

Dans ce chapitre, on étudie les *calculs locaux sur les arêtes non-étiquetées*. Dans ce modèle, on considère des graphes simples où seuls les sommets peuvent être étiquetés, comme dans le Chapitre 4. Un pas de calcul est décrit par une règle de réétiquetage de la forme présentée sur la Figure 18. Si dans un graphe G, deux sommets voisins sont étiquetés X et Y, alors lors de l'application de cette règle, on remplace X par X' et Y par Y'. Les étiquettes de tous les autres sommets de G ne sont pas prises en compte lors de

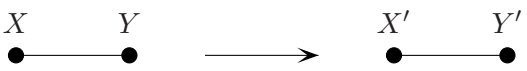


FIG. 18 – Forme générique d'une règle de calcul pour les calculs locaux sur les arêtes non-étiquetées.

l'application de la règle et ne sont pas modifiées. Les sommets de G dont les étiquettes sont modifiées sont dits *actifs* et les autres sommets de G sont dits *inactifs*. Les calculs réalisés en utilisant uniquement ce type de règles de réétiquetage sont appelés *calculs locaux sur les arêtes non-étiquetées*. Il faut noter, que comme dans le modèle du Chapitre 3, les règles ne sont pas symétriques, i.e., l'application d'une règle de réétiquetage sur une arête dont les deux extrémités ont la même étiquette peut attribuer des nouvelles étiquettes différentes aux deux sommets.

La différence entre ce modèle et le modèle des calculs locaux sur les arêtes étiquetées est que les arêtes ne peuvent pas être étiquetées (et réétiquetées) ; cela donne un modèle strictement plus faible. On montre aussi que les calculs locaux sur les arêtes non-étiquetées et les calculs locaux cellulaires sur les étoiles ont des puissances de calcul incomparables. En effet, il existe des graphes admettant un algorithme de nommage et d'élection utilisant des calculs locaux sur les arêtes non-étiquetées pour lesquels il n'existe pas d'algorithme d'élection ou de nommage utilisant des calculs locaux cellulaires sur les étoiles, et réciproquement. Ainsi, lorsque les arêtes ne peuvent pas être étiquetées, il n'existe pas de résultat correspondant à la Proposition 3.42.

5.1.1 Résultats

Dans le modèle des calculs locaux sur les arêtes non-étiquetées, on caractérise les graphes admettant un algorithme de nommage et d'élection. Comme dans les Chapitres 2 et 3, on peut élire dans un graphe si et seulement si on peut nommer dans ce graphe. On introduit la notion de *pseudo-revêtements* pour pouvoir exprimer cette caractérisation.

On montre qu'un graphe admet un algorithme de nommage et d'élection utilisant des calculs locaux sur les arêtes non-étiquetées si et seulement s'il est minimal pour les pseudo-revêtements (Théorème 5.19). L'algorithme d'énumération qu'on présente dans la Section 5.4 est adapté de l'algorithme de Mazurkiewicz.

Cependant, dans le modèle considéré ici, un sommet ne peut pas distinguer ses voisins. Lorsqu'un sommet échange son numéro avec un de ses voisins, il doit aussi détruire de sa vue locale l'information relative à l'ancien numéro de ce voisin. Ainsi, dans notre algorithme, à chaque fois que la vue locale d'un sommet v est modifiée, on supprime beaucoup d'information de la vue locale de v afin de s'assurer que les informations erronées ont bien été effacées. Toutefois, on peut aussi supprimer de cette manière des informations valides qui devront être acquises à nouveau par le sommet v.

De plus, dans le modèle étudié ici, l'algorithme doit assurer que tous les numéros attribués au sommets apparaissent avec la même cardinalité dans la configuration finale. La solution proposée est d'associer un identifiant à chaque transaction entre deux voisins de telle sorte que deux transactions impliquant un même sommet aient deux identifiants distincts. On montre qu'avec cette méthode, dans la configuration finale, tous les numéros associés aux sommets apparaissent avec la même cardinalité. Cependant, ce mécanisme implique que lorsqu'un sommet v met à jour sa vue locale en se synchronisant avec un

voisin v', alors la vue locale de v' est aussi modifiée et des informations valides en sont effacées. Le sommet v' doit alors aussi remettre à jour sa vue locale, etc. Il faut donc montrer que la présence des identifiants associées aux transactions entre voisins ne génère pas d'exécution infinie.

Le mécanisme utilisé pour s'assurer que tous les numéros associés aux sommets apparaissent avec la même cardinalité dans la configuration finale a un impact non-négligeable sur la complexité. En effet, certaines exécutions de l'algorithme nécessitent un nombre d'étapes de réétiquetage exponentiel en la taille du graphe.

On étudie ensuite l'influence de la connaissance initiale du degré et on montre que si initialement, chaque sommet connaît son degré, alors il suffit de connaître une borne sur le diamètre pour pouvoir nommer ou élire dans un graphe minimal pour les pseudo-revêtements (Théorème 5.28). Contrairement aux modèles étudiés dans les Chapitres 2, 3 et 4, la connaissance initiale du degré ne permet pas d'obtenir un algorithme effectif d'élection dans ce modèle même si on connaît la taille exacte du graphe.

Une partie des résultats présentés dans ce chapitre a été publiée dans [Cha05].

5.1.2 Travaux Liés

Dans [AAD$^+$04, AAD$^+$06, AAE06b, AAE06a], Angluin et al. considèrent un modèle utilisant le même type de règles. Ils supposent que la mémoire de chaque processus est finie : l'ensemble des étiquettes apparaissant sur les sommets au cours de toute exécution est donc fini et indépendant de la taille du graphe.

De plus, ils supposent que si une configuration globale peut être infiniment souvent atteinte en un nombre fini d'étapes à partir de la configuration courante, alors cette configuration globale sera atteinte. De cette manière, il n'est pas nécessaire de considérer des relations de réétiquetage noetheriennes. Ils montrent que dans ce contexte, la topologie du graphe n'entre pas en compte et qu'ils peuvent toujours considérer que le graphe est complet.

Ils supposent qu'il existe une fonction de sortie qui permet d'attribuer à chaque configuration globale du graphe une valeur, qui est le résultat du calcul. Ils considèrent des exécutions qui calculent un résultat de manière *stabilisante* : ce sont les exécutions telles qu'à partir d'une certaine étape, la fonction de sortie renvoie toujours la même valeur. Cette notion de terminaison est à rapprocher de la terminaison implicite, au sens où les sommets ne savent pas si l'exécution est terminée. Il faut toutefois noter que la fonction de sortie est globale et par conséquent, les étiquettes peuvent être modifiées, tant que globalement, le résultat reste le même.

En supposant que chaque sommet v à initialement une valeur $\mathbf{input}(v)$ dans un aphabet d'entrée Σ, Angluin et al. déterminent quels prédicats satisfaits par le multi-ensemble des entrées peuvent être calculés de manière stabilisante. Ils montrent que dans ce modèle, les prédicats calculables de manière stabilisante sont les prédicats semi-linéaires portant sur la multiplicité de chaque symbole de Σ.

5.2 Pseudo-revêtements

Dans ce chapitre, les pseudo-revêtements sont les homomorphismes qui permettent de caractériser les graphes où on peut résoudre l'élection et le nommage avec des calculs

locaux sur les arêtes non-étiquetées.

Définition 5.1 *Un graphe simple G est un un* pseudo-revêtement *d'un graphe simple H à travers un homomorphisme $\gamma : G \to H$ et respectivement à G' si G' est un sous-graphe partiel de G (i.e., $V(G') = V(G)$ et $E(G') \subseteq E(G)$) et si la restriction $\gamma_{|G'}$ de γ à G' est un homomorphisme localement bijectif de G' dans H.*

Un graphe simple G est un pseudo-revêtement propre *de H si γ n'est pas un isomorphisme et G est* minimal pour les pseudo-revêtements *si G n'est un pseudo-revêtement propre d'aucun autre graphe simple.*

Naturellement, un graphe simple étiqueté (G, λ) est un pseudo-revêtement d'un graphe simple (H, η) à travers γ et respectivement à (G', λ) si γ conserve l'étiquetage.

Proposition 5.2 *Si le graphe $\mathbf{G}$ est un pseudo-revêtement d'un graphe $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$, alors pour chaque arête $\{v_1, v_2\} \in E(H)$, chaque sommet $u_1 \in \gamma^{-1}(v_1)$ (resp. $u_2 \in \gamma^{-1}(v_2)$) est adjacent à exactement un sommet $u_2 \in \gamma^{-1}(v_2)$ (resp. $u_1 \in \gamma^{-1}(v_1)$) dans le graphe G'.*

Par conséquent, le sous-graphe de G' induit par $\gamma^{-1}(v_1)$ et $\gamma^{-1}(v_2)$ est un couplage parfait dans le sous-graphe de G induit par $\gamma^{-1}(v_1)$ et $\gamma^{-1}(v_2)$.

Preuve : On considère un graphe $\mathbf{G}$ qui est un pseudo-revêtement d'un graphe $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$.

On considère deux sommets voisins $v_1, v_2 \in V(H)$. Pour tout $u_1 \in \gamma^{-1}(v_1)$, il existe un unique $u_2 \in \gamma^{-1}(v_2) \cap N_{G'}(v_1)$ puisque $\gamma_{|G'}$ est un homomorphisme localement bijectif de G' dans H. Par symétrie, pour tout $u_2 \in \gamma^{-1}(v_2)$, il existe un unique $u_1 \in \gamma^{-1}(v_1) \cap N_{G'}(v_2)$. □

Exemple 5.3 *Le graphe $\mathbf{G}$ de la Figure 19 est un pseudo-revêtement de $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$.*

Les arêtes représentées en gras dans la représentation de $\mathbf{G}$ (resp. $\mathbf{G'}$) sont les arêtes de $\mathbf{G}$ (resp. $\mathbf{G'}$) que γ envoie sur l'arête $\{1, 3\}$ de $\mathbf{H}$. On remarque sur cet exemple qu'un homomorphisme γ qui définit un pseudo-revêtement ne conserve pas forcément le degré, contrairement à un homomorphisme localement bijectif.

Le graphe $\mathbf{G}$ est un pseudo-revêtement propre de $\mathbf{H}$ et le graphe $\mathbf{H}$ est minimal pour les pseudo-revêtements.

On note que le graphe $\mathbf{G}$ de la Figure 19 est minimal pour les fibrations discrètes. Ainsi, il existe des graphes minimaux pour les fibrations discrètes qui ne sont pas minimaux pour les pseudo-revêtements. Réciproquement, le graphe de la Figure 15 est minimal pour les pseudo-revêtements (il a un nombre premier de sommets) mais il n'est pas minimal pour les fibrations discrètes non-triviales.

Si $\mathbf{G}$ est un pseudo-revêtement de $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$, les propositions suivantes sont des corollaires des Propositions 2.5 et 2.6, puisque $\mathbf{G'}$ est un revêtement de $\mathbf{H}$ à travers $\gamma_{|G'}$ et puisque $V(G') = V(G)$.

Proposition 5.4 *Si un graphe simple $\mathbf{G}$ est un pseudo-revêtement d'un graphe simple connexe $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$, alors γ est surjectif.*

Proposition 5.5 *Si un graphe simple $\mathbf{G}$ est un pseudo-revêtement d'un graphe simple connexe $\mathbf{H}$ à travers γ et respectivement à $\mathbf{G'}$, alors il existe une constante q telle que pour tout sommet $v \in V(H)$, $|\gamma^{-1}(v)| = q$.*

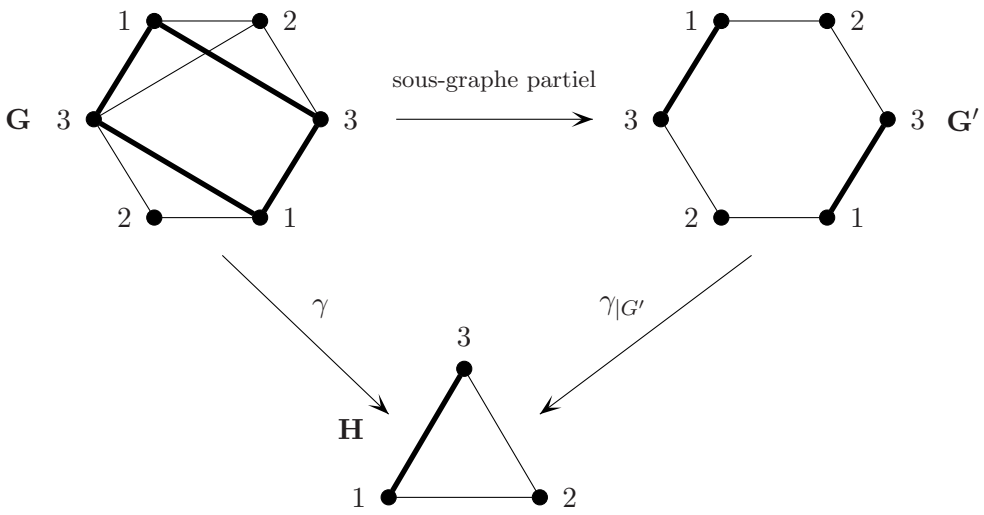


Fig. 19 – Le graphe **G** est un pseudo-revêtement de **H** à travers l'homomorphisme γ et respectivement à $\mathbf{G}'$ où γ envoie chaque sommet de **G** étiqueté i sur l'unique sommet de **H** étiqueté i.

Ainsi, les graphes simples ayant un nombre premier de sommets sont minimaux pour les pseudo-revêtements. Cependant, on va voir dans la Section 5.5 que les arbres ne sont pas minimaux pour les pseudo-revêtements, bien qu'ils soient minimaux pour les revêtements et pour les fibrations discrètes non-triviales.

La notion de coloration pseudo-régulière permet de caractériser les graphes minimaux pour les pseudo-revêtements en termes de colorations de graphes. Un étiquetage ℓ d'un graphe non-étiqueté est une coloration pseudo-régulière si deux sommets voisins ont des couleurs distinctes et si le graphe induit par deux classes de couleurs adjacentes admet un coupage parfait.

Définition 5.6 *Une* coloration pseudo-régulière *d'un graphe simple non-étiqueté G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un stable,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i, j]$ est un stable, ou alors $G[i, j]$ admet un couplage parfait.*

Dans la proposition suivante, on caractérise les graphes qui sont minimaux pour les pseudo-revêtements à l'aide de la notion de coloration pseudo-régulière. On rappelle qu'une coloration ℓ est dite propre si $|\ell(V(G))| < |V(G)|$.

Proposition 5.7 *Un graphe G est minimal pour les pseudo-revêtements si et seulement si G n'admet aucune coloration pseudo-régulière propre.*

Preuve : Étant donné un graphe G qui n'est pas minimal pour les pseudo-revêtements, on considère un sous graphe partiel G' de G, un graphe H et un homomorphisme $\gamma : G \to H$ tels que G est un pseudo-revêtement propre de H à travers γ et respectivement à G'. On va montrer que γ est une coloration pseudo-régulière de G et que l'ensemble des couleurs utilisés est $V(H)$. Puisque H est un graphe simple, pour toute arête $\{u, u'\} \in V(G)$,

$\gamma(u) \neq \gamma(u')$. De plus, pour tous $v, v' \in V(H)$, si $\{v, v'\} \notin E(H)$, alors $G[v, v']$ est un stable puisque γ est un homomorphisme. Et si $\{v, v'\} \in E(H)$, alors d'après la Proposition 5.2, on sait que les arêtes du sous-graphe induit par $\gamma^{-1}(v) \cup \gamma^{-1}(v')$ dans G' est un couplage parfait de $G[v, v']$. Finalement, puisque $|V(H)| < |V(G)|$, γ est une coloration pseudo-régulière propre de G.

Réciproquement, étant donnée une coloration pseudo-régulière propre ℓ d'un graphe G, on va définir un graphe H tel que G soit un pseudo-revêtement de H. On pose $V(H) = \ell(V(G))$. Pour tous $v, v' \in V(H)$, l'arête $\{v, v'\}$ appartient à $E(H)$ si et seulement si $G[v, v']$ n'est pas un stable. Le graphe H ainsi défini est un graphe simple. On pose $\gamma = \ell$ et pour toute arête $\{u, u'\} \in E(G)$, $G[\ell(u), \ell(u')]$ n'est pas un stable et par conséquent, $\{\gamma(u), \gamma(u')\} \in E(H)$. Ainsi, γ est un homomorphisme de G dans H. Pour définir G', il suffit de définir $E(G')$. Pour toute arête $\{v, v'\} \in E(H)$, il existe un ensemble d'arêtes $M_{\{v,v'\}} \in E(G)$ qui est un couplage parfait de $\ell^{-1}(v) \cup \ell^{-1}(v')$. On pose $E(G') = \bigcup_{\{v,v'\} \in E(H)} M_{\{v,v'\}}$. Étant donné un sommet $u \in V(G)$, on note $v = \gamma(u)$. Pour tout $v' \in N_H(v)$, il existe exactement une arête $\{u, u'\} \in M_{\{v,v'\}}$ et par conséquent, u a un unique voisin $u' \in \gamma^{-1}(v')$. Ainsi $\gamma_{|G'}$ est un homomorphisme localement bijectif de G' dans H et G n'est donc pas minimal pour les pseudo-revêtements. □

5.3 Calculs Locaux sur les Arêtes Non-Étiquetées

On rappelle qu'informellement, les calculs locaux sur les arêtes non-étiquetées sont les calculs réalisés en utilisant uniquement des règles de la forme présentée sur la Figure 18 : à chaque pas de calcul, l'étiquette de deux sommets voisins sont modifiées par l'application d'une règle qui ne dépend que de l'étiquette de ces deux sommets. On présente maintenant un définition plus formelle du modèle.

5.3.1 Définitions

Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les arêtes non-étiquetées si l'application d'une règle ne dépend que des étiquettes des extrémités d'une arête et si seulement ces deux étiquettes sont modifiées.

Définition 5.8 *Une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les arêtes non-étiquetées si la condition suivante est satisfaite : Pour tous graphes (G, λ), (G, λ'), (H, η), (H, η') et toutes arêtes $\{v_1, v_2\} \in E(G)$ et $\{w_1, w_2\} \in E(H)$, si les trois conditions suivantes sont vérifiées :*

1. *$\lambda(v_1) = \eta(w_1)$, $\lambda(v_2) = \eta(w_2)$, $\lambda'(v_1) = \eta'(w_1)$ et $\lambda'(v_2) = \eta'(w_2)$,*
2. *$\lambda(v) = \lambda'(v)$, pour tout $v \notin \{v_1, v_2\}$,*
3. *$\eta(w) = \eta'(w)$, pour tout $w \notin \{w_1, w_2\}$,*

alors $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ si et seulement si $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

Par définition, les *calculs locaux sur les arêtes non-étiquetées* correspondent aux relations de réétiquetage localement engendrées sur les arêtes non-étiquetées.

Une relation de réétiquetage localement engendrée sur les arêtes non-étiquetées peut être décrite par un ensemble récursif de règles de la forme présentées sur la Figure 18.

Réciproquement, un tel ensemble de règles induit une relation de réétiquetage localement engendrée sur les arêtes non-étiquetées. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

5.3.2 Pseudo-Revêtements et Calculs Locaux sur les Arêtes Non-Étiquetées

Le lemme suivant met en évidence le lien entre les calculs locaux sur les arêtes non-étiquetées et les pseudo-revêtements. C'est l'équivalent du lemme de relèvement d'Angluin [Ang80] pour les pseudo-revêtements.

Lemme 5.9 *On considère un graphe simple $\mathbf{G}_0 = (G, \lambda_0)$ qui est un pseudo-revêtement d'un graphe simple $\mathbf{H}_0 = (H, \eta_0)$ à travers un homomorphisme γ et respectivement à un sous graphe partiel $\mathbf{G}'_0 = (G', \lambda_0)$ de $\mathbf{G}$. Pour toute relation $\mathcal{R}$ de réétiquetage localement engendrée sur les arêtes, si $\mathbf{H}_0 \ \mathcal{R}^* \ \mathbf{H}_1 = (H, \eta_1)$, alors il existe $\mathbf{G}_1 = (G, \lambda_1)$ tel que $\mathbf{G}_0 \ \mathcal{R}^* \ \mathbf{G}_1$ et $\mathbf{G}_1$ est un pseudo-revêtement de $\mathbf{H}_1$ à travers γ et respectivement à $\mathbf{G}'_1 = (G', \lambda_1)$.*

Preuve : On considère deux graphes $(G, \lambda_0), (H, \nu_0)$ tels que (G, λ_0) est un pseudo-revêtement de (H, ν_0) à travers φ et respectivement à (G', λ_0). Il suffit de prouver le lemme pour un pas de calcul. On considère un pas de réétiquetage sur H qui modifie les étiquettes des extrémités d'une arête $\{v, v'\} \in E(H)$. On note η_1 l'étiquetage de H obtenu après ce pas de réétiquetage.

Puisque G' est un revêtement de H à travers $\gamma_{G'}$, tout sommet $u \in \gamma^{-1}(v)$ (resp. $u' \in \gamma^{-1}(v')$) a un unique voisin $u \in \gamma^{-1}(v') \cap N_{G'}(u)$ (resp. $u \in \gamma^{-1}(v) \cap N_{G'}(u')$) et on peut donc appliquer la règle de réétiquetage sur toutes les arêtes $\{u, u'\} \in E(G')$ telles que $\gamma(u) = v$ et $\gamma(u') = v'$. Ainsi, tout sommet $u \in \gamma^{-1}(v)$ a l'étiquette $\eta_1(v)$ et tout sommet $u' \in \gamma^{-1}(v')$ a l'étiquette $\eta_1(v')$; les étiquettes des autres sommets ne sont pas modifiées. On note λ_1, l'étiquetage de G ainsi obtenu et on sait que (G', λ_1) est toujours un revêtement de (H, η_1) à travers $\gamma_{G'}$. De plus, γ est toujours un homomorphisme de (G, λ_1) dans (H, λ_1) et par conséquent (G, λ_1) est un pseudo-revêtement de (H, η_1) à travers γ et respectivement à (G', λ_1). □

Le diagramme suivant représente la propriété du Lemme 5.9.

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow{\mathcal{R}^*} & \mathbf{G}' \\ \text{pseudo-revêtement}\downarrow & & \downarrow\text{pseudo-revêtement} \\ \mathbf{H} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{H}' \end{array}$$

5.4 Énumération, Nommage et Élection

On s'intéresse maintenant aux problèmes d'élection et de nommage dans le cadre des calculs locaux sur les arêtes non-étiquetées. Dans ce modèle, les graphes où on peut résoudre l'élection sont exactement les graphes où on peut résoudre le nommage et l'énumération. Cela est du à la Proposition 5.5 : si on arrive à distinguer un sommet en lui

attribuant une étiquette spéciale, alors on arrivera à donner des étiquettes uniques à tous les sommets.

On montre d'abord qu'il ne peut pas exister d'algorithme d'élection ou de nommage pour un graphe simple $\mathbf{G}$ qui n'est pas minimal pour les pseudo-revêtements. On présente ensuite un algorithme d'énumération à la Mazurkiewicz pour les graphes minimaux pour les pseudo-revêtements.

5.4.1 Résultats d'Impossibilité

Proposition 5.10 *Soit* $\mathbf{G}$ *un graphe simple étiqueté qui n'est pas minimal pour les pseudo-revêtements. Il n'existe pas d'algorithme de nommage, d'énumération ou d'élection pour le graphe* $\mathbf{G}$ *utilisant des calculs locaux sur les arêtes non-étiquetées.*

Preuve : On considère un graphe simple étiqueté $\mathbf{H}$ qui n'est pas isomorphe à $\mathbf{G}$ et tel que $\mathbf{G}$ soit un pseudo-revêtement de $\mathbf{H}$ à travers un homomorphisme γ et respectivement à un sous-graphe $\mathbf{G}'$ de $\mathbf{G}$. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux sur les arêtes non-étiquetées, on considère une exécution de $\mathcal{R}$ sur $\mathbf{H}$. Si cette exécution est infinie sur $\mathbf{H}$, alors d'après le Lemme 5.9, il existe une exécution infinie de $\mathcal{R}$ sur $\mathbf{G}$; auquel cas, $\mathcal{R}$ n'est ni un algorithme d'énumération, ni de nommage, ni d'élection.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur $\mathbf{H}$ et on considère la configuration finale $\mathbf{H}_1$. D'après le Lemme 5.9, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui permet d'atteindre une configuration $\mathbf{G}_1$ telle que $\mathbf{G}_1$ est un pseudo-revêtement de $\mathbf{H}_1$ à travers γ et respectivement à $\mathbf{G}'_1$. Si $\mathbf{G}_1$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe une arête $\{u, u'\} \in E(G)$ telle qu'on puisse appliquer une règle de réétiquetage sur $\{u, u'\}$. Dans ce cas là, on peut aussi appliquer cette même règle de réétiquetage sur l'arête $\{\gamma(u), \gamma(u')\} \in E(H)$ et la configuration $\mathbf{H}_1$ n'est pas une configuration finale de $\mathcal{R}$. Par conséquent, $\mathbf{G}_1$ est une configuration finale de $\mathcal{R}$. Mais puisque $\mathbf{G}_1$ n'est pas isomorphe à $\mathbf{H}_1$, on sait d'après la Proposition 5.5 que chaque étiquette de $\mathbf{H}_1$ apparaît au moins deux fois dans $\mathbf{G}_1$ et par conséquent, $\mathcal{R}$ n'est ni un algorithme de nommage, ni un algorithme d'énumération, ni un algorithme d'élection. □

5.4.2 Un Algorithme d'Énumération

On va maintenant décrire un algorithme d'énumération $\mathcal{M}$ pour les graphes minimaux pour les pseudo-revêtements. Cet algorithme s'inspire de l'algorithme de Mazurkiewicz.

Durant l'exécution de l'algorithme, chaque sommet v essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. Chaque sommet va ensuite échanger son numéro avec ses voisins. À chacune de ces transactions où deux sommets échangent leurs numéros, on va associer un identifiant tel que deux transactions impliquant le même sommet aient des identifiant différents. Grâce à ces échanges de numéros, chaque sommet peut construire sa *vue locale*, qui est ici un ensemble fini de couples d'entiers : à chaque échange de numéro avec un de ses voisins, le sommet ajoute le numéro de ce voisin associé à l'identifiant de la transaction à sa vue locale. Ensuite, chaque sommet va diffuser dans tout le graphe son numéro et sa vue locale. Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette $\lambda(u)$ et sa vue locale avec l'étiquette $\lambda(v)$ et la vue locale de v : si l'étiquette

de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre proche de l'ordre utilisé dans l'algorithme de Mazurkiewicz), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire) et la diffuse à nouveau avec sa vue locale. Lorsque l'exécution est terminée, si le graphe $\mathbf{G}$ est minimal pour les pseudo-revêtements, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

Une première difficulté par rapport aux versions précédentes est qu'un sommet ne peut pas faire la distinction entre ses voisins qui ont le même numéro. Ainsi, lorsqu'un sommet échange son numéro avec un de ses voisins, il doit aussi détruire de sa vue locale l'information relative à l'ancien numéro de ce voisin. Pour cela, l'algorithme qu'on présente détruit des informations qui peuvent être valides de la vue locale du sommet afin de s'assurer que l'information obsolète a bien été supprimée. Cette méthode permet de s'assurer que dans la configuration finale, la vue locale de chaque sommet ne contient que des informations pertinentes.

Par ailleurs, dans le modèle étudié ici, l'algorithme doit assurer que tous les numéros attribués au sommets apparaissent avec la même cardinalité dans la configuration finale. Pour cela, dans notre algorithme, on associe un identifiant à chaque transaction entre deux voisins. Cette méthode permet d'assurer que dans la configuration finale, tous les numéros associés aux sommets apparaissent avec la même cardinalité. Cependant, ce mécanisme implique que lorsqu'un sommet v met à jour sa vue locale en se synchronisant avec un voisin v', alors la vue locale de v' est aussi modifiée et des informations valides en sont effacées. Le sommet v' doit alors aussi remettre à jour sa vue locale, etc. Il faut donc montrer que l'utilisation des identifiants associées aux transactions entre voisins ne génère pas d'exécution infinie.

L'utilisation de ces identifiants pour assurer un résultat correct a un impact non-négligeable sur la complexité. En effet, certaines exécutions de l'algorithme nécessitent un nombre d'étapes de réétiquetage exponentiel en la taille du graphe.

Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda\colon V(G) \to L$ est un étiquetage initial, qui ne sera pas modifié par l'algorithme. Lors de l'exécution, chaque sommet va obtenir une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ est la *vue locale* du sommet v. Informellement, la vue locale contient l'information la plus récente que v a de ses voisins. Si le sommet v a un voisin v', un pas de réétiquetage sur $\{v, v'\}$ va permettre de générer un identifiant o et d'ajouter $(n(v'), o)$ à $N(v)$ ainsi que $(n(v), o)$ à $N(v')$. Ainsi $N(v)$ est toujours un ensemble fini de couples d'entiers.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

La différence entre les étiquettes utilisées ici et les étiquettes de l'algorithme de Mazurkiewicz [Maz97] sont les identifiants o qui apparaissent dans les vues locales des sommets. Dans le Chapitre 3, les numéros associés aux arêtes permettaient de distinguer les différents voisins d'un sommet et permettaient de définir les arêtes du graphe construit à partir de l'étiquetage final. Ici, les identifiants o vont permettre de déterminer quelles sont les arêtes de G qui devront être conservées dans $E(G')$ afin que G' soit un revêtement du graphe construit à partir de l'étiquetage final.

Un Ordre sur les Vues Locales

Comme pour l'algorithme de Mazurkiewicz [Maz97], les bonnes propriétés de l'algorithme reposent sur un ordre sur les vues locales, i.e., sur les ensembles finis de couples d'entiers. Pour cela, on considère que $\mathbb{N}^2$ est muni de l'ordre lexicographique usuel : $(n, o) < (n', o')$ si $n < n'$ ou si $n = n'$ et si $o < o'$.

Ensuite, on utilise le même ordre sur les ensembles que Mazurkiewicz : étant donnés deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ distincts, on dit que $N_1 \prec N_2$ si le maximum pour l'ordre lexicographique sur $\mathbb{N}^2$ de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2.

Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre $\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(\mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

Les Règles de Réétiquetage

On décrit maintenant l'algorithme d'énumération grâce à des règles de réétiquetage. La première règle $\mathcal{M}_0$ est une règle spéciale qui permet à chaque sommet de modifier son étiquette initiale $\lambda(v)$ pour obtenir l'étiquette $(\lambda(v), 0, \emptyset, \emptyset)$.

Les règles $\mathcal{M}_1$ et $\mathcal{M}_2$ sont proches des deux règles de l'algorithme de Mazurkiewicz [Maz97]. La première règle permet à deux sommets voisins v et v' de partager les informations contenues dans leurs boîtes-aux-lettres à propos des étiquettes apparaissant dans le graphe.

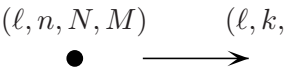
$\mathcal{M}_1$:

(ℓ_1, n_1, N_1, M_1) (ℓ_2, n_2, N_2, M_2) $\longrightarrow$ (ℓ_1, n_1, N_1, M') (ℓ_2, n_2, N_2, M')

Si $M_1 \neq M_2$ alors $M' := M_1 \cup M_2$.

La deuxième règle $\mathcal{M}_2$ ne dépend que de l'étiquette d'un seul sommet v. Elle permet à v de modifier son numéro si v n'a pas encore choisi de numéro (son numéro courant est 0), ou si la boîte-aux-lettres de v contient un message indiquant qu'il existe un autre sommet dans le graphe ayant le même numéro que v et qui a une étiquette supérieure à celle de v ou qui a une vue locale plus forte que celle de v.

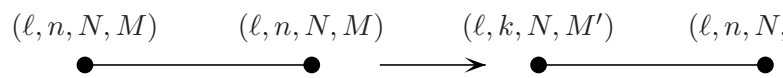
$\mathcal{M}_2$:



Si $n = 0$ ou $\exists (n, \ell', N') \in M$ tel que $(\ell, N) \prec (\ell', N')$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$ et
$M' := M \cup \{(k, \ell, N)\}$.

La troisième règle permet à un sommet v de modifier son numéro s'il a un voisin v' qui est exactement dans le même état, i.e., $(\lambda(v), n(v), N(v), M(v)) = (\lambda(v'), n(v'), N(v'), M(v'))$. Cette règle ne peut être appliquée que si on ne peut pas appliquer la règle $\mathcal{M}_2$ à v ou à v'.

$\mathcal{M}_3$:



Si $n > 0$ et
$\forall (n, \ell', N') \in M, (\ell', N') \preceq (\ell, N)$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$ et
$M' := M \cup \{(k, \ell, N)\}$.

La quatrième règle permet à deux sommets voisins v et v' d'échanger leurs numéros si une mise à jour de leurs vues locales est nécessaires, i.e., s'il n'existe pas de o tel que $(n(v), o) \in N(v')$ et $(n(v'), o) \in N(v)$. Cette règle ne peut être appliquée sur une arête $\{v, v'\}$ que si les trois règles précédentes ne peuvent pas être appliquées sur $\{v, v'\}$.

Le rôle de l'identifiant o associé à cet échange est d'assurer qu'à la fin de l'exécution, si un couple (n', o) appartient à la vue locale d'un sommet v alors v a un voisin v' tel que $n(v') = n'$ et $(n(v), o) \in N(v')$. De plus, dans la configuration finale, cet identifiant o permet de construire un couplage parfait entre les sommets étiquetés $n(v)$ et ceux étiquetés $n(v')$. Ainsi, dans la configuration finale, tous les numéros apparaissant dans le graphe apparaissent avec la même cardinalité.

On remarque que si un couple (n, o) est ajouté à la vue locale $N(v)$ de v, tous les couples (n', o') avec $n' \leq n$ sont supprimés de $N(v)$. La justification de toutes ces suppressions est la suivante. Si deux voisins v et v' se synchronisent et se rendent compte qu'ils n'ont pas un identifiant commun o tels que $(n(v), o) \in N(v')$ et $(n(v'), o) \in N(v)$, alors ils doivent se mettre d'accord sur un identifiant o et ajouter $(n(v), o)$ à $N(v')$ et $(n(v'), o)$ à $N(v)$. Mais on peut se retrouver dans deux situations. Si v échange son numéro avec v' pour la première fois, il suffit d'ajouter le couple $(n(v'), o)$ à $N(v)$. Cependant, si v et v' avaient déjà échangé leurs numéros auparavant et qu'entre-temps v' a changé de numéro, alors il faut non-seulement ajouter $(n(v'), o)$ à $N(v)$, mais il faut aussi supprimer l'ancien couple (n', o') correspondant au précédent échange de numéro avec v'. Le problème est que v n'a pas moyen de savoir lequel des couples apparaissant dans $N(v)$ correspond à v' et donc il ne peut pas savoir quel couple il doit effacer. Cependant, l'algorithme assure que lorsqu'un sommet change de numéro, son numéro ne peut qu'augmenter. Par conséquent, on sait que si on supprime tous les couples (n', o') tels que $n' \leq n(v')$, toute l'information obsolète relative à v' aura été effacée. Avec cette méthode, on peut aussi effacer des informations

valides sur certains voisins de v, mais ce n'est pas un problème puisque v peut récupérer cette information en se resynchronisant avec ces voisins là.

$\mathcal{M}_4$:

(ℓ_1, n_1, N_1, M) (ℓ_2, n_2, N_2, M) $\longrightarrow$ (ℓ_1, n_1, N'_1, M') (ℓ_2, n_2, N'_2, M')

Si $n_1, n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell'_1, N'_1) \in M, (\ell'_1, N'_1) \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell'_2, N'_2) \in M, (\ell'_2, N'_2) \preceq (\ell_2, N_2)$ et
$\nexists o \mid (n_2, o) \in N_1$ et $(n_1, o) \in N_2$
alors $o := 1 + \max\{o' \mid \exists (n', o') \in N_1 \cup N_2\}$,
$N'_1 := N_1 \setminus \{(n', o') \in N_1 \mid n' \leq n_2\} \cup \{(n_2, o)\}$,
$N'_2 := N_2 \setminus \{(n', o') \in N_2 \mid n' \leq n_1\} \cup \{(n_1, o)\}$ et
$M' := M \cup \{(n_1, \ell_1, N'_1), (n_2, \ell_2, N'_2)\}$.

5.4.3 Correction de l'Algorithme d'Énumération

On considère un graphe simple étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage de l'algorithme $\mathcal{M}$ décrit ci-dessus. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Propriétés Satisfaites lors de l'Exécution

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur le nombre d'étapes, rappelle quelque propriétés simples qui sont toujours satisfaites par l'étiquetage.

Lemme 5.11 *Pour chaque sommet v et chaque étape i,*

1. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
2. $\forall (n', o') \in N_i(v), n' \neq 0$ *et* $\exists (n', \ell', N') \in M_i(v)$,
3. $\nexists (n_i(v), o) \in N_i(v)$,
4. $\forall (n', o'), (n'', o'') \in N_i(v), n' \neq n''$.

L'algorithme $\mathcal{M}$ a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 5.12 *Pour chaque sommet v et chaque étape i,*
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

De plus, à chaque étape i, il existe un sommet v telle qu'au moins une de ces inégalités (ou inclusions) est stricte pour v.

Preuve : La propriété est trivialement vraie pour les sommets qui ne sont pas réétiquetés lors de la $(i+1)$ème étape. De plus, il est facile de voir que quelque soit la règle appliquée à l'étape $i+1$, on a toujours $M_i(v) \subseteq M_{i+1}(v)$ pour tout sommet $v \in V(G)$.

Pour chaque sommet v tel que $n_i(v) \neq n_{i+1}(v)$, alors la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ a été appliquée à v et on sait que $n_{i+1}(v) = 1 + \max\{n' \mid \exists(n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 5.11, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Pour chaque sommet v tel que $N_i(v) \neq N_{i+1}(v)$, la règle $\mathcal{M}_4$ a été appliquée au sommet v et à un de ses voisins v' ; on note o l'identifiant associé à cet échange de numéro. Pour chaque $(n', o') \in N_i(v)$, si $n' > n_{i+1}(v')$, alors $(n', o') \in N_{i+1}(v)$ et on sait que $(n_{i+1}(v), o) \in N_{i+1}(v) \setminus N_i(v)$ puisque $o = 1 + \max\{o' \mid \exists(n', o') \in N_i(v) \cup N_i(v')\}$. Par conséquent, on a $\max(N_i(v) \triangle N_{i+1}(v)) = (n_{i+1}(v'), o) \in N_{i+1}(v)$ et donc $N_i(v) \prec N_{i+1}(v)$.

Puisque chaque application d'une règle modifie l'étiquette d'au moins un sommet v, on sait que l'une de ces inégalités est stricte pour v. □

Les informations dont dispose chaque sommet v dans sa boîte-aux-lettres permettent d'obtenir des informations vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet v' tel que $n_i(v') = m'$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet v' tel que $n_i(v') = m'$.

Lemme 5.13 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid \forall(u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, il existe exactement un élément $(u, i_0) \in U'$ puisqu'à chaque étape, le numéro d'au plus un sommet peut être modifié. Le numéro $n_{i_0}(u) = m$ a donc été modifié à l'étape $i_0 + 1$, mais puisque à cette étape, le sommet u n'avait aucun voisin avec le même numéro m, la règle $\mathcal{M}_3$ n'a pu être appliquée, et par maximalité de $(\lambda(u), N_{i_0}(u))$, la règle $\mathcal{M}_2$ n'a pas non plus pu être appliquée à u à l'étape i_0. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 5.14 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. La propriété est trivialement vraie à l'étape $i + 1$ pour tout sommet $w \in V(G)$ dont l'étiquette n'est pas modifiée à l'étape $i + 1$. Soit v un sommet dont l'étiquette est modifiée à l'étape $i + 1$.

Si la règle $\mathcal{M}_1$ est appliquée à l'étape $i+1$ à v et à un de ses voisins v', alors $M_{i+1}(v) = M_i(v) \cup M_i(v')$ et la propriété est vérifiée à l'étape $i + 1$ puisqu'elle était vraie pour v et

v' à l'étape i.

Si la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ est appliquée à v à l'étape $i+1$, alors $M_{i+1}(v) = M_i(v) \cup \{(n_{i+1}(v) = 1+\max\{m \mid (m,\ell,N) \in M_i(v)\}, \lambda(v), N_i(v))\}$, et par conséquent pour chaque $m \in M_{i+1}(v)$, la propriété reste vraie.

Si la règle $\mathcal{M}_4$ est appliquée à v à l'étape $i+1$, pour tout $(m,\ell,N) \in M_{i+1}(v)$, il existe $(m,\ell,N') \in M_i(v)$ et la propriété reste donc vraie. □

Dans le lemme suivant, on montre que si un couple (n,o) apparaît dans la vue locale $N(v)$ d'un sommet v, alors ou bien v a un voisin v' tel que $n(v') = n$, ou bien la règle $\mathcal{M}_4$ peut être appliquée au sommet v et à un de ses voisins v'.

Lemme 5.15 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(n,o) \in N_i(v)$, il existe $v' \in N_G(v)$ tel que, ou bien $n_i(v') = n$, ou bien $n_i(v') > \max\{n' \mid \exists(n',o') \in N_i(v)\}$.*

Preuve : Soit i_0 l'étape où (n,o) a été ajouté à $N(v)$: $\forall j \geq i_0, (n,o) \in N_j(v)$ et $(n,o) \notin N_{i_0-1}(v)$. Cela signifie qu'à l'étape i_0, la règle $\mathcal{M}_4$ a été appliquée à v et v' avec $n_{i_0}(v') = n$. D'après le Lemme 5.11, il existe $(n,\ell,N) \in M_{i_0}(v')$. Si $n_i(v') = n_{i_0}(v') = n$, alors la propriété est vérifiée.

Dans le cas contraire, pour chaque étape $j \in [i_0,i]$, on pose $m(j) = \max\{n' \mid \exists(n',o') \in N_j(v)\}$. On sait que $m_{i_0}(v) \geq n$ et qu'il existe $(m_{i_0}(v),\ell,N) \in M_{i_0}(v')$. Si un couple (n',o') est ajouté à $N(v)$ lors d'une étape $i_1 \in [i_0,i]$, alors la règle $\mathcal{M}_4$ a été appliquée à v et à un de ses voisins lors de l'étape i_1. De plus $n' < n$, puisque $(n,o) \in N_{i_1}(v)$ et par conséquent, $m_{i_0}(v) = m_i(v)$.

Puisque $n_i(v') \neq n_{i_0}(v')$, une des règles $\mathcal{M}_2$ ou $\mathcal{M}_3$ a été appliquée à v' lors d'une étape $i_2 \in [i_0,i]$. Par ailleurs, on sait qu'il existe $(m_{i_0}(v),\ell,N) \in M_{i_0}(v') \subseteq M_{i_2}(v')$ et par conséquent, $n_i(v') \geq n_{i_2}(v') > m_{i_0}(v) = m_i(v) = \max\{n' \mid \exists(n',o') \in N_i(v)\}$. La propriété est donc vérifiée. □

Terminaison

On veut maintenant montrer que toute exécution de l'algorithme $\mathcal{M}$ termine sur $\mathbf{G}$. D'après les Lemmes 5.13 et 5.14, on voit qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1,k]$ ou un ensemble $[0,k]$ avec $k \leq |V(G)|$. Par conséquent, d'après le Lemme 5.12, on sait qu'il existe une étape i_0 telle que pour tout sommet v et toute étape $i \geq i_0$, $n_{i+1}(v) = n_i(v)$.

Pour montrer que l'algorithme termine toujours, il est suffisant de montrer que la règle $\mathcal{M}_4$ ne peut pas être appliquée infiniment souvent. En effet, cela implique alors que les identifiants o associés à chaque échange de numéro ne peuvent prendre qu'un nombre fini de valeurs. Et dans ce cas, les vues locales et les boîtes-aux-lettres des sommets ne peuvent prendre qu'un nombre fini de valeurs et on peut assurer que l'algorithme termine d'après le Lemme 5.12.

On commence par définir un pré-ordre $\trianglelefteq$ sur les arêtes de $\mathbf{G}$ de la manière suivante. Étant données deux arêtes $\{v,w\},\{v',w'\} \in E(G)$ telles que $\{n_{i_0}(v), n_{i_0}(w)\} \neq \{n_{i_0}(v'), n_{i_0}(w')\}$, on dit que $\{v,w\} \triangleleft \{v',w'\}$ si $\max\{n_{i_0}(v), n_{i_0}(w)\} \triangle \{n_{i_0}(v'), n_{i_0}(w')\} \in \{n_{i_0}(v), n_{i_0}(w)\}$. Si $\{v,w\} \triangleleft \{v',w'\}$ ou si $\{n_{i_0}(v), n_{i_0}(w)\} = \{n_{i_0}(v'), n_{i_0}(w')\}$, on dit que $\{v,w\} \trianglelefteq \{v',w'\}$.

Dans le lemme suivant, on montre qu'une fois que les sommets ne modifient plus leurs numéros, la règle $\mathcal{M}_4$ ne peut pas être appliquée infiniment souvent sur chaque arête.

Lemme 5.16 *Pour toute arête $\{v_0, w_0\} \in E(G)$ et toute étape $i_1 \geq i_0$ telles que la règle $\mathcal{M}_4$ n'est pas appliquée lors d'une étape $i \geq i_1$ sur une arête $\{v, w\} \rhd \{v_0, w_0\}$, il existe au plus une étape $i \geq i_1$ lors de laquelle la règle $\mathcal{M}_4$ est appliquée sur l'arête $\{v_0, w_0\}$.*

Preuve : On considère une arête $\{v_0, w_0\} \in E(G)$ et une étape $i_1 \geq i_0$ telles que la règle $\mathcal{M}_4$ n'est pas appliquée lors d'une étape $i \geq i_1$ sur une arête $\{v, w\} \rhd \{v_0, w_0\}$. On suppose que $n_{i_0}(v_0) > n_{i_0}(w_0)$ et on note $n_0 = n_{i_0}(v_0)$ et $m_0 = n_{i_0}(w_0)$.

On remarque que pour toute arête $\{v, w\}$ sur laquelle la règle $\mathcal{M}_4$ est appliquée à une étape $i > i_1$, on a $n_i(v) = n_{i_0}(v) \leq n_0$ et $n_{i_0}(w) \leq n_0$. Ainsi, pour toute étape $i \geq i_1$ et tout sommet $v \in V(G)$, pour tout $n > n_{i_0}(v)$, $(n, o) \in N_i(v)$ si et seulement si $(n, o) \in N_{i_1}(v)$.

De plus pour tout sommet v tel que $n_{i_0}(v) = n_0$, si la règle $\mathcal{M}_4$ est appliqué a v et à un de ses voisins w à une étape $i \geq i_1$, alors $n_i(w) = n_{i_0}(w) \leq m_0$. Ainsi pour toute étape i, pour tout $n > m_0$, $(n, o) \in N_i(v)$ si et seulement si $(n, o) \in N_{i_1}(v)$.

On considère maintenant une arête $\{v, w\} \in E(G)$ telle que $n_{i_0}(v) = n_0$ et $m_{i_0}(v) = m_0$ et telle qu'il existe une étape $j_1 > i_1$ lors de laquelle la règle $\mathcal{M}_4$ a été appliquée sur l'arête $\{v, w\}$.

On montre par induction sur j que pour toute étape $j \geq j_1$, il existe un identifiant o tel que $(m_0, o) \in N_j(v)$. De par la définition de j_1, il existe $(m_0, o) \in N_{j_1}(v)$ et $(n_0, o) \in N_{j_1}(w)$. De plus, si la vue locale de v est modifiée lors de l'étape $j + 1$, alors la règle $\mathcal{M}_4$ a été appliqué à v et à l'un de ses voisins w' lors de cette étape. On sait déjà que $n_{i_0}(w') \leq m_0$ et si $n_{i_0}(w') < m_0$, alors il existe $(m_0, o) \in N_j(v)$ et puisque $n_{i_0}(w') < m_0$, $(m_0, o) \in N_{j+1}(v)$. Si $n_j(w') = m_0$, alors un identifiant o est créé lors de cette étape et $(m_0, o) \in N_{j+1}(v)$. De la même manière, on peut montrer que pour toute étape $j \geq j_1$, il existe un identifiant o tel que $(n_0, o) \in N_j(w)$.

On suppose maintenant que la règle $\mathcal{M}_4$ est appliquée lors de deux étapes $j_2 > j_1 \geq i_1$ sur l'arête $\{v_0, w_0\}$. On sait qu'il existe un identifiant o_0 tel que $(m_0, o_0) \in N_{j_1}(v_0)$ et $(n_0, o_0) \in N_{j_1}(w_0)$. Puisque la règle $\mathcal{M}_4$ peut être appliquée lors de l'étape j_2, il existe deux identifiants distincts $o_1, o_2 \geq o_0$ tels que $(n_0, o_1) \in N_{j_2-1}(w_0)$ et $(m_0, o_2) \in N_{j_2-1}(v_0)$.

On considère le cas où $o_2 > o_1$. Puisque $o_2 > o_1 \geq o_0$, il existe une étape j telle que $j_1 < j < j_2$ lors de laquelle la règle $\mathcal{M}_4$ a été appliquée entre le sommet v_0 et un de ses voisins w tel que $n_{i_0}(w) = m_0$. Puisque la règle $\mathcal{M}_4$ a pu être appliquée et que $(m_0, \lambda(w_0), N_{j_1}(w_0)) \in M_{j_1}(v_0) \subseteq M_j(v_0) = M_j(w)$, cela signifie que $(\lambda(w_0), N_{j_1}(w_0)) \preceq (\lambda(w), N_{j-1}(w)) \prec (\lambda(w), N_j(w))$. De plus, on sait que $(n_0, o_2) \in N_j(w)$ et que $(n_0, \lambda(w), N_j(w)) \in M_j(v_0) \subseteq M_{j_2-1}(v_0) = M_{j_2-1}(w_0)$. On rappelle que pour tout $n > n_0$, $(n, o) \in N_{j_2-1}(w_0)$ si et seulement si $(n, o) \in N_{j_1}(w_0)$ et on sait que $(n_0, o_2) \in N_j(w) \setminus N_{j_2-1}(w_0)$. Ainsi puisque $o_2 > o_1$ et que $(\lambda(w_0), N_{j_1}(w_0)) \prec (\lambda(w), N_j(w))$, on sait que $(\lambda(w_0), N_{j_2}(w_0)) \prec (\lambda(w), N_j(w))$. Par conséquent, la règle $\mathcal{M}_4$ ne peut pas être appliquée sur l'arête $\{v_0, w_0\}$ lors de l'étape j_2.

Dans le cas où $o_1 > o_2$, on peut de la même manière montrer qu'il existe $(n_0, \ell, N) \in M_{j_2-1}(v_0)$ tel que $(\lambda(v_0), N_{j_2-1}(v_0)) \prec (\ell, N)$ et la règle $\mathcal{M}_4$ ne peut donc pas non plus être appliquée sur l'arête $\{v_0, w_0\}$ lors de l'étape j_2. □

On peut désormais montrer que toute exécution de l'algorithme $\mathcal{M}$ sur $\mathbf{G}$ termine. En effet, s'il existe une exécution infinie de $\mathcal{M}$ sur $\mathbf{G}$, alors on considère l'ensemble E_1 des

arêtes sur lesquels la règle $\mathcal{M}_4$ est appliquée infiniment souvent. On considère une étape i_1 telle que pour toute étape $i > i_1$, les règles $\mathcal{M}_2$ et $\mathcal{M}_3$ ne sont pas appliquées à l'étape i et si la règle $\mathcal{M}_4$ est appliquée sur une arête $\{v, w\}$ à l'étape i, alors $\{v, w\} \in E_1$. On considère alors une arête de E_1 qui est un élément maximal pour l'ordre $\trianglelefteq$ et d'après le Lemme 5.16, on sait qu'on ne peut pas appliquer la règle $\mathcal{M}_4$ sur $\{v, w\}$ infiniment souvent : l'ensemble E_1 est donc vide et toute exécution de $\mathcal{M}$ sur $\mathbf{G}$ termine.

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final.

Lemme 5.17 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. *$M_\rho(v) = M_\rho(v')$,*
3. *$(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,*
4. *si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et $N_\rho(v) = N_\rho(v')$,*
5. *$(n, o) \in N_\rho(v)$ si et seulement s'il existe $w \in N_G(v)$ tel que $n_\rho(w) = n$; auquel cas, $(n_\rho(v), o) \in N_\rho(w)$.*

Preuve :

1. D'après les Lemmes 5.13 et 5.14 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
2. Dans le cas contraire, la règle $\mathcal{M}_1$ peut être appliquée.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 5.11.
4. Dans le cas contraire, la règle $\mathcal{M}_2$ peut être appliquée à v ou à v'.
5. D'après le Lemme 5.15 et puisque les règles $\mathcal{M}_3$ et $\mathcal{M}_4$ ne peuvent plus être appliquées.

□

Grâce au Lemme 5.17, on va montrer comment construire un graphe $\mathbf{H}$ à partir de l'étiquetage final et on va montrer que $\mathbf{G}$ est un pseudo-revêtement de $\mathbf{H}$.

Proposition 5.18 *Étant donné un graphe simple $\mathbf{G}$, on peut construire, à partir de l'étiquetage final obtenu après une exécution ρ de $\mathcal{M}$, un graphe simple $\mathbf{H}$ tel que $\mathbf{G}$ est un pseudo-revêtement de $\mathbf{H}$.*

Preuve : On utilise les notations du Lemme 5.17.

On considère le graphe H défini par $V(H) = \{m \in \mathbb{N} \mid \exists v \in V(G), n_\rho(v) = m\}$ et $E(H) = \{\{m, m'\} \mid \exists v, v' \in V(G); n_\rho(v) = m, n_\rho(v') = m' \text{ et } \{v, v'\} \in E(G)\}$. On définit un étiquetage η de H en posant $\eta(n_\rho(v)) = \lambda(v)$; d'après le Lemme 5.17, si deux sommets $v, v' \in V(G)$ ont le même numéro, alors $\lambda(v) = \lambda(v')$ et par conséquent, cet étiquetage est bien défini.

D'après le Lemme 5.17, $\{m, m'\} \in E(H)$ si et seulement s'il existe $v, v' \in V(G)$ et un identifiant o tels que $n_\rho(v) = m, n_\rho(v') = m'$, $(m', o) \in N_\rho(v)$ et $(m, o) \in N_\rho(v')$. D'après

le Lemme 5.11, on sait qu'il n'existe aucun $\{n, n\} \in E(H)$: le graphe H ne contient pas de boucle. De plus, par définition, $E(H)$ ne contient pas d'arêtes multiples.

On considère maintenant la fonction $n_\rho : V(G) \to V(H)$. Par définition de H, on sait que si $\{v, v'\} \in E(G)$, alors $\{n_\rho(v), n_\rho(v')\} \in E(G)$. De plus, pour tout $v \in V(G)$, $\eta(n_\rho(v)) = \lambda(v)$ et par conséquent, n_ρ est un homomorphisme de $\mathbf{G}$ dans $\mathbf{H}$.

On va maintenant définir l'ensemble d'arêtes $E(G')$. Pour chaque arête $\{m, m'\} \in E(H)$, on veut définir un couplage parfait entre les sommets de $n_\rho^{-1}(m)$ et les sommets de $n_\rho^{-1}(m')$. Étant donné un sommet v tel que $n_\rho(v) = m$, on sait qu'il existe un identifiant o tel que $(m', o) \in N_\rho(v)$ et on note $i_{m'}(v)$ l'étape où (m', o) a été ajouté à $N(v)$, i.e., $(m', o) \in N_{i_{m'}(v)}(v) \setminus N_{i_{m'}(v)-1}(v)$. Cela signifie que lors de l'étape $i_{m'}(v)$, la règle $\mathcal{M}_4$ a été appliquée à v et à un de ses voisins v' tel que $n_{i_{m'}(v)}(v') = m'$. D'après le Lemme 5.15 et puisque l'exécution est terminée, on sait que $n_\rho(v') = n_{i_{m'}(v)}(v') = m'$, que $(m, o) \in N_{i_m(v')}(v') \setminus N_{i_m(v')-1}(v')$ et que $(m, o) \in N_\rho(v')$. Par conséquent, on sait que $i_{m'}(v) = i_m(v')$.

On définit les arêtes de $E(G')$ comme ceci : une arête $\{v, v'\}$ appartient à $E(G)$ si et seulement si $i_{n_\rho(v')}(v) = i_{n_\rho(v)}(v')$. En d'autres termes, une arête $\{v, v'\}$ appartient à $E(G)$ si et seulement si la règle $\mathcal{M}_4$ a été appliquée sur $\{v, v'\}$ et que l'identifiant créé à ce moment là apparaît toujours dans les vues locales des sommets v et v' dans la configuration finale. Puisque lors d'une étape de calcul, seules les étiquettes des extrémités d'une arête sont modifiées, il est clair que pour tout $\{m, m'\} \in E(H)$, chaque sommet de $n_\rho^{-1}(m)$ est adjacent à exactement un sommet de $n_\rho^{-1}(m')$ dans G'. Par conséquent, G' est bien un revêtement de H à travers n_ρ et puisque n_ρ préserve l'étiquetage, $\mathbf{G}'$ est un revêtement de $\mathbf{H}$ à travers n_ρ.

Ainsi, le graphe $\mathbf{G}$ est un pseudo-revêtement de $\mathbf{H}$ à travers n_ρ et respectivement à $\mathbf{G}'$. □

On considère maintenant un graphe $\mathbf{G}$ qui est minimal pour les pseudo-revêtements. Pour chaque exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, le graphe obtenu à partir de l'étiquetage final est isomorphe à $\mathbf{G}$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique. L'algorithme $\mathcal{M}$ permet de résoudre le nommage sur la famille des graphes minimaux pour les pseudo-revêtements, mais si aucune information à propos de $\mathbf{G}$ n'est disponible, les sommets ne peuvent pas détecter la terminaison.

Cependant, il est possible de détecter la terminaison pour un graphe $\mathbf{G}$ donné (l'algorithme dépend alors de $\mathbf{G}$). En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 5.13 et 5.14, il sait que tous les sommets de $\mathbf{G}$ ont un numéro unique qui ne va plus être modifié. Dans ce cas là, on peut aussi résoudre le problème de l'élection, puisque ce sommet peut prendre l'étiquette ÉLU et diffuser ensuite l'information qu'un sommet a été élu.

Par ailleurs, d'après la Proposition 5.10, on sait que pour tout graphe $\mathbf{G}$ qui n'est pas minimal pour les pseudo-revêtements, il n'existe aucun algorithme utilisant des calculs locaux sur les arêtes non-étiquetées qui permettent de résoudre les problèmes du nommage ou de l'énumération sur $\mathbf{G}$. On a donc prouvé le théorème suivant.

Théorème 5.19 *Pour tout graphe simple étiqueté* $\mathbf{G}$*, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour* $\mathbf{G}$ *utilisant des calculs locaux sur les arêtes non-étiquetées,*
2. *il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour* $\mathbf{G}$ *utilisant des calculs locaux sur les arêtes non-étiquetées,*
3. $\mathbf{G}$ *est minimal pour les pseudo-revêtements.*

Remarque 5.20 *Étant donné un graphe* $\mathbf{G}$ *minimal pour les pseudo-revêtements, pour détecter que l'algorithme* $\mathcal{M}$ *a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de* $\mathbf{G}$*. Ainsi, l'algorithme* $\mathcal{M}$ *permet de résoudre l'élection ainsi que le nommage avec détection de la terminaison sur les graphes minimaux pour les pseudo-revêtements de taille donnée.*

5.4.4 Complexité

On s'intéresse à la complexité de l'algorithme $\mathcal{M}$ présenté ci-dessus. Dans le cadre des calculs locaux, on s'intéresse au nombre de pas de réétiquetages effectuées lors d'une exécution. La proposition suivante donne une borne supérieure sur le nombre de pas de réétiquetages de toute exécution de l'algorithme $\mathcal{M}$ sur un graphe à m arêtes.

Proposition 5.21 *Pour tout graphe* $\mathbf{G}$ *à* m *arêtes, durant toute exécution de l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes non-étiquetées,* $2^{O(m)}$ *règles sont appliquées.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et à m arêtes ainsi qu'une exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$. D'après les Lemmes 5.13 et 5.14, on sait que les règles $\mathcal{M}_2$ et $\mathcal{M}_3$ ne peuvent pas être appliquées plus de $\frac{n(n-1)}{2}$ fois durant l'exécution ρ.

Entre deux étapes où une des règles $\mathcal{M}_2, \mathcal{M}_3$ est appliquée, d'après le Lemme 5.16, la règle $\mathcal{M}_4$ est appliquée au plus 2^m fois. Ainsi, la règle $\mathcal{M}_4$ est appliquée $O(n^2 2^m) = 2^{0(m)}$ fois durant l'exécution ρ.

À chaque fois qu'un sommet v modifie son numéro ou sa vue locale, un couple (n_0, ℓ, N) est ajouté $M(v)$. Pour chacun de ces couples, la règle $\mathcal{M}_1$ est appliquée au plus n fois.

Ainsi, durant toute exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, $2^{0(m)}$ règles sont appliquées. □

Contrairement aux modèles étudiés dans les Chapitres 2, 3 et 4, on remarque que certaines exécutions de l'adaptation de l'algorithme de Mazurkiewicz qu'on a présenté dans ce chapitre nécessitent un nombre de pas de réétiquetages qui peut être exponentiel en la taille du graphe. Cela est du au fait que dans les Chapitres 2, 3 et 4, lorsqu'un sommet met à jour sa vue locale, il sait quelle information n'est plus à jour et il peut alors supprimer seulement cette information. Au contraire, dans le cadre des calculs locaux sur les arêtes non-étiquetées, à chaque fois qu'un sommet met à jour sa vue locale, il supprime les informations obsolètes, mais il supprime aussi des informations qui peuvent être valides. De plus, à chaque fois qu'un sommet v applique la règle $\mathcal{M}_4$ avec un de ses voisins v' pour mettre à jour sa vue locale, la vue locale de v' est modifiée pour qu'il existe un entier o tel que $(n(v'), o) \in N(v)$ et $(n(v), o) \in N(v')$. Certaines informations pertinentes peuvent alors être supprimées de la vue locale du sommet v' qui doit alors remettre à jour sa vue locale. Ainsi, à chaque fois qu'un sommet modifie son numéro, cela peut nécessiter que la règle $\mathcal{M}_4$ soit appliquée sur toutes les arêtes de graphes et que cette règle soit appliquée un nombre exponentiel de fois sur les arêtes minimales pour l'ordre $\trianglelefteq$.

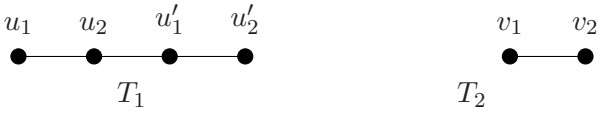


Fig. 20 – L'arbre T_1 n'est pas minimal pour les pseudo-revêtements puisque T_1 est un pseudo-revêtement de T_2.

On s'intéresse maintenant à la mémoire nécessaire à chaque sommet pour stocker son étiquette. On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets et à toutes les arêtes de G). On obtient ici aussi une borne supérieure sur la mémoire nécessaire à chaque sommet plus élevée que dans les Chapitres 2, 3 et 4.

Proposition 5.22 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets et* m *arêtes dont le degré maximal est* Δ*, l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes étiquetées nécessite* $O(\Delta nm)$ *bits de mémoire par sommet.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et m arêtes dont le degré maximal est Δ. D'après le Lemme 5.16, la valeur maximale des numéros o générés par l'application de la règle $\mathcal{M}_4$ sont en $2^{O(m)}$. Ainsi, la vue locale de chaque sommet peut être représentée en utilisant $O(\Delta m)$ bits.

Chaque sommet peut ne conserver dans sa boîte-aux-lettres que l'information utile, i.e., l'ensemble $\{(n_0, \ell, N) \in M(v) \mid \forall (n_0, \ell', N') \in M(v), (\ell', N') \preceq (\ell, N)\}$. Ainsi, dans la boîte-aux-lettres de chaque sommet, il existe au plus n triplets (n_0, ℓ, N) dont la taille est en $O(\Delta m)$ bits. Par conséquent, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta nm)$ bits. □

5.5 Connaissance Initiale du Degré

On étudie dans cette partie l'influence de la connaissance initiale du degré sur les calculs locaux sur les arêtes non-étiquetées. C'est à dire qu'initialement, chaque sommet v d'un graphe $\mathbf{G} = (G, \lambda)$ connaît son degré, i.e., son degré fait partie de son étiquette initiale $\lambda(v)$.

5.5.1 Élection dans la Famille des Arbres

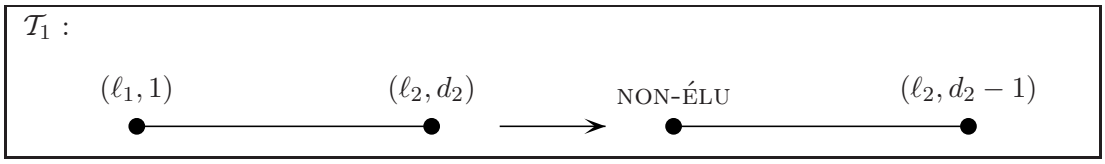
Il existe des arbres qui ne sont pas minimaux pour les pseudo-revêtements. Par exemple, considèrons le graphe non-étiqueté T_1 de la Figure 20. On note T'_1 le sous graphe partiel de T_1 défini par $E(T'_1) = \{\{u_1, u_2\}, \{u'_1, u'_2\}\}$ et γ l'homomorphisme de T_1 dans T_2 qui envoie u_1 (resp. u'_1) sur v_1 et u_2 (resp. u'_2) sur v_2. Il est facile de voir que T'_1 est un revêtement simple de T_2 et par conséquent, T_1 est un pseudo-revêtement de T_2 à travers γ et respectivement à T'_1. On a ainsi exhibé un arbre qui n'est pas minimal pour les pseudo-revêtements et il est facile de voir que toutes les chaines de longueurs paires ne sont pas minimales pour les pseudo-revêtements.

En utilisant la même preuve que dans le Chapitre 3, on peut montrer maintenant que même si on considère seulement les arbres minimaux pour les pseudo-revêtements, il n'existe pas d'algorithme utilisant des calculs locaux sur les arêtes non-étiquetées qui permette de résoudre le problème de l'élection sur la famille des arbres minimaux pour les pseudo-revêtements. En effet, les arbres de la Figure 12 considérés dans la preuve du Chapitre 3 sont tous les deux minimaux pour les revêtements puisqu'ils ont tous les deux un nombre premier de sommets.

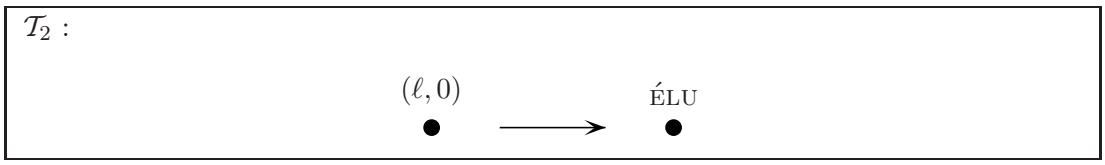
On va montrer maintenant que si initialement, tous les sommets connaissent leur degré, il existe un algorithme d'élection pour la famille des arbres utilisant des calculs locaux sur les arêtes non-étiquetées. L'algorithme est inspiré de l'algorithme d'élection dans les arbres présenté dans le Chapitre 1.

On considère la famille des arbres $\mathcal{T}$ étiquetés de telle sorte que pour tout arbre $\mathbf{T} \in \mathcal{T}$, $\mathbf{T} = (T, (\lambda, d))$ où λ est une fonction d'étiquetage habituel et d est la fonction qui associe à chaque sommet son degré dans l'arbre.

Pour tout sommet $v \in V(G)$, l'étiquette initiale d'un sommet est donc $(\lambda(v), d(v))$ où $d(v)$ est le degré du sommet v dans $\mathbf{G}$. On va montrer qu'avec la connaissance du degré, il est facile de coder l'algorithme d'élection dans les arbres présenté dans le Chapitre 1. La première règle est une règle «d'élagage» qui permet à un sommet qui n'a qu'un seul voisin actif (i.e. dont l'étiquette n'est pas terminale) dans l'arbre de prendre l'étiquette NON-ÉLU et d'informer son voisin qu'il a un voisin actif de moins



La seconde règle permet à un sommet qui n'a plus aucun voisin actif dans l'arbre de prendre l'étiquette ÉLU.



On considère un arbre $\mathbf{T}$ et une exécution de l'algorithme décrit précédemment sur $\mathbf{T}$. Pour chaque étape i de l'exécution, on note $(\lambda(v), d_i(v))$ l'étiquette du sommet v après le $i$ème pas de réétiquetage.

On observe qu'à chaque application d'une des deux règles, le nombre de sommets qui n'ont pas d'étiquettes finales diminue strictement. On est donc assuré que toute exécution de l'algorithme termine. On dit qu'un sommet est *actif* si son étiquette n'est pas finale (i.e., son étiquette est différente de ÉLU et de NON-ÉLU). On montre dans le lemme suivant un invariant qui permet de prouver la correction de l'algorithme.

Lemme 5.23 *Pour toute étape i, le graphe induit par l'ensemble des sommets actifs est un arbre et pour tout sommet v, $d_i(v)$ est le nombre de voisins actifs de v.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, tous les sommets sont actifs et par conséquent, la propriété est bien vérifiée puisque $d(v)$ correspond au degré de v dans T. On suppose que la propriété est vérifiée à l'étape i.

Si la règle $\mathcal{T}_1$ est appliquée à l'étape $i+1$ sur une arête $\{v, v'\}$ lors de laquelle le sommet v prend l'étiquette NON-ÉLU. On sait que $d_i(v) = 1$ est le nombre de voisins actifs de v et par conséquent v est une feuille dans l'arbre induit par les sommets actifs à l'étape i. Par conséquent, à l'étape $i + 1$, le graphe induit par les sommets actifs est toujours un arbre. De plus, v' a exactement un voisin actif de moins à l'étape $i+1$ qu'à l'étape i et par conséquent $d_{i+1}(v') = d_i(v') - 1$ est bien le nombre de voisins actifs de v' à l'étape $i + 1$.

Si la règle $\mathcal{T}_2$ est appliquée lors de l'étape $i + 1$ et modifie l'étiquette d'un sommet v, cela signifie que v n'a aucun voisin actif et puisque le graphe induit par les sommets actifs est connexe, cela implique qu'à l'étape $i + 1$, il n'y aucun sommet actif dans $\mathbf{T}$: la propriété reste vraie. □

On considère la configuration finale d'une exécution de l'algorithme sur $\mathbf{T}$. S'il existe encore des sommets actifs, alors d'après le Lemme 5.23, le graphe induit par les sommets actifs est un arbre et ou bien, cet arbre est réduit à un sommet et la règle $\mathcal{T}_2$ peut être appliquée, ou bien il existe un sommet actif qui a un unique voisin actif et la règle $\mathcal{T}_1$ peut être appliquée. Par conséquent, dans la configuration finale, tous les sommets ont l'étiquette ÉLU ou NON-ÉLU. De plus, d'après le Lemme 5.23, si la règle $\mathcal{T}_2$ est appliquée lors d'une étape i, alors il n'y a plus aucun sommet actif après l'étape i. Par conséquent, il y au plus un sommet qui a l'étiquette ÉLU dans la configuration finale. De plus, si on considère le dernier sommet qui a pris une étiquette finale, ce sommet n'a pas pu appliquer la règle $\mathcal{T}_1$ et il a par conséquent l'étiquette ÉLU dans la configuration finale. L'algorithme décrit ci-dessus est donc bien un algorithme d'élection pour la famille des arbres.

Théorème 5.24 *Il existe un algorithme d'élection pour la famille des arbres utilisant des calculs locaux sur les arêtes non-étiquetées avec connaissance initiale du degré.*

Un corollaire intéressant de ce théorème et de l'exemple donné sur la Figure 20 est que les calculs locaux sur les arêtes non-étiquetées avec connaissance initiale du degré permettent de résoudre le problème de l'élection sur strictement plus de graphes que les calculs locaux sur les arêtes non-étiquetées sans connaissance initiale du degré. Cela est du au fait que, contrairement aux revêtements, les pseudo-revêtements ne conserve pas forcément le degré. En effet, si on considère le graphe T_1 de la Figure 20, la connaissance initiale du degré rend le graphe minimal pour les pseudo-revêtements puisque les deux seuls sommets dont l'étiquette initiale (i.e. leur degré) est 2 sont adjacents et ils ne peuvent donc pas avoir la même image à travers un homomorphisme de graphes simples.

5.5.2 Élection dans les Familles de Diamètre Borné

On montre dans cette partie que pour pouvoir élire dans un graphe $\mathbf{G}$ minimal pour les pseudo-revêtements, il n'est pas nécessaire de connaître la taille de $\mathbf{G}$, mais qu'une borne sur le diamètre suffit si chaque sommet connaît initialement son degré. Pour cela, on explique comment implémenter l'algorithme GSSP décrit dans le Chapitre 2 utilisant des calculs locaux sur les arêtes non-étiquetées avec connaissance du degré.

On va utiliser les mêmes idée que dans les Chapitres 2 et 3 pour implémenter l'algorithme GSSP dans le modèle considéré ici. Comme dans le Chapitre 3, chaque sommet va devoir stocker des informations à propos du rayon de confiance de ses voisins. Cependant, dans le Chapitre 3, chaque sommet pouvait distinger ses voisins en utilisant les numéros distincts attribuées à chacune des arêtes qui lui étaient incidentes. On ne peut pas utiliser

ce principe ici, puisque les arêtes ne peuvent pas être étiquetées. Néanmoins, si le graphe $\mathbf{G}$ est minimal pour les pseudo-revêtements, on sait que pour toute exécution de l'algorithme $\mathcal{M}$ sur $\mathbf{G}$, il existe une étape lors de laquelle tous les sommets de $\mathbf{G}$ ont un numéro différent. Par conséquent, s'il y a moins de $\deg_G(v)$ couples (n, o) dans la vue locale de v, on sait que la vue locale d'un sommet v ne peut pas être sa vue locale finale. En revanche, si un sommet v a $\deg_G(v)$ couples (n, o) dans sa vue locale et que chaque voisin v' de v n'a pas modifié sa boîte-aux-lettres depuis la dernière étape lors de laquelle une règle de $\mathcal{M}$ a été appliquée sur l'arête $\{v, v'\}$, tous les voisins de v ont des numéros différents et le sommet v les connaît. Par conséquent, le sommet v peut distinguer ses voisins grâce à leurs numéros et on peut donc implémenter l'algorithme GSSP en utilisant des calculs locaux sur les arêtes non-étiquetées.

Comme dans le Chapitre 3, l'étiquette de chaque sommet v va être de la forme $(\lambda(v), d(v), n(v), N(v), M(v), a(v), A(v))$ où $\lambda(v), n(v), N(v)$ et $M(v)$ jouent le même rôle que dans l'algorithme $\mathcal{M}$. Le champ $d(v)$ code le degré de v dans le graphe et ne sera pas modifié. Le champ $a(v)$ correspond au rayon de confiance du sommet v et le champ $A(v)$ est un ensemble de couples d'entiers qui contient l'information dont v dispose à propos du rayon de confiance de ses voisins. Comme dans l'algorithme $\mathcal{M}$, l'étiquette initiale de chaque sommet $v \in V(G)$ est $(\lambda(v), d(v), 0, \emptyset, \emptyset, -1, \emptyset)$.

Il est à noter qu'ici, on considère que l'étiquette initiale du sommet est $(\lambda(v), d(v))$ et que les triplets ajoutés à $M(v)$ sont de la forme $(n, (\ell, d), N)$. De cette manière, on s'assure que dans la configuration finale, deux sommets qui ont le même numéro ont aussi le même degré.

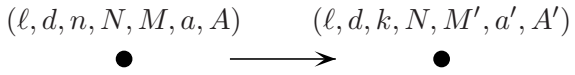
Les règles $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$ restent les mêmes que dans l'algorithme $\mathcal{M}$ a la seule différence que si la boîte-aux-lettres d'un sommet v est modifiée par l'application d'une de ces règles, alors $a(v)$ est réinitialisé à -1 et $A(v)$ est réinitialisé à $\emptyset$. En effet, si un sommet modifie sa boîte-aux-lettres, cela signifie que ses voisins peuvent avoir changer de numéro et les numéros apparaissant dans $A(v)$ sont alors obsolètes. On note $\mathcal{M}'_1$, $\mathcal{M}'_2$, $\mathcal{M}'_3$ et $\mathcal{M}'_4$ les règles ainsi obtenues.

$\mathcal{M}'_1$:

$(\ell_1, d_1, n_1, N_1, M_1, a_1, A_1)$ $(\ell_2, d_2, n_2, N_2, M_2, a_2, A_2)$ $\longrightarrow$ $(\ell_1, d_1, n_1, N_1, M', a'_1, A'_1)$ $(\ell_2, d_2, n_2, N_2, M_2, a_2, A_2)$

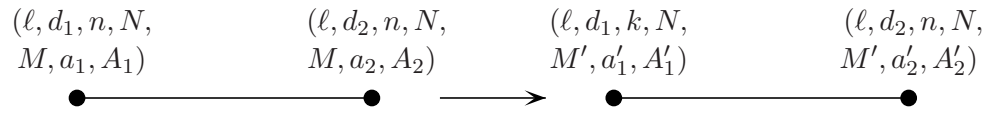
Si $M_2 \setminus M_1 \neq \emptyset$
alors $M' := M'_1 \cup M'_2$;
$a'_1 := -1$;
$A'_1 := \emptyset$.

$\mathcal{M}'_2$:



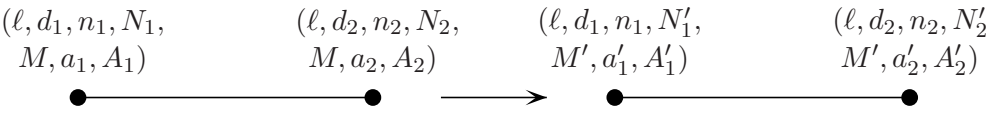
Si $n = 0$ ou $\exists (n, (d', \ell'), N') \in M$ tel que $((d, \ell), N) \prec ((d', \ell'), N')$
alors $k := 1 + \max\{n' \mid \exists (n', (d', \ell'), N') \in M\}$;
$M' := M \cup \{(k, (d, \ell), N)\}$;
$a' := -1$;
$A' := \emptyset$.

$\mathcal{M}'_3$:



Si $n > 0$ et
$\forall (n, (d', \ell'), N') \in M$, $((d', \ell'), N') \preceq ((d, \ell), N)$
alors $k := 1 + \max\{n' \mid \exists (n', (d', \ell'), N') \in M\}$;
$M' := M \cup \{(k, (d, \ell), N)\}$;
$a'_1 := -1$;
$A'_1 := \emptyset$;
$a'_2 := -1$;
$A'_2 := \emptyset$.

$\mathcal{M}'_4$:



Si $n_1, n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, (d'_1, \ell'_1), N'_1) \in M, ((d'_1, \ell'_1), N'_1) \preceq ((d_1, \ell_1), N_1)$,
$\forall (n_2, (d'_2, \ell'_2), N'_2) \in M, ((d'_2, \ell'_2), N'_2) \preceq ((d_2, \ell_2), N_2)$ et
$\nexists o \mid (n_2, o) \in N_1$ et $(n_1, o) \in N_2$
alors $o := 1 + \max\{o' \mid \exists (n', o') \in N_1 \cup N_2\}$;
$N'_1 := N_1 \setminus \{(n', o') \in N_1 \mid n' \leq n_2\} \cup \{(n_2, o)\}$;
$N'_2 := N_2 \setminus \{(n', o') \in N_2 \mid n' \leq n_1\} \cup \{(n_1, o)\}$;
$M' := M \cup \{(n_1, (d_1, \ell_1), N'_1), (n_2, (d_2, \ell_2), N'_2)\}$;
$a'_1 := -1$;
$A'_1 := \emptyset$;
$a'_2 := -1$;
$A'_2 := \emptyset$;

La règle $\mathcal{M}'_5$ permet à un sommet v qui ne peut pas appliquer la règle $\mathcal{M}'_2$, dont le rayon de confiance est -1 et dont la vue locale contient $d(v)$ exactement couples (n, o) de prendre le rayon de confiance 0 et de construire un ensemble $A(v)$ qui contient un couple $(n, -1)$ pour chaque couple (n, o) apparaissant dans sa vue locale. Cet ensemble $A(v)$ va

permettre à chaque sommet de stocker le rayon de confiance de ses voisins.

$\mathcal{M}'_5$:

$(\ell, d, n, N, M, -1, A)$ ● ⟶ ● $(\ell, d, k, N, M, 0, A')$

Si $n > 0$,
$\forall (n, (d', \ell'), N') \in M, ((d', \ell'), N') \preceq ((d, \ell), N)$ et
$|N| = d$
Alors $A' := \{(n', -1) \mid \exists (n', o) \in N\}$.

La règle $\mathcal{M}'_6$ permet à un sommet dont le rayon de confiance est supérieur ou égal à 0 d'augmenter son rayon de confiance si chacun de ses voisins a un rayon de confiance supérieur ou égal au sien. On remarque que si un sommet a un rayon de confiance supérieur ou égal à 0, on sait que la règle $\mathcal{M}'_2$ ne peut pas être appliquée sur le sommet v.

$\mathcal{M}'_6$:

(ℓ, d, n, N, M, a, A) ● ⟶ ● $(\ell, d, k, N, M, a+1, A)$

Cette règle est applicable si $n > 0$, $a \geq 0$ et
$\forall (n', a') \in A, a' \geq a$.

La règle $\mathcal{M}'_7$ permet à deux sommets d'échanger leurs rayons de confiance une fois que ceux ci ont été modifiés par la règle précédente. Cette règle ne peut être appliquée sur une arête e que si aucune des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4$ ne peut être appliquée sur e.

$\mathcal{M}'_7$:

$(\ell_1, d_1, n_1, N_1, M, a_1, A_1)$ ● — ● $(\ell, d_2, n_2, N_2, M, a_2, A_2)$ ⟶ $(\ell_1, d_1, n_1, N'_1, M, a_1, A'_1)$ ● — ● $(\ell, d_2, n_2, N'_2, M, a_2, A'_2)$

Si $p > 0$, $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$), $a_1 \geq 0$, $a_2 \geq 0$
$\exists o$ tel que$(n_2, o) \in N_1$ et $(n_1, o) \in N_2$ et
$(n_1, a_1) \notin A_2$ ou $(n_2, a_2) \notin A_1$
alors $A'_1 := A_1 \setminus \{(n'_2, a') \mid n'_2 = n_2\} \cup \{(n_2, a_2)\}$;
$A'_2 := A_2 \setminus \{(n'_1, a') \mid n'_1 = n_1\} \cup \{(n_1, a_1)\}$;

On s'intéresse maintenant aux propriétés satisfaites par toute exécution de l'algorithme $\mathcal{M}'$. Comme précedemment, on considère une exécution de l'algorithme $\mathcal{M}'$ sur un graphe étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), d(v), n_i(v), N_i(v), M_i(v), a_i(v), A_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage.

Le lemme suivant qui peut être facilement prouvé par induction sur le nombre d'étapes montre que le rayon de confiance d'un sommet ne peut qu'augmenter tant que sa boîte-aux-lettres n'est pas modifiée.

Lemme 5.25 *Pour tout sommet v et toute étape i, si $M_i(v) = M_{i+1}(v)$, alors $a_{i+1}(v) \geq a_i(v)$ et pour tout $(n, a) \in M_i(v)$, il existe $(n', a') \in M_{i+1}(v)$ et $a' \geq a \geq a_{i+1}(v) - 1$. De plus, si $a_i(v) \geq 0$, alors la règle $\mathcal{M}'_2$ ne peut être appliquée sur le sommet v.*

Dans le lemme suivant, on montre que le rayon de confiance d'un sommet permet d'obtenir des informations sur le contenu des boîtes-aux-lettres d'autres sommets du graphe lors d'étapes précédentes de l'exécution. Cette propriété est la même que la propriété de l'algorithme du Chapitre 3.

Lemme 5.26 *Pour tout sommet $v \in V(G)$ et toute étape i, pour tout sommet $w \in V(G)$ tel que* $\text{dist}_G(v,w) \leq a_i(v)$*, il existe une étape $j \geq i$ telle que $a_j(w) \geq a_i(v) - \text{dist}_G(v,w)$ et $M_j(v) = M_i(v)$.*

Preuve : On fait une démonstration par récurrence sur la distance k entre v et w dans G. Si $k = 0$, la propriété est trivialement vraie. On suppose maintenant que la propriété est vraie pour tous sommets v, w tels que $\text{dist}_G(v,w) \leq k$.

On considère deux sommets v, w et une étape i tels que $a_i(v) \geq k+1 \geq 1$ et $\text{dist}_G(v,w) = k+1$. Il existe une étape $i_0 < i$ lors de laquelle la règle $\mathcal{M}'_6$ a été appliquée sur le sommet v telle que $M_i(v) = M_{i_0}(v)$. Il existe aussi une étape $i' \in [i_0, i]$ lors de laquelle la règle $\mathcal{M}'_7$ a été appliquée sur v. On note i_1 la dernière étape lors de laquelle la règle $\mathcal{M}'_7$ a été appliquée sur v et on sait que $M_i(v) = M_{i_0}(v) \subseteq M_{i_1}(v) \subseteq M_i(v)$.

D'après le Lemme 5.15, puisque $N_{i_0}(v) = N_i(v)$, on sait que pout toute étape $i' \in [i_0, i_1]$, pour tout couple $(n,o) \in N_{i'}(v)$, ou bien il existe $u \in N_G(v)$ tel que $n_{i'}(u) = n$, ou bien il existe $u \in N_G(v)$ tel que $n_{i'}(u) \geq \max\{n' \mid \exists (n', o') \in N_{i_0}(v)\}$. De plus, pour tout $(n,a) \in A_i(v)$, $a \geq a_i(v) - 1 \geq 0$ et par conséquent, pour tout $(n,o) \in N_i(v)$, il existe une étape $i' \in [i_0, i_1]$ lors de laquelle règle $\mathcal{M}'_7$ a été appliquée sur une arête $\{v, v'\}$ telle que $n_{i'}(v') = n$ et $a_{i'}(v') = a \geq 0$. Par conséquent, puisque les numéros des sommets ne peuvent qu'augmenter et que $|N_{i_0}(v)| = \deg_G(v)$, on sait que pour tout $u \in N_G(v)$, il existe $(n_{i_0}(u), o) \in N_{i_0}(v)$. De plus, d'après le Lemme 5.15, pour toute étape $i' \in [i_0, i]$ lors de laquelle la règle $\mathcal{M}'_7$ est appliquée sur l'arête $\{u, v\}$, $n_{i'}(u) = n_{i_0}(u)$ et $M_{i'}(u) = M_i(v)$.

On sait qu'il existe un sommet $u \in N_G(v)$ tel que $\text{dist}_G(u,w) = k$. On considère la dernière étape j' lors de laquelle la règle règle $\mathcal{M}'_7$ a été appliquée sur l'arête $\{u, v\}$. On sait qu'il existe $(n_{j'}(u), a_{j'}(u)) \in N_i(v)$. Par conséquent, d'après le Lemme 5.25, on sait que $a_{j'}(u) \geq a_{i'}(v) - 1$ et $M_{j'}(u) = M_i(v)$. Par hypothèse de récurrence, on sait qu'il existe une étape $j < j'$ telle que $a_j(w) \geq a_{j'}(u) - k \geq a_i(v) - (k+1)$ et telle que $M_j(w) = M_{j'}(u) = M_i(v)$. Ainsi, la propriété est vérifiée pour tous sommets v, w à distance $k+1$ dans G. □

Comme dans l'algorithme GSSP des Chapitres 2 et 3, on montre que si à un moment donné, un sommet a un rayon de confiance supérieur au diamètre du graphe, alors tous les sommets du graphe ont la même boîte-aux-lettres et ont tous un rayon de confiance supérieur à 0.

Lemme 5.27 *S'il existe un sommet v et une étape i telle que $a_i(v) \geq D(G) + 1$, alors pour tout $w \in V(G)$, $M_i(w) = M_i(v)$ et $a_i(v) \geq 0$.*

Preuve : Puisque $a_i(v) > D(G)$, on sait d'après le Lemme 5.26 que pour tout sommet $w \in V(G)$, il existe une étape $i_w < i$ telle que $a_{i_w} \geq 1$ et $M_{i_w}(w) = M_i(v)$.

Supposons qu'il existe un sommet w tel que $M_i(w) \neq M_{i_w}(w)$. Soit j l'étape de l'exécution lors de laquelle pour la première fois, la boîte-aux-lettres d'un sommet w qui valait $M_i(v)$ a été modifiée. Autrement dit, pour tout sommet $w \in V(G)$, il existe une étape $j' \geq j - 1$ telle que $M_{j'}(w) = M_i(v)$ et il existe un sommet w tel que

$M_{j-1}(w) = M_i(v) \subsetneq M_j(w)$. Cela signifie qu'une des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4$ a été appliquée lors de l'étape j. Soit w un sommet tel que $M_{j-1}(w) \subsetneq M_j(w)$; on sait d'après le Lemme 5.25 que $M_{j-1}(w) = M_i(v)$ et $a_{j-1}(w) \geq 0$. Par conséquent, d'après le Lemme 5.25, la règle $\mathcal{M}'_2$ n'a pas pu être appliquée sur le sommet w.

On considère un sommet w dont l'étiquette a été modifiée lors de l'étape j. En raison du choix de j, on sait que la règle $\mathcal{M}'_1$ n'a pas pu être appliquée à l'étape j. Par ailleurs, on sait que $a_{j-1}(w) \geq 1$ et par conséquent, pour tout sommet $w' \in N_G(w)$, il existe une étape $j' \leq j$ telle que $M_{j'}(w') = M_j(w) = M_i(v)$ lors de laquelle la règle $\mathcal{M}'_7$ a été appliquée sur l'arête $\{w, w'\}$. De plus, en raison du choix de j, on sait que $n_{j'}(w) = n_{j-1}(w)$, $n_{j'}(w') = n_{j-1}(w')$, $N_{j'}(w) = N_{j-1}(w)$, $N_{j'}(w') = N_{j-1}(w')$ et $M_{j'}(w) = M_{j'}(w') = M_i(v)$. De plus, aucune des règles $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3, \mathcal{M}'_4, \mathcal{M}'_5$ n'a peut être appliquée sur l'arête $\{w', w\}$ à l'étape $j' + 1$ et par conséquent, aucune de ces règles ne peut être appliquée sur l'arête $\{w', w\}$ à l'étape j. Ainsi pour tout sommet $w \in V(G)$, $M_i(w) = M_i(v)$. □

Ainsi, si on connaît une borne B sur le diamètre de $\mathbf{G}$, l'algorithme $\mathcal{M}'$ permet de détecter que l'exécution de l'algorithme $\mathcal{M}$ sous-jacent est terminé. En effet, une fois qu'un sommet a un rayon de confiance supérieur ou égal à $B + 1$, il sait que tous les sommets de $\mathbf{G}$ ont la même boîte-aux-lettres et qu'ils ont leurs numéros et vues locales finaux.

Si on sait que le graphe $\mathbf{G}$ est minimal pour les pseudo-revêtements, on sait alors d'après la Proposition 5.18 que tous les sommets ont un identifiant unique et dans ce cas là, le sommet dont le numéro est 1 peut prendre l'étiquette ÉLU et diffuser l'information. On a par conséquent montré le théorème suivant.

Théorème 5.28 *Pour tout entier B, il existe un algorithme d'élection et un algorithme de nommage avec détection de la terminaison utilisant des calculs locaux sur les arêtes non-étiquetées avec connaissance initiale du degré pour la famille des graphes minimaux pour les pseudo-revêtements dont le diamètre est bornée par B.*

Ainsi, comme dans les modèles considérés dans les Chapitres 2 et 3, il n'est pas nécessaire de connaître la taille pour pouvoir élire dans un graphe $\mathbf{G}$ que l'on sait minimal pour les pseudo-revêtements : une borne sur la taille ou le diamètre est suffisante.

Cependant, on peut montrer en utilisant la même preuve que dans le Chapitre 3, que si les sommets ne connaissent pas initialement leur degré, il est impossible de résoudre le problème de l'élection sur les graphes minimaux pour les pseudo-revêtements en connaissant seulement une borne sur la taille.

5.5.3 Impossibilité de Détecter la Non-Minimalité

On montre maintenant que même si initialement les sommets connaissent leurs degrés ainsi que la taille n du graphe $\mathbf{G}$, si celle-ci n'est pas un nombre premier (sinon, d'après la Remarque 5.20, on peut élire dans la famille des graphes de taille n puisque ceux-ci sont tous minimaux pour les pseudo-revêtements), il n'existe pas d'algorithme utilisant des calculs locaux sur les arêtes non-étiquetées qui permet de résoudre le problème de l'élection sur $\mathbf{G}$ ou de détecter que $\mathbf{G}$ n'est pas minimal pour les pseudo-revêtements.

Pour tout entier n qui n'est pas premier, on suppose qu'il existe un algorithme effectif d'élection $\mathcal{R}$ pour la classe des graphes de taille n utilisant des calculs locaux sur les arêtes non-étiquetées avec connaissance initiale du degré. On considère deux entiers p, q tels que $n = pq$.

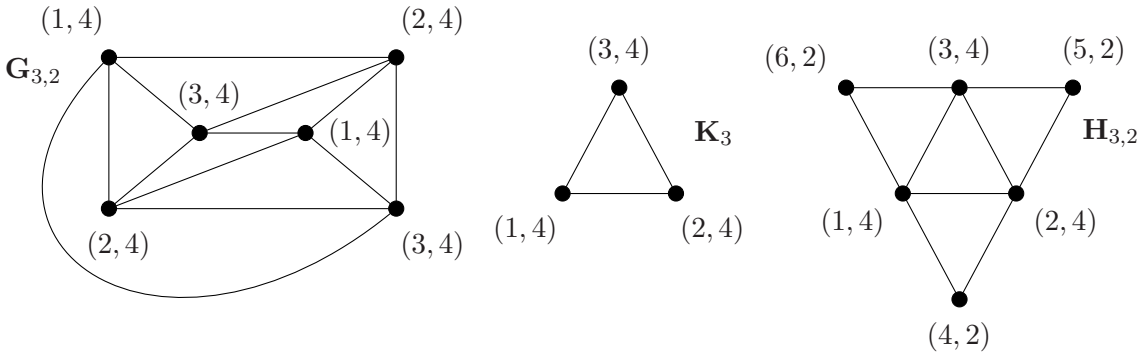


Fig. 21 – Le graphe $\mathbf{G}_{3,2}$ est un pseudo-revêtement de $\mathbf{K}_3$ et $\mathbf{K}_3$ est un sous-graphe de $\mathbf{H}_{3,2}$. Dans les graphes $\mathbf{G}_{3,2}$ et $\mathbf{H}_{3,2}$, l'étiquette de chaque sommet est de la forme (ℓ, d) où ℓ est l'étiquette initiale du sommet et d est le degré du sommet dans le graphe.

On note K_p le graphe complet à p sommets et on considère un étiquetage μ de K_p tel que chaque sommet ait un numéro unique. On construit ensuite le graphe $(G_{p,q}, \lambda)$ de la manière suivante. On considère q copies $(K_{p,1}, \mu), \ldots, (K_{p,q}, \mu)$ distinctes de (K_p, μ) et pour tous entiers distincts $i, j \in [1, q]$, pour tous sommets $v_i \in K_{p,i}$ et $v_j \in K_{p,j}$, on ajoute une arête entre v_i et v_j si $\mu(v_i) \neq \mu(v_j)$. Le graphe ainsi obtenu est un graphe multiparti complet avec p parties de taille q (chaque étiquette définit une partie). On remarque que chaque sommet $v \in V(G_{p,q})$ a exactement $q(p-1)$ voisins dans $(G_{p,q}, \lambda)$. On considère le graphe étiqueté $\mathbf{G}_{p,q} = (G_{p,q}, (\lambda, d))$ où pour tout sommet $v \in V(G_{p,q})$, $d(v) = q(p-1)$ est le degré de v dans $G_{p,q}$. On considère le graphe $\mathbf{K}_p = (K, (\mu, d))$ où pour tout sommet $v \in V(K_p)$, $d(v) = q(p-1)$. Sur la Figure 21, les graphes $\mathbf{G}_{3,2}$ et $\mathbf{K}_3$ sont représentées.

On remarque que dans $\mathbf{K}_p$, les étiquettes initiales des sommets ne contiennent pas le degré de chaque sommet. On peut toutefois exécuter l'algorithme $\mathcal{R}$ sur $\mathbf{K}$, même si l'algorithme n'est pas censé terminer dans une configuration correcte, puisque l'information initiale dont dispose chaque sommet n'est pas correcte. D'après le Lemme 5.9, il existe une exécution ρ_G de $\mathcal{R}$ sur $\mathbf{G}_{p,q}$ qui est relevée d'une exécution ρ_K sur $\mathbf{K}_p$. De plus, puisque l'exécution ρ_G est relevée de l'exécution ρ_K, on sait que dans la configuration finale $\mathbf{G}'_{p,q}$ de ρ_G, les sommets n'ont pas des étiquettes uniques et par conséquent, dans $\mathbf{G}'_{p,q}$, tous les sommets ont une étiquette finale indiquant que $\mathbf{G}_{p,q}$ n'est pas minimal pour les revêtements. Par ailleurs, puisque la configuration finale $\mathbf{G}'_{p,q}$ de ρ_G est relevée de la configuration finale $\mathbf{K}'_p$ de ρ_K, on sait que dans $\mathbf{K}'_p$, tous les sommets ont une étiquette finale indiquant que le graphe n'est pas minimal pour les revêtements.

On construit maintenant un graphe $(H_{p,q}, \eta)$ minimal pour les revêtements qui a pq sommets. Pour cela, on considère une copie de (K_p, λ) qui contient p sommets qui ont tous un degré égal à $p-1$. Pour tout ensemble de $p-1$ sommets de (K_p, λ), on ajoute $q-1$ sommets qui sont tous adjacents à tous ces $p-1$ sommets. Le graphe ainsi obtenu est le graphe $H_{p,q}$, on étend l'étiquetage de (K_p, λ) de telle sorte que tous les sommets du graphe aient des étiquettes différentes. Le graphe étiqueté ainsi obtenu est le graphe (H, η) et ce graphe à $p + p(q-1) = pq$ sommets. On remarque que chaque sommet qui est dans la copie de K_p a $p-1$ voisins dans la copie de K_p et $(p-1)(q-1)$ voisins parmi les sommets ajoutés pour construire $H_{p,q}$: tous ces sommets ont un degré égal à $q(p-1)$. On note $\mathbf{H}_{p,q}$ le graphe $(H_{p,q}, (\eta, d))$ où pour tout sommet $v \in V(H_{p,q})$, $d(v)$ est le degré de v dans $H_{p,q}$. La construction du graphe $\mathbf{H}_{3,2}$ est présenté sur la Figure 21. Dans le graphe

$\mathbf{H}_{p,q}$, tous les sommets ont une étiquette initiale différente et ce graphe est donc mimimal pour les pseudo-revêtements.

Le graphe $\mathbf{H}_{p,q}$ a la même taille que $\mathbf{G}_{p,q}$ et tous les sommets de $\mathbf{H}_{p,q}$ connaissent initialement leur degré : par conséquent, toute exécution de l'algorithme $\mathcal{R}$ doit élire un sommet de $\mathbf{H}_{p,q}$ puisqu'il est minimal pour les pseudo-revêtements. On sait que le graphe $\mathbf{K}_p$ est un sous-graphe de $\mathbf{H}_{p,q}$, puisqu'on a construit le graphe $\mathbf{H}_{p,q}$ de telle sorte que tous les sommets de la copie de K_p aient un degré égal à $q(p-1)$ dans $\mathbf{H}_{p,q}$. Par conséquent on peut trouver une exécution de $\mathcal{R}$ sur $\mathbf{H}_{p,q}$ dont le préfixe est ρ_K. Dans la configuration finale de cette exécution, les sommets de la copie de K_p dans $\mathbf{H}_{p,q}$ ont une étiquette finale indiquant que le graphe $\mathbf{H}_{p,q}$ n'est pas minimal pour les pseudo-revêtements, ce qui est faux.

Par conséquent, dans le modèle des calculs locaux sur les arêtes non-étiquetées, il n'existe pas de théorème correspondant aux Théorèmes 3.50 et 2.31.

5.6 Conclusion et Perspectives

Dans ce chapitre, on a caractérisé les graphes admettant un algorithme de nommage et d'élection utilisant des calculs locaux sur les arêtes non-étiquetées (Théorème 5.19). Cette caractérisation s'exprime à l'aide de la notion de pseudo-revêtements. Contrairement aux algorithmes des Chapitres 2, 3 et 4, l'algorithme présenté dans la Section 5.4 peut nécessiter un nombre d'étapes de réétiquetages exponentiel en la taille du graphe. Le problème de déterminer s'il existe un algorithme polynomial de nommage pour tous les graphes minimaux de taille donnée est une question qui n'a pas été résolue pour le moment.

Ce résultat nous permet de penser qu'il est possible d'utiliser les techniques présentées dans [GM03, GMM04] afin de caractériser les classes de graphes qui peuvent être reconnues par des calculs locaux sur les arêtes non-étiquetées avec ou sans connaissance initiale (la terminaison est alors implicite). Cependant, on peut reconnaître strictement moins de classes de graphes en utilisant des calculs locaux sur les arêtes étiquetées (avec ou sans connaissance initiale) que dans les modèles étudiés dans les Chapitres 2, 3 et 4.

Dans la Section 5.5, on a étudié l'importance de la connaissance initiale du degré. On a montré qu'avec la connaissance initiale du degré, la connaissance initiale d'une borne sur le diamètre du graphe permettait de résoudre élection et nommage dans un graphe minimal pour les revêtements (Théorème 5.28).

Ce résultat nous permet de penser qu'on peut étendre les techniques utilisés dans [GM02] aux calculs locaux sur les arêtes non-étiquetées afin de caractériser les classes de graphes qui admettent un algorithme universel d'élection dans ce modèle.

Cependant, le fait qu'il n'existe pas d'algorithme effectif d'élection pour les graphes de taille donnée utilisant des calculs locaux sur les arêtes non-étiquetées et plus particulièrement, le fait qu'on ne puisse pas implémenter l'algorithme GSSP pour les graphes qui ne sont pas minimaux pour les pseudo-revêtements dans ce modèle nous laissent supposer qu'on ne peut pas employer dans ce modèle les techniques utilisés dans [MT00] pour caractériser ce qui peut être calculé avec détection de la terminaison.

Par ailleurs, même si chaque sommet connaît initialement son degré, l'algorithme d'énumération auto-stabilisant de Godard présenté dans [God02b] ne semble pas pouvoir être étendu au modèle considéré dans ce chapitre. Contrairement au modèles étudiés

dans les Chapitres 2 et 4, dans le modèle des calculs locaux sur les arêtes non-étiquetées, un sommet ne peut pas forcément se rendre compte que l'information dont il dispose à propos de ses voisins (sa vue locale) est erronnée.

Chapitre 6

Calculs Locaux Cellulaires sur les Arêtes Non-Étiquetées

Sommaire

Fig. 22 – Forme générique d'une règle de calcul pour les calculs locaux cellulaires sur les arêtes non-étiquetées.

6.1 Introduction

Dans ce chapitre, on étudie les *calculs locaux cellulaires sur les arêtes non-étiquetées*. Dans ce modèle, on considère des graphes simples où seuls les sommets peuvent être étiquetés, comme dans les Chapitres 4 et 5. Un pas de calcul est décrit par une règle de réétiquetage de la forme présentée sur la Figure 22. Si dans un graphe G, il existe un sommet étiqueté X qui a un voisin étiqueté Y, alors l'application de la règle de la Figure 22 permet de remplacer l'étiquette X par l'étiquette X'. Les étiquettes de tous les autres sommets ne sont pas prises en compte pour l'application de la règle de réétiquetage et restent inchangées. Le sommet de G dont l'étiquette est modifiée est dit *actif* (et est représenté en noir sur les figures), le voisin du sommet actif qui est utilisé pour pouvoir appliquer la règle est dit *passif* (et est représenté en blanc sur les figures). Tous les autres sommets de G qui ne participent pas à l'application de la règle sont dits *inactifs*. Les calculs réalisés en utilisant uniquement ce type de réétiquetage sont appelés *calculs locaux cellulaires sur les arêtes non-étiquetées*.

La différence entre ce modèle et le modèle des calculs locaux cellulaires sur les arêtes étiquetées est que les arêtes ne peuvent pas être étiquetées (et réétiquetées) ; cela donne un modèle strictement plus faible. On va aussi montrer que les calculs locaux cellulaires sur les arêtes non-étiquetées sont strictement plus faibles que les calculs locaux sur les arêtes non-étiquetées et que les calculs locaux cellulaires sur les étoiles. Ainsi, lorsque les arêtes ne peuvent pas être étiquetées, il n'existe pas de résultats correspondant aux Propositions 3.20 et 3.42.

6.1.1 Résultats

Dans le modèle des calculs locaux cellulaires sur les arêtes non-étiquetées, on caractérise les graphes admettant un algorithme de nommage et les graphes admettant un algorithme d'élection. Comme dans le Chapitre 4, les deux problèmes ne sont pas équivalents dans le modèle considéré dans ce chapitre. On utilise la notion de *submersions* (qui correspondent aux homomorphismes de graphes simples localement surjectifs) pour pouvoir exprimer ces caractérisations.

On montre qu'un graphe admet un algorithme de nommage utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées si et seulement s'il est minimal pour les submersions (Théorème 6.16). L'algorithme de nommage qu'on présente dans la Section 6.4.2 est inspiré de l'algorithme de Mazurkiewicz [Maz97] et utilise les idées présentées dans le Chapitre 5.

La caractérisation des graphes admettant un algorithme d'élection utilise aussi la notion d'homomorphismes localement surjectifs, mais est un peu plus difficile à exprimer (Théorème 6.28). L'algorithme d'élection qu'on présente dans la Section 6.5 repose à la fois sur l'algorithme de Mazurkiewicz [Maz97] et sur une adaptation de l'algorithme de Szymansky, Shi et Prywes [SSP85] différente de l'algorithme GSSP présentée dans les Chapitres 2, 3 et 5. Dans le cadre des calculs locaux cellulaires sur les arêtes non-étiquetées, il

n'est pas possible d'assurer que tous les voisins d'un sommet ont des étiquettes distinctes à partir d'une certaine étape, contrairement à l'algorithme d'élection pour les familles de diamètre borné présenté dans le Chapitre 5.

On étudie ensuite l'influence de la connaissance initiale du degré et on montre que si initialement, chaque sommet connaît son degré, alors il suffit de connaître une borne sur le diamètre pour pouvoir nommer dans un graphe minimal pour les submersions (Théorème 6.31). Cependant, on n'obtient pas de résultats similaires pour le problème de l'élection et on montre que contrairement aux modèles étudiés dans les Chapitres 3 et 5, la connaissance initiale du degré ne permet pas d'élire dans la famille des arbres.

Les résultats présentés dans ce chapitre ont été obtenus en collaboration avec Yves Métiver et Wiesław Zielonka. Une version préliminaire de certains de ces résultats est parue dans [CMZ04] et une version complète est parue dans [CMZ06].

6.1.2 Travaux Liés

Processus à Mémoire Finie

Dans [AAER05], Angluin et al. considèrent un modèle utilisant le même type de règles. Cependant, ils supposent que la mémoire de chaque processus est finie : l'ensemble des étiquettes apparaissant sur les sommets au cours de toute exécution est donc fini et indépendant de la taille du graphe. Dans leurs travaux, Angluin et al. ne considèrent que le cas où le graphe est un graphe complet.

De plus, ils supposent que si une configuration globale peut être infiniment souvent atteinte en un nombre fini d'étapes à partir de la configuration courante, alors cette configuration globale sera atteinte. De cette manière, il n'est pas nécessaire de considérer des relations de réétiquetage noethériennes.

Ils supposent qu'il existe une fonction de sortie qui permet d'attribuer à chaque configuration globale du graphe une valeur, qui est le résultat du calcul. Ils considèrent des exécutions qui calculent un résultat de manière *stabilisante* : ce sont les exécutions telles qu'à partir d'une certaine étape, la fonction de sortie renvoie toujours la même valeur. Cette notion de terminaison est à rapprocher de la terminaison implicite, au sens où les sommets ne savent pas si l'exécution est terminée. Cependant, il faut noter que la fonction de sortie est globale et par conséquent, les étiquettes peuvent être modifiée, tant que globalement, le résultat reste le même.

En supposant que chaque sommet v a initialement une valeur $\mathbf{input}(v)$ dans un alphabet d'entrée Σ, Angluin et al. déterminent quels prédicats satisfaits par le multiensemble des entrées peuvent être calculés de manière stabilisante. Ils montrent que dans ce modèle particulier, les prédicats calculables de manière stabilisante sont les combinaisons booléennes finies de prédicats de la forme $|\{v|\mathbf{input}(v) = a\}| \geq r$ où $a \in \Sigma$ et r est une constante.

Assignation de Rôles

Les homomorphismes localement surjectifs ont été étudiés précédemment dans le cadre des problèmes de l'assignation de rôles dans les réseaux sociaux [EB91, FP03, FP05, KT00]. En particulier, Fiala et Paulusma ont montré [FP05] que pour tout graphe H fixé, il est

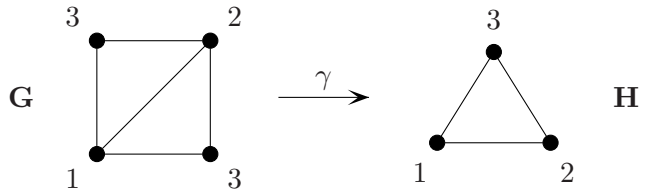


Fig. 23 – Le graphe simple **G** est une submersion de **H** à travers l'homomorphisme γ qui envoie chaque sommet de **G** étiqueté i sur l'unique sommet de **H** dont l'étiquette est i. Cette submersion est propre et donc le graphe **G** n'est pas minimal pour les submersions, mais le graphe **H** est minimal pour les submersions.

NP-complet de décider s'il existe un homomorphisme localement surjectif d'un graphe G donné dans H.

6.2 Submersions

Dans ce chapitre, les submersions, i.e., les homomorphismes de graphes localement surjectifs sont les homomorphismes qui permettent de donner des conditions nécessaires que doivent vérifier les graphes admettant un algorithme de nommage ou d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

Définition 6.1 *Un graphe simple G est une* submersion *d'un graphe simple H à travers un homomorphisme $\gamma\colon G \to H$ si pour tout sommet $v \in V(G)$, γ induit une surjection du voisinage $N_G(v)$ sur le voisinage $N_H(\gamma(v))$, i.e., $\gamma(N_G(v)) = N_H(\gamma(v))$. On dit alors que l'homomorphisme γ est localement surjectif.*

Un graphe simple G est une submersion propre *de H si γ n'est pas un isomorphisme et G est* minimal pour les submersions *si G n'est une submersion propre d'aucun autre graphe simple.*

Naturellement, un graphe simple étiqueté (G, λ) est une submersion d'un graphe simple étiqueté (H, η) à travers γ si G est une submersion de H à travers γ et si γ conserve l'étiquetage.

Un exemple de submersion est présenté sur la Figure 23.

Comme les homomorphismes localement bijectifs, un homomorphisme localement surjectif d'un graphe **G** dans un graphe connexe **H** est surjectif.

Proposition 6.2 *Si* **G** *est une submersion d'un graphe connexe* **H** *à travers γ, alors γ est surjectif.*

Preuve : On considère un graphe **G** qui est une submersion d'un graphe **H** à travers un homomorphisme γ. Pour tout sommet $v \in \gamma(V(G))$, il existe $u \in V(G)$ tel que $\gamma(u) = v$. Puisque γ est localement surjectif, pour tout sommet $v' \in N_H(v)$, il existe $u' \in N_u(v)$ tel que $\gamma(u') = v'$. Ainsi, puisque **H** est connexe, on sait que γ est surjectif. □

La notion de coloration connexe permet de caractériser les graphes minimaux pour les submersions en termes de colorations de graphes. Un étiquetage ℓ d'un graphe non-étiqueté est une coloration connexe si deux sommets voisins ont des couleurs distinctes et

si deux sommets qui ont la même couleur ont les mêmes couleurs dans leur voisinage.

Définition 6.3 *Une* coloration connexe *d'un graphe simple non-étiqueté G est un étiquetage ℓ de G tel que*
- *pour tout $i \in \ell(V(G))$, $G[i]$ est un stable,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ n'a pas de sommet isolé.*

Dans la proposition suivante, on utilise la notion de coloration connexe pour caractériser les graphes minimaux pour les submersions. On rappelle qu'une coloration ℓ est dite propre si $|\ell(V(G))| < |V(G)|$.

Proposition 6.4 *Un graphe simple non-étiqueté G est minimal pour les submersions si et seulement si G n'admet aucune coloration connexe propre.*

Preuve : Étant donné un graphe G qui n'est pas minimal pour les submersions, soit H un graphe tel que G est une submersion propre de H à travers un homomorphisme γ. Alors γ est une coloration connexe propre de G et l'ensemble des couleurs utilisées est $V(H)$. De plus, puisque H est un graphe simple, pour toute arête $\{u,u'\} \in E(G)$, $\gamma(u) \neq \gamma(u')$. Enfin, pour tous $v, v' \in V(H)$, si $\{v,v'\} \notin E(H)$, alors $G[v,v']$ est un stable puisque γ est un homomorphisme. Et si $\{v,v'\} \in E(H)$, alors pour tout $u \in \gamma^{-1}(v)$ (resp. $u' \in \gamma^{-1}(v')$), il existe $u' \in N_G(u) \cap \gamma^{-1}(v')$ (resp. $u \in N_G(u') \cap \gamma^{-1}(v)$) : par conséquent, $G[v,v']$ n'a pas de sommets isolés. Par ailleurs, puisque G est une submersion propre de H, on sait que $|\ell(V(G))| = |V(H)| < |V(G)|$: G admet donc une coloration connexe propre.

On considère maintenant un graphe G et une coloration connexe propre ℓ de G. On va maintenant définir un graphe H et un homomorphisme localement surjectif γ de G dans H. On pose $V(H) = \ell(V(G))$. Pour tous $v, v' \in V(H)$, l'arête $\{v,v'\}$ appartient à $E(H)$ si et seulement si $G[v,v']$ n'est pas un stable. Le graphe H ainsi défini est un graphe simple. On pose $\gamma = \ell$. Pour toute arête $\{u,u'\} \in E(G)$, $\gamma(u) \neq \gamma(u')$ et $G[\gamma(u),\gamma(u')]$ n'est pas un stable et donc $\{\gamma(u),\gamma(u')\} \in E(H)$: γ est donc un homomorphisme de G dans H. Enfin, pour tout $u \in V(G)$, on note $v = \gamma(u)$ et pour tout $v' \in N_H(v)$, $V(G[v,v'])$ n'est pas un stable et donc u qui appartient à $G[v,v']$ a un voisin $u' \in \gamma^{-1}(v')$. Par conséquent, γ est un homomorphisme localement surjectif de G dans H. □

Exemple 6.5 *Le graphe complet à n sommets K_n ne peut pas être colorié avec moins de n couleurs et par conséquent K_n est minimal pour les submersions.*

6.3 Calculs Locaux Cellulaires sur les Arêtes Non-Étiquetées

Dans cette partie, on présente les définitions formelles des calculs locaux cellulaires sur les arêtes non-étiquetées puis on étudie leurs relations avec les submersions.

6.3.1 Définitions

On rappelle qu'informellement, les calculs locaux cellulaires sur les arêtes non-étiquetées sont les calculs réalisés en utilisant uniquement des règles de la forme présentée sur la Figure 22 : à chaque pas de calcul, l'étiquette d'un sommet est modifiée par l'application

d'une règle qui dépend de l'étiquette du sommet et de l'étiquette d'un de ses voisins. On présente maintenant un définition plus formelle du modèle.

D'après la Définition 5.8, une relation de réétiquetage $\mathcal{R}$ est localement engendrée sur les arêtes non-étiquetées si l'application d'une règle ne dépend que des étiquettes des extrémités d'une arête, i.e., si la condition suivante est satisfaite. Pour tous graphes (G, λ), (G, λ'), (H, η), (H, η') et toutes arêtes $\{v_1, v_2\} \in E(G)$ et $\{w_1, w_2\} \in E(H)$, si les trois conditions suivantes sont vérifiées :

1. $\lambda(v_1) = \eta(w_1)$, $\lambda(v_2) = \eta(w_2)$, $\lambda'(v_1) = \eta'(w_1)$ et $\lambda'(v_2) = \eta'(w_2)$,
2. $\lambda(v) = \lambda'(v)$, pour tout $v \notin \{v_1, v_2\}$,
3. $\eta(w) = \eta'(w)$, pour tout $w \notin \{w_1, w_2\}$,

alors $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ si et seulement si $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

Une relation de réétiquetage $\mathcal{R}$ localement engendrée sur les arêtes non-étiquetées est cellulaire si lors de l'application d'une règle de réétiquetage, l'étiquette d'un seul sommet est modifiée.

Définition 6.6 *Une relation de réétiquetage $\mathcal{R}$ localement engendrée sur les arêtes non-étiquetées est* cellulaire *si lorsque $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ alors il existe un sommet $v \in V(G)$ tel que $\forall w \in V(G), w \neq v \implies \lambda(w) = \lambda'(w)$.*

Par définition, les *calculs locaux cellulaires sur les arêtes non-étiquetées* correspondent aux relations de réétiquetage cellulaires localement engendrées sur les arêtes non-étiquetées.

Une relation de réétiquetage cellulaire localement engendrée sur les arêtes non-étiquetées peut être décrite par un ensemble récursif de règles de la forme présentées sur la Figure 22. Réciproquement, un tel ensemble de règles induit une relation de réétiquetage cellulaire localement engendrée sur les arêtes non-étiquetées. Ainsi, on notera $\mathcal{R}$ l'ensemble de règles de réétiquetage aussi bien que la relation de réétiquetage correspondante.

6.3.2 Submersions et Calculs Locaux Cellulaires sur les Arêtes Non-Étiquetées

On présente maintenant le lemme qui met en évidence le lien entre les calculs locaux cellulaires sur les arêtes non-étiquetées et les submersions. C'est l'équivalent du lemme de relèvement d'Angluin [Ang80] pour les submersions.

Lemme 6.7 (Lemme de relèvement) *On considère un graphe simple $\mathbf{G}$ qui est une submersion d'un graphe simple $\mathbf{H}$ à travers un homomorphisme γ et une relation $\mathcal{R}$ de réétiquetage cellulaire localement engendrée sur les arêtes non-étiquetées. Si $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$, alors il existe $\mathbf{G}'$ tel que $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$ et $\mathbf{G}'$ est une submersion de $\mathbf{H}'$ à travers γ.*

Preuve : Il suffit de prouver ce lemme pour un pas de calcul. On considère deux graphes simples (G, λ) et (H, η) tels que (G, λ) est une submersion de (H, ν) à travers γ. On considère un pas de réétiquetage sur H qui implique un sommet actif w et un sommet passif w' et on note η' le nouvel étiquetage de H obtenu après ce pas de réétiquetage.

Puisque γ est localement surjectif, pour chaque sommet $v \in \gamma^{-1}(w)$, il existe $v' \in \varphi^{-1}(w') \cap N_G(v)$. Puisque les sommets de $\varphi^{-1}(w)$ forment un stable, $v' \notin \varphi^{-1}(w)$ et on peut donc appliquer la règle de réétiquetage à chaque sommet $v \in \varphi^{-1}(w)$. On obtient ainsi

un étiquetage λ' de G tel que $\gamma : (G, \lambda') \to (H, \eta')$ reste un homomorphisme localement surjectif.

On note que chaque pas de réétiquetage dans $\mathbf{H}$ est simulé par plusieurs pas de réétiquetage dans $\mathbf{G}$ où la même règle de réétiquetage est appliquée. □

Le diagramme suivant représente la propriété du Lemme 6.7.

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow{\mathcal{R}^*} & \mathbf{G'} \\ {\scriptstyle \text{submersion}}\downarrow & & \downarrow{\scriptstyle \text{submersion}} \\ \mathbf{H} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{H'} \end{array}$$

6.4 Énumération et Nommage

On s'intéresse aux problèmes d'énumération et du nommage dans le cadre des calculs locaux cellulaires sur les arêtes non-étiquetées. On montre d'abord que dans ce modèle, il n'existe pas d'algorithme d'énumération ou de nommage pour un graphe $\mathbf{G}$ qui n'est pas minimal pour les submersions. On donne ensuite un algorithme d'énumération pour les graphes minimaux pour les submersions qui est inspiré de l'algorithme de Mazurkiewicz [Maz97].

6.4.1 Résultats d'Impossibilité pour l'Énumération et le Nommage

Proposition 6.8 *Soit* $\mathbf{G}$ *un graphe simple étiqueté qui n'est pas minimal pour les submersions. Il n'existe pas d'algorithme d'énumération ou de nommage pour le graphe* $\mathbf{G}$ *utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.*

Preuve : Soit $\mathbf{H}$ un graphe étiqueté non-isomorphe à $\mathbf{G}$ et tel qu'il existe un homomorphisme localement surjectif γ de $\mathbf{G}$ dans $\mathbf{H}$. Étant donné un algorithme $\mathcal{R}$ utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées, on considère une exécution de $\mathcal{R}$ sur $\mathbf{H}$. Si cette exécution est infinie sur $\mathbf{H}$, alors d'après le Lemme 6.7, il existe une exécution infinie de $\mathcal{R}$ sur $\mathbf{G}$; auquel cas, $\mathcal{R}$ n'est ni un algorithme d'énumération, ni de nommage.

On suppose maintenant qu'il existe une exécution finie de $\mathcal{R}$ sur $\mathbf{H}$ et on considère la configuration finale $\mathbf{H'}$. D'après le Lemme 6.7, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui permet d'atteindre une configuration $\mathbf{G'}$ telle que $\mathbf{G'}$ est une submersion de $\mathbf{H'}$ à travers γ. Si $\mathbf{G'}$ n'est pas une configuration finale de $\mathcal{R}$, alors il existe un sommet $u \in V(G)$ et un voisin u' de u tels qu'on puisse appliquer une règle de réétiquetage où u est le sommet actif et u' est le sommet passif. Dans ce cas là, on peut appliquer la même règle dans $\mathbf{H'}$ aux sommets $\gamma(u)$ et $\gamma(u')$, ce qui est impossible. Par conséquent, $\mathbf{G'}$ est une configuration finale de $\mathcal{R}$. Mais puisque $\mathbf{G'}$ n'est pas isomorphe à $\mathbf{H'}$, cela implique qu'il existe deux sommets de G qui ont la même étiquette dans $\mathbf{G'}$. Par conséquent, $\mathcal{R}$ ne permet pas d'attribuer de noms distincts à tous les sommets de G et donc $\mathcal{R}$ n'est pas un algorithme de nommage ou d'énumération pour le graphe $\mathbf{G}$. □

6.4.2 Un Algorithme d'Énumération

On va maintenant décrire un algorithme à la Mazurkiewicz $\mathcal{M}$ qui utilise des calculs locaux cellulaires sur les arêtes non-étiquetées et qui permet de résoudre le problème de l'énumération sur un graphe $\mathbf{G} = (G, \lambda)$ qui est minimal pour les submersions.

Durant l'exécution de l'algorithme, chaque sommet v essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. Chaque sommet va observer les numéros choisis par ses voisins pour construire sa *vue locale*, i.e., l'ensemble des numéros de ses voisins et va ensuite diffuser dans tout le graphe son numéro et sa vue locale. Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette $\lambda(u)$ et sa vue locale avec l'étiquette $\lambda(v)$ et la vue locale de v : si l'étiquette de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre similaire à l'ordre utilisé dans l'algorithme de Mazurkiewicz), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire) et la diffuse à nouveau avec sa vue locale. Lorsque l'exécution est terminée, si le graphe $\mathbf{G}$ est minimal pour les submersions, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

Comme dans le Chapitre 5, un sommet ne peut pas faire la distinction entre ses voisins qui ont le même numéro. Ainsi, lorsqu'un sommet ajoute le numéro d'un de ses voisins à sa vue locale, il doit aussi supprimer l'ancien numéro de ce voisin de sa vue locale. Puisque le sommet ne peut pas toujours distinguer ses voisins, il va supprimer des informations qui peuvent être valides de sa vue locale afin de s'assurer que l'ancien numéro du voisin en question a bien été supprimé. Cette méthode permet de s'assurer que dans la configuration finale, la vue locale de chaque sommet ne contient que des informations pertinentes.

Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda \colon V(G) \to L$ est un étiquetage initial, qui ne sera pas modifié par l'algorithme. Lors de l'exécution, chaque sommet va obtenir une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ est la *vue locale* du sommet v. Informellement, la vue locale contient l'information la plus récente que v a de ses voisins. Lors de l'exécution, l'algorithme va mettre à jour cette vue locale pour qu'elle contienne les numéros courants des voisins de v. Ainsi, $N(v)$ va toujours être un ensemble fini d'entiers.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N})$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

Un Ordre sur les Vues Locales

Comme pour l'algorithme de Mazurkiewicz [Maz97], les bonnes propriétés de notre algorithme reposent sur un ordre sur les vues locales, i.e., sur les ensembles finis d'entiers. L'ordre choisi doit être tel que lors de l'exécution de l'algorithme, la vue locale de chaque sommet ne puisse pas diminuer.

Étant donnés deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ distincts, on dit que $N_1 \prec N_2$ si le maximum de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2. Cet ordre est un ordre total et en particulier, l'ensemble vide est minimum pour l'ordre $\prec$.

Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre $\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(\mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

Les Règles de Réétiquetage

On décrit maintenant les règles de réétiquetages qui définissent l'algorithme d'énumération.

Pour lancer l'algorithme, il y a une règle spéciale $\mathcal{M}_0$ qui permet à chaque sommet de modifier son étiquette initiale $\lambda(v)$ pour obtenir l'étiquette $(\lambda(v), 0, \emptyset, \emptyset)$.

Les deux premières règles $\mathcal{M}_1$ et $\mathcal{M}_2$ sont proches des deux règles de l'algorithme de Mazurkiewicz. La première règle $\mathcal{M}_1$ permet à un sommet v de mettre à jour sa boîte-aux-lettres en observant la boîte-aux-lettres de l'un de ses voisins.

$\mathcal{M}_1$:

(ℓ_1, n_1, N_1, M_1) (ℓ_2, n_2, N_2, M_2) $\longrightarrow$ (ℓ_1, n_1, N_1, M_1') (ℓ_2, n_2, N_2, M_2)

Si $M_2 \setminus M_1 \neq \emptyset$ alors $M_1' := M_1 \cup M_2$.

La deuxième règle $\mathcal{M}_2$ ne dépend que de l'étiquette d'un seul sommet. Elle permet à un sommet v de modifier son numéro si son numéro courant est 0 (il n'a pas encore choisi de numéro) ou si la boîte-aux-lettres de v contient un message indiquant qu'il existe un autre sommet ayant le même numéro que v et qui a une étiquette plus grande ou une vue locale plus forte.

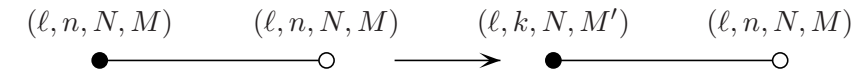
$\mathcal{M}_2$:

$(\ell, n, N, M) \longrightarrow (\ell, k, N, M')$

Si $n = 0$ ou $\exists (n, \ell', N') \in M$ tel que $(\ell, N) \prec (\ell', N')$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$ et
$M' := M \cup \{(k, \ell, N)\}$.

La troisième règle $\mathcal{M}_3$ permet à un sommet v de modifier son numéro s'il a un voisin v' qui est exactement dans le même état, i.e., $(\lambda(v), n(v), N(v), M(v)) = (\lambda(v'), n(v'), N(v'), M(v'))$. Cette règle ne peut être appliquée que si la règle $\mathcal{M}_2$ ne peut pas être appliquée par v ou v'.

$\mathcal{M}_3$:

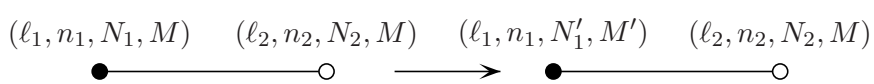


Si $n > 0$ et
$\forall (n, \ell', N') \in M,\ (\ell', N') \preceq (\ell, N)$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', N') \in M\}$ et
$M' := M \cup \{(k, \ell, N)\}$.

La quatrième règle $\mathcal{M}_4$ permet à un sommet v d'ajouter le numéro $n(v')$ d'un de ses voisins v' a sa vue locale. Cette règle ne peut être appliquée que si les trois règles précédentes ne peuvent pas être appliquées.

Lorsque le numéro $n(v')$ est ajouté à la vue locale $N(v)$ de v, tous les éléments $m < n(v')$ sont supprimés de $N(v)$. La justification de cette suppression est la suivante. Si un sommet v se synchronise avec un de ses voisins v' et qu'il remarque que le numéro $n(v')$ de v' n'apparaît pas dans sa vue locale, il doit ajouter ce numéro à $N(v)$, puisque le rôle de $N(v)$ est de stocker les numéros des voisins de v. Mais on peut alors se retrouver dans deux situations. Si v se synchronise pour la première fois avec v', il suffit d'ajouter $n(v')$ à $N(v)$. Mais, il se peut aussi que v se soit déjà synchronisé avec v' auparavant et qu'entre-temps, v' ait changé de numéro. Dans ce cas là, on doit non seulement ajouter le numéro courant de v' dans $N(v)$ mais on doit aussi effacer l'ancien numéro de v' de $N(v)$. Le problème est que v n'a a priori aucun moyen de savoir lequel des numéros de $N(v)$ il doit effacer et en fait, il n'est même pas capable de savoir s'il s'est déjà synchronisé avec v' auparavant ou non. Cependant, notre algorithme assure que lorsqu'un sommet change de numéro, son numéro ne peut qu'augmenter. Par conséquent, on sait que l'ancien numéro de v' est inférieur à son numéro actuel $n(v')$ et il est donc suffisant d'effacer tous les $m < n(v')$ pour assurer que l'ancien numéro de v' a bien été effacé. Avec cette méthode, on peut aussi effacer les numéros courants d'autres voisins de v de la vue locale de v. Mais ce n'est pas un problème puisque v peut récupérer cette information en se resynchronisant avec ces voisins là.

$\mathcal{M}_4$:



Si $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell_1', N_1') \in M, (\ell_1', N_1') \preceq (\ell_1, N_1)$,
$\forall (n_2, \ell_2', N_2') \in M,\ (\ell_2', N_2') \preceq (\ell_2, N_2)$ et
$n_2 \notin N_1$
alors $N_1' := N_1 \setminus \{n' \in N_1 \mid n' < n_2\} \cup \{n_2\}$ et
$M' := M \cup \{(n_1, \ell_1, N_1')\}$.

6.4.3 Correction de l'Algorithme d'Énumération

On considère un graphe simple étiqueté $\mathbf{G}$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ l'étiquette du sommet v après la $i$ème étape de réétiquetage

de l'algorithme $\mathcal{M}$ décrit ci-dessus. On présente d'abord quelques propriétés qui sont satisfaites par n'importe quelle exécution de l'algorithme.

Propriétés Satisfaites lors de l'Exécution

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur le nombre d'étapes, rappelle quelque propriétés simples qui sont toujours satisfaites par l'étiquetage.

Lemme 6.9 *Pour chaque sommet v et chaque étape i,*

1. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
2. $\forall n' \in N_i(v), n' \neq 0$ *et* $\exists \ell' \in L, \exists N' \in \mathcal{P}_{\text{fin}}(\mathbb{N}), (n', \ell', N') \in M_i(v)$,
3. $n_i(v) \notin N_i(v)$.

L'algorithme $\mathcal{M}$ a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 6.10 *Pour chaque sommet v et chaque étape i,*
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

De plus, à chaque étape i, il existe un sommet v telle qu'au moins une de ces inégalités (ou inclusions) soit stricte pour v.

Preuve :

On note v le sommet dont l'étiquette est modifiée lors de la $(i+1)$ème étape de réétiquetage. Pour tout sommet $w \in V(G)$ différent de v, la propriété est trivialement vraie. De plus, quelle que soit la règle appliquée, on a toujours $M_i(v) \subsetneq M_{i+1}(v)$.

Si $n_i(v) \neq n_{i+1}(v)$, alors la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ a été appliquée, et on a $n_{i+1}(v) = 1 + \max\{n' \mid (n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 6.9, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Si $N_i(v) \neq N_{i+1}(v)$, la règle $\mathcal{M}_4$ a été appliquée à v et à un de ses voisins v'. Pour chaque $n \in N_i(v)$ tel que $n > n_{i+1}(v')$, $n \in N_i(v)$ et puisque $n_{i+1}(v) \in N_{i+1}(v) \setminus N_i(v)$, on a $\max(N_i(v) \triangle N_{i+1}(v)) = n_{i+1}(v') \in N_{i+1}(v)$. Par conséquent $N_i(v) \prec N_{i+1}(v)$. □

Les informations dont dispose chaque sommet dans sa boîte-aux-lettres reflètent des propriétés vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet v' tel que $n_i(v') = m'$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet v' tel que $n_i(v) = m'$.

Lemme 6.11 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid$

$\forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, il existe exactement un élément $(u, i_0) \in U'$ puisqu'à chaque étape, le numéro d'au plus un sommet peut être modifié. Le numéro $n_{i_0}(u) = m$ a donc été modifié à l'étape $i_0 + 1$, mais puisque à cette étape, le sommet u n'avait aucun voisin avec le même numéro m, la règle $\mathcal{M}_3$ n'a pu être appliquée, et par maximalité de $(\lambda(u), N_{i_0}(u))$, la règle $\mathcal{M}_2$ n'a pas non plus pu être appliquée à u à l'étape i_0. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 6.12 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. Soit v le sommet dont l'étiquette est modifiée à l'étape $i + 1$. La propriété est trivialement vraie à l'étape $i + 1$ pour tout sommet $w \in V(G)$ distinct de v.

Si la règle $\mathcal{M}_1$ est appliquée à l'étape $i+1$ à v et à un de ses voisins v', alors $M_{i+1}(v) = M_i(v) \cup M_i(v')$ et la propriété est vérifiée à l'étape $i + 1$ puisqu'elle était vraie pour v et v' à l'étape i.

Si la règle $\mathcal{M}_2$ ou $\mathcal{M}_3$ est appliquée à v à l'étape $i + 1$, alors $M_{i+1}(v) = M_i(v) \cup \{(n_{i+1}(v) = 1 + \max\{m \mid (m, \ell, N) \in M_i(v)\}, \lambda(v), N_i(v))\}$, et par conséquent pour chaque $m \in M_{i+1}(v)$, la propriété reste vraie.

Si la règle $\mathcal{M}_4$ est appliquée à v à l'étape $i+1$, pour tout $(m, \ell, N) \in M_{i+1}(v)$, il existe $(m, \ell, N') \in M_i(v)$ et la propriété reste donc vraie. □

D'après les Lemmes 6.11 et 6.12, on voit qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1, k]$ ou un ensemble $[0, k]$ avec $k \leq |V(G)|$.

De plus, pour chaque sommet v et chaque étape i, si $N_i(v)$ contient un numéro n', alors d'après les Lemmes 6.9 et 6.11, il existe $v' \in V(G)$ tel que $n_i(v') = n'$. Par conséquent $N(v)$ ne peut prendre qu'un nombre fini de valeurs et il en est de même pour $M(v)$. Ainsi le nombre de valeurs différentes que peut prendre l'étiquette de chaque sommet est fini (mais dépend de la taille du graphe). Par ailleurs, on sait que les étiquettes consécutives de chaque sommet v forment une suite croissante et puisqu'on sait qu'à chaque étape i, l'étiquette d'au moins un sommet est modifiée, on sait que toute exécution de l'algorithme termine : la relation $\mathcal{M}$ est noethérienne.

Dans le lemme suivant, on montre que si un numéro n apparaît dans la vue locale d'un sommet v, alors ou bien v a un voisin v' tel que $n(v') = n$, ou bien la règle $\mathcal{M}_4$ peut être appliquée au sommet v et à un de ses voisins v'.

Lemme 6.13 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $n_0 \in N_i(v)$, il existe $v' \in N_G(v)$ tel que, ou bien $n_i(v') = n_0$, ou bien $n_i(v') > \max N_i(v)$.*

Preuve : Soit i_0 la dernière étape où n_0 a été ajouté à $N(v)$: $\forall j \geq i_0, n_0 \in N_j(v)$ et $n_0 \notin N_{i_0-1}(v)$. Cela signifie qu'à l'étape i_0, la règle $\mathcal{M}_4$ a été appliquée à v et v' avec

$n_{i_0}(v') = n_0$. D'après le Lemme 6.9, il existe $(n_0, \ell, N) \in M_{i_0}(v')$. Si $n_i(v') = n_{i_0}(v') = n_0$, alors la propriété est vérifiée.

Dans le cas contraire, pour chaque étape $j \in [i_0, i]$, on note $m(j) = \max N_j(v)$. On sait que $m_{i_0}(v) \geq n_0$ et qu'il existe $(m_{i_0}(v), \ell, N) \in M_{i_0}(v')$. Si un numéro n_1 est ajouté à $N(v)$ lors d'une étape $i_1 \in [i_0, i]$, alors la règle $\mathcal{M}_4$ a été appliquée à v et à un de ses voisins lors de l'étape i_1. De plus $n_1 \leq n_0$, puisque $n_0 \in N_{i_1}(v)$ et par conséquent, $m_{i_0}(v) = m_i(v)$.

Puisque $n_i(v') \neq n_{i_0}(v')$, une des règles $\mathcal{M}_2$ ou $\mathcal{M}_3$ a été appliquée à v' lors d'une étape $i_2 \in [i_0, i]$. Par ailleurs, on sait qu'il existe $(m_{i_0}(v), \ell, N) \in M_{i_0}(v') \subseteq M_{i_2}(v')$ et par conséquent, $n_i(v') \geq n_{i_2}(v') > m_{i_0}(v) = m_i(v) = \max N_i(v)$. La propriété est donc vérifiée. □

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final.

Lemme 6.14 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. *$M_\rho(v) = M_\rho(v')$,*
3. *$(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,*
4. *si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et $N_\rho(v) = N_\rho(v')$,*
5. *$n \in N_\rho(v)$ si et seulement s'il existe $w \in N_G(v)$ tel que $n_\rho(w) = n$; auquel cas, $n_\rho(v) \in N_\rho(w)$.*

Preuve :

1. D'après les Lemmes 6.11 et 6.12 et puisque la règle $\mathcal{M}_2$ ne peut pas être appliquée.
2. Dans le cas contraire, la règle $\mathcal{M}_1$ peut être appliquée.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 6.9.
4. Dans le cas contraire, la règle $\mathcal{M}_2$ peut être appliquée à v ou à v'.
5. D'après le Lemme 6.13 et puisque les règles $\mathcal{M}_3$ et $\mathcal{M}_4$ ne peuvent plus être appliquées.

□

Grâce au Lemme 6.14, on peut prouver que l'étiquetage final permet de construire un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est une submersion de $\mathbf{H}$.

Proposition 6.15 *Étant donné un graphe simple $\mathbf{G}$, on peut construire, à partir de l'étiquetage final obtenu après une exécution ρ de $\mathcal{M}$, un graphe simple $\mathbf{H}$ tel qu'il existe un homomorphisme localement surjectif de $\mathbf{G}$ dans $\mathbf{H}$.*

Preuve : On utilise les notations du Lemme 6.14.

On considère le graphe H défini par $V(H) = \{m \in \mathbb{N} \mid \exists v \in V(G), n_\rho(v) = m\}$ et $E(H) = \{\{m, m'\} \mid \exists v, v' \in V(G); n_\rho(v) = m, n_\rho(v') = m' \text{ et } \{v, v'\} \in E(G)\}$. On définit

un étiquetage η de H en posant $\eta(n_\rho(v)) = \lambda(v)$; d'après le Lemme 6.14, si deux sommets $v, v' \in V(G)$ ont le même numéro, alors $\lambda(v) = \lambda(v')$ et par conséquent, cet étiquetage est bien défini.

D'après le Lemme 6.14, $\{m, m'\} \in E(H)$ si et seulement s'il existe $v, v' \in V(G)$ tels que $n_\rho(v) = m, n_\rho(v') = m'$, $m' \in N_\rho(v)$ et $m \in N_\rho(v')$. D'après le Lemme 6.9, on sait qu'il n'existe aucun $\{n, n\} \in E(H)$: le graphe H ne contient pas de boucle. De plus, par définition, $E(H)$ ne contient pas d'arêtes multiples.

On considère maintenant la fonction $n_\rho : V(G) \to V(H)$. Par définition de H, on sait que si $\{v, v'\} \in E(G)$, alors $\{n_\rho(v), n_\rho(v')\} \in E(G)$. De plus, pour tout $v \in V(G)$, $\eta(n_\rho(v)) = \lambda(v)$ et par conséquent, n_ρ est un homomorphisme de $\mathbf{G}$ dans $\mathbf{H}$. D'après le Lemme 6.14, $m \in N_H(n_\rho(v))$ si et seulement s'il existe $w \in N_G(v)$ tel que $n_\rho(w) = m$ et par conséquent n_ρ est un homomorphisme localement surjectif de $\mathbf{G}$ dans $\mathbf{H}$. □

On considère maintenant un graphe $\mathbf{G}$ qui est minimal pour les submersions. Pour chaque exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, le graphe obtenu à partir de l'étiquetage final est isomorphe à $\mathbf{G}$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique. L'algorithme $\mathcal{M}$ permet de résoudre le nommage sur la famille des graphes minimaux pour les submersions, mais si aucune information à propos de $\mathbf{G}$ n'est disponible, les sommmets ne peuvent pas détecter la terminaison.

Cependant, la détection de la terminaison est possible pour un graphe $\mathbf{G}$ donné. En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 6.11 et 6.12, il sait que tous les sommets de $\mathbf{G}$ ont un numéro unique qui ne va plus être modifié : il peut donc détecter que le nommage est effectué.

Par ailleurs, d'après la Proposition 6.8, on sait que pour tout graphe $\mathbf{G}$ qui n'est pas minimal pour les submersions, il n'existe aucun algorithme utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées qui permettent de résoudre les problèmes du nommage ou de l'énumération sur $\mathbf{G}$. On a donc prouvé le théorème suivant.

Théorème 6.16 *Pour tout graphe simple étiqueté* $\mathbf{G}$*, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour* $\mathbf{G}$ *utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées,*
2. *il existe un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour* $\mathbf{G}$ *utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées,*
3. $\mathbf{G}$ *est minimal pour les submersions.*

Remarque 6.17 *Étant donné un graphe* $\mathbf{G}$ *minimal pour les submersions, pour détecter que l'algorithme* $\mathcal{M}$ *a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de* $\mathbf{G}$*. Ainsi, l'algorithme* $\mathcal{M}$ *permet de résoudre l'énumération et le nommage avec détection de la terminaison sur les graphes minimaux pour les submersions de taille donnée.*

6.4.4 Complexité

On s'intéresse à la complexité de l'algorithme $\mathcal{M}$ présenté ci-dessus. Dans le cadre des calculs locaux, on s'intéresse au nombre de pas de réétiquetages effectuées lors d'une

exécution. La proposition suivante donne une borne supérieure sur le nombre de pas de réétiquetages de toute exécution de l'algorithme $\mathcal{M}$ sur un graphe à n sommets de degré maximal Δ.

Proposition 6.18 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, durant toute exécution de l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes étiquetées,* $O(n^3 2^\Delta)$ *règles sont appliquées.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et une exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$. D'après les Lemmes 6.11 et 6.12, on sait que les règles $\mathcal{M}_2$ et $\mathcal{M}_3$ ne peuvent pas être appliquées plus de $\frac{n(n-1)}{2}$ fois durant l'exécution ρ.

Entre deux étapes où une des règles $\mathcal{M}_2, \mathcal{M}_3$ est appliquée, la règle $\mathcal{M}_4$ est appliquée au plus 2^Δ fois, puisqu'il existe 2^Δ sous-ensembles de $n(N_G(v))$. La règle $\mathcal{M}_4$ est donc appliquée $O(n^2 2^\Delta)$ fois durant l'exécution ρ.

À chaque fois qu'un sommet v modifie son numéro ou sa vue locale, un couple (n_0, ℓ, N) est ajouté $M(v)$. Pour chacun de ces couples, la règle $\mathcal{M}_1$ est appliquée au plus n fois.

Ainsi, durant toute exécution ρ de $\mathcal{M}$ sur $\mathbf{G}$, $O(n^3 2^\Delta)$ règles sont appliquées. □

Remarque 6.19 *Si on considère les graphes de degré maximum* Δ *borné, l'algorithme* $\mathcal{M}$ *présenté dans ce Chapitre s'exécute* $O(n^3)$ *étapes de réétiquetage sur un graphe* $\mathbf{G}$ *de taille* n*, ce qui est comparable au nombre d'étapes nécessaires à l'exécution de l'algorithme de Mazurkiewicz sur* $\mathbf{G}$*. Malgré un pouvoir de synchronisation beaucoup plus faible, l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux cellulaires sur les arêtes permet de nommer sur les graphes minimaux pour les submersions de degré borné aussi «rapidement» que l'algorithme de Mazurkiewicz.*

Comme dans le Chapitre 5, certaines exécutions de l'adaptation de l'algorithme de Mazurkiewicz qu'on a présenté dans ce chapitre nécessitent un nombre de pas de réétiquetages qui peut être exponentiel. Cela est du au fait que lorsqu'un sommet met à jour sa vue locale, il supprime les informations obsolètes, mais il supprime aussi des informations qui peuvent être valides.

On s'intéresse à la mémoire nécessaire à chaque sommet pour stocker son étiquette. On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets de G).

Proposition 6.20 *Pour tout graphe* $\mathbf{G}$ *à* n *sommets de degré maximal* Δ*, l'algorithme* $\mathcal{M}$ *utilisant des calculs locaux sur les arêtes étiquetées nécessite* $O(\Delta n \log n)$ *bits de mémoire par sommet.*

Preuve : On considère un graphe $\mathbf{G}$ à n sommets et m arêtes dont le degré maximal est Δ. 0n sait que la vue locale de chaque sommet contient au plus Δ entiers n_0 qui peuvent être représentés avec $O(\log n)$ bits. Ainsi, pour chaque sommet v, $N(v)$ peut être représenté avec $O(\Delta \log n)$ bits.

Chaque sommet peut ne conserver dans sa boîte-aux-lettres que l'information utile, i.e., l'ensemble $\{(n_0, \ell, N) \in M(v) \mid \forall (n_0, \ell', N') \in M(v), (\ell', N') \preceq (\ell, N)\}$. Ainsi, dans la boîte-aux-lettres de chaque sommet, il existe au plus n triplets (n_0, ℓ, N) dont la taille est en $O(\Delta \log n)$ bits. Par conséquent, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta n \log n)$ bits. □

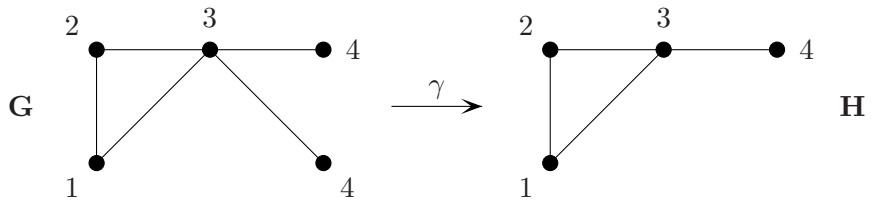


Fig. 24 – Le graphe **G** est une submersion de **H** à travers l'homomorphisme γ qui envoie chaque sommet de G étiqueté i sur l'unique sommet de H étiqueté i. Il n'existe pas d'algorithme d'énumération pour **G**, mais il existe un algorithme d'élection pour **G**.

6.5 Élection

On remarque que s'il existe un algorithme d'énumération avec détection de la terminaison pour un graphe **G**, alors on peut facilement obtenir un algorithme d'élection pour **G**. En effet, une fois qu'un sommet a détecté que tous les sommets ont un identifiant unique, il prend l'étiquette élu si son numéro est égal à 1.

Cependant, dans le modèle des calculs locaux cellulaires sur les arêtes non-étiquetées, les problèmes d'énumération et d'élection ne sont pas équivalents. En effet, le graphe **G** de la Figure 24 n'est pas minimal pour les submersions puisque c'est une submersion de **H**. Par conséquent, il n'y a pas d'algorithme de nommage ou d'énumération pour **G** utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées. Mais si on considère n'importe quelle exécution de l'algorithme $\mathcal{M}$ sur **G**, on atteint une configuration finale, où seul le sommet étiqueté 3 sur la Figure 24 a au moins trois numéros différents dans sa vue locale et il est donc le seul sommet qui sait qu'il a au moins trois voisins : il peut donc se déclarer élu et diffuser l'information aux autres sommets.

6.5.1 Conditions Nécessaires pour l'Élection

On présente ici des conditions que doivent satisfaire les graphes pour lesquels il existe un algorithme d'élection. Étant donné un graphe **G**, on note $\mathcal{S}_{\mathbf{G}}$ l'ensemble des graphes **H** tels que **G** soit une submersion de **H**. D'après le Lemme 6.7, si un algorithme utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées permet d'élire dans le graphe **G**, alors ce même algorithme permet aussi d'élire dans n'importe quel graphe $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$.

Remarque 6.21 *On considère un algorithme utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées $\mathcal{R}$ qui permet d'élire dans* **G**. *Supposons qu'il existe un sous-graphe* $\mathbf{G}'$ *de* **G** *tel que* $\mathbf{G}'$ *soit une submersion d'un graphe* $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$ *à travers un homomorphisme γ. S'il existe une exécution de $\mathcal{R}$ sur* **H** *qui élit un sommet $v \in V(H)$ tel que $|\gamma^{-1}(v)| > 1$, alors d'après le Lemme 6.7, il existe une exécution de $\mathcal{R}$ sur* $\mathbf{G}'$ *tel que l'étiquette* élu *apparaissent au moins deux fois. Puisque chaque exécution de $\mathcal{R}$ sur* $\mathbf{G}'$ *peut être étendue pour obtenir une exécution de $\mathcal{R}$ sur* **G**, *il existe une exécution de $\mathcal{R}$ sur* **G** *tel que l'étiquette* élu *apparaisse au moins deux fois et dans ce cas, $\mathcal{R}$ n'est pas un algorithme d'élection pour* **G**.

Étant donné un graphe $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$ et un sous-graphe $\mathbf{G}'$ de **G** qui est une submersion de **H** à travers un homomorphisme γ, on définit $P_{\mathbf{H}}(\mathbf{G}', \gamma) = \{v \in V(H) \mid |\gamma^{-1}(v)| > 1\}$ et

on sait, d'après la Remarque 6.21 que chaque exécution de $\mathcal{R}$ sur $\mathbf{H}$ ne peut pas élire un sommet $v \in P_{\mathbf{H}}(\mathbf{G}', \gamma)$. On définit alors $P_{\mathbf{H}}(\mathbf{G})$ comme l'union de tous les $P_{\mathbf{H}}(\mathbf{G}', \gamma)$ pour tous les sous-graphes $\mathbf{G}'$ de $\mathbf{G}$ et pour tous les homomorphismes localement surjectifs γ de $\mathbf{G}'$ dans $\mathbf{H}$. On définit aussi l'ensemble $C_{\mathbf{H}}(\mathbf{G}) = V(H) \setminus P_{\mathbf{H}}(\mathbf{G})$ et les éléments de cet ensemble sont appelés les *candidats* de $\mathbf{H}$ pour $\mathbf{G}$.

D'après la Remarque 6.21, un algorithme d'élection pour le graphe $\mathbf{G}$ doit permettre d'élire un sommet de $C_{\mathbf{H}}(\mathbf{G})$ dans n'importe quel graphe $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$. Par conséquent, s'il existe un algorithme d'élection pour le graphe $\mathbf{G}$, alors pour tout graphe $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$, l'ensemble $C_{\mathbf{H}}(\mathbf{G})$ des candidats de $\mathbf{H}$ pour $\mathbf{G}$ est non-vide.

Par ailleurs, s'il existe deux sous-graphes disjoint (qui n'ont aucun sommet en commun) $\mathbf{G}_1$ et $\mathbf{G}_2$ de $\mathbf{G}$ qui sont respectivement des submersions de deux graphes $\mathbf{H}_1$ et $\mathbf{H}_2$ appartenant à $\mathcal{S}_{\mathbf{G}}$, alors il n'existe pas d'algorithme d'élection pour le graphe $\mathbf{G}$. En effet, s'il existe un algorithme $\mathcal{R}$ d'élection pour $\mathbf{G}$, cet algorithme permet d'élire dans $\mathbf{H}_1$ et dans $\mathbf{H}_2$. Par conséquent, d'après le Lemme 6.7, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ tel que l'étiquette ÉLU apparaisse une fois dans $\mathbf{G}_1$ et une fois dans $\mathbf{G}_2$: $\mathcal{R}$ n'est pas un algorithme d'élection pour $\mathbf{G}$.

La proposition suivante résume les conditions nécessaires que doit satisfaire tout graphe $\mathbf{G}$ pour lequel il existe un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

Proposition 6.22 *Pour tout graphe $\mathbf{G}$ pour lequel il existe un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées, les conditions suivantes sont satisfaites.*

1. *pour tout $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$, $C_{\mathbf{H}}(\mathbf{G}) \neq \emptyset$,*
2. *il n'existe pas deux sous-graphes disjoints $\mathbf{G}_1, \mathbf{G}_2$ de $\mathbf{G}$ qui sont respectivement des submersions de deux graphes $\mathbf{H}_1, \mathbf{H}_2 \in \mathcal{S}_{\mathbf{G}}$.*

6.5.2 Un Algorithme d'Élection

On considère dans cette partie un graphe $\mathbf{G}$ qui vérifie les conditions énoncées dans la Proposition 6.22. On veut décrire par des règles de réétiquetage un algorithme tel que lors de toute exécution sur $\mathbf{G}$, on détecte un graphe $\mathbf{H} \in \mathcal{S}_{\mathbf{G}}$ tel qu'il existe un sous-graphe $\mathbf{G}'$ qui est une submersion de $\mathbf{H}$. Pour ce faire, on réutilise l'algorithme d'énumération présentée dans la partie précédente afin de reconstruire le graphe $\mathbf{H}$ et on utilise une adaptation de l'algorithme de détection de la terminaison de Szymansky, Shi et Prywes [SSP85] (SSP) afin de s'assurer qu'un tel sous-graphe $\mathbf{G}'$ existe.

Le principe est le suivant : à partir des boîtes-aux-lettres des sommets obtenues par l'algorithme d'énumération, on reconstruit un graphe $\mathbf{H}$ et ensuite, les sommets de $\mathbf{G}'$ vont vérifier qu'ils sont tous d'accord sur ce graphe.

On va utiliser une adaptation de l'algorithme SSP en utilisant les idées de l'algorithme GSSP introduite dans [MT00, GM03] et présenté dans le Chapitre 2.

Dans le modèle considéré ici, la principale difficulté est qu'un sommet ne peut pas distinguer ses voisins et on ne peut donc pas assurer qu'un sommet observe les valeurs des niveaux de confiance de tous ses voisins avant de modifier son propre niveau de confiance $a(v)$. Afin de récolter le plus d'informations possibles, un sommet va utiliser les numéros qui apparaissent dans sa vue locale afin d'obtenir le plus d'information possible à propos de

ses voisins. Ainsi, un sommet ne pourra augmenter son niveau de confiance $a(v)$ seulement si pour chaque numéro n apparaissant dans sa vue locale, il a un voisin v' tel que $n(v') = n$ et $a(v') \geq a(v)$.

Contrairement à l'adaptation de l'algorithme GSSP présenté dans le Chapitre 5, chaque sommet ne connaît pas son degré et n'est pas assuré que tous ses voisins vont avoir des numéros distincts à partir d'un certain moment. Par conséquent, les sommets doivent pouvoir commencer l'algorithme GSSP à presque tout moment de l'exécution et ils ne peuvent pas attendre d'avoir un certain type d'informations à propos de leurs voisinages comme dans les Chapitres 3 et 5.

Le principe de l'algorithme est de détecter un sous-graphe $\mathbf{G}'$ de $\mathbf{G}$ qui est une submersion d'un graphe $\mathbf{H}' \in \mathcal{S}_{\mathbf{G}}$. Ce sous-graphe peut être $\mathbf{G}$ lui-même, mais les sommets n'ont pas moyen de savoir si c'est le cas ou pas. Une fois qu'un tel sous graphe $\mathbf{G}'$ a été détecté, les sommets doivent élire un sommet de $\mathbf{G}'$, puisque dans le cas où $\mathbf{G}' \simeq \mathbf{G}$, les sommets ne peuvent pas obtenir plus d'information sur $\mathbf{G}$. Cependant, une fois qu'un sommet de $\mathbf{G}'$ a été élu, il ne faut pas qu'un autre sous-graphe $\mathbf{G}''$ de $\mathbf{G}$ qui est une submersion d'un graphe $\mathbf{H}'' \in \mathcal{S}_{\mathbf{G}}$. Puisqu'on suppose que le graphe $\mathbf{G}$ satisfait les conditions de la Proposition 6.22, cela signifie qu'avant d'élire un sommet de $\mathbf{G}'$, il faut s'assurer que les sommets de $\mathbf{G}'$ ne vont pas ensuite participer à la détection d'un autre sous-graphe $\mathbf{G}''$ de $\mathbf{G}$ qui est une submersion d'un graphe $\mathbf{H}'' \in \mathcal{S}_{\mathbf{G}}$.

Ainsi, contrairement aux adaptations de l'algorithme GSSP présenté dans les Chapitres 3 et 5, un sommet ne peut pas toujours réinitialiser son rayon de confiance à -1 s'il a un voisin avec une boîte-aux-lettres différente de la sienne. De cette manière, on assure que si lors d'une étape un sommet de $\mathbf{G}'$ prend l'étiquette élu, les autres sommets de $\mathbf{G}'$ ne modifieront plus leurs boîtes-aux-lettres après cette étape et qu'ils ne pourront alors modifier leur étiquette que pour prendre l'étiquette non-élu. Les restrictions qu'on pose pour qu'un sommet puisse modifier son rayon de confiance nous assurent qu'au plus un sommet de $\mathbf{G}$ prendra l'étiquette élu si $\mathbf{G}$ satisfait les conditions de la Proposition 6.22, mais il faut prouver que ces restrictions ne mènent pas à des situations d'inter-blocage lors de l'exécution de l'algorithme.

Étiquettes

Comme dans la partie précédente, on considère un graphe $\mathbf{G} = (G, \lambda)$ où λ est un étiquetage initial qui ne sera pas modifié par l'algorithme. Lors de l'exécution, chaque sommet v va avoir une étiquette $(\lambda(v), n(v), N(v), M(v), a(v), H(v))$ qui représente les informations suivantes.

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $a(v) \in \mathbb{N}$ est le *niveau de confiance* du sommet v,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$ est la *vue locale* du sommet v. Si un sommet v a un voisin v', les règles de réétiquetage vont permettre à v d'ajouter le couple $(n(v'), a(v'))$ à $N(v)$. Ainsi, $N(v)$ est toujours un ensemble fini de paires d'entiers. Pour tout $N \in \mathcal{P}_{\text{fin}}(\mathbb{N}^2)$, on note $\Pi_1(N) = \{n \mid \exists (n, a) \in N\}$ la projection sur la première composante de N.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N})$ est la *boîte-aux-lettres* du sommet v. Elle va contenir

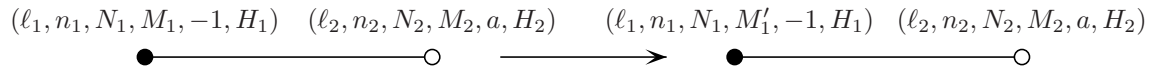
toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les paires de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.
– $H(v)$ est l'*historique* du sommet v. Si à un moment donné $(n, N, M, a) \in H(v)$, cela signifie qu'auparavant le sommet v avait un niveau de confiance a pour la valeur M et qu'à ce moment là, il avait le numéro n et la vue locale N.

Les Règles de Réétiquetage

La première règle de réétiquetage $\mathcal{E}_0$ remplace juste l'étiquette initiale $\lambda(v)$ par $(\lambda(v), 0, \emptyset, \emptyset, -1, \emptyset)$.

Les quatre règles de réétiquetage suivantes correspondent aux quatre règles de l'algorithme d'énumération.

$\mathcal{E}_1$:

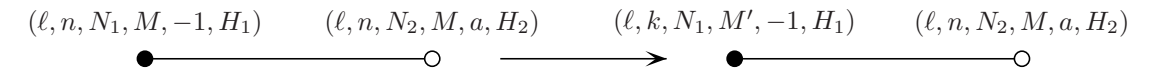


Si $M_2 \setminus M_1 \neq \emptyset$ alors $M_1' := M_1 \cup M_2$.

$\mathcal{E}_2$:

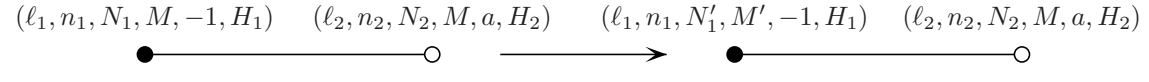
Si $n = 0$ ou $\exists (n, \ell', K') \in M$ tel que $(\ell, \Pi_1(N)) \prec (\ell', K')$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', K') \in M\}$ et
$M' := M \cup \{(k, \ell, \Pi_1(N))\}$.

$\mathcal{E}_3$:



Si $n > 0$,
$\Pi_1(N_1) = \Pi_1(N_2)$ et
$\forall (n, \ell', K') \in M$, $(\ell', K') \preceq (\ell, \Pi_1(N_1))$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', K') \in M\}$ et
$M' := M \cup \{(k, \ell, \Pi_1(N_1'))\}$.

$\mathcal{E}_4$:



Si $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$,
$\forall (n_1, \ell_1', K_1') \in M$, $(\ell_1', K_1') \preceq (\ell_1, \Pi_1(N_1))$,
$\forall (n_2, \ell_2', K_2') \in M$, $(\ell_2', K_2') \preceq (\ell_2, \Pi_1(N_2))$ et
$(n_2, -1) \notin N_1$
alors $N_1' := N_1 \setminus \{(n', a') \in N_1 \mid n' < n_2\} \cup \{(n_2, -1)\}$ et
$M' := M \cup \{(n_1, \ell_1, N_1')\}$.

La cinquième règle permet à un sommet v d'incrémenter son niveau de confiance s'il détecte que tous les voisins qu'il connaît ont un niveau de confiance $a \geq a(v)$.

On explique d'abord comment construire un graphe $\mathbf{H}$ à partir du contenu de la boîte-aux-lettres d'un sommet. À partir d'une boîte-aux-lettres M, pour chaque $n > 0$, on définit $\pi_n(M)$ comme l'ensemble des triplets $(n, \ell, N) \in M$ dont la première composante est n et on pose $u(M) = \{(n, \ell, N) \in M \mid \forall (n, \ell', N') \in \pi_n(M), (n, \ell, N) \preceq (n, \ell', N')\}$; Ainsi, $u(M)$ contient le triplet (n, ℓ, N) de $\pi_n(M)$ pour lequel le couple (ℓ, N) est maximum pour l'ordre $\preceq$.

Ensuite, on définit le graphe $\mathbf{H}_M = (H_M, \eta_M)$ de la manière suivante. S'il existe $(n_1, \ell_1, N_1), (n_2, \ell_2, N_2) \in u(M)$ tels que $n_2 \in N_1$ et $n_1 \notin N_2$, alors le graphe $\mathbf{H}_M$ n'est pas défini. Sinon, le graphe H_M est le graphe dont l'ensemble de sommets est $V(H_M) = \{n \mid (n, \ell, N) \in u(M)\}$ et l'ensemble d'arêtes est $E(H_M) = \{\{n_1, n_2\} \mid \exists (n_1, \ell_1, N_1), (n_2, \ell_2, N_2) \in u(M), n_2 \in N_1 \text{ et } n_1 \in N_2\}$. L'étiquetage η_M de H_M est aussi obtenu à partir de $u(M)$: pour tout $(n, \ell, N) \in u(M)$, on pose $\eta_M(n) = \ell$.

$\mathcal{E}_5$:

$(\ell, n, N, M, a, H) \quad \longrightarrow \quad (\ell, n, N, M, a+1, H)$

Cette règle est applicable si
$n > 0$,
$\forall (n, \ell', K') \in M, (\ell', K') \preceq (\ell, \Pi_1(N))$,
$\mathbf{H}_M \in \mathcal{S}_{\mathbf{G}}$,
$\forall (n', a') \in N, a \leq a'$ et
$a \leq |V(\mathbf{G})| + 1$.

La sixième règle permet à un sommet v de mettre à jour les informations dont il dispose à propos du niveau de confiance de l'un de ses voisins si ce niveau de confiance a augmenté.

$\mathcal{E}_6$:

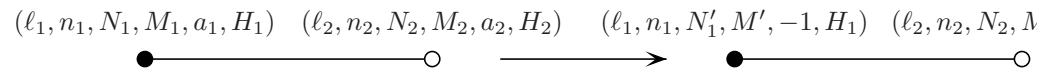
$(\ell_1, n_1, N_1, M, a_1, H_1) \quad (\ell_2, n_2, N_2, M, a_2, H_2) \quad \longrightarrow \quad (\ell_1, n_1, N_1', M, a_1, H_1) \quad (\ell_2, n_2, N_2, M, a_2, H_2)$

Si $a_1 \geq 0$,
$\forall (n_2, \ell_2', N_2') \in M, (\ell_2', N_2') \preceq (\ell_2, \Pi_1(N_2))$ et
$\exists (n_2, a) \in N_1$ tel que $a_2 > a$
alors $N_1' := N_1 \setminus \{(n_2, a)\} \cup \{(n_2, a_2)\}$.

Les règles $\mathcal{E}_7, \mathcal{E}_8$ et $\mathcal{E}_9$ ont pour but d'éviter tout inter-blocage lors de l'exécution de l'algorithme. Si l'une de ces règles est appliquée à un sommet v, alors la boîte-aux-lettres de v est modifiée. Par conséquent, le niveau de confiance de v, qui est associée à une valeur de sa boîte-aux-lettres, est réinitialisée, ainsi que les niveaux de confiance associés aux voisins de v dans sa vue locale.

La règle $\mathcal{E}_7$ permet à un sommet v de modifier sa boîte-aux-lettres M s'il a un voisin v' qui a eu auparavant un niveau de confiance associé à M supérieur à $a(v) - 1$ et qui a entre-temps modifié sa boîte-aux-lettres. Puisque le but de l'historique est de conserver une trace des niveaux de confiance associés aux précédents contenus de la boîte-aux-lettres de v, le sommet v ajoute $(n(v), N(v), M(v), a(v))$ à son historique.

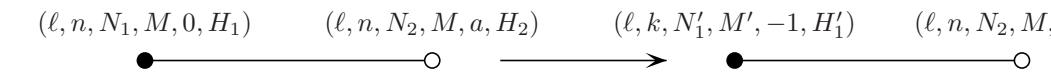
$\mathcal{E}_7$:



Si $M_2 \setminus M_1 \neq \emptyset$ et
ou bien $a_1 = 0$, ou bien $(a_1 \geq 0$ et $\exists (n, N, M_1, a) \in H_2, \exists (n, a') \in N_1, a \geq a')$
alors $N_1' := \{(n, -1) \mid \exists (n, a) \in N_1\}$,
$M' := M_1 \cup M_2$ et
$H_1' := H_1 \cup \{(n_1, N_1, M_1, a_1)\}$.

La règle $\mathcal{E}_8$ permet à un sommet v dont le niveau de confiance est égal à 0 de modifier son état s'il découvre l'existence d'un voisin ayant le même numéro que lui. Ici encore, le sommet v modifie son historique en conséquence.

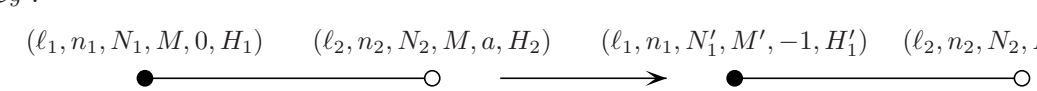
$\mathcal{E}_8$:



Si $n > 0$,
$\Pi_1(N_1) = \Pi_1(N_2)$ et
$\forall (n, \ell', K') \in M$, $(\ell', K') \preceq (\ell, \Pi_1(N_1))$
alors $k := 1 + \max\{n' \mid \exists (n', \ell', K') \in M\}$,
$N_1' := \{(n, -1) \mid \exists (n, a) \in N_1\}$,
$M' := M \cup \{(k, \ell, \Pi_1(N_1'))\}$ et
$H_1' := H_1 \cup \{(n, N_1, M, 0)\}$.

La règle $\mathcal{E}_9$ permet à un sommet v dont le niveau de confiance est 0 de modifier son état s'il découvre un voisin qu'il ne connaît pas, i.e., le numéro de ce voisin n'apparaît pas dans sa vue locale. Ici encore, le sommet v modifie son historique $H(v)$ en conséquence.

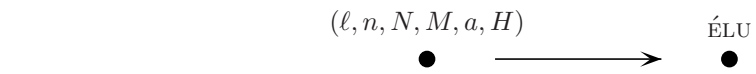
$\mathcal{E}_9$:



Si $n_1 > 0$, $n_2 > 0$, $n_1 \neq n_2$
$\forall (n_1, \ell_1', K_1') \in M$, $(\ell_1', K_1') \preceq (\ell_1, \Pi_1(N_1))$,
$\forall (n_2, \ell_2', K_2') \in M$, $(\ell_2', K_2') \preceq (\ell_2, \Pi_1(N_2))$ et
$\nexists (n_2, a') \in N_1$
alors $N_1' := \{(n, -1) \mid \exists (n, a') \in N_1; n > n_2\} \cup \{(n_2, -1)\}$,
$M' := M \cup \{(n_1, \ell_1, N_1')\}$ et
$H_1' := H_1 \cup \{(n_1, N_1, M, 0)\}$.

La règle $\mathcal{E}_{10}$ permet a un sommet dont le niveau de confiance est égal à $|V(G)| + 2$ de prendre l'étiquette ÉLU si son numéro est le plus grand numéro qui correspond à un candidat du graphe $\mathbf{H}_M$ pour $\mathbf{G}$.

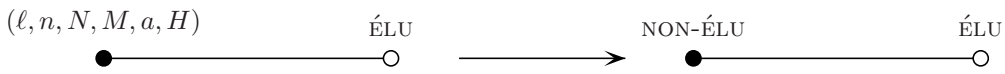
$\mathcal{E}_{10}$:



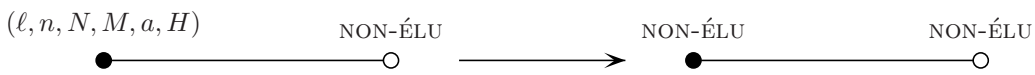
Cette règle est applicable si $n = \max\{n \in C_{\mathbf{H}_M}(\mathbf{G})\}$ et $a = |V(G)| + 2$.

Les deux dernières règles permettent de diffuser l'information que l'élection a été effectuée, une fois qu'un sommet a pris l'étiquette ÉLU.

$\mathcal{E}_{11}$:



$\mathcal{E}_{12}$:



6.5.3 Correction de l'Algorithme d'Élection

L'algorithme décrit par les règles présentées ci-dessus est appelé $\mathcal{E}$ et on appelle $\mathcal{E}'$ l'algorithme décrit par les règles $\mathcal{E}_1, \ldots, \mathcal{E}_9$. Si on prouve que l'algorithme $\mathcal{E}'$ termine, alors il est évident que l'algorithme $\mathcal{E}$ termine puisque l'étiquette de chaque sommet ne peut être modifiée qu'une fois par l'application d'une des règles $\mathcal{E}_{10}, \mathcal{E}_{11}, \mathcal{E}_{12}$. De plus, il est clair que les étiquettes ÉLU et NON-ÉLU sont finales. Il est donc suffisant de prouver la terminaison de $\mathcal{E}'$ et de prouver qu'exactement un sommet peut appliquer la règle $\mathcal{E}_{10}$. En effet, dans ce cas là, il y aura exactement un sommet avec l'étiquette ÉLU dans la configuration finale et tous les autres sommets auront l'étiquette NON-ÉLU.

On note $(\lambda(v), n_i(v), N_i(v), M_i(v), a_i(v), H_i(v))$ l'étiquette d'un sommet v après le $i$ème pas de calcul de l'algorithme $\mathcal{E}'$. Les propriétés de croissance des étiquettes de l'algorithme $\mathcal{M}$ sont conservées.

Lemme 6.23 *Pour chaque sommet v et chaque étape i,*

1. $n_i(v) \leq n_{i+1}(v)$,
2. $M_i(v) \subseteq M_{i+1}(v)$,
3. $H_i(v) \subseteq H_{i+1}(v)$,
4. *si* $H_i(v) = H_{i+1}(v)$, *alors* $\Pi_1(N_i(v)) \preceq \Pi_1(N_{i+1}(v))$, $a_i(v) \leq a_{i+1}(v)$ *et* $\forall (n, a_1) \in N_i(v)$, $(n, a_2) \in N_{i+1}(v), a_1 \leq a_2$.

Et pour chaque étape i, il existe exactement un sommet v tel que l'une de ces inégalités soit stricte pour v.

Comme pour l'algorithme $\mathcal{M}$, on peut montrer que pour tout sommet v et pour toute étape i, la valeur de $n_i(v)$ est toujours inférieure ou égale à $|V(G)|$. Puisque par ailleurs, la valeur de $a_i(v)$ est toujours comprise entre -1 et $|V(G)| + 2$, on sait que les valeurs que

peuvent prendre $N_i(v), M_i(v)$ et $H_i(v)$ sont aussi en nombre fini. D'après le Lemme 6.23, on peut donc déduire que chaque exécution de $\mathcal{E}'$ termine.

Dans le lemme suivant, on montre que pour tout sommet v et pour toute étape i, si un niveau de confiance a est associée à un numéro n dans la vue locale de v, alors $a \geq a_i(v)-1$.

Lemme 6.24 *Pour chaque sommet v et chaque étape i, pour tout $(n,a) \in N_i(v)$, $a \geq a_i(v) - 1$.*

Preuve : On prouve cette propriété par récurrence sur le nombre d'étapes. Initialement, $N(v) = \emptyset$ pour tout sommet v et la propriété est trivialement vraie.

On suppose maintenant que la propriété est vraie après la $i$ème étape. Soit v le sommet dont l'étiquette est modifiée lors de la $(i+1)$ème étape ; pour tout sommet $w \in V(G)$ distinct de v, la propriété est conservée. Si une des règles $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_7, \mathcal{E}_8, \mathcal{E}_9$ est appliquée à v lors de l'étape $i+1$, alors $a_{i+1}(v) = -1$ et la propriété est conservée.

Si la règle $\mathcal{E}_5$ est appliquée à v à l'étape $i+1$, pour tout $(n,a) \in N_i(v) = N_{i+1}(v)$, $a \geq a_i(v) = a_{i+1}(v) - 1$: la propriété est conservée.

Si la règle $\mathcal{E}_6$ est appliquée à l'étape i à v en fonction de l'étiquette d'un de ses voisins v', alors $N_{i+1}(v) = N_i(v) \setminus \{(n_i(v'), a_1)\} \cup \{(n_i(v'), a_i(v'))\}$ et $a_i(v') > a_1$. Puisque par hypothèse de récurrence, $a_1 \geq a_{i+1}(v) - 1$, on a alors $a_i(v') > a_{i+1}(v) - 1$ et la propriété est conservée. □

Dans le lemme suivant, on montre que si la règle $\mathcal{E}_7$ est appliquée à une étape $i+1$ à un sommet v dont le niveau de confiance était supérieur ou égal à 1, alors il existe un voisin v' de v tel que l'une des règles $\mathcal{E}_7, \mathcal{E}_8, \mathcal{E}_9$ a été appliqué à v' à une étape $i'+1 \leq i$ telle que $M_i(v) = M_{i'}(v')$ et $a_{i'}(v') \geq a_i(v) - 1$.

Lemme 6.25 *Pour chaque sommet $v \in V(G)$ et chaque étape i tels que $a_i(v) \geq k+1 \geq 1$ et $M_i(v) \subsetneq M_{i+1}(v)$, il existe un sommet $v' \in N_G(v)$ et une étape $i' < i$ tels que $M_{i'}(v') = M_i(v) \subsetneq M_{i'+1}(v')$ et $a_{i'}(v') \geq k$.*

Preuve : Puisque $a_i(v) \geq k+1$ et $M_i(v) \subsetneq M_{i+1}(v)$, cela signifie que $a_{i+1}(v) = -1$ et que la règle $\mathcal{E}_7$ a été appliquée à l'étape $i+1$ à v en fonction de l'étiquette de l'un de ses voisins v'.

Par conséquent, il existe $(n, N, M_i(v), a) \in H_i(v')$ tel que $a \geq a_i(v) - 1 \geq k$. Si $a \geq 1$, alors il existe une étape $i'+1 \leq i$ où la règle $\mathcal{E}_7$ a été appliqué à v' et par conséquent, $a_{i'}(v') = a \geq k$ et $M_{i'}(v') = M_i(v) \subsetneq M_{i'+1}(v')$.

Si $a = 0$, alors $k = 0$ et il existe une étape $i'+1$ où une des règles $\mathcal{E}_7, \mathcal{E}_8, \mathcal{E}_9$ a été appliquée à v'. Par conséquent, $a_{i'}(v') = 0 \geq k$ et $M_{i'}(v') = M_i(v) \subsetneq M_{i'+1}(v')$. □

Dans la proposition suivante, on montre que chaque exécution de $\mathcal{E}'$ sur $\mathbf{G}$ ne s'arrête pas avant qu'un sommet ait un niveau de confiance égal à $|V(G)| + 2$.

Proposition 6.26 *Pour toute exécution de $\mathcal{E}'$ sur $\mathbf{G}$, il existe un sommet $v \in V(G)$ et une étape i telle que $a_i(v) = |V(G)| + 2$.*

Preuve : On sait déjà que toute exécution de $\mathcal{E}'$ sur $\mathbf{G}$ termine : on considère donc une exécution ρ de $\mathcal{E}'$ sur $\mathbf{G}$ et on note $(\lambda, n_\rho, N_\rho, M_\rho, a_\rho, H_\rho)$ l'étiquetage final obtenu. On suppose que pour tout sommet v, $a_\rho(v) \leq |V(G)| + 1$.

On suppose d'abord que pour tout sommet v, $a_\rho(v) = -1$. D'après la proposition 6.15, on sait que l'étiquetage final n_ρ permet de construire un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est

une submersion de $\mathbf{H}$ à travers n_ρ, i.e., $\mathbf{H} \in \mathcal{S}_\mathbf{G}$. Par conséquent, pour chaque sommet v, $\mathbf{H}_{M_\rho(v)} = \mathbf{H} \in \mathcal{S}_\mathbf{G}$ et l'une des deux règles $\mathcal{E}_2, \mathcal{E}_5$ peut être appliquée à v : l'exécution de $\mathcal{E}'$ n'est donc pas terminée.

On considère maintenant un sommet v tel que $a_\rho(v) \geq 1$. Puisqu'on ne peut pas appliquer la règle $\mathcal{E}_5$ à v, on sait d'après le Lemme 6.24 qu'il existe $(n', a_\rho(v)-1) \in N_\rho(v)$. Considérons alors la dernière étape i où la règle $\mathcal{E}_6$ a été appliquée à v en fonction de l'un de ses voisins v' tel que $n_i(v') = n'$. Ainsi, $a_i(v') = a_\rho(v)-1 \geq 0$ et donc $M_\rho(v) = M_i(v) = M_i(v')$. Si $M_i(v') \subsetneq M_\rho(v')$, alors on peut appliquer la règle $\mathcal{E}_7$ à v en fonction de l'étiquette de v'. Par conséquent, $M_i(v') = M_\rho(v')$, $n_i(v') = n_\rho(v')$ et $\Pi_1(N_i(v')) = \Pi_1(N_\rho(v'))$. Puisque la règle $\mathcal{E}_6$ ne peut pas être appliquée à v en fonction de l'étiquette de v', cela signifie que $a_\rho(v') = a_\rho(v) - 1$. Par conséquent, pour chaque sommet v tel que $a_\rho(v) \geq 1$, il existe un sommet v' tel que $M_\rho(v) = M_\rho(v')$ et $a_\rho(v') = 0$.

On considère maintenant un sommet v dont le niveau de confiance $a_\rho(v)$ est égal à 0 et tel que pour tout sommet $v' \in V(G)$, si $a_\rho(v') \geq 0$, alors $M_\rho(v) = M_\rho(v')$ ou $M_\rho(v') \setminus M_\rho(v) \neq \emptyset$. Puisque la règle $\mathcal{E}_5$ ne peut pas être appliquée à v, il existe $(n', -1) \in N_\rho(v)$. Si on considère la dernière étape i où la règle $\mathcal{E}_6$ a été appliquée à v en fonction de l'étiquette d'un voisin v' tel que $n_i(v') = n'$, alors $a_i(v') = -1$.

Supposons que $a_\rho(v') = -1$. Puisqu'on ne peut appliquer ni la règle $\mathcal{E}_1$ à v', ni la règle $\mathcal{E}_7$ à v, on sait que $M_\rho(v) = M_\rho(v')$ et donc $\mathbf{H}_{M_\rho(v')} \in \mathcal{S}_\mathbf{G}$. Par conséquent, une des deux règles $\mathcal{E}_2, \mathcal{E}_5$ peut être appliquée à v' et l'exécution n'est pas terminée.

Supposons que $a_\rho(v') \geq 0$. Puisque la règle $\mathcal{E}_7$ ne peut pas être appliquée à v, on sait que $M_\rho(v) = M_\rho(v')$. Si $n_\rho(v) = n_\rho(v')$, alors $\Pi_1(N_\rho(v)) = \Pi_1(N_\rho(v'))$ et on peut donc appliquer la règle $\mathcal{E}_8$ à v en fonction de l'étiquette de v'. Par conséquent, $n_\rho(v) \neq n_\rho(v')$ et d'après le Lemme 6.13, ou bien $n_\rho(v') = n_i(v')$, ou bien $n_\rho(v') \notin \Pi_1(N_\rho(v))$. Dans le premier cas, la règle $\mathcal{E}_6$ peut être appliquée à v en fonction de l'étiquette de v' ; dans le second cas, on peut appliquer la règle $\mathcal{E}_9$ à v en fonction de l'étiquette de v'.

Par conséquent, il n'existe pas d'exécution ρ de $\mathcal{E}'$ qui termine dans une configuration finale où il n'existe aucun sommet $v \in V(G)$ avec $a_\rho(v) = |V(G)| + 2$. □

La propriété cruciale de l'algorithme est donnée dans la proposition suivante. On montre que si le niveau de confiance d'un sommet v est $|V(G)| + 2$, alors $\mathbf{G}$ a un sous graphe $\mathbf{G}'$ qui est une submersion de $\mathbf{H}_{M(v)}$.

Proposition 6.27 *S'il existe un sommet $v_0 \in V(G)$ et une étape i_0 tels que $a_{i_0}(v_0) = |V(G)| + 2$, alors il existe un sous-graphe $\mathbf{G}'$ de $\mathbf{G}$ qui est une submersion de $\mathbf{H}_{M_{i_0}(v_0)}$.*

De plus pour tout sommet $v \in V(G')$ et pour toute étape $i \geq i_0$, $M_i(v) = M_{i_0}(v_0)$ et il existe une étape $i_1 \geq i_0$ où un sommet peut appliquer la règle $\mathcal{S}_{10}$.

Preuve : On considère un sommet $v_0 \in V(G)$ et une étape i_0 tels que $a_{i_0}(v_0) = |V(G)|+2$; on pose $M_0 = M_{i_0}(v_0)$.

On définit une fonction partielle $i : V(G) \to \mathbb{N}$ de la manière suivante. Étant donné un sommet $v \in V(G)$, s'il existe une étape $j \leq i_0$ telle que $M_j(v) = M_0$, alors $i(v) = \max\{j \leq i_0 \mid M_j(v) = M_0\}$; sinon $i(v)$ n'est pas défini.

On peut maintenant définir récursivement une suite d'ensembles de sommets $(V_j)_{j\in\mathbb{N}}$ à l'aide de la fonction i. On pose $V_0 = \{v_0\}$ et $V_{k+1} = V_k \cup \{w \mid \exists v \in N_G(w) \cap V_k ; a_{i(w)}(w) \geq a_{i(v)}(v) - 1\}$.

Soit $\mathbf{G}' = (G', \nu)$ le sous-graphe de $\mathbf{G}$ défini par $V(G') = \bigcup_{j\in\mathbb{N}} V_j = V_{|V(G)|}$ et $E(G') =$

$\big\{\{v,v'\} \in E(G) \mid v,v' \in V(G');\exists (n_{i(v)}(v),a) \in N_{i(v')}(v');a_{i(v)}(v) \geq a\big\}$. L'étiquetage ν de G' est la restriction de l'étiquetage λ de G à G'. On note γ l'homomorphisme de $\mathbf{G}'$ dans $\mathbf{H}_{M_0}$ défini par $\gamma(v) = n_{i(v)}(v)$.

Pour chaque sommet $v \in V_p \setminus V_{p-1}$, il existe un chemin élémentaire de longueur p de v à v_0. Ainsi, pour chaque sommet $v \in V(G')$ tel que $a_{i(v)}(v) = |V(G)| + 2 - p$, il existe un chemin élémentaire de v à v_0 de longueur supérieure ou égale à p, puisqu'un tel sommet v appartient à $V_k \setminus V_{k-1}$ où $k \geq p$. De plus, pour tout sommet $v \in V(G')$, $a_{i(v)}(v) \geq |V(G)| + 2 - |V(G')| \geq 2$.

On considère maintenant un sommet $v \in V(G')$. Si $(n,a) \in N_{i(v)}(v)$, cela signifie que $a \geq a_{i(v)}(v) - 1 \geq 1$ et par conséquent, la règle $\mathcal{E}_6$ a été appliquée à une étape $j \leq i(v)$ à v en fonction de l'étiquette de l'un de ses voisins w tel que $n_j(w) = n, M_j(w) = M_{i(v)}(v) = M_0$ et $a_j(w) = a$. Par conséquent $i(w)$ est défini et $a_{i(w)}(w) \geq a_j(w) = a \geq a_{i(v)}(v) - 1$. Ainsi, $w \in V(G')$ et l'homomorphisme γ est localement surjectif pour tout $v \in V(G')$: le graphe $\mathbf{G}'$ est une submersion de $\mathbf{H}_{M_0}$.

On suppose qu'il existe un sommet $v \in V(G')$ et une étape j telle que $M_j(v) = M_0 \subsetneq M_{j+1}(v)$. On considère alors un sommet $v_1 \in V(G')$ et une étape j_1 tels que $M_{j_1}(v_1) = M_0 \subsetneq M_{j_1+1}(v_1)$ et pour tout $v \in V(G')$, il existe une étape $j \geq j_1$ telle que $M_j(v) = M_0(v)$.

Puisque $a_{j_1}(v_1) \geq |V(G)| + 2 - |V(G')|$, on peut appliquer le Lemme 6.25 afin de trouver une suite de sommets $(v_1, \ldots, v_k)$ et une suite décroissante d'étapes $(j_1, \ldots, j_k)$ telles que $k = |V(G)| - |V(G')| + 1$ et pour tout $1 \leq p \leq k$, $M_{j_p}(v_p) = M_0 \subsetneq M_{j_p+1}(v_p)$ et $a_{j_p}(v_p) \geq k+1-p$. En raison de notre choix de v_1, on sait que pour tout $p \geq 2$, $v_p \notin V(G')$. Puisque $k = |V(G)| - |V(G')| + 1$, il existe deux étapes $p > q$ telles que $v_p = v_q$. Mais alors, $M_0 = M_{j_p}(v_p) \subsetneq M_{j_p+1}(v_p) \subseteq M_{j_q}(v_q) = M_0$, ce qui est impossible. Par conséquent, pour tout sommet $v \in V(G')$ et pour toute étape $i > i_0$, $M_i(v) = M_0$.

On va maintenant montrer qu'il existe une étape $i_1 \geq i_0$ où la règle $\mathcal{E}_{10}$ peut être appliquée à un sommet v. Supposons qu'on arrive à une configuration finale où la règle $\mathcal{E}_{10}$ ne peut être appliquée à aucun sommet $v \in V(G)$. On considère alors l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho, a_\rho, H_\rho)$ de $\mathbf{G}$.

On sait que $M_\rho(v_0) = M_{i_0}(v_0)$ et $a_\rho(v_0) = a_{i_0}(v_0) = |V(G)|+2$. On peut donc redéfinir le graphe $\mathbf{G}'$ comme précédemment en considérant la dernière étape de l'exécution à la place de l'étape i_0. Si dans la configuration finale, $a(v) = |V(G)| + 2$ pour tout sommet $v \in V(G')$, il existe un unique sommet $v_1 \in V(G')$ tel que $n(v_1) = \max\{n \in C_{\mathbf{H}_{M_0}}(\mathbf{G})\}$ puisque $\mathbf{G}$ vérifie les conditions de la Proposition 6.22 : on peut donc appliquer la règle $\mathcal{E}_{10}$ au sommet v_1.

Dans le cas contraire, on considère un sommet $v \in V(G')$ dont le niveau de confiance est minimum. Puisqu'on ne peut pas appliquer la règle $\mathcal{E}_5$ à v, cela signifie qu'il existe $(n, a_\rho(v)-1) \in N_\rho(v)$. On considère alors la dernière étape j où la règle $\mathcal{E}_6$ a été appliquée à v en fonction de l'étiquette d'un de ses voisins v' tel que $n_j(v') = n$. Comme expliqué précédemment, $v' \in V(G')$ et $\{v,v'\} \in E(G')$. Par conséquent, $M_\rho(v) = M_j(v) = M_j(v') = M_\rho(v')$ et $a_j(v') = a_\rho(v) - 1$. Ainsi, puisque $n_\rho(v') = n_j(v')$ et que $a_\rho(v') \geq a_\rho(v) > a_\rho(v) - 1$, on peut appliquer la règle $\mathcal{E}_6$ à v en fonction de l'étiquette de v' et l'exécution n'est pas terminée.

Il existe donc toujours une étape $i_1 > i_0$ où la règle $\mathcal{E}_{10}$ peut être appliquée à un sommet $v \in V(G)$. □

D'après la Proposition 6.27, si un sommet v_0 a un niveau de confiance égal à $|V(G)|+2$ après une étape i_0, il existe un sous-graphe $\mathbf{G}'$ de $\mathbf{G}$ qui est une submersion de $\mathbf{H}_{M_{i_0}(v_0)} \in \mathcal{S}_\mathbf{G}$ et tel que pour tout sommet $v \in V(G')$ et toute étape $i > i_0$, $\mathbf{H}_{M_i(v)} = \mathbf{H}_{M_{i_0}(v_0)}$ et $a_i(v) \geq 1$.

On sait qu'il n'existe pas deux sous-graphes disjoints $\mathbf{G}_1, \mathbf{G}_2$ de $\mathbf{G}$ tels que $\mathbf{G}_1$ (resp. $\mathbf{G}_2$) soit une submersion d'un graphe $\mathbf{H}_1 \in \mathcal{S}_\mathbf{G}$ (resp. $\mathbf{H}_2 \in \mathcal{S}_\mathbf{G}$). Par conséquent, il y a exactement un sommet de $\mathbf{G}$ qui peut appliquer la règle $\mathcal{E}_{10}$ et l'algorithme $\mathcal{E}$ est bien un algorithme d'élection pour $\mathbf{G}$.

On a donc montré qu'il existe un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées pour tout graphe qui satisfait les conditions nécessaires de la Proposition 6.22.

Théorème 6.28 *Il existe un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées pour un graphe simple étiqueté* $\mathbf{G}$ *si et seulement si les conditions suivantes sont vérifiées :*

1. *pour tout* $\mathbf{H} \in \mathcal{S}_\mathbf{G}$, $C_\mathbf{H}(\mathbf{G}) \neq \emptyset$,
2. *il n'existe pas deux sous-graphes disjoints* $\mathbf{G}_1, \mathbf{G}_2$ *de* $\mathbf{G}$ *qui sont respectivement des submersions de deux graphes* $\mathbf{H}_1, \mathbf{H}_2 \in \mathcal{S}_\mathbf{G}$.

Remarque 6.29 *Contrairement à l'algorithme* $\mathcal{M}$*, l'algorithme* $\mathcal{E}$ *nécessite de connaître exactement le graphe* $\mathbf{G}$*, afin de pouvoir calculer la famille* $\mathcal{S}_\mathbf{G}$.

6.6 Exemples

Si dans un graphe $\mathbf{G}$, chaque sommet a un identifiant unique, lors le graphe $\mathbf{G}$ est minimal pour les submersions. La connaissance de la taille du graphe permet à l'algorithme $\mathcal{M}$ de résoudre le nommage avec détection de la terminaison sur $\mathbf{G}$, et par conséquent de résoudre l'élection sur $\mathbf{G}$.

On sait que tout graphe complet est minimal pour les submersions, et par conséquent, la connaissance de la taille du graphe permet de résoudre l'élection et le nommage avec détection de la terminaison avec des calculs locaux cellulaires sur les arêtes non-étiquetées.

6.6.1 Arbres, Grilles et Graphes Bipartis

Étant donné un graphe biparti non-étiqueté G avec au moins trois sommets, G peut être colorié (au sens classique) avec deux couleurs. Une telle coloration montre que G est une submersion du graphe complet à deux sommets K_2. Puisque G a au moins trois sommets, G est une submersion propre de K_2 et il n'existe donc pas d'algorithme de nommage pour G utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

De plus, étant donné un homomorphisme localement surjectif $\gamma_1 : G \to K_2$, on peut aisément construire un autre homomorphisme localement surjectif $\gamma_2 : G \to K_2$ obtenu en inversant $\gamma_1^{-1}(v_1)$ et $\gamma_1^{-1}(v_2)$, où v_1 et v_2 sont les sommets de K_2. Par conséquent, pour tout graphe biparti G ayant au moins trois sommets, ou bien $|\gamma_1^{-1}(v_1)| > 1$ et $|\gamma_2^{-1}(v_2)| > 1$, ou bien $|\gamma_2^{-1}(v_1)| > 1$ et $|\gamma_1^{-1}(v_2)| > 1$. Ainsi, il n'existe pas d'algorithme d'élection pour le graphe G utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

On ne peut donc pas résoudre l'élection ou le nommage avec des calculs locaux cellulaires sur les arêtes non-étiquetées dans un graphe biparti non-étiqueté. Par conséquent,

on ne peut pas résoudre l'élection et le nommage dans ce modèle sur les arbres ou les grilles ayant au moins trois sommets.

6.6.2 Anneaux de Taille Première

Il est connu qu'on ne peut pas résoudre les problèmes de nommage ou d'élection sur un anneau anonyme dans un modèle asynchrone où les processus communiquent en échangeant des messages [Tel00]. Par conséquent, il est nécessaire d'avoir un mécanisme de synchronisation entre voisins pour pouvoir résoudre ces problèmes sur un anneau anonyme. Néanmoins, même dans le modèle des calculs locaux sur les étoiles fermées (le modèle impliquant le plus de synchronisation parmi ceux étudiés dans ce mémoire), on ne peut pas résoudre ce problème sur un anneau de taille quelconque : on ne peut pas résoudre le problème si la taille de l'anneau n'est pas un nombre premier (sauf pour l'anneau à quatre sommets). On va montrer ici que les calculs locaux cellulaires sur les arêtes non-étiquetées sont un modèle de calcul assez puissant pour résoudre élection et nommage sur les anneaux de taille première. Ce résultat vient de la proposition suivante.

Proposition 6.30 *Un anneau non-étiqueté de taille p est minimal pour les submersions si et seulement si p est un nombre premier.*

Preuve : On considère un anneau non-étiqueté C_p de taille p où $V(C_p) = \{c_0, c_1, \ldots, c_{p-1}\}$, $E(C_p) = \{\{c_i, c_{i+1 \mod p}\} \mid 0 \leq i < p\}$.

Si 2 divise p, alors le graphe C_p est biparti et C_p est une submersion de K_2. Si $q > 2$ divise p, alors C_p est un revêtement simple propre de C_q et par conséquent, C_p est une submersion propre de C_q.

Réciproquement, on considère un homomorphisme localement surjectif $\gamma : C_p \to H$ tel que H n'est pas isomorphe à C_p. Puisque l'image d'un sommet de degré 2 par un homomorphisme localement surjectif est un sommet de degré 1 ou 2, le graphe H est ou bien un anneau C_q, ou un chemin P_q (avec $V(P_q) = \{p_0, \ldots, p_{q-1}\}$ et $E(P_q) = \{\{p_i, p_{i+1}\} \mid i \in [0, q-1]\}$). Dans le premier cas, pour tout $v \in V(C_p)$, $|\gamma(N_{C_p}(v))| = |N_{C_p}(v)|$ et γ est donc un revêtement : q divise donc p. Dans le second cas, γ envoie un sommet $v \in V(C_p)$ sur l'extrémité p_0 du chemin P_q et pour tout $1 \leq i < p/2$, γ envoie les deux sommets de C_p à distance i de v sur c_i. Si p est impair, alors les deux sommets à distance $\lfloor p/2 \rfloor$ de v dans C_p sont voisins et sont tous les deux envoyés sur le même sommet par γ, ce qui est impossible. Par conséquent, C_p est un anneau de taille paire. □

Pour toute anneau de taille première C_p, il existe donc des algorithmes de nommage et d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées. C'est un corollaire intéressant des Théorèmes 6.16 et 6.28, puisque le modèle de réétiquetage utilisé ici est le modèle induisant une synchronisation minimale entre deux voisins. De plus, l'algorithme $\mathcal{M}$ ne nécessite aucun sens de la direction sur les anneaux considérés, contrairement à d'autres algorithmes existants.

Par ailleurs, on remarque que d'après la Proposition 6.18, toute exécution de l'algorithme $\mathcal{M}$ ne nécessite que $O(p^3)$ étapes de réétiquetages sur un anneau de taille première C_p, ce qui est comparable au nombre de pas de réétiquetages nécessaire à l'algorithme de Mazurkiewicz pour élire dans un anneau de taille p.

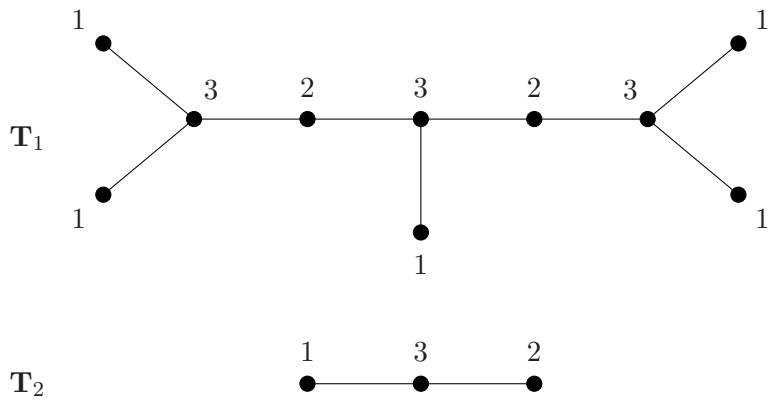


Fig. 25 – Le graphe $\mathbf{T}_1$ est une submersion de $\mathbf{T}_2$. Dans le graphe $\mathbf{T}_1$, l'étiquette de chaque sommet est son degré.

6.7 Connaissance Initiale du Degré

On étudie dans cette partie l'influence de la connaissance initiale du degré sur les calculs locaux cellulaires sur les arêtes non-étiquetées. C'est à dire qu'initialement, chaque sommet v d'un graphe $\mathbf{G} = (G, \lambda)$ connaît son degré, i.e., son degré fait partie de son étiquette initiale $\lambda(v)$.

6.7.1 Pas d'Algorithme d'Élection ou de Nommage pour la Famille des Arbres

Contrairement aux modèles étudiés dans les Chapitres 3 et 5, la connaissance initiale du degré ne permet pas de résoudre le problème de l'élection ou du nommage dans la famille des arbres. En effet, il existe des arbres qui ne vérifient pas les conditions du Théorèmes 6.28, même si les sommets connaissent initialement leurs degrés. Un tel exemple est présenté sur la Figure 25.

6.7.2 Nommage dans les Familles de Diamètre Borné

On étudie dans cette partie l'importance de la connaissance initiale du degré pour pouvoir trouver un algorithme utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées qui permet de nommer sur les familles de graphes minimaux pour les submersions dont le diamètre est borné.

En fait, il est facile de voir qu'on peut modifier l'algorithme $\mathcal{M}$ présenté ci-dessus de la même manière que dans le modèle étudié dans le Chapitre 5 pour obtenir le théorème suivant.

Théorème 6.31 *Pour tout entier B, il existe un algorithme de nommage avec détection de la terminaison utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées avec connaissance initiale du degré pour la famille des graphes minimaux pour les submersions dont le diamètre est bornée par B.*

Ainsi, comme dans les modèles considérés dans les Chapitres 2, 3 et 5, lorsque les sommets connaissent initialement leurs degrés, il n'est pas nécessaire de connaître la taille pour pouvoir nommer dans un graphe $\mathbf{G}$ que l'on sait minimal pour les submersions : une borne sur la taille ou le diamètre est suffisante. Cependant, on ne connaît pas de résultat équivalent pour le problème de l'élection, que les sommets connaissent initialement leurs degrés ou non.

Toutefois, on peut montrer en utilisant la même preuve que dans le Chapitre 3, que si les sommets ne connaissent pas initialement leur degré, il est impossible de résoudre le problème de l'élection ou du nommage sur les graphes minimaux pour les submersions en connaissant seulement une borne sur la taille.

De plus, en utilisant la même preuve, que dans le Chapitre 5, on peut montrer que pour tout entier n qui n'est pas premier, il n'existe pas d'algorithme effectif d'élection ou de nommage pour la classe des graphes de taille n, même si les sommets connaissent initialement leurs degrés. En effet, si on considère les graphes $\mathbf{G}_{p,q}$, $\mathbf{K}_p$ et $\mathbf{H}_{p,q}$ construits dans la preuve du Chapitre 5, on observe que le graphe $\mathbf{G}_{p,q}$ est une submersion de $\mathbf{K}_p$ et que le graphe $\mathbf{H}_{p,q}$ est minimal pour les submersions.

6.8 Conclusion et Perspectives

Dans ce chapitre, on a caractérisé les graphes admettant un algorithme de nommage (Théorème 6.16) et ceux admettant un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées (Théorème 6.28). Ces caractérisations sont basées sur la notion de submersions.

La caractérisation pour le problème du nommage s'exprime simplement en termes de submersions. Cela nous permet de penser qu'il est possible d'utiliser les techniques présentées dans [GM03, GMM04] afin de caractériser les classes de graphes qui peuvent être reconnues par des calculs locaux cellulaires sur les arêtes non-étiquetées avec ou sans connaissance initiale (la terminaison est alors implicite).

Par ailleurs, on a montré que la connaissance initiale du degré permet de résoudre le problème du nommage sur les familles de graphes minimaux pour les submersions (Théorème 6.31). Ce résultat nous permet de penser que les techniques de [GM02] peuvent être utilisées pour caractériser les familles de graphes admettant un algorithme universel de nommage utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

La caractérisation des graphes où on peut résoudre l'élection est un peu plus complexe. De plus, l'algorithme d'élection pour un graphe $\mathbf{G}$ nécessite la connaissance du graphe lui-même (et pas seulement de sa taille) afin de pouvoir construire l'ensemble $\mathcal{S}_{\mathbf{G}}$. Ainsi, pour deux graphes différents, la connaissance initiale est différente. Il pourrait être intéressant de réussir à caractériser les familles de graphe admettant un algorithme d'élection, comme cela a été fait dans le modèle des calculs locaux sur les étoiles fermées [GM02], mais les méthodes utilisés dans [GM02, GM03, MT00] ne semblent pas pouvoir être adaptées facilement dans le modèle des calculs locaux cellulaires sur les arêtes non-étiquetées.

Chapitre 7

Échanges de Messages en Mode Asynchrone

Sommaire

7.1 Introduction

Dans ce chapitre, on considère des réseaux modélisés par des graphes simples disposant d'étiquetages des ports où les processus communiquent en échangeant des messages.

Lorsqu'un processus envoie (ou reçoit) un message, il connaît le numéro du port par lequel le message a été envoyé (ou reçu). Le modèle est asynchrone : il n'y a pas d'horloge globale, les messages parviennent à leurs destinataires en un temps fini mais arbitraire et les processus exécutent les instructions à des vitesses arbitraires. Ce modèle correspond au modèle étudié par Yamashita et Kameda dans [YK96b] et au modèle port-à-port de [YK99].

On donne une caractérisation des graphes disposant d'un étiquetage des ports dans lesquelles on peut résoudre l'élection et le nommage (Théorème 7.32). Cette caractérisation repose sur les notions de fibrations et de revêtements dirigés. Elle peut être obtenue à partir des résultats de Boldi et al [BCG$^+$96] en interprétant leurs résultats dans le modèle étudié ici. Il convient cependant de noter que les techniques présentées dans ce chapitre sont différentes des techniques employées dans [BCG$^+$96]. On présente ensuite une caractérisation des graphes admettant un algorithme d'élection quel que soit l'étiquetage des ports (Théorème 7.34).

Pour obtenir des conditions nécessaires, on représente un graphe avec un étiquetage des ports par un graphe simple étiqueté et on obtient ainsi un résultat d'impossibilité directement à partir du lemme de relèvement d'Angluin [Ang80] présenté dans le Chapitre 3 (Proposition 7.24).

L'algorithme d'énumération présenté dans la Section 7.5 est inspiré de l'algorithme de Mazurkiewicz. Il est très différent de l'algorithme de Yamashita et Kameda [YK96b] (et de l'algorithme de Boldi et al. [BCG$^+$96] qui utilise sur les mêmes techniques que l'algorithme de Yamashita et Kameda) et a des propriétés intéressantes que l'algoritme de Yamashita et Kameda n'a pas. L'algorithme qu'on obtient est un algorithme qui utilise un nombre polynomial de messages de taille polynomiale, alors que l'algorithme de Yamashita et Kameda nécessite des messages de taille exponentielle. Par ailleurs, pour tout graphe G avec un étiquetage des ports, il existe une exécution de notre algorithme qui permet de résoudre l'élection et le nommage. Ainsi, notre algorithme permet parfois d'exploiter l'asymétrie qui peut provenir de l'exécution de l'algorithme et non du graphe sur lequel l'algorithme est exécuté.

On étudie ensuite l'importance des connaissances initiales dans ce modèle (Section 7.6). On montre qu'il suffit de connaître une borne sur le diamètre pour pouvoir nommer ou élire dans un graphe minimal pour les revêtements dirigés symétriques (Théorème 7.42) et que la connaissance d'une borne serrée sur la taille permet d'obtenir un algorithme effectif d'élection et de nommage (Théorème 7.43).

Dans la Section 7.7, on considère d'autres modèles où les processus communiquent par échanges de messages qui ont été étudiés par Yamashita et Kameda [YK99] et on montre que des résultats similaires à ceux présentés pour le modèle considéré dans ce chapitre peuvent être obtenus. On présente aussi les caractérisations obtenues par Yamashita et Kameda [YK99] qui sont exprimées en termes d'étiquetage.

Les résultats présentés dans ce chapitre ont été obtenus en collaboration avec Yves Métivier et une partie de ces résultats a été publiée dans [CM05]. Dans ce modèle, on a obtenu d'autres résultats [CDS06, CGMT07] qui ne sont pas détaillés ici mais qui sont présentés dans la conclusion de ce chapitre.

7.2 Fibrations et Revêtements Dirigés

Dans ce chapitre, on a besoin de considérer des graphes dirigés afin de pouvoir exprimer des conditions nécessaires et suffisantes pour qu'un graphe admette un algorithme de nommage ou d'élection dans les différents modèles considérés.

Les homomorphismes qu'on considère dans ce Chapitre sont les fibrations et les revêtements dirigés. Ces objets ont été introduits par Boldi et al. [BCG$^+$96]. Les définitions et propriétés présentées ici sont dues à Boldi et Vigna [BV02a].

On rappelle ici la définition des fibrations donnée dans le Chapitre 4

Définition 7.1 *Un graphe dirigé D est* fibré *sur un graphe dirigé D' à travers un homomorphisme $\gamma\colon D \to D'$ si pour tout arc $a' \in A(D')$ et pour tout sommet $v \in \gamma^{-1}(t(a'))$, il existe un unique arc $a \in A(D)$ tel que $t(a) = v$ et $\varphi(a) = a'$.*

On dit alors que l'homomorphisme γ est une fibration *de D dans D'.*

Un graphe dirigé D est fibré proprement *sur D' si γ n'est pas un isomorphisme. Naturellement, un graphe dirigé étiqueté (D, λ) est fibré sur un graphe dirigé étiqueté (D', λ') à travers γ si D est fibré sur D' à travers γ et si γ conserve l'étiquetage.*

Comme dans le Chapitre 4, on ne considère que des graphes dirigés fortement connexes, mais les graphes dirigés considérés dans ce chapitre peuvent avoir des boucles. On définit maintenant les graphes minimaux pour les fibrations.

Définition 7.2 *Un graphe dirigé fortement connexe* $\mathbf{D}$ *est* minimal pour les fibrations *si* $\mathbf{D}$ *n'est fibré proprement sur aucun autre graphe dirigé fortement connexe.*

Un graphe simple $\mathbf{G}$ *est* minimal pour les fibrations *si $Dir(\mathbf{G})$ est minimal pour les fibrations discrètes.*

On rappelle ici la définition des fibrations non-triviales donnée dans le Chapitre 4.

Définition 7.3 *Si un graphe dirigé* $\mathbf{D}$ *est fibré sur un graphe dirigé fortement connexe* $\mathbf{D}'$ *à travers une fibration γ, pour tout sommet $v \in V(D')$, l'ensemble $\gamma^{-1}(v)$ est* la fibre *de v. La fibre d'un sommet v est* triviale *si elle est réduite à un singleton, i.e., $|\gamma^{-1}(v)| = 1$ et* non-triviale *sinon.*

La fibration γ est non-triviale *si toutes ses fibres sont non-triviales. On dit alors que* $\mathbf{D}$ *est* non-trivialement *fibré sur* $\mathbf{D}'$.

On donne maintenant la définition des graphes minimaux pour les fibrations non-triviales. Ici aussi, les graphes considérés sont toujours fortement connexes et peuvent avoir des boucles. On remarque que tout graphe minimal pour les fibrations est aussi minimal pour les fibrations non-triviales.

Définition 7.4 *Un graphe dirigé fortement connexe* $\mathbf{D}$ *est* minimal pour les fibrations non-triviales *si* $\mathbf{D}$ *n'est fibré non-trivialement sur aucun autre graphe dirigé fortement connexe.*

Un graphe simple $\mathbf{G}$ *est* minimal pour les fibrations non-triviales *si $Dir(\mathbf{G})$ est minimal pour les fibrations non-triviales.*

On montre dans la proposition suivante que si D fibré sur D', alors on peut relever un étiquetage codéterministe de D' en un étiquetage codéterministe de D.

Proposition 7.5 *Si un graphe dirigé D est fibré sur un graphe dirigé D' à travers γ, pour tout étiquetage codéterministe λ' de D', il existe un étiquetage codéterministe λ de D tel que (D, λ) est fibré sur (D', λ') à travers γ.*

Preuve : On considère un graphe dirigé D fibré sur un graphe dirigé D' à travers γ et un étiquetage codéterministe λ' de D'. On définit un étiquetage λ de D de la manière suivante : pour tout $x \in V(D) \cup A(D)$, $\lambda(x) = \lambda'(\gamma(x))$. Puisque γ est une fibration qui préserve l'étiquetage, (D, λ) est fibré sur (D', λ') à travers γ. De plus, puisque γ est une fibration, pour tout arcs distincts a, a' tels que $t(a) = t(a')$, on a $\gamma(a) \neq \gamma(a')$. Puisque $t(\gamma(a)) = t(\gamma(a'))$ et que λ' est un étiquetage codéterministe, $\lambda(a) \neq \lambda(a')$. Ainsi, λ est aussi un étiquetage codéterministe. □

On introduit maintenant les définitions de revêtements dirigés et de revêtements dirigés symétriques qui sont des fibrations particulières. On va aussi voir les liens entre ces homomorphismes et les revêtements introduits dans le Chapitre 3.

Un graphe dirigé $\mathbf{D}$ est un revêtement d'un graphe dirigé $\mathbf{D'}$ à travers un homomorphisme γ si pour tout sommet $v \in V(D)$, γ induit une bijection entre les arcs incidents à v dans $\mathbf{D}$ et les arcs incidents à $\gamma(v)$ dans $\mathbf{D'}$.

Définition 7.6 *Un graphe dirigé D est un* revêtement dirigé *d'un graphe dirigé D' à travers un homomorphisme $\gamma\colon D \to D'$ si pour tout arc $a' \in A(D')$ et pour tout sommet $v \in \gamma^{-1}(t(a'))$ (resp. $v \in \gamma^{-1}(s(a'))$, il existe un unique arc $a \in A(D)$ tel que $t(a) = v$ (resp. $s(a) = v$) et $\varphi(a) = a'$.*

On dit alors que l'homomorphisme γ est un homomorphisme localement bijectif. *Un graphe dirigé D est un* revêtement dirigé propre *de D' si γ n'est pas un isomorphisme.*

Naturellement, un graphe dirigé étiqueté (D, λ) est un revêtement dirigé d'un graphe dirigé étiqueté (D', λ') à travers γ si D est un revêtement dirigé de D' à travers γ et si γ conserve l'étiquetage.

D'après les définitions précédentes, on remarque que si un graphe dirigé $\mathbf{D}$ est un revêtement dirigé d'un graphe dirigé $\mathbf{D'}$, alors $\mathbf{D}$ est fibré sur $\mathbf{D'}$. On définit maintenant les graphes minimaux pour les revêtements dirigés.

Définition 7.7 *Un graphe dirigé fortement connexe $\mathbf{D}$ est* minimal pour les revêtements dirigés *si $\mathbf{D}$ n'est un revêtement dirigé propre d'aucun autre graphe dirigé fortement connexe.*

Un graphe simple $\mathbf{G}$ est minimal pour les revêtements dirigés *si $Dir(\mathbf{G})$ est minimal pour les revêtements dirigés.*

On montre dans la proposition suivante que si D est un revêtement dirigé de D', alors on peut relever un étiquetage déterministe de D' en un étiquetage déterministe de D.

Proposition 7.8 *Si un graphe dirigé D est un revêtement dirigé d'un graphe dirigé D' à travers γ, pour tout étiquetage déterministe λ' de D', il existe un étiquetage déterministe λ de D tel que (D, λ) est un revêtement dirigé de (D', λ') à travers γ.*

Preuve : On considère un graphe dirigé D qui est un revêtement dirigé d'un graphe dirigé D' à travers γ et un étiquetage déterministe λ' de D'. On définit un étiquetage λ de D de la manière suivante : pour tout $x \in V(D) \cup A(D)$, $\lambda(x) = \lambda'(\gamma(x))$. Puisque γ est un revêtement qui préserve l'étiquetage, (D, λ) est un revêtement de (D', λ') à travers γ. De plus, puisque γ est un revêtement dirigé, pour tout arcs distincts a, a' tels que $s(a) = s(a')$,

on a $\gamma(a) \neq \gamma(a')$. Puisque $s(\gamma(a)) = s(\gamma(a'))$ et que λ' est un étiquetage déterministe, $\lambda(a) \neq \lambda(a')$. Ainsi, λ est aussi un étiquetage codéterministe. □

Si D est un revêtement dirigé de D', on sait d'après les Propositions 7.5 et 7.8, qu'on peut relever tout étiquetage déterministe et codéterministe de D' en un étiquetage déterministe et codéterministe de D.

Dans la proposition suivante, on montre que si un graphe $\mathbf{D}$ a un étiquetage déterministe, alors $\mathbf{D}$ est fibré sur un graphe $\mathbf{D'}$ si et seulement si $\mathbf{D}$ est un revêtement de $\mathbf{D'}$.

Proposition 7.9 *On considère un graphe dirigé $\mathbf{D} = (D, \lambda)$ est fibré sur un graphe dirigé fortement connexe $\mathbf{D'} = (D', \lambda')$ à travers γ. Si λ est un étiquetage déterministe, alors $\mathbf{D}$ est un revêtement de $\mathbf{D'}$ à travers γ.*

Preuve : On considère un graphe dirigé (D, λ) dont l'étiquetage est déterministe qui est fibré sur un graphe dirigé (D', λ') à travers γ.

Pour tout arc $a' \in A(D')$, pour tout sommet $u \in \gamma^{-1}(t(a'))$, il existe un arc $a \in A(D)$ tel que $\gamma(a) = a'$ et $t(a) = u$. De plus, puisque l'étiquetage des arcs de D est déterministe, pour tous arcs distincts $a_1, a_2 \in \gamma^{-1}(a')$, $\lambda(a_1) = \lambda(a_2)$ et par conséquent, $s(a_1) \neq s(a_2)$. Ainsi, pour tout arc $a' \in A(D)$, on peut associer à tout sommet $u \in \gamma^{-1}(t(a'))$, un unique sommet $v \in \gamma^{-1}(s(a'))$ et par conséquent, $|\gamma^{-1}(t(a'))| \leq |\gamma^{-1}(s(a'))|$. Puisque D est un graphe fortement connexe, on sait donc que pour tout $u', v' \in V(D')$, $|\gamma^{-1}(u')| = |\gamma^{-1}(v')|$.

Pour tous arcs distincts $a_1, a_2 \in A(D)$ tels que $s(a_1) = s(a_2)$, $\lambda(a_1) \neq \lambda(a_2)$ et par conséquent, $\gamma(a_1) \neq \gamma(a_2)$. Ainsi, pour tout arc $a' \in A(D')$, on peut associer à tout arc $a \in \gamma^{-1}(a')$, un unique sommet $u \in \gamma^{-1}(s(a'))$. Par ailleurs, puisque γ est une fibration, pour tout arc $a' \in A(D')$, $|\gamma^{-1}(a)| = |\gamma^{-1}(t(a))|$ et ainsi, $|\gamma^{-1}(a)| = |\gamma^{-1}(s(a))|$. Par conséquent, pour tout sommet $u \in \gamma^{-1}(s(a'))$, il existe un unique arc $a \in \gamma^{-1}(a')$ tel que $\gamma(a) = a'$.

Puisque γ est une fibration, on a donc montré que (D, λ) était un revêtement dirigé de (D', λ') à travers γ. □

Les revêtements dirigés symétriques sont les revêtements dirigés qui préservent la fonction Sym sur les arcs.

Définition 7.10 *Un graphe dirigé symétrique D est un* revêtement dirigé symétrique *d'un graphe dirigé symétrique D' à travers un homomorphisme $\gamma\colon D \to D'$ si D est un revêtement dirigé de D' à travers γ et si γ préserve Sym, i.e., pour tout arc $a \in A(D)$, $Sym(\gamma(a)) = \gamma(Sym(a))$.*

Un graphe dirigé D est un revêtement dirigé symétrique propre *de D' si γ n'est pas un isomorphisme.*

Naturellement, un graphe dirigé symétrique étiqueté (D, λ) est un revêtement dirigé symétrique d'un graphe dirigé symétrique étiqueté (D', λ') à travers γ si D est un revêtement dirigé symétrique de D' à travers γ et si γ conserve l'étiquetage.

D'après les définitions précédentes, on remarque que si un graphe dirigé $\mathbf{D}$ est un revêtement dirigé symétrique d'un graphe dirigé $\mathbf{D'}$, alors $\mathbf{D}$ est un revêtement dirigé de $\mathbf{D'}$.

On définit maintenant les graphes minimaux pour les revêtements dirigés symétriques et puisque pour tout graphe simple $\mathbf{G}$, $Dir(\mathbf{G})$ est un graphe dirigé symétrique, on peut étendre cette définition aux graphes simples comme précédemment.

Définition 7.11 *Un graphe dirigé symétrique fortement connexe* $\mathbf{D}$ *est* minimal pour les revêtements dirigés symétriques *si* $\mathbf{D}$ *n'est un revêtement dirigé symétrique propre d'aucun autre graphe dirigé symétrique fortement connexe.*

Un graphe simple $\mathbf{G}$ *est* minimal pour les revêtements dirigés symétriques *si* $Dir(\mathbf{G})$ *est minimal pour les revêtements dirigés symétriques.*

On montre dans la proposition suivante que si D est un revêtement dirigé symétrique de D', alors on peut relever un étiquetage déterministe, codéterministe et symétrique de D' en un étiquetage déterministe, codéterministe et symétrique de D.

Proposition 7.12 *Si un graphe dirigé symétrique* D *est un revêtement dirigé symétrique d'un graphe dirigé symétrique* D' *à travers* γ*, pour tout étiquetage déterministe, codéterministe et symétrique* λ' *de* D'*, il existe un étiquetage déterministe, codéterministe et symétrique* λ *de* D *tel que* (D, λ) *est un revêtement dirigé symétrique de* (D', λ') *à travers* γ*.*

Preuve : On considère un graphe dirigé symétrique D qui est un revêtement dirigé symétrique d'un graphe dirigé symétrique D' à travers γ et un étiquetage déterministe, codéterministe et symétrique λ' de D'. On définit un étiquetage λ de D de la manière suivante : pour tout $x \in V(D) \cup A(D)$, $\lambda(x) = \lambda'(\gamma(x))$. Puisque γ est un revêtement dirigé symétrique qui préserve l'étiquetage, (D, λ) est un revêtement de (D', λ') à travers γ. De plus, puisque γ est un revêtement dirigé, pour tout arcs a, a' tels que $s(a) = s(a')$ (resp. $t(a) = t(a')$), $\gamma(a) \neq \gamma(a')$ et $s(\gamma(a)) = s(\gamma(a'))$ (resp. $t(\gamma(a)) = t(\gamma(a'))$). Puisque λ' est un étiquetage déterministe et codéterministe, λ est aussi un étiquetage déterministe et codéterministe. Par ailleurs, puisque γ est un revêtement dirigé symétrique, pour tout arc $a \in A(D)$, $Sym(\lambda(a)) = Sym(\lambda'(\gamma(a))) = \lambda'(Sym(\gamma(a))) = \lambda'(\gamma(Sym(a))) = \lambda(Sym(a))$ et ainsi, λ est un étiquetage symétrique de D. □

Dans la proposition suivante, on montre le lien entre les revêtements de graphes non-dirigés et et les revêtements dirigés symétriques.

Proposition 7.13 *Si un graphe* $\mathbf{G}$ *est un revêtement d'un graphe* $\mathbf{H}$*, alors* $Dir(\mathbf{G})$ *est un revêtement dirigé symétrique d'un graphe* $Dir(\mathbf{H})$*.*

Preuve : On considère deux graphes $\mathbf{G} = (G, \lambda), \mathbf{H} = (H, \eta)$. On sait que $Dir(\mathbf{G})$ et $Dir(\mathbf{H})$ sont deux graphes dirigés fortement connexes symétriques. De plus, pour toute arête $e \in E(G)$ dont les extrémités sont u et u', il existe deux arcs $a_{e,u,u'}, a_{e,u',u} \in A(Dir(\mathbf{G}))$ tels que $\lambda(a_{e,u,u'}) = \lambda(a_{e,u',u}) = \lambda(e)$, $s(a_{e,u,u'}) = t(a_{e,u',u}) = u$ et tels que $Sym(a_{e,u,u'}) = a_{e,u',u}$. De même, pour toute arête $f \in E(H)$ dont les extrémités sont v et v', il existe deux arcs $a_{f,v,v'}, a_{f,v',v} \in A(Dir(\mathbf{H}))$ tels que $\eta(a_{f,v,v'}) = \eta(a_{f,v',v}) = \eta(f)$, $s(a_{f,v,v'}) = t(a_{f,v',v}) = v$ et $Sym(a_{f,v,v'}) = a_{f,v',v}$.

On suppose que $\mathbf{G}$ est un revêtement de $\mathbf{H}$ à travers un homomorphisme γ. On définit un homomorphisme φ de $Dir(\mathbf{G})$ dans $Dir(\mathbf{H})$ de la manière suivante : pour tout $u \in V(Dir(G)) = V(G)$, $\varphi(u) = \gamma(u)$ et pour tout arc $a_{e,u,u'} \in A(Dir(G))$, $\varphi(a_{e,u,u'}) = a_{\gamma(e),\gamma(u),\gamma(u')}$. Pour tout sommet $u \in V(Dir(G))$, $\eta(\varphi(u)) = \eta(\gamma(u)) = \lambda(u)$ et pour tout arc $a_{e,u,u'} \in A(Dir(G))$, $\eta(\varphi(a_{e,u,u'})) = \eta(\gamma(e)) = \lambda(e) = \lambda(a_{e,u,u'})$. Ainsi, φ préserve l'étiquetage. De plus, pour tout $a_{e,u,u'} \in A(Dir(G))$, $s(\varphi(a_{e,u,u'})) = s(a_{\gamma(e),\gamma(u),\gamma(u')}) = \gamma(u) = \varphi(s(a_{e,u,u'}))$ et $Sym(\varphi(a_{e,u,u'})) = Sym(a_{\gamma(e),\gamma(u),\gamma(u')}) = a_{\gamma(e),\gamma(u'),\gamma(u)} = \varphi(a_{e,u',u'}) = \varphi(Sym(a_{e,u,u'}))$. Ainsi, φ est un homomorphisme de $Dir(\mathbf{G})$ dans $Dir(\mathbf{H})$ qui préserve la fonction Sym. Par ailleurs, étant donnés un sommet $u \in V(Dir(G))$ et

un arc $a_{f,\gamma(u),v'} \in A(Dir(H))$ (resp. $a_{f,v,\gamma(u)} \in A(Dir(H))$), il existe $e \in E(G)$ tel que $\gamma(e) = f$ et $u \in ext(e)$: ainsi, il existe $a_{e,u,u'} \in A(G)$ (resp. $a_{e,u',u} \in A(G)$) tel que $\varphi(a_{e,u,u'}) = a_{f,\gamma(u),v'}$ et $s(a_{e,u,u'}) = u$ (resp. $\varphi(a_{e,u',u}) = a_{f,v,\gamma(u)}$ et $t(a_{e,u',u}) = u$). Ainsi $Dir(\mathbf{G})$ est un revêtement de $\mathbf{H}$ à travers φ. □

Comme pour les revêtements de graphes non-dirigés, si un graphe dirigé $\mathbf{D}$ est un revêtement d'un graphe dirigé $\mathbf{D}'$ fortement connexe, alors tous les sommets et les arcs de $\mathbf{D}'$ ont le même nombre d'antécédents, qui est appelé le nombre de feuillets du revêtement.

Proposition 7.14 *Si un graphe dirigé $\mathbf{D}$ est un revêtement dirigé d'un graphe connexe $\mathbf{D}'$ à travers γ, alors il existe une constante q telle que pour tout $x \in V(D') \cup A(D')$, $|\gamma^{-1}(x)| = q$.*

Cette constante q est appelée le nombre de feuillets *du revêtement.*

Preuve : On considère un graphe dirigé $\mathbf{D}$ qui est un revêtement d'un graphe dirige fortement connexe $\mathbf{D}'$ à travers un homomorphisme γ. On considère un sommet $v' \in V(D')$ et un arc $a' \in A(D')$ tel que $s(a') = v'$.

Puisque γ est un homomorphisme, pour tout arc $a \in \gamma^{-1}(a')$, $\gamma(s(a)) = s(\gamma(a)) = v'$. De plus, puisque γ est localement bijectif, on sait que pour tout arc a'' différent de a tel que $s(a'') = s(a)$, $\gamma(a'') \neq a'$. Par conséquent, $|\gamma^{-1}(a')| \leq |\gamma^{-1}(v')|$.

Réciproquement, pour tout sommet $v \in \gamma^{-1}(v')$, il existe un unique arc $a \in \gamma^{-1}(a')$ tel que $s(a) = v$. Par conséquent $|\gamma^{-1}(a')| \geq |\gamma^{-1}(v')|$.

De la même manière, on montre que pour tout sommet $v' \in V(D')$ et tout arc $a' \in A(D')$ tel que $t(a') = v'$, $|\gamma^{-1}(a')| = |\gamma^{-1}(v')|$. Ainsi, puisque le graphe $\mathbf{D}'$ est fortement connexe, pour tout $x, x' \in V(D') \cup A(D')$, $|\gamma^{-1}(x)| = |\gamma^{-1}(x')|$. □

Grâce à la Proposition 7.14, on voit que si un graphe dirigé $\mathbf{D}$ est un revêtement propre de $\mathbf{D}'$, alors $\mathbf{D}$ est fibré non-trivialement sur $\mathbf{D}'$.

Exemple 7.15 *Des exemples de fibrations et de revêtements sont présentés sur la Figure 26. Le graphe dirigé $\mathbf{D}_1$ est minimal pour les fibrations non-triviales, mais est fibré proprement sur $\mathbf{D}'_1$. Le graphe dirigé $\mathbf{D}_2$ est minimal pour les revêtements dirigés, mais est fibré non-trivialement sur $\mathbf{D}'_2$. Le graphe dirigé D_3 est minimal pour les revêtements symétriques dirigés, mais est un revêtement dirigé propre de D'_3.*

Étant donné un graphe dirigé $\mathbf{D}$, il existe un graphe dirigé $\mathbf{D}_0$ canonique sur lequel $\mathbf{D}$ est fibré. Ce graphe dirigé $\mathbf{D}_0$ est appelé la *base minimale* de $\mathbf{D}$; c'est le «plus petit» graphe dirigé sur lequel $\mathbf{D}$ est fibré. La proposition suivante est due à Boldi et Vigna [BV02a].

Proposition 7.16 ([BV02a]) *Pour tout graphe dirigé fortement connexe $\mathbf{D}$, il existe un graphe dirigé fortement connexe $\mathbf{D}_0$ tel que $\mathbf{D}$ est fibré sur $\mathbf{D}_0$ et pour tout graphe dirigé fortement connexe $\mathbf{D}'$ tel que $\mathbf{D}$ est fibré sur $\mathbf{D}$, $\mathbf{D}'$ est fibré sur $\mathbf{D}_0$.*

Le graphe $\mathbf{D}_0$ est la base minimale *de $\mathbf{D}$.*

La base minimale d'un graphe dirigé $\mathbf{D}$ peut être calculé en temps polynomial en utilisant la méthode du raffinement de degré [BV02a, Lei82]. Cette méthode est similaire à la technique classique utilisée pour minimiser un automate déterministe qu'on peut trouver par exemple dans [HU79].

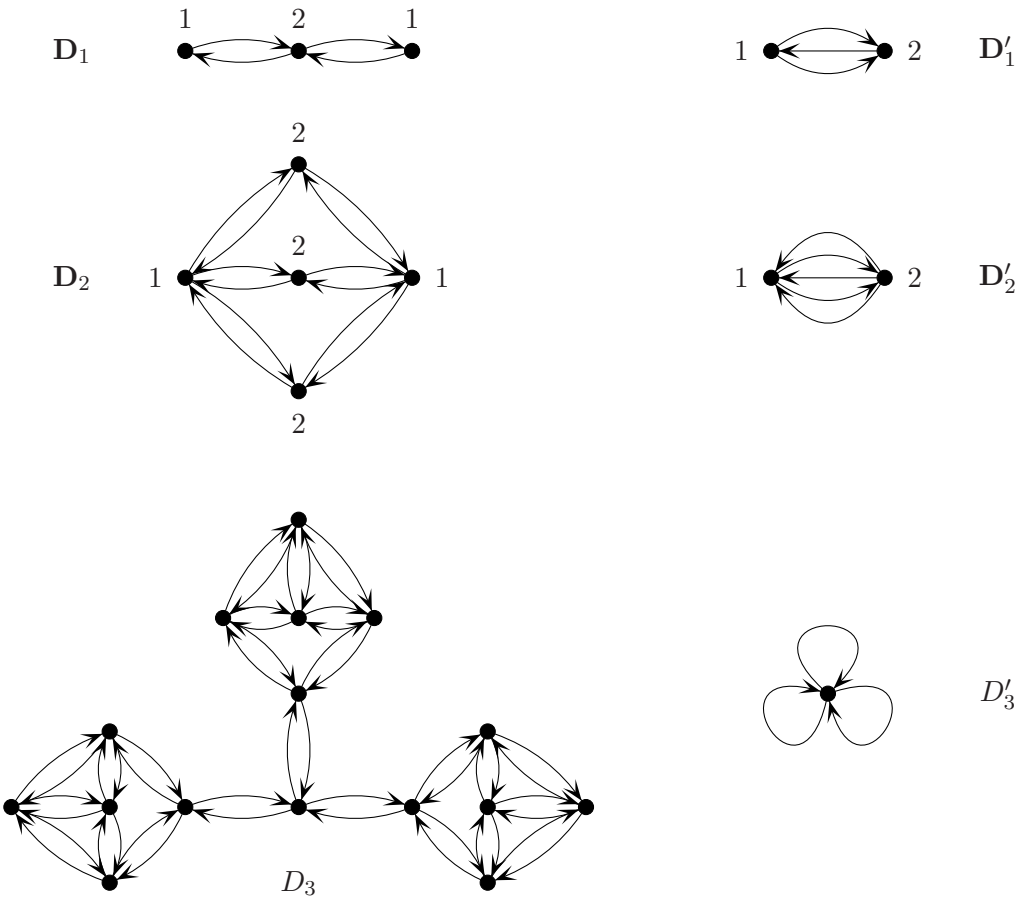


FIG. 26 – Le graphe dirigé $\mathbf{D}_1$ est fibré proprement sur $\mathbf{D}'_1$, le graphe $\mathbf{D}_2$ est fibré non-trivialement sur $\mathbf{D}'_2$ et le graphe $\mathbf{D}_3$ est un revêtement dirigé propre de $\mathbf{D}'_3$.

7.3 Le Modèle

Les définitions présentées dans cette section correspondent aux définitions données par Tel dans [Tel00] (pp. 45-47). On peut aussi se référer au livre d'Attiya et Welch [AW04] (pp. 10-12).

Étant donné un graphe simple G, un étiquetage des ports est un ensemble de fonctions locales qui permettent à chaque sommet de distinguer ses voisins.

Définition 7.17 *Étant donné un graphe simple G, un étiquetage des ports δ est un ensemble de fonctions $\{\delta_u \mid u \in V(G)\}$ tel que pour tout sommet u, δ_u est une bijection entre $N_G(u)$ et $[1, \deg_G(u)]$.*

On représente un graphe G avec un étiquetage δ en représentant le numéro $\delta_u(v)$ sur l'incidence entre le sommet u et l'arête $\{u, v\}$ comme représenté sur la Figure 27.

7.3.1 Systèmes Distribués avec Communication par Échange de Messages

Un système distribué avec communication par échange de message (P, C) est défini par un ensemble P de processus et un sous-système de communication C qui est représenté par un graphe simple connexe G où les sommets représentent les processus et les arêtes représentent des liens de communications bidirectionnels. Le système est asynchrone : il n'y a pas d'horloge globale, les messages parviennent à leurs destinataires en un temps fini mais arbitraire et les processus exécutent les instructions à des vitesses arbitraires. Les processus communiquent en échangeant des messages en mode asynchrone et chaque processus distingue les canaux par lesquels il reçoit ou envoie chaque message, i.e., il existe une fonction δ d'étiquetage des ports du graphe G. Ainsi, le système de communication C est représenté par un graphe simple G avec un étiquetage des ports δ. Chaque processus a un état initial défini par une fonction d'étiquetage des sommets λ. Le système distribué (P, C) est alors représenté par un graphe étiqueté $\mathbf{G} = (G, \lambda)$ avec un étiquetage des ports δ. On appellera parfois ce système distribué un *réseau* on le notera $(\mathbf{G}, \delta)$.

Remarque 7.18 *Si le réseau est anonyme, tous les sommets doivent avoir le même état initial, et l'étiquetage λ est alors uniforme (tous les sommets ont la même étiquette). Si les processus du réseau ont tous des identités distinctes, l'étiquetage λ est tel que pour tout $u, v \in V(G)$, $\lambda(u) \neq \lambda(v)$. S'il existe un sommet distingué dans le réseau, alors il existe un sommet $v \in V(G)$ tel que pour tout $u \in V(G)$ différent de v, $\lambda(u) \neq \lambda(v)$.*

7.3.2 Algorithmes utilisant des Messages

Un système de transition est associé à chaque processus, et ce système de transition interagit avec le système de communication. Les différents types de transitions (ou d'événements) associés à chaque processus sont des transitions *internes*, des transitions d'*envoi* et des transitions de *réception*. Si une transition d'envoi (resp. de réception) est effectuée, un message est créé (resp. consommé).

Étant donné un processus p représenté par le sommet v, l'algorithme local du processus p, dénoté $\mathcal{D}_p$ est défini par

- l'ensemble récursif Q des états possibles de p,
- l'état initial $\lambda(v)$ de p,

– une relation récursive $\vdash_p$ de transitions (transitions internes, transitions d'envoi et transitions de réception).

Pour toute arête $\{v, v'\} \in E(G)$, on note $\mathtt{M}(v, v')$ (resp. $\mathtt{M}(v', v)$) le multi-ensemble des messages en transit entre le processus p (resp. p') correspondant au sommet v (resp. v') et le processus p' (resp. p) correspondant au sommet v' (resp. v). Initialement, tous les ensembles $\mathtt{M}(v, v')$ sont vides.

Pour tout processus p représenté par un sommet v, l'état de p est noté $\mathtt{state}(p)$ et une transition associée au processus p est notée sous la forme

$$(c, in, m) \vdash_p (d, out, m'),$$

où c et d sont des états, in et out sont des entiers, et m et m' sont des messages. Une telle transition a la signification suivante.

– Si $in = out = 0$, alors $m = m' = \perp$ (m et m' ne sont pas définis) et cela représente une transition interne. Cette transition ne peut être effectuée que si l'état de p est c. Lorsque cette transition est effectuée, l'état de p devient d.
– Si $in \neq 0$, alors $out = 0$ et $m' = \perp$ (m' n'est pas défini) et cela représente une transition de réception. Cette transition ne peut être appliquée que si l'état de p est c et qu'il existe un message $m \in \mathtt{M}(v', v)$ tel que $\delta_v(v') = in$. Lorsque cette transition est effectuée, l'état du processus devient d et une occurrence de m est effacée de $\mathtt{M}(v', v)$, où v' est le sommet tel que $\delta_v(v') = in$.
– Si $out \neq 0$, alors $in = 0$ et $m = \perp$ (m n'est pas défini) et cela représente une transition d'envoi. Cette transition ne peut être effectuée que si l'état du processus p est c. Lorsque cette transition est effectuée, l'état de p devient d et un message m' est ajouté à $\mathtt{M}(v, v')$, où v' est le sommet tel que $\delta_v(v') = out$.

Un algorithme distribué utilisant des échanges de messages $\mathcal{D}$ pour le réseau $(\mathbf{G}, \delta)$ est une collection d'algorithmes locaux $\mathcal{D}_p$ (un pour chaque processus p associé à chaque sommet de $\mathbf{G}$). Un tel algorithme est noté $\mathcal{D} = (\mathcal{D}_p)_{p \in P}$. Une *transition* de l'algorithme distribué est une transition effectuée par un processus du système.

Remarque 7.19 *On considère généralement des algorithmes distribués tels que deux processus p, p' dans le même état puissent effectuer les mêmes transitions. Cependant, si p peut communiquer avec plus de processus que p', p' ne pourra pas envoyer ou recevoir de messages par le port* $\deg(v)$ *où v est le sommet correspondant à p.*

Ainsi, pour tous processus p, p' correspondant respectivement à des sommets v, v' de même degré, $\vdash_p = \vdash_{p'}$. Si on ne veut pas que deux processus correspondant à des sommets de même degré exécutent les mêmes transitions, on utilise l'étiquetage initial des sommets pour que les processus aient des comportements différents.

De cette manière, un algorithme n'a pas besoin d'être défini pour un réseau particulier, mais il suffit de décrire pour tout entier d, les transitions que peuvent effectuer les processus de degré d.

Par la suite, sauf si c'est explicitement indiqué, on ne considère que des algorithmes respectant cette propriété.

Remarque 7.20 *Si on considère des systèmes distribués dont les canaux de communication préservent l'ordre des messages (canaux FIFO), il faut voir chaque ensemble $\mathtt{M}(v, v')$ comme une file.*

Lorsqu'une transition d'envoi $(c, in, m) \vdash_p (d, out, m')$ *est effectuée par un processus* p *correspondant à un sommet* v*, une occurrence de* m' *est ajouté en queue de la file correspondant à* $\mathtt{M}(v, v')$ *où* v' *est le sommet tel que* $\delta_v(v') = out$*. Une transition de réception* $(c, in, m) \vdash_p (d, out, m')$ *peut être effectuée si* m *est le message en tête de la file* $\mathtt{M}(v', v)$ *où* v' *est le sommet tel que* $\delta_v(v') = in$ *; auquel cas, la tête de la file* $\mathtt{M}(v', v)$ *est supprimée.*

Dans ce chapitre, sauf si c'est explicitement indiqué, on ne considère que des systèmes distribués dont les canaux de communication préservent l'ordre des messages.

Exécution d'un Algorithme utilisant des Échanges de Messages

Une exécution ρ d'un algorithme utilisant des messages est défini par une suite $(\mathtt{state}_0, \mathtt{M}_0), (\mathtt{state}_1, \mathtt{M}_1), \ldots, (\mathtt{state}_i, \mathtt{M}_i), \ldots$ telle que :

- pour tout i, $\mathtt{M}_i$ est l'ensemble des ensembles $\mathtt{M}_i(v, v')$ de messages en transit entre les sommets v et v',
- pour toute $\{v, v'\} \in E(G)$, $\mathtt{M}_0(v, v') = \mathtt{M}_0(v', v) = \emptyset$,
- pour tout i et tout processus p, $\mathtt{state}_i(p)$ est l'état du processus p,
- pour tout processus p, $\mathtt{state}_0(p) = \lambda(p)$ est l'état initial de p,
- pour toute étape i, il existe un unique processus p tel que :
  - si $p' \neq p$, alors $\mathtt{state}_{i+1}(p') = \mathtt{state}_i(p')$,
  - $\mathtt{state}_{i+1}(p)$ et $\mathtt{M}_{i+1}$ sont obtenus à partir de $\mathtt{state}_i(p)$ et $\mathtt{M}_i$ par une transition effectuée par le processus p.

Par définition, $(\mathtt{state}_i, \mathtt{M}_i)$ est une *configuration* du réseau. L'exécution ρ de l'algorithme est définie par $\mathcal{E} = (\mathtt{state}_i, \mathtt{M}_i)_{i \geq 0}$.

Une configuration *finale* est une configuration dans laquelle aucune transition n'est applicable. On suppose que dans une configuration finale, aucun message n'est en transit. Autrement dit, pour tout processus p correspondant à un sommet v, pour tout état c, s'il n'existe aucune transition de la forme $(c, 0, \perp) \vdash_p (d, out, m)$ (i.e., lorsque p est dans l'état c, il ne peut ni envoyer de message, ni effectuer de transition interne), alors pour tout message m et pour tout entier $in \in [1, \deg(v)]$, il existe une transition $(c, in, m) \vdash_p (d, 0, \perp)$.

Étant donnée une exécution qui atteint une configuration finale, la *longueur* de l'exécution est la longueur de la séquence $(\mathtt{state}_0, \mathtt{M}_0), (\mathtt{state}_1, \mathtt{M}_1), \ldots, (\mathtt{state}_i, \mathtt{M}_i), \ldots$

Remarque 7.21 *Comme Tel l'explique ([Tel00] p. 46), les systèmes de transition sont des modèles théoriques de systèmes distribués. Cependant, les algorithmes qu'on va étudier ne seront pas décrits par l'énumération de leurs états et de leurs transitions, mais en utilisant le pseudo-code habituel. La mise à jour de l'état d'un sommet se fera à l'aide de variables et les transitions d'envois et de réception de messages seront exprimées respectivement à l'aide des primitives* **envoyer** *et* **recevoir**.

7.4 Codage du Réseau

On explique dans cette section comment représenter un réseau par un graphe dirigé symétrique afin de pouvoir exprimer simplement les conditions nécessaires que doit vérifier un réseau admettant un algorithme d'élection. On présente ensuite un codage du réseau sous la forme d'un graphe simple qui permet d'obtenir des résultats d'impossibilité en utilisant les résultats du Chapitre 3.

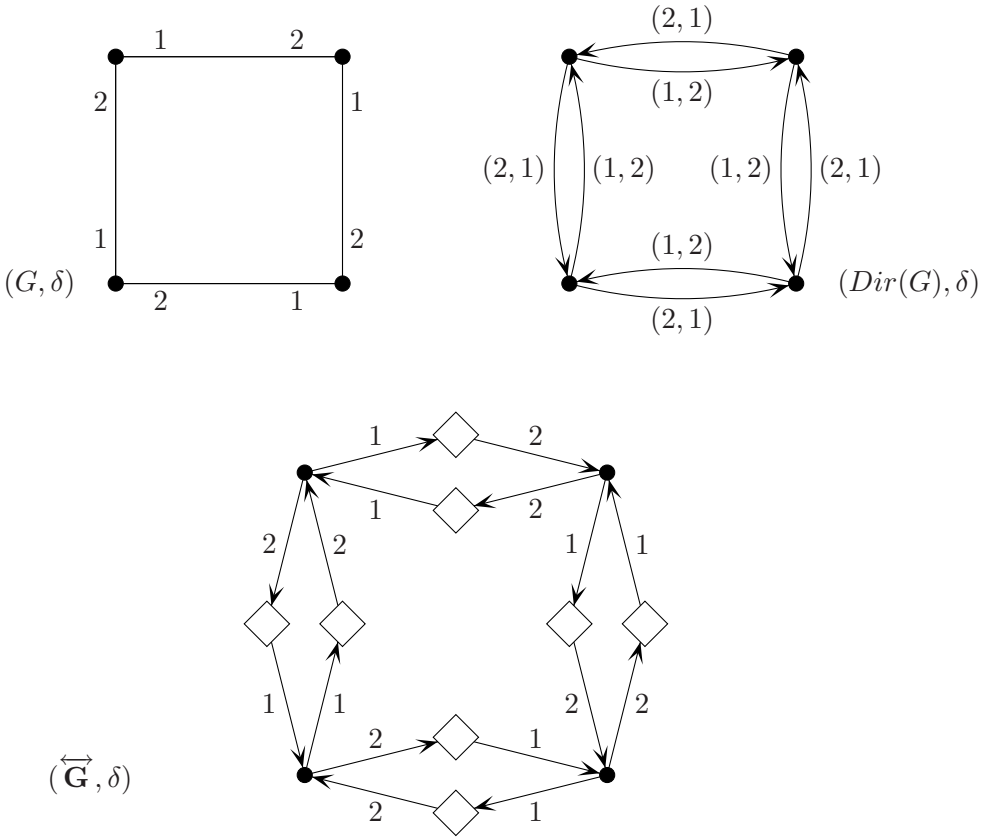


FIG. 27 – Un graphe G avec un étiquetage de ports δ, sa représentation sous forme de graphe dirigé et sa représentation sous forme de graphe simple étiqueté.

7.4.1 Étiquetage des Ports et Graphes Dirigés Symétriques

Un réseau est représenté par un graphe simple étiqueté $(\mathbf{G}, \delta)$ où $\mathbf{G} = (G, \lambda)$ est un graphe simple dont seuls les sommets sont étiquetés et où δ est une fonction d'étiquetage des ports.

À chaque réseau $(\mathbf{G}, \delta)$, on associe le graphe dirigé symétrique $(Dir(\mathbf{G}), \delta)$ où tout sommet $u \in V(Dir(\mathbf{G}))$ est étiqueté comme dans $\mathbf{G}$ et où tout arc $a_{u,u'} \in A(Dir(G))$ tel que $s(a) = u$ et $t(a) = u'$ est étiqueté par $(\delta_u(u'), \delta_{u'}(u))$. On remarque que pour tout arc $a_{u,u'} \in A(Dir(G))$ étiqueté (p, q), l'arc symétrique $Sym(a_{u,u'})$ est étiqueté (q, p) : l'étiquetage des arcs du graphe dirigé obtenu est symétrique. Par ailleurs, puisque δ est une fonction d'étiquetage des ports, on remarque que l'étiquetage des arcs ainsi obtenu est déterministe et codéterministe. Une telle représentation d'un réseau (G, δ) est présentée sur la Figure 27.

7.4.2 Codage du Réseau avec un Graphe Simple Étiqueté

On considère un graphe dirigé étiqueté $\mathbf{D} = (D, \lambda)$ où seuls les sommets sont étiquetés. On va maintenant associer un graphe simple étiqueté à $\mathbf{D}$.

À chaque arc a dont on note la source u et la cible v, on associe un sommet canal(a)

et deux arêtes $\{u, \text{canal}(a)\}$, $\{\text{canal}(a), v\}$. On considère maintenant le graphe simple non-étiqueté $\overleftrightarrow{D}$ définit de la manière suivante :

- $V(\overleftrightarrow{D}) = V(D) \cup \bigcup_{a \in A(D)} \{\text{canal}(a)\}$
- $E(\overleftrightarrow{D}) = \bigcup_{a \in A(D)} \{\{s(a), \text{canal}(a)\}, \{\text{canal}(a), t(a)\}\}$.

On note $\overleftrightarrow{\mathbf{D}}$ le graphe étiqueté $(\overleftrightarrow{D}, (\kappa, \lambda))$ où λ et κ sont définies de la manière suivante. Pour tout $x \in V(\overleftrightarrow{D}) \cup E(\overleftrightarrow{D})$, $\lambda(x)$ est l'étiquette de x dans $\mathbf{D}$ si $x \in V(D)$ et $\lambda(x) = \epsilon$ sinon. La fonction d'étiquetage κ code la fonction de chaque sommet ou arête de $\overleftrightarrow{D}$ et est définie de la manière suivante :

- $\forall u \in V(D)$, $\kappa(u) =$ **process**,
- $\forall a \in A(D)$, $\kappa(\text{canal}(a)) =$ **transmission**,
- $\forall a \in A(D)$, $\kappa(\{s(a), \text{canal}(a)\}) =$ **send**,
- $\forall a \in A(D)$, $\kappa(\{tran(a), s(a)\}) =$ **receive**.

On suppose maintenant qu'il existe une fonction δ d'étiquetage des arcs de $\mathbf{D}$ telle que pour tout arc $a \in A(D)$, il existe p, q tels que $\delta(a) = (p, q)$. On étend δ en une fonction d'étiquetage partielle des arêtes de $\overleftrightarrow{D}$ de la manière suivante. Pour tout arc $a \in A(D)$ étiqueté (p, q) dont la source est u et la cible est v, on pose $\delta(\{u, \text{canal}(a)\}) = p$ et $\delta(\{\text{canal}(a), v\}) = q$.

On suppose maintenant que $\mathbf{D}$ est un graphe dirigé symétrique $\mathbf{D}$ et on considère une fonction δ d'étiquetage des arcs symétrique telle que pour tout arc $a \in A(D)$, il existe p, q tels que $\delta(a) = (p, q)$ et $\delta(Sym(a)) = (q, p)$. Dans ce cas, on remarque que pour tout arc a dont la source est u et la cible est v, $\delta(\{u, \text{canal}(a)\}) = \delta(\{\text{canal}(Sym(a)), u\})$ et $\delta(\{\text{canal}(a), v\}) = \delta(\{v, \text{canal}(Sym(a))\})$.

Étant donné un réseau $(\mathbf{G}, \delta)$ où $\mathbf{G}$ est un graphe simple et δ est une fonction d'étiquetage des ports, on note $\overleftrightarrow{\mathbf{G}}$ le graphe simple obtenu lorsque la construction décrite précédemment est appliquée sur le graphe dirigé $(Dir(G), \delta)$.

Remarque 7.22 *Sur les figures, on représente les sommets avec des formes particulières dépendant de leurs images par κ :*

- *chaque sommet u tel que $\kappa(u) =$* **process** *est représenté par un sommet rond,*
- *chaque sommet u tel que $\kappa(u) =$* canal *est représenté par un sommet carré,*

De plus, pour tout arête $\{u, v\} \in E(\overleftrightarrow{\mathbf{D}})$ telle que $\kappa(u) =$ **process** *et $\kappa(v) =$* canal*, on représente l'arête $\{u, v\}$ par un arc allant de u à v si $\kappa(\{u, v\}) =$* **send** *et par un arc allant de v à u si $\kappa(\{u, v\}) =$* **receive***.*

Sur les Figures 27 et 28, des exemples de graphes obtenus par cette construction sont présentés.

7.4.3 Coder un Algorithme utilisant des échanges de Messages en Mode Asynchrone

On explique dans cette partie comment tout algorithme utilisant des échanges de messages en mode asynchrone sur un réseau $(\mathbf{G}, \delta)$ peut être transformer en un algorithme utilisant des calculs locaux sur les arêtes étiquetées sur le graphe $(\overleftrightarrow{\mathbf{G}}, \delta)$.

On considère un réseau $(\mathbf{G}, \delta)$ et le graphe simple étiqueté $(\overleftrightarrow{\mathbf{G}}, \delta)$ obtenu à partir de $(\mathbf{G}, \delta)$. On explique maintenant comment coder les différentes instructions que peut

exécuter un processus de G ainsi que les événements de transmission de messages en utilisant des calculs locaux sur les arêtes. L'étiquette de chaque sommet v de $\overleftrightarrow{\mathbf{G}}$ est de la forme $(\kappa(v), \lambda(v))$ et chaque règle de réétiquetage ne peut modifier que $\lambda(v)$.

- Une instruction qui modifie l'état d'un processus est codé par une règle de réétiquetage pouvant être appliquée sur sommet v tel que $\kappa(v) = \mathbf{process}$. Cette règle ne modifie que l'étiquette d'un sommet v et ne dépend que de l'étiquette de ce sommet.
- Une instruction d'envoi de message est codé par une règle de réétiquetage sur une arête $\{u, v\}$ telle que $\kappa(u) = \mathbf{process}$, $\kappa(v) =$ canal et $\kappa(\{u, v\}) = \mathbf{send}$. Cette règle ne modifie pas l'étiquette de l'arête, mais seulement $\lambda(u)$ et $\lambda(v)$.
- Une instruction de réception de message est codé par une règle de réétiquetage sur une arête $\{u, v\}$ telle que $\kappa(u) = \mathbf{process}$, $\kappa(v) =$ canal et $\kappa(\{u, v\}) = \mathbf{receive}$. Cette règle ne modifie pas l'étiquette de l'arête, mais seulement $\lambda(u)$ et $\lambda(v)$.

La méthode qu'on a présenté permet de coder un algorithme distribué où les processus communiquent par échange de messages en mode asynchrone en utilisant des calculs locaux sur les arêtes. Cependant, il existe des algorithmes utilisant des calculs locaux sur les arêtes dont les exécutions sur le graphe $(\overleftrightarrow{\mathbf{G}}, \delta)$ ne correspondent pas forcément à des exécutions sur G d'un algorithme où les processus communiquent par échange de message. Néanmoins, cette construction nous permet d'obtenir un lemme de relèvement et des résultats d'impossibilité.

On montre d'abord que si un graphe dirigé $(\mathbf{D}, \delta)$ est un revêtement dirigé d'un graphe dirigé $(\mathbf{D'}, \delta')$, alors $(\overleftrightarrow{\mathbf{D}}, \delta)$ est un revêtement de $(\overleftrightarrow{\mathbf{D}'}, \delta')$.

Lemme 7.23 *Pour tous graphes dirigés $(\mathbf{D}, \delta)$, $(\mathbf{D'}, \delta')$ tels que $(\mathbf{D}, \delta)$ est un revêtement dirigé de $(\mathbf{D'}, \delta')$, le graphe simple $(\overleftrightarrow{\mathbf{D}}, \delta)$ est un revêtement de $(\overleftrightarrow{\mathbf{D}'}, \delta')$.*

Preuve : On considère un graphe dirigé $((D, \lambda), \delta)$ qui est un revêtement dirigé de $((D', \lambda'), \delta')$ à travers un homomorphisme γ. On définit un homomorphisme φ de $\overleftrightarrow{D}$ dans $\overleftrightarrow{D}'$ de la manière suivante. Pour tout sommet $u \in V(\overleftrightarrow{D})$ tel que $u \in V(D)$, alors $\varphi(u) = \gamma(u)$. Pour tout arc $a \in A(D)$, on pose $\varphi(\text{canal}(a)) = \text{canal}(\gamma(a))$. Il est clair que φ est un homomorphisme de $\overleftrightarrow{D}$ dans $\overleftrightarrow{D}'$.

De plus, pour tout sommet $u \in V(\overleftrightarrow{\mathbf{D}})$ tel que $u \in V(D)$, $\kappa(u) = \kappa(\gamma(u)) = \kappa(\varphi(u)) =$ **process** et $\lambda(u) = \lambda(\gamma(u)) = \lambda(\varphi(u))$. Puisque pour tout arc a, $\kappa(\text{canal}(a)) = \kappa(\varphi(\text{canal}(a))) =$ **transmission** et que $\lambda(\text{canal}(a)) = \lambda(\varphi(\text{canal}(a))) = \epsilon$, l'homomorphisme φ préserve l'étiquetage des sommets.

Par ailleurs, pour toute arête $\{u, v\} \in E(\overleftrightarrow{\mathbf{D}})$ telle que $\kappa(u) = \mathbf{process}$, $\kappa(v) =$ canal et $\kappa(\{u, v\}) = \mathbf{send}$ (resp. $\kappa(\{u, v\}) = \mathbf{send}$), il existe un arc $a \in A(D)$ tel que $s(a) = u$ (resp. $t(a) = u$) et $\text{canal}(a) = v$. Puisque $\gamma(u) = s(\gamma(a))$ (resp. $\gamma(u) = t(\gamma(a))$), $\kappa(\{\gamma(u), \text{canal}(\gamma(a))\}) = \mathbf{send}$ (resp. $\kappa(\{\gamma(u), \text{canal}(\gamma(a))\}) = \mathbf{receive}$) et $\kappa(\varphi(\{u, v\})) = \kappa(\{u, v\})$. De plus, pour tout arc $a \in A(D)$, il existe (p, q) tels que $\delta(a) = (p, q)$ et on sait que $\delta(\gamma(a)) = (p, q)$. Ainsi, $\delta(\{s(a), \text{canal}(a)\}) = \delta(\{\varphi(s(a)), \varphi(\text{canal}(a))\}) = p$ et $\delta(\{\text{canal}(a), t(a)\}) = \delta(\{\varphi(\text{canal}(a)), \varphi(t(a))\}) = q$. L'homomorphisme φ préserve donc l'étiquetage.

D'autre part, pour tout sommet $u \in V(D)$, pour tout arc $a \in A(D)$ tel que $s(a) = u$ (resp. $t(a) = u$), il existe exactement un arc $a' = \gamma(a) \in A(D')$ tel que $s(a') = \gamma(u)$ (resp. $t(a') = \gamma(u)$).

Ainsi pour tout sommet $u \in V(\overleftrightarrow{D})$ tel que $u \in V(D)$, $\varphi(N_{\overleftrightarrow{D}}(u)) = \varphi(\{\text{canal}(a) \mid$

Fig. 28 – Un graphe dirigé D avec un étiquetage symétrique des arcs et sa représentation sous forme de graphe simple étiqueté.

$s(a) = u$ ou $t(a) = u\}) = \text{canal}(\{\gamma(a) \mid s(a) = u$ ou $t(a) = u\}) = \text{canal}(\{a' \mid s(a') = \gamma(u)$ ou $t(a') = \gamma(u)\}) = N_{\overleftrightarrow{D}'}(\gamma(u))$ et puisque $|N_{\overleftrightarrow{D}'}(\gamma(u))| = |\{a' \in A(D') \mid s(a) = \gamma(u)$ ou $t(a) = \gamma(u)\}| = |\{a \in A(D) \mid s(a) = u$ou $t(a) = u\}| = |N_{\overleftrightarrow{D}}(u)|$, l'homomorphisme φ est bien localement bijectif.

Ainsi, $(\overleftrightarrow{\mathbf{D}}, \delta)$ est un revêtement de $(\overleftrightarrow{\mathbf{D}}', \delta')$ à travers φ. □

Un corollaire intéressant du Lemme 7.23 est le résultat d'impossibilité suivant qui est obtenu à partir de la Proposition 3.22.

Proposition 7.24 *Soit* $\mathbf{G}$ *un graphe simple et* δ *une fonction d'étiquetage des ports de* $\mathbf{G}$ *tel que* $(Dir(\mathbf{G}), \delta)$ *n'est pas minimal pour les revêtements dirigés symétriques. Il n'existe pas d'algorithme où les processus communiquent par échange de message qui permette de résoudre l'élection, l'énumération ou le nommage dans le réseau* $(\mathbf{G}, \delta)$.

Exemple 7.25 *Si on considère le graphe* G *de la Figure 27 avec l'étiquetage des ports* δ *indiqués sur la figure. Il est facile de voir que* $(Dir(G), \delta)$ *est un revêtement dirigé symétrique du graphe* (D, δ') *de la Figure 28. Ainsi, le graphe* $(\overleftrightarrow{\mathbf{G}}, \delta)$ *est un revêtement* $(\overleftrightarrow{\mathbf{D}}, \delta')$ *et il n'existe donc pas d'algorithme d'élection ou de nommage utilisant des échanges de messages pour le graphe* G *avec l'étiquetage des ports* δ.

7.5 Un Algorithme d'Énumération

On présente maintenant un algorithme d'énumération inspiré de l'algorithme de Mazurkiewicz et adapté au modèle étudié dans ce chapitre.

Durant l'exécution de l'algorithme, chaque sommet u essaie d'obtenir une identité qui est un numéro entre 1 et $|V(G)|$. Chaque sommet u va ensuite envoyer son numéro à chacun de ses voisins v, en leur indiquant le numéro du port $\delta_u(v)$. Lorsqu'un sommet u reçoit le numéro n d'un de ses voisins v, il mémorise le triplet $(n, \delta_u(v), \delta_v(u))$ afin de construire sa *vue locale*. Ensuite, chaque sommet va diffuser dans tout le graphe son numéro et sa vue locale. Si un sommet u découvre qu'un autre sommet v a le même numéro que lui, alors le sommet u doit décider s'il modifie son identité. Pour cela, il compare son étiquette initiale $\lambda(u)$ et sa vue locale avec l'étiquette initiale $\lambda(v)$ et la vue locale de v : si l'étiquette de u est plus faible que l'étiquette de v ou si les deux sommets ont la même étiquette et que la vue locale de u est plus «faible» (pour un ordre qu'on expliquera par la suite), alors le sommet u choisit un nouveau numéro (sa nouvelle identité temporaire), en informe ses voisins et diffuse à nouveau son numéro et sa vue locale. Lorsque l'exécution est terminée,

si le graphe $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés, alors chaque sommet a un numéro unique : l'algorithme permet de résoudre le problème du nommage.

7.5.1 Étiquettes

On considère un graphe $\mathbf{G} = (G, \lambda)$ où $\lambda \colon V(G) \to L$ est un étiquetage initial des sommets, qui ne sera pas modifié par l'algorithme. On suppose qu'il existe une fonction δ étiquetage des ports de $\mathbf{G}$. Lors de l'exécution, l'état de chaque sommet va être codé par une étiquette de la forme $(\lambda(v), n(v), N(v), M(v))$ qui représente les informations suivantes :

- la première composante $\lambda(v)$ est l'étiquette initiale et ne sera pas modifiée lors de l'exécution.
- $n(v) \in \mathbb{N}$ est le *numéro* courant du sommet v qui est modifié lors de l'exécution de l'algorithme,
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N}^3)$ est la *vue locale* du sommet v. Informellement, la vue locale contient l'information la plus récente que v a de ses voisins. Si le sommet v a un voisin v' tel que $\delta_{v'}(v) = p$ et $\delta_v(v') = q$, alors le couple (n, p, q) apparaît dans $N(v)$ si n est le numéro qu'avait v' dans le dernier message reçu par v et envoyé par v'. Ainsi $N(v)$ est toujours un ensemble fini de triplets de $\mathbb{N}^3$.
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{P}_{\text{fin}}(\mathbb{N}^3)$ est la *boîte-aux-lettres* de v. Elle va contenir toute l'information reçue par v lors de l'exécution de l'algorithme, i.e., les couples de numéros et de vues locales qui auront été diffusées par tous les sommets du graphe.

Initialement, chaque sommet a une étiquette de la forme $(\lambda(v), 0, \emptyset, \emptyset)$ qui signifie qu'au début de l'algorithme, v n'a pas choisi de numéro et qu'il n'a aucune information à propos de ses voisins, ni à propos des autres sommets du graphe.

Dans notre algorithme, les processus communiquent en échangeant des messages de la forme $< (n, n_{old}, M), p >$. Si un processus v envoie un message à un de ses voisins v', alors $< (n, n_{old}, M), p >$ contient les informations suivantes.

- n est le numéro courant $n(v)$ de v.
- n_{old} est l'ancien numéro de v, i.e., le numéro que v avait lorsqu'il a envoyé le message précédent à v'. Dans le cas où v n'a pas modifié son message entre-temps, $n = n_{old}$.
- M est la boîte-aux-lettres de v.
- p est le port par lequel le message a été envoyé, i.e., $p = \delta_v(v')$.

7.5.2 Un Ordre sur les Vues Locales

Comme pour l'algorithme de Mazurkiewicz [Maz97] et les algorithmes présentés dans les chapitres précédents, les bonnes propriétés de l'algorithme reposent sur un ordre sur les vues locales, i.e., sur les ensembles finis de triplets de $\mathbb{N}^3$. Pour cela, on considère que $\mathbb{N}^3$ est muni de l'ordre lexicographique usuel : $(n, p, q) < (n', p', q')$ si $n < n'$, ou si $n = n'$ et $p < p'$, ou si $n = n'$, $p = p'$ et $q < q'$.

Ensuite, on utilise le même ordre sur les ensembles que Mazurkiewicz : étant donnés deux ensembles $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N}^3)$ distincts, on dit que $N_1 \prec N_2$ si le maximum pour l'ordre lexicographique sur $\mathbb{N}^3$ de la différence symétrique $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ appartient à N_2.

Si $N(u) \prec N(v)$, alors on dit que la vue locale $N(v)$ de v est *plus forte* que celle de u et que $N(u)$ est *plus faible* que $N(v)$. En utilisant l'ordre total $<_L$ de L, on étend l'ordre

$\prec$ pour obtenir un ordre total sur $L \times \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N})$: $(\ell, N) \prec (\ell', N')$ si $\ell <_L \ell'$ ou bien si $\ell = \ell'$ et $N \prec N'$. Par la suite, on notera $\preceq$ la clôture réflexive de $\prec$.

7.5.3 L'Algorithme

L'algorithme d'énumération est L'Algorithme 1. Cet algorithme est décrit par deux règles. La première règle **I** ne peut être appliquée que par un processus v_0 qui lorsqu'il se réveille n'a encore reçu aucun message. Dans ce cas là, le sommet choisit le numéro 1, met à jour sa boîte-aux-lettres et informe ses voisins de son nouveau numéro et de sa nouvelle boîte-aux-lettres.

La seconde règle **R** explique les opérations effectuées par un processus v_0 lorsqu'il reçoit un message $< (n', n'_{old}, M'), p >$ d'un de ses voisins par le port q. Il met d'abord à jour sa boîte-aux-lettres en ajoutant à sa boîte-aux-lettres les éléments de M'. Il modifie ensuite son numéro s'il existe un triplet $(n(v_0), \ell, N) \in M(v_0)$ tel que $(\lambda(v_0), N(v_0)) \prec (\ell, N)$. Puis, il met à jour sa vue locale en effaçant le triplet (n'_{old}, p, q) de $N(v_0)$ (s'il était présent) et en y ajoutant le triplet (n', p, q). Il ajoute ensuite le triplet $(n(v_0), \lambda(v_0), N(v_0))$ à sa boîte-aux-lettres $M(v_0)$. Finalement, si sa boîte-aux-lettres a bien été modifiée lors de l'exécution de cette suite d'instruction, il envoie son numéro et sa boîte-aux-lettres à tous ses voisins.

Si la boîte-aux-lettres d'un sommet n'est pas modifiée lors de l'exécution de la règle **R**, cela signifie que l'information à propos du numéro du voisin dont il a reçu un message était correcte, que toutes les informations contenues dans M' apparaissait déjà dans $M(v_0)$ et que pour tout $(n(v_0), \ell, N) \in M(v_0)$, $(\ell, N) \preceq (\lambda(v_0), N(v_0))$.

7.5.4 Propriétés de l'Algorithme 1

On considère un graphe $\mathbf{G}$ avec un étiquetage des ports δ et une exécution ρ de l'Algorithme 1 sur $(\mathbf{G}, \delta)$. Pour chaque sommet v, une transition interne est le «traitement» d'un message, i.e., l'exécution des instructions de mise à jour de son état une fois qu'un message a été reçu, mais pas les exécutions d'envoi de message. Si un sommet exécute la règle **I**, la transition interne correspondante est la mise à jour de son numéro et de sa boîte-aux-lettres.

L'exécution ρ est donc décrite par une suite $(\texttt{state}_i, \texttt{M}_i)$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v))$ les valeurs des variables $\lambda(v)$, $n(v)$, $N(v)$ et $M(v)$ après la $i$ème étape de l'exécution ρ. On remarque que si lors d'une étape i, une transition d'envoi ou de réception est effectuée, la valeur de $(\lambda(v), n(v), N(v), M(v))$ n'est modifiée pour aucun sommet.

Le lemme suivant, qui peut être facilement prouvé par une récurrence sur la longueur de l'exécution, rappelle quelque propriétés simples qui sont toujours satisfaites par l'état de chaque sommet.

Lemme 7.26 *Pour tout sommet $v \in V(G')$, et pour toute transition i,*

1. $\exists (n, p, q) \in N_i(v) \iff \exists v' \in N_G(v)$ *tel que* $\delta_v(v') = q$ *et* $\delta_{v'}(v) = p$,
2. $n_i(v) \neq 0 \implies (n_i(v), \lambda(v), N_i(v)) \in M_i(v)$,
3. $\forall (n, p, q) \in N_i(v), n \neq 0$ *et* $\exists (n, \ell', N') \in M_i(v)$,
4. $\forall (n, p, q), (n', p', q') \in N_i(v), q \neq q'$,

Algorithme 1 : L'algorithme d'énumération.

I : $\{n(v_0) = 0$ et aucun message n'a été reçu par $v_0\}$
début
  $n(v_0) := 1$;
  $M(v_0) := \{(n(v_0), \lambda(v_0), \emptyset)\}$;
  pour $i := 1$ **à** $\deg(v_0)$ **faire**
    envoyer $< (n(v_0), 0, M(v_0)), i >$ par le port i ;
fin

R : {Un message $< (n', n'_{old}, M'), p >$ est arrivé à v_0 par le port $q\}$
début
  $M_{old} := M(v_0)$;
  $n_{old} := n(v_0)$;
  $M(v_0) := M(v_0) \cup M'$;
  si $n(v_0) = 0$ *ou* $\exists (n(v_0), \ell, N) \in M(v_0)$ *tel que* $(\lambda(v_0), N(v_0)) \prec (\ell, N)$ **alors**
    $n(v_0) := 1 + \max\{n \mid \exists (n, \ell, N) \in M(v_0)\}$;
  $N(v_0) := N(v_0) \setminus \{(n'_{old}, p, q)\} \cup \{(n', p, q)\}$;
  $M(v_0) := M(v_0) \cup \{(n(v_0), \lambda(v_0), N(v_0))\}$;
  si $M(v_0) \neq M_{old}$ **alors**
    pour $i := 1$ **à** $\deg(v_0)$ **faire**
      envoyer $< (n(v_0), n_{old}, M(v_0)), i >$ par le port i;
fin

5. $\forall (n(v_0), \ell, N) \in M_i(v)$, $(\ell, N) \preceq (\lambda(v_0), N(v_0))$,
6. $(n, p, q) \in N_i(v)$ *si et seulement si le dernier message reçu par* v *par le port* q *était de la forme* $< (n, M), p >$.

L'Algorithme 1 a des propriétés de monotonicité intéressantes qui sont données dans le lemme suivant.

Lemme 7.27 *Pour chaque sommet* v *et chaque transition* i,
- $n_i(v) \leq n_{i+1}(v)$,
- $N_i(v) \preceq N_{i+1}(v)$,
- $M_i(v) \subseteq M_{i+1}(v)$.

Preuve : On suppose qu'une transition interne est appliquée à l'étape $i + 1$ sur un sommet v. La propriété est trivialement vraie pour les sommets distincts de v. qui ne sont pas réétiquetés lors de la $(i+1)$ème étape. De plus, il est facile de voir que $M_i(v) \subseteq M_{i+1}(v)$.

Si $n_i(v) \neq n_{i+1}(v)$, alors $n_{i+1}(v) > \max\{n' \mid \exists (n', \ell', N') \in M_i(v)\}$. De plus, ou bien $n_i(v) = 0 < n_{i+1}(v)$, ou alors d'après le Lemme 7.26, $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ et donc $n_i(v) < n_{i+1}(v)$.

Si $N_i(v) \neq N_{i+1}(v)$, alors v a reçu un message **mes** $=< (n', n'_{old}, M'), p >$ par un port q et $N_{i+1}(v) = N_i(v) \setminus \{(n'_{old}, p, q)\} \cup \{(n', p, q)\}$. On note v' le voisin de v tel que $\delta_v(v') = q$ et on sait que $\delta_{v'}(v) = p$. Si $(n'_{old}, p, q) \notin N_i(v)$, cela signifie que **mes** est le premier message reçu par v à travers le port q et alors $\max N_{i+1}(v) \triangle N_i(v) = (n', p, q) \in N_{i+1}(v)$ et $N_i(v) \prec N_{i+1}(v)$.

Si $(n'_{old}, p, q) \in N_i(v)$, alors $n'_{old'} \neq n'$. Si on considère l'étape $j < i$ lors de laquelle le sommet v' a envoyé le message **mes**, on sait que $n'_{old} \leq n' = n_j(v')$. Par conséquent, $\max N_{i+1}(v) \triangle N_i(v) = (n', p, q) \in N_{i+1}(v)$ et $N_i(v) \prec N_{i+1}(v)$. □

Les informations dont dispose chaque sommet v dans sa boîte-aux-lettres permettent d'obtenir des informations vérifiées par la configuration globale du graphe. Les deux lemmes suivants permettent de prouver que si un sommet v connaît un numéro m à une étape i (i.e., il existe ℓ, N tels que $(m, \ell, N) \in M_i(v)$), alors pour chaque $m' \leq m$, il existe un sommet v' tel que $n_i(v') = m'$. On montre d'abord que si v connaît un numéro m, alors il existe un sommet v' tel que $n_i(v) = m'$.

Lemme 7.28 *Pour chaque sommet $v \in V(G)$ et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, il existe un sommet $w \in V(G)$ tel que $n_i(w) = m$.*

Preuve : On remarque d'abord qu'un triplet (m, ℓ, N) est ajouté à une étape i dans $\bigcup_{v \in V(G)} M_i(v)$ seulement s'il existe un sommet v tel que $n_i(v) = m$, $\lambda(v) = \ell$ et $N_i(v) = N$.

Étant donné un sommet v, une étape i et un triplet $(m, \ell, N) \in M_i(v)$, on note $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$. On considère ensuite l'ensemble $U' = \{(u, j) \in U \mid \forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ ou $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ et $j' \leq j\}$. Puisque $(m, \ell, N) \in M_i(v)$, U et U' sont deux ensembles non-vides. On remarque aisément qu'il existe i_0 tel que pour tout $(u, j) \in U'$, $j = i_0$.

Si $i_0 < i$, on considère un élément $(u, i_0) \in U'$. Le numéro $n_{i_0}(u) = m$ de u a donc été modifié à l'étape $i_0 + 1$, mais par maximalité de $(\lambda(u), N_{i_0}(u))$, le sommet u n'a pas pu modifié son numéro. Par conséquent, $i_0 = i$ et il existe donc un sommet w tel que $n_i(w) = m$. □

Dans le lemme suivant, on montre que si un sommet v connaît un numéro m, alors il connaît tous les numéros inférieurs à m.

Lemme 7.29 *Pour chaque sommet v et chaque étape i, pour tout $(m, \ell, N) \in M_i(v)$, pour tout $m' \in [1, m]$, il existe $(m', \ell', N') \in M_i(v)$.*

Preuve : On montre ce lemme par récurrence sur i. Initialement, la propriété est trivialement vraie. On suppose que la propriété est vérifiée pour $i \geq 0$. On suppose qu'une transition interne est effectuée par un sommet v lors de l'étape $i + 1$.

Si le sommet v a appliquée la règle **I**, alors, $M_i(v) = \{(1, \lambda(v_0), \emptyset)\}$ et la propriété est trivialement vraie.

Si le sommet v a appliquée la règle **R**, alors le sommet v a reçu un message **mes** $=< (n', n'_{old}, M'), p >$ d'un sommet v'. On note j l'étape lors de laquelle le sommet v' a envoyé le message et on sait que $M' = M_j(v')$. Si le sommet v ne modifie pas son numéro lors de l'étape $i+1$, alors $M_{i+1}(v) = M_i(v) \cup M_j(v')$ et la propriété est conservée par hypothèse de récurrence. Si le sommet v' modifie son numéro, alors $n_{i+1}(v) = 1 + \max\{n \mid \exists (n, \ell, N) \in M_i(v) \cup M_j(v')\}$ et $M_{i+1}(v) = M_i(v) \cup M_j(v') \cup \{(n_{i+1}(v), \lambda(v), N_{i+1}(v)\}$. Dans ce cas aussi, la propriété est donc conservée. □

On veut maintenant montrer que toute exécution de l'algorithme $\mathcal{M}$ termine sur $\mathbf{G}$. On sait qu'un sommet v peut envoyer un message après avoir effectuer une transition interne seulement s'il a modifié son étiquette lors de cette transition interne. Ainsi, il est suffisant

de montrer qu'il existe une étape i à partir de laquelle les sommets ne modifient plus leurs étiquettes. D'après les Lemmes 7.28 et 7.29, on voit qu'à chaque étape de l'exécution, les numéros des sommets forment un ensemble $[1, k]$ ou un ensemble $[0, k]$ avec $k \leq |V(G)|$. Par conséquent, d'après le Lemme 7.27, on sait qu'il existe une étape i_0 telle que pour tout sommet v et toute étape $i \geq i_0$, $n_{i+1}(v) = n_i(v)$. De plus, on sait que pour tout sommet v, $N_i(v)$ ne peut prendre qu'un nombre fini de valeurs et par conséquent, pour tout sommet v, $M_i(v)$ ne peut prendre qu'un nombre fini de valeurs. Ainsi, d'après le Lemme 7.27, on sait que toute exécution de $\mathcal{M}$ sur $\mathbf{G}$ termine.

Propriétés Satisfaites par l'Étiquetage Final

Puisqu'on sait que l'algorithme termine toujours, on s'intéresse maintenant aux propriétés satisfaites par l'étiquetage final. On sait que lorsque l'algorithme est terminé, tous les messages envoyés sont arrivés à destination.

Lemme 7.30 *Toute exécution ρ de l'algorithme $\mathcal{M}$ sur un graphe simple étiqueté $\mathbf{G} = (G, \lambda)$ avec un étiquetage des ports δ termine et l'étiquetage final $(\lambda, n_\rho, N_\rho, M_\rho)$ des sommets vérifie les propriétés suivantes :*

1. *il existe un entier $k \leq |V(G)|$ tel que $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$,*

et pour tous sommets v, v' :

2. *$M_\rho(v) = M_\rho(v')$,*
3. *$(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$,*
4. *si $n_\rho(v) = n_\rho(v')$, alors $\lambda(v) = \lambda(v')$ et $N_\rho(v) = N_\rho(v')$,*
5. *$(n, p, q) \in N_\rho(v)$ si et seulement s'il existe un sommet $w \in N_G(v)$ tel que $\delta_v(w) = q$, $\delta_w(v) = p$ et $n_\rho(w) = n$.*

Preuve :

1. D'après les Lemmes 7.28 et 7.29 et puisque tout sommet a appliqué au moins une des règles $\mathbf{I}, \mathbf{R}$.
2. Puisque tous les messages envoyés par chaque sommet sont arrivés et qu'à chaque fois qu'un sommet modifie sa boîte-aux-lettres, il l'envoie à ses voisins.
3. C'est une conséquence directe de la propriété précédente d'après le Lemme 7.26.
4. D'après le Lemme 7.26.
5. D'après le Lemme 7.26 et puisque tous les messages envoyés sont arrivés.

□

Grâce au Lemme 7.30, on peut prouver que l'étiquetage final permet de construire un graphe dirigé symétrique $(\mathbf{D}, \delta')$ tel que $(Dir(\mathbf{G}), \delta)$ est un revêtement dirigé symétrique de $(\mathbf{D}, \delta')$.

Proposition 7.31 *Étant donné un graphe $\mathbf{G} = (G, \lambda)$ avec un étiquetage des ports δ, on peut construire, à partir de l'étiquetage final obtenu après une exécution ρ de l'Algorithme 1, un graphe $\mathbf{D}$ avec un étiquetage symétrique des arcs δ' tel que $(\mathbf{G}, \delta)$ est un revêtement dirigé symétrique de $(\mathbf{D}, \delta')$.*

Preuve : On utilise les notations du Lemme 7.30.

On considère le graphe dirigé D défini par $V(D) = \{m \in \mathbb{N} \mid \exists v \in V(G), n_\rho(v) = m\}$, $A(D) = \{a_{(n,p,q,m)} \mid \exists \{v,w\} \in E(G)$ telle que $n_\rho(v) = n, n_\rho(w) = m, \delta_v(w) = p, \delta_w(v) = q\}$ et pour tout arc $a_{(n,p,q,m)}$, $s(a_{(n,p,q,m)}) = n$ et $t(a_{(n,p,q,m)}) = m$. On définit la fonction $Sym : A(D) \to A(D)$ de la manière suivante : pour tout arc $a_{(n,p,q,m)} \in A(D)$, $Sym(a_{(n,p,q,m)}) = a_{(m,q,p,n)}$. On remarque que pour tout arc $a_{(n,p,p,n)}$, $Sym(a_{(n,p,p,n)}) = a_{(n,p,p,n)}$.

On définit un étiquetage η des sommets de D de la manière suivante. Pour tout $v \in V(G)$, $\eta(n_\rho(v)) = \lambda(v)$. D'après le Lemme 7.30, si deux sommets $v, v' \in V(G)$ ont le même numéro, alors $\lambda(v) = \lambda(v')$ et par conséquent, cet étiquetage est bien défini. On définit un étiquetage δ' des arcs de D de la manière suivante : pour tout arc $a_{(n,p,q,m)} \in A(D)$, $\delta'(a_{(n,p,q,m)}) = (p,q)$. Ainsi, pour tout arc $a_{(n,p,q,m)}$, $\delta'(Sym(a_{(n,p,q,m)})) = \delta'(a_{(m,q,p,n)}) = (q,p)$ et l'étiquetage des arcs de D est bien un étiquetage symétrique.

On définit un homomorphisme γ de $(Dir(\mathbf{G}), \lambda)$ dans $(\mathbf{D}, \delta')$ de la manière suivante. Pour tout sommet $v \in V(G)$, $\gamma(v) = n_\rho(v)$ et pour tout arc $a \in A(Dir(\mathbf{G}))$, $\gamma(a) = a_{(n_\rho(s(a)),\delta_{s(a)}(t(a)),\delta_{t(a)}(s(a)),n_\rho(t(a)))}$. Puisque pour tout arc $a \in A(Dir(\mathbf{G}))$, $s(\gamma(a)) = n_\rho(s(a)) = \gamma(s(a))$ et $t(\gamma(a)) = n_\rho(t(a)) = \gamma(t(a))$, γ est un homomorphisme de $Dir(G)$ dans D. Puisque pour tout arc $a \in A(Dir(G))$, $\delta(a) = (\delta_{s(a)}(t(a)), \delta_{t(a)}(s(a))) = \delta'(\gamma(a))$ et puisque pour tout sommet $v \in V(Dir(G))$, $\lambda(v) = \eta(\gamma(v))$, l'homomorphisme γ est un homomorphisme de $(Dir(\mathbf{G}), \delta)$ dans $(\mathbf{D}, \delta')$. De plus, pour tout arc $a \in A(Dir(G))$, $\gamma(Sym(a)) = a_{(n_\rho(t(a)),\delta_{t(a)}(s(a)),\delta_{s(a)}(t(a)),n_\rho(s(a)))} = Sym(\gamma(a))$.

Pour tout arc $a_{(n,p,q,m)} \in A(D)$, il existe une arête $\{v,w\}$ telle que $n_\rho(v) = n$, $\delta_v(w) = q$, $\delta_w(v) = p$ et $n_\rho(w) = m$. D'après le Lemme 7.30, on sait que pour tout sommet $u \in \gamma^{-1}(n)$ (resp. $u \in \gamma^{-1}(m)$), $N_\rho(u) = N_\rho(v)$ (resp. $N_\rho(u) = N_\rho(w)$) et par conséquent, d'après le Lemme 7.30 et puisque δ est une bijection entre $N_G(u)$ et $[1, \deg_G(u)]$, il existe un unique $u' \in N_G(u)$ tel que $n_\rho(u') = m$ (resp. $n_\rho(u') = n$), $\delta_u(u') = p$ ($\delta_u(u') = q$) et $\delta_{u'}(u) = q$ ($\delta_{u'}(u) = p$). Ainsi, pour tout arc $a_{(n,p,q,m)} \in A(D)$, pour tout sommet $u \in \gamma^{-1}(s(a))$ (resp. $u \in \gamma^{-1}(t(a))$), il existe un unique arc $a \in Dir(G)$ tel que $\gamma(a) = a_{(n,p,q,m)}$ et $s(a) = u$ (resp. $t(a) = u$). Par conséquent, $(Dir(\mathbf{G}), \delta)$ est un revêtement symétrique de $(\mathbf{D}, \delta')$. □

On considère maintenant un graphe $\mathbf{G}$ avec un étiquetage des ports δ tel que $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés symétriques. Pour chaque exécution ρ de l'Algorithme 1 sur $(\mathbf{G}, \delta)$, le graphe dirigé obtenu à partir de l'étiquetage final est isomorphe à $(Dir(\mathbf{G}), \delta)$. Par conséquent, l'ensemble des numéros des sommets est exactement $[1, |V(G)|]$: chaque sommet a un identifiant unique.

L'Algorithme 1 permet donc résoudre le nommage sur tout graphe $\mathbf{G}$ avec un étiquetage des ports δ tel que $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés symétriques. Cependant, si aucune information n'est disponible à propos de $(\mathbf{G}, \delta)$, les sommets ne peuvent pas détecter la terminaison.

Néanmoins, il est possible de détecter la terminaison pour un réseau $(\mathbf{G}, \delta)$ donné (l'algorithme dépend alors de $(\mathbf{G}, \delta)$). En effet, une fois qu'un sommet a obtenu le numéro $|V(G)|$, d'après les Lemmes 7.28 et 7.29, il sait que tous les sommets de $\mathbf{G}$ ont un numéro unique qui ne va plus être modifié. Dans ce cas là, on peut aussi résoudre le problème de l'élection, puisque ce sommet peut prendre l'étiquette élu et diffuser ensuite l'information qu'un sommet a été élu.

Par ailleurs, d'après la Proposition 7.24, pour tout graphe $\mathbf{G}$ avec un étiquetage des ports δ tel que $(Dir(\mathbf{G}), \delta)$ n'est pas minimal pour les revêtements dirigés, il n'existe aucun algorithme utilisant des échanges de messages qui permette de résoudre les problèmes du nommage ou de l'élection sur $\mathbf{G}$. On a donc prouvé le théorème suivant.

Théorème 7.32 *Pour tout graphe simple étiqueté $\mathbf{G}$ avec un étiquetage des ports δ, les assertions suivantes sont équivalentes :*

1. *il existe un algorithme de nommage (ou d'énumération) pour $(\mathbf{G}, \delta)$ utilisant des échanges de message,*
2. *il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $(\mathbf{G}, \delta)$ utilisant des échanges de messages,*
3. *$(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés symétriques.*

Remarque 7.33 *Étant donné un graphe $\mathbf{G}$ avec un étiquetage des ports δ tel que $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés, pour détecter que l'Algorithme 1 a attribué un identifiant unique à chaque sommet, il suffit de connaître le nombre de sommets de $\mathbf{G}$. Ainsi, l'Algorithme 1 permet de résoudre l'élection ainsi que le nommage avec détection de la terminaison sur les réseaux $(\mathbf{G}, \delta)$ de taille donnée tels que $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés.*

Comme Yamashita et Kameda, on souhaite caractériser les graphes $\mathbf{G}$ dans lesquels les problèmes d'élection et de nommage peuvent être résolus quel que soit la fonction d'étiquetage des ports choisie. Cette caractérisation, présentée dans le théorème suivant, est un corollaire de la Proposition 7.12 et du Théorème 7.32. On rappelle qu'un algorithme d'élection pour un graphe $\mathbf{G}$ permet de résoudre l'élection sur $\mathbf{G}$ quel que soit l'étiquetage des ports.

Théorème 7.34 *Pour tout graphe simple étiqueté $\mathbf{G}$, il existe un algorithme qui résout l'élection (ou le nommage) sur $\mathbf{G}$ si et seulement si $Dir(\mathbf{G})$ est minimal pour les revêtements dirigés symétriques.*

7.5.5 Remarques sur l'Algorithme

Contrairement aux algorithmes de Yamashita et Kameda [YK99] et de Boldi et al. [BCG+96], l'Algorithme 1 est un algorithme totalement asynchrone, au sens où une fois qu'un sommet a envoyé un message à tous ses voisins, il n'a pas besoin d'attendre d'avoir reçu un message de chacun de ses voisins avant de pouvoir envoyer à nouveau des messages. Une conséquence intéressante est que certaines exécutions de l'Algorithme 1 permettent d'élire dans des graphes qui ne sont pas minimaux pour les revêtements dirigés comme le montre la proposition suivante.

Proposition 7.35 *Pour tout réseau $(\mathbf{G}, \delta)$, il existe une exécution de l'Algorithme 1 sur $(\mathbf{G}, \delta)$ qui permet de résoudre nommage et élection sur $(\mathbf{G}, \delta)$.*

Preuve : Étant donné un réseau $(\mathbf{G}, \delta)$, on considère une exécution où un seul sommet v_0 applique la règle $\mathbf{I}$ et où tous les autres sommets commencent à exécuter l'algorithme uniquement lorsqu'ils reçoivent un message. En d'autres termes, tous les sommets sont initialement endormis, et il y a un unique sommet v_0 qui initie le calcul. Ainsi, pour tout sommet $v \in V(G)$ différent de v_0, à chaque fois que v modifie son numéro, il existe un triplet

$(1, \lambda(v_0), \emptyset)$ dans $M(v)$ et par conséquent, le sommet v ne prend jamais le numéro 1. Ainsi, si on considère l'étiquetage final obtenu à la fin de l'exécution, il existe un seul sommet dont le numéro est 1. Par conséquent, d'après la Proposition 7.14 et la Proposition 7.31, on sait que chaque numéro apparaissant dans l'étiquetage final des sommets est unique. Ainsi, l'exécution décrite permet de résoudre le nommage et l'élection sur $(\mathbf{G}, \delta)$. □

Cependant, pour tout réseau $(\mathbf{G}, \delta)$, il existe une exécution «canonique» de l'Algorithme 1 sur $(\mathbf{G}, \delta)$ qui permet de résoudre le nommage sur $(\mathbf{G}, \delta)$ si et seulement si $(Dir(\mathbf{G}), \delta)$ est minimal pour les revêtements dirigés symétriques.

Une exécution est *synchrone* est exécution en *rondes*. Lors d'une ronde, chaque sommet reçoit tous les messages qui lui ont été envoyés lors de la ronde précédente, puis il modifie son état en fonction des messages reçus et il envoie ensuite autant de messages que nécessaires à ses voisins. En particulier, pour toute exécution synchrone de l'Algorithme 1 sur un réseau $(\mathbf{G}, \delta)$, tous les sommets de $\mathbf{G}$ exécutent la règle **I** lors de la première ronde.

La proposition suivante est similaire aux résultats d'impossibilité présentés par Yamashita et Kameda [YK99] et Boldi et al. [BCG+96].

Proposition 7.36 *Pour tout réseau $(\mathbf{G}, \delta)$, si on considère une exécution synchrone ρ de l'Algorithme 1 sur $(\mathbf{G}, \delta)$, le graphe dirigé reconstruit à partir de l'étiquetage final est la base minimale de $(Dir(\mathbf{G}), \delta)$.*

Preuve : Étant donné un réseau $(\mathbf{G}, \delta)$, on note $(\mathbf{D}, \delta')$ la base minimale de $(Dir(\mathbf{G}), \delta)$ et γ l'homomorphisme localement bijectif de $(Dir(\mathbf{G}), \delta)$ dans $(\mathbf{D}, \delta')$.

Pour tous sommets v, v' tels que $\gamma(v) = \gamma(v')$, les sommets v et v' ont initialement la même étiquette et appliquent tous les deux la règle **I** et envoient les mêmes messages à leurs voisins. De plus, pour tout sommet $u \in N_G(v)$, il existe un sommet $u' \in N_G(v')$ tel que $\delta_{v'}(u') = \delta_v(u)$ et $\gamma(u) = \gamma(u')$. Par conséquent, lors de toute ronde i, les sommets v et v' reçoivent les mêmes messages de leurs voisins, modifient leurs étiquettes de la même manière et envoient les mêmes messages à leurs voisins.

Par conséquent, dans la configuration finale, pour tous sommets $v, v' \in V(G)$ tels que $\gamma(v) = \gamma(v')$, $n(v) = n(v')$ et $N(v) = N(v')$. Ainsi, le graphe dirigé reconstruit à partir de l'étiquetage final est $(\mathbf{D}, \delta')$. □

Comme dans les modèles étudiés dans les chapitres précédents, on se rend compte que la boîte-aux-lettres de chaque sommet contient beaucoup d'informations inutiles. En effet, si un triplet (n, ℓ, N) apparaît dans la boîte-aux-lettres $M(v)$ d'un sommet v, on peut supprimer de $M(v)$ tous les triplets $(n, \ell', N') \in M(v)$ tels que $(\ell', N') \prec (\ell, N)$. De cette manière, la boîte-aux-lettres de chaque sommet contient au plus $|V(G)|$ triplets (n, ℓ, N).

Par ailleurs, il n'est pas forcément nécessaire d'assurer que les canaux de communication préservent l'ordre des messages. En effet, lorsqu'un sommet v reçoit un message $< (n, M), p >$ par le port q, s'il existe un triplet $(n', p, q) \in N(v)$ tel que $n' > n$, alors v sait qu'il a déjà reçu un message $< (n', M'), p >$ à travers le port q qui contenait une information plus à jour à propos de l'état de son voisin v' tel que $\delta_v(v') = q$. De plus, on sait aussi que $M \subseteq M'$ et par conséquent, le sommet v peut tout simplement ignorer un message $< (n, M), p >$ par le port q s'il existe un triplet $(n', p, q) \in N(v)$ tel que $n' > n$.

Cependant, le fait de supposer que les canaux de communication préservent l'ordre des messages permet d'obtenir un algorithme nécessitant des messages de plus petite taille. En effet, à chaque fois qu'un sommet modifie sa boîte-aux-lettres, il envoie tout le contenu

de sa boîte-aux-lettres à tous ses voisins. Il est toutefois suffisant de ne renvoyer que les informations qui ont été ajoutées lorsque le sommet v_0 a modifié sa boîte-aux-lettres, i.e., dans l'Algorithme 1, il suffit d'envoyer les éléments de $M(v_0) \setminus M_{old}$.

L'Algorithme 1 a été implémenté sous ViSiDiA en utilisant le fait que les canaux de communication préservent l'ordre des messages et en réduisant la taille des boîte-aux-lettres et des messages comme on l'a indiqué.

7.5.6 Complexité

On s'intéresse ici à la complexité de l'Algorithme 1. Pour cela, on étudie la complexité en message et la complexité en temps. On s'intéresse aussi à la taille des messages et à la taille de la mémoire utilisé par chaque sommet.

Comme dans [Tel00] (p.71), la complexité en temps est mesurée en supposant que les transitions effectuées par chaque sommet nécessitent zéro unités de temps et que le temps de transmission d'un message (i.e., le temps entre l'émission et la réception d'un message) se fait en une unité de temps. Cela correspond au nombre de rondes effectuées lors d'une exécution synchrone de l'algorithme. On note que les propriétés de correction de l'algorithme sont indépendantes de ces suppositions.

On suppose que l'étiquetage initial λ du graphe G est tel que chaque étiquette initiale ℓ a une taille en $O(\log |V(G)|)$ bits (ce qui est suffisant pour attribuer des étiquettes différentes à tous les sommets de G).

On considère une exécution, où chaque sommet n'envoie à ses voisins que les triplets qu'il a rajouté dans sa boîte-aux-lettres, et non pas toute sa boîte-aux-lettres à chaque étape. De plus, on suppose qu'il envoie ses triplets un par un, i.e., les messages échangés sont de la forme $< (n, \{(n', \ell', N')\}), p >$ et si un sommet doit envoyer plusieurs triplets à ses voisins, il envoie plusieurs messages.

Proposition 7.37 *Pour tout réseau $(\mathbf{G}, \delta)$ à n sommets, m arêtes, de degré maximum Δ et de diamètre D, toute exécution de l'Algorithme 1 sur $(\mathbf{G}, \delta)$ nécessite $O(Dn^2)$ unités de temps et $O(m^2 n)$ messages de taille $O(\Delta \log n)$ bits. De plus, chaque processus utilise $O(\Delta n \log n)$ bits de mémoire.*

Preuve : On considère un réseau $(\mathbf{G}, \delta)$ à n sommets, m arêtes, de degré maximum Δ et de diamètre D. On considère une exécution ρ de l'Algorithme 1 sur $(\mathbf{G}, \delta)$. D'après les Lemmes 7.28 et 7.29, on sait que chaque sommet ne modifie pas son numéro plus de n fois.

Pour tout sommet v, puisque les numéros de v et de ses voisins ne peuvent qu'augmenter, le couple $(n(v), N(v))$ peut prendre au plus $(d(v) + 1)n$ valeurs différentes. À chaque fois que le sommet v modifie son numéro ou sa vue locale, cela peut générer au plus $O(m)$ messages, puisqu'un sommet qui à déjà le triplet $(n(v), \lambda(v), N(v))$ dans sa boîte-aux-lettres ne diffuse plus cette information. Ainsi, toute exécution de l'algorithme nécessite $O(m^2 n)$ messages. Puisqu'on suppose que tous les messages utilisés sont de la forme $< (n, \{(n', \ell', N')\}), p >$ et que pour tout sommet v, $N(v)$ contient au plus Δ éléments qui sont tous inférieurs à n, la taille des messages utilisés est $O(\Delta \log n)$ bits.

À chaque fois qu'un sommet v modifie son numéro ou sa vue locale, en D unités de temps, tous les sommets de $\mathbf{G}$ connaissent le triplet $(n(v), \lambda(v), N(v))$. De plus, à chaque fois qu'un sommet v modifie son numéro, les voisins de v mettent à jour leur vues locales en

une unité de temps. Ainsi, une fois qu'un sommet a modifié son numéro, tous les sommets de $\mathbf{G}$ ont la même boîte-aux-lettres au bout de $D+1$ unités de temps si aucun sommet ne modifie son numéro entre-temps. Si tous les sommets ont la même boîte-aux-lettres et s'il n'y a plus de messages en transit, on sait d'après le Lemme 7.26 que l'exécution est terminée. Par conséquent, puisqu'il y a au plus $\frac{n(n-1)}{2}$ changements de numéros lors de l'exécution de l'algorithme, il faut au plus $O(Dn^2)$ unités de temps pour que l'algorithme termine.

Pour chaque sommet v, $n(v)$ peut être représenté avec $\log n$ bits et $N(v)$ peut être représenté avec $O(\Delta \log n)$ bits. Puisque chaque sommet v conserve que l'information utile dans sa boîte-aux-lettres, il existe au plus n triplets (n_0, ℓ, N) dans $M(v)$ et chacun de ces triplets peut être représenté avec $O(\Delta \log n)$ bits. Par conséquent, on peut représenter la boîte-aux-lettres de chaque sommet avec $O(\Delta n \log n)$ bits. Ainsi, chaque sommet a besoin de $O(\Delta n \log n)$ bits de mémoire pour stocker son état. □

Remarque 7.38 *Pour tout réseau $(\mathbf{G}, \delta)$, les algorithmes de Yamashita et Kameda [YK96b] et de Boldi et al. [BCG+96] nécessitent $O(n)$ unités de temps et $O(mn)$ messages de taille $2^{O(n)}$ bits ; de plus, chaque processus nécessite $2^{O(n)}$ bits de mémoire.*

Ainsi, l'Algorithme 1 est un algorithme qui nécessite des messages et une mémoire de tailles polynomiales, alors que les algorithmes connus jusqu'à présent utilisaient des messages de taille exponentielle. Par ailleurs, l'Algorithme 1 utilise des structures de données simples et a donc pu être implémenté sous ViSiDiA.

7.6 Élection dans les Familles de Diamètre Borné

On montre dans cette partie que pour pouvoir élire dans un réseau $(\mathbf{G}, \delta)$ minimal pour les revêtements dirigés symétriques, il n'est pas nécessaire de connaître la taille de $\mathbf{G}$, mais qu'une borne sur le diamètre suffit. On présente une adaptation de l'algorithme GSSP décrit dans le Chapitre 2 qui permet de vérifier que tous les sommets du graphe ont la même boîte-aux-lettres. L'algorithme SSP [SSP85] a été décrit dans le cadre des échanges de messages et il n'est donc pas surprenant qu'on puisse obtenir un algorithme GSSP dans ce cadre.

Comme dans les chapitres précédents, chaque sommet dispose d'un rayon de confiance $a(v)$ et d'un ensemble $A(v)$ contenant l'information dont v dispose à propos du rayon de confiance de ses voisins. Comme précédemment, à chaque fois qu'un sommet v modifie sa boîte-aux-lettres, le sommet v réinitialise $a(v)$ et $A(v)$. Si un sommet v s'aperçoit que son rayon de confiance est inférieur ou égal au rayon de confiance de tous ses voisins, alors il peut augmenter son propre rayon de confiance. De plus, à chaque fois qu'un sommet v modifie son rayon de confiance, il en informe ses voisins, afin que ceux-ci mettent à jour l'information dont ils disposent à propos de v. Ainsi, les messages échangés sont de la forme $< (n, n_{old}, M, a), p >$ où la seule différence avec l'Algorithme 1 est le champ a qui contient le rayon de confiance du sommet qui envoie le message. L'adaptation de l'Algorithme 1 prenant en compte ces modifications est l'Algorithme 2.

On s'intéresse maintenant aux propriétés satisfaites par toute exécution de l'Algorithme 2. Comme précédemment, on considère une exécution de l'algorithme sur un réseau $(\mathbf{G}, \delta)$. Pour tout sommet $v \in V(G)$, on note $(\lambda(v), n_i(v), N_i(v), M_i(v), a_i(v), A_i(v))$ les valeurs des variables $\lambda(v)$, $n(v)$, $N(v)$, $M(v)$, $a(v)$ et $A(v)$ après la $i$ème étape de l'exécution.

Algorithme 2 : L'algorithme d'énumération utilisant GSSP.

I : $\{n(v_0) = 0$ et aucun message n'a été reçu par $v_0\}$
début
  $n(v_0) := 1$;
  $M(v_0) := \{(n(v_0), \lambda(v_0), \emptyset)\}$;
  $a(v_0) := 0$;
  $A(v_0) := \{(q, -1) \mid q \in [1, \deg_G(v_0)]\}$;
  pour $i := 1$ **à** $\deg(v_0)$ **faire**
    envoie $< (n(v_0), 0, M(v_0), a(v_0)), i >$ par le port i ;
fin

R : {Un message $< (n', n'_{old}, M', a'), p >$ est arrivé à v_0 par le port $q\}$
début
  $M_{old} := M(v_0)$;
  $n_{old} := n(v_0)$;
  $a_{old} := a(v_0)$;
  $M(v_0) := M(v_0) \cup M'$;
  si $n(v_0) = 0$ *ou* $\exists (n(v_0), \ell, N) \in M(v_0)$ *tel que* $(\lambda(v_0), N(v_0)) \prec (\ell, N)$ **alors**
    $n(v_0) := 1 + \max\{n \mid \exists (n, \ell, N) \in M(v_0)\}$;
  $N(v_0) := N(v_0) \setminus \{(n'_{old}, p, q)\} \cup \{(n', p, q)\}$;
  $M(v_0) := M(v_0) \cup \{(n(v_0), \lambda(v_0), N(v_0))\}$;
  si $M(v_0) \neq M_{old}$ **alors**
    $a(v_0) := 0$;
    $A(v_0) := \{(q, -1) \mid q \in [1, \deg_G(v_0)]\}$;
  si $M(v_0) = M'$ **alors**
    $A(v_0) := A(v_0) \setminus \{(q, a'') \mid (q, a'') \in A(v_0)\} \cup \{(q, a')\}$;
  si $\forall (p, a) \in A(v), a(v_0) \leq a$ **alors**
    $a(v_0) := 1 + \max\{a \mid \exists (p, a) \in A(v_0)\}$;
  si $M(v_0) \neq M_{old}$ *ou* $a(v_0) \neq a_{old}$ **alors**
    pour $i := 1$ **à** $\deg(v_0)$ **faire**
      envoie $< (n(v_0), n_{old}, M(v_0), a(v_0)), i >$ par le port i;
fin

Le lemme suivant qui peut être facilement prouvé par récurrence sur le nombre d'étapes montre que le rayon de confiance d'un sommet ne peut qu'augmenter tant que sa boîte-aux-lettres n'est pas modifiée.

Lemme 7.39 *Pour tout sommet v et toute étape i, si $M_i(v) = M_{i+1}(v)$, alors $a_{i+1}(v) \geq a_i(v)$ et si $n_i(v) \neq 0$, pour tout $p \in [1, \deg_G(v)]$, il existe $(p, a) \in A_i(v)$ et $(p, a') \in A_{i+1}(v)$ tels que $a' \geq a \geq a_{i+1}(v) - 1$.*

Dans le lemme suivant, on montre que le rayon de confiance d'un sommet permet d'obtenir des informations sur le contenu des boîtes-aux-lettres d'autres sommets du graphe lors d'étapes précédentes de l'exécution. Cette propriété correspond aux propriétés décrites dans les Lemmes 2.28 et 3.47 pour les algorithmes des Chapitres 2 et 3.

Lemme 7.40 *Pour tout sommet $v \in V(G)$ et toute étape i, pour tout sommet $w \in V(G)$ tel que $\text{dist}_G(v, w) \leq a_i(v)$, il existe une étape $j \geq i$ telle que $a_j(w) \geq a_i(v) - \text{dist}_G(v, w)$ et $M_j(v) = M_i(v)$.*

Preuve : On fait une démonstration par récurrence sur la distance k entre v et w dans G. Si $k = 0$, la propriété est trivialement vraie. On suppose maintenant que la propriété est vraie pour tous sommets v, w tels que $\text{dist}_G(v, w) \leq k$.

On considère deux sommets v, w et une étape i tels que $a_i(v) \geq k + 1 \geq 1$ et $\text{dist}_G(v, w) = k + 1$. Il existe un sommet $u \in N_G(v)$ tels que $\text{dist}_G(u, w) = k$. On sait qu'il existe $(\delta_v(u), a) \in A_i(v)$ tel que $a \geq a_i(v) - 1 \geq 0$. On considère l'étape $j' < i$ lors de laquelle le dernier message reçu par v par le port $\delta_v(u)$ a été envoyé par u. On sait que $M_{j'}(u) = M_i(v)$ et $a_{j'}(u) = a \geq a_i(v) - 1$. Par hypothèse de récurrence, on sait qu'il existe une étape $j < j'$ telle que $a_j(w) \geq a_{j'}(u) - k \geq a_i(v) - (k+1)$ et $M_j(w) = M_{j'}(u) = M_i(v)$. Ainsi, la propriété est vérifiée pour tous sommets v, w à distance $k + 1$ dans G. □

Comme dans l'algorithme GSSP des Chapitres 2 et 3, on montre que si à un moment donné, un sommet a un rayon de confiance supérieur au diamètre du graphe, alors tous les sommets du graphe ont la même boîte-aux-lettres et ont tous un rayon de confiance supérieur à 0.

Lemme 7.41 *S'il existe un sommet v et une étape i telle que $a_i(v) \geq D(G) + 1$, alors pour tout $w \in V(G)$, $M_i(w) = M_i(v)$ et $a_i(v) \geq 0$.*

Preuve : Puisque $a_i(v) > D(G)$, on sait d'après le Lemme 7.40 que pour tout sommet $w \in V(G)$, il existe une étape $i_w < i$ telle que $a_{i_w} \geq 1$ et $M_{i_w}(w) = M_i(v)$.

Supposons qu'il existe un sommet w tel que $M_i(w) \neq M_{i_w}(w)$. Soit j l'étape de l'exécution lors de laquelle pour la première fois, la boîte-aux-lettres d'un sommet w qui valait $M_i(v)$ a été modifiée. Autrement dit, pour tout sommet $w \in V(G)$, il existe une étape $j' \geq j - 1$ telle que $M_{j'}(w) = M_i(v)$ et il existe un sommet w tel que $M_{j-1}(w) = M_i(v) \subsetneq M_j(w)$. On considère le message $\textbf{mes} =< (n, n_{old}, M, a), p >$ reçu par w qui a été traité à l'étape j et on note q le port par lequel il est arrivé et u le voisin de w tel que $\delta_w(u) = q$. En raison du choix de j, on sait que lors de l'étape j' lors de laquelle le message été envoyé par u, $M_{j'}(u) = M \subseteq M_j(w)$.

On sait que $a_{j-1}(w) \geq 1$ et il existe donc un couple $(\delta_w(u), a) \in A(w)$ tel que $a \geq 0$. Ainsi, il existe une étape $j'' \leq j - 1$ lors de laquelle le sommet u a envoyé un message $\textbf{mes}' =< (n, n_{old}, M_{j-1}(w), a), p >$ et alors, $M_{j''}(u) = M_{j-1}(w)$. Puisque le message $\textbf{mes}'$ est arrivé avant le message **mes** et que les canaux de communication préservent l'ordre

des messages, on sait que $j'' < j'$. Ainsi, $M_{j''}(u) = M_{j'}(u)$ et alors $n_{j''}(u) = n_{j'}(u)$. Par conséquent, lorsque le sommet w a reçu le message **mes**, $M(w)$ et $N(w)$ n'ont pas été modifiés et d'après le Lemme 7.26, $n(w)$ n'a pas pu être modifié non plus. Par conséquent, $M_j(w) = M_{j-1}(w)$ et pour tout sommet $w \in V(G)$, $M_i(w) = M_i(v)$. □

Ainsi, si on connaît une borne B sur le diamètre de $\mathbf{G}$, l'Algorithme 2 permet de détecter que l'exécution de l'algorithme à la Mazurkiewicz sous-jacent est terminé. En effet, une fois qu'un sommet a un rayon de confiance supérieur ou égal à $B+1$, il sait que tous les sommets de $\mathbf{G}$ ont la même boîte-aux-lettres et qu'ils ont leurs numéros et vues locales finaux.

Si on sait que le réseau $(\mathbf{G}, \delta)$ est minimal pour les revêtements dirigés symétriques, on sait d'après la Proposition 7.31 que tous les sommets ont un identifiant unique et dans ce cas là, le sommet dont le numéro est 1 peut prendre l'étiquette ÉLU et diffuser l'information. On a par conséquent montré le théorème suivant.

Théorème 7.42 *Pour tout entier B, il existe un algorithme d'élection et un algorithme de nommage avec détection de la terminaison utilisant des échanges de messages pour la famille des réseaux $(\mathbf{G}, \delta)$ minimaux pour les revêtements dirigés symétriques dont le diamètre est bornée par B.*

Ainsi, comme dans les modèle considéré dans les Chapitres précédents, il n'est pas nécessaire de connaître la taille pour pouvoir élire dans un réseau $(\mathbf{G}, \delta)$ que l'on sait minimal pour les revêtements dirigés symétriques : une borne sur la taille ou le diamètre est suffisante.

Par ailleurs, en utilisant la même preuve que dans le Chapitre 3 et la Proposition 7.14, on montre que la connaissance d'une borne serrée sur la taille d'un réseau $(\mathbf{G}, \delta)$ permet ou bien de résoudre le problème de l'élection sur $(\mathbf{G}, \delta)$, ou bien de détecter que $(\mathbf{G}, \delta)$ n'est pas minimal pour les revêtements dirigés symétriques.

Théorème 7.43 *Pour tout entier B, il existe un algorithme effectif d'élection et de nommage avec détection de la terminaison utilisant des échanges de messages pour la classe des réseaux $(\mathbf{G}, \delta)$ tels que $V(G) \leq B < 2|V(G)|$.*

Remarque 7.44 *L'Algorithme 2 utilise le fait que les canaux préservent l'ordre des messages, mais cette hypothèse n'est pas forcément nécessaire. On peut adapter l'Algorithme 2 de la même façon que l'Algorithme 1 afin que les propriétés des Lemmes 7.27 et 7.39 soient préservées. En effet, si un sommet v reçoit un message* **mes** $=< (n', n'_{old}, M', a'), p >$ *par le port q et s'il existe $(q, a) \in A(v)$ et $(n, q, p) \in N(v)$ tels que $M' \subseteq M(v)$, $n' \leq n$ et $a' \leq a$, alors v peut ignorer le message* **mes**.

7.7 Des Étiquetages des Ports plus Faibles

Dans [YK99], Yamashita et Kameda considèrent le modèle étudié dans ce chapitre ainsi que des modèles légèrement différents. Les différences entre ces modèles résident dans l'existence ou la non-existence de ports d'entrées et de sorties. Dans le modèle considéré ici, chaque sommet distingue les ports par lesquels il envoie des messages et les ports par lesquels il reçoit les messages ; ce modèle est donc appelé un modèle *port-à-port* (qu'on appelle PP en abrégé) par Yamashita et Kameda.

Dans l'algorithme de Yamashita et Kameda pour élire dans le modèle port-à-port, à chaque fois qu'un sommet envoie un message, il envoie un message à tous ses voisins (les messages diffèrent uniquement par le port de sortie). On remarque que l'Algorithme 1 fonctionne de la même manière. Ainsi, dans [YK99], Yamashita et Kameda considère un autre type d'émissions de messages, qu'ils appellent *diffusion*. Dans ce modèle, à chaque fois qu'un sommet envoie un message, il envoie le même message à tous ses voisins (il ne peut pas ajouter un numéro de port au message). Le modèle correspondant est le modèle *diffusion-à-port* (qu'on appelle DP en abrégé).

Dans [YK99], Yamashita et Kameda considère aussi des modèles où lorsqu'un sommet reçoit un message, il ne connaît pas le numéro du port par lequel ce message est arrivé : tous les messages reçus par un sommet v arrive dans une unique *boîte*. Les modèles correspondants sont appelés *port-à-boîte* (qu'on appelle PB en abrégé) et *diffusion-à-boîte* (qu'on appelle DB en abrégé).

7.7.1 Nommage

De la même manière que dans le modèle port-à-port, on peut associer un graphe dirigé étiqueté à tout graphe $\mathbf{G}$ dans les autres modèles. Étant donné un graphe $\mathbf{G}$ avec un étiquetage des ports δ, on considère le graphe $Dir(\mathbf{G})$ et pour chaque modèle, on lui associe une fonction d'étiquetage des arcs définie de la manière suivante. Pour tout arc $a \in A(Dir(G))$, $\delta_{PP}(a) = (p, q)$, $\delta_{PB}(a) = (p, 0)$, $\delta_{DP}(a) = (0, q)$, $\delta_{DB}(a) = (0, 0)$ où $p = \delta_{s(a)}(t(a))$ et $q = \delta_{t(a)}(s(a))$. Comme leurs noms l'indiquent, les fonctions δ_{PP}, δ_{PB}, δ_{DP} et δ_{DB} correspondent respectivement aux modèles port-à-port, port-à-boîte, diffusion-à-port, diffusion-à-boîte.

Remarque 7.45 *Dans le modèle port-à-port, comme expliqué précédemment, l'étiquetage des arcs ainsi obtenu est déterministe, codéterministe et symétrique. Dans le modèle port-à-boîte, l'étiquetage des arcs ainsi obtenu est déterministe. Dans le modèle diffusion-à-boîte, l'étiquetage des arcs ainsi obtenu est codéterministe.*

Dans le modèle diffusion-à-boîte, l'étiquetage des arcs ainsi obtenu assigne la même étiquette à tous les arcs, i.e., l'étiquetage des ports initial est inutile, puisque les processus n'ont pas accès aux numéros de ports.

Avec un argument similaire à celui utilisé dans la Proposition 7.36, on obtient la proposition suivante, qui dit que pour chaque modèle considéré, il n'existe pas d'algorithme de nommage (resp. d'élection) pour un graphe $\mathbf{G}$ avec un étiquetage des ports δ si le graphe dirigé correspondant n'est pas minimal pour les fibrations (resp. minimal pour les fibrations non-triviales).

Proposition 7.46 *Étant donné un graphe $\mathbf{G}$ avec un étiquetage des ports δ, pour tout modèle $M \in \{DP, PB, DB\}$, il n'existe pas d'algorithme de nommage (resp. d'élection) dans le modèle M pour le réseau $(\mathbf{G}, \delta)$ si $(Dir(\mathbf{G}), \delta_M)$ n'est pas minimal pour les fibrations (resp. minimal pour les fibrations non-triviales).*

Dans le modèle port-à-boîte, on sait que $(Dir(\mathbf{G}), \delta_{PB})$ a un étiquetage des ports déterministes et par conséquent, si $(Dir(\mathbf{G}), \delta_{PB})$ est fibré sur un graphe dirigé $(\mathbf{D}, \delta')$, alors $(Dir(\mathbf{G}), \delta_{PB})$ est un revêtement de $(\mathbf{D}, \delta')$. Par conséquent, dans ce modèle, les conditions nécessaires pour l'élection et le nommage sont les mêmes.

Réciproquement, il n'est pas difficile que l'Algorithme 1 peut être adapté à chacun des modèles port-à-boîte, diffusion-à-port et diffusion-à-boîte. La seule différence réside dans le fait où dans les modèles port-à-boîte et diffusion-à-boîte, la vue locale $N(v)$ de chaque sommet v est un multi-ensemble, puisque les sommets ne peuvent pas distinguer les ports par lesquels les messages arrivent. On remarque cependant que pour que chaque sommet puisse avoir une information correcte à propos de ses voisins, on est obligé d'assurer que dans ces modèles, les canaux de communication préservent l'ordre des messages : un sommet ne peut pas savoir d'où vient un message et il ne peut donc pas détecter s'il est inutile ou pas. À l'exception de cette propriété, les algorithmes obtenus dans les modèles port-à-boîte, diffusion-à-port et diffusion-à-boîte ont les même propriétés que l'Algorithme 1 décrites dans les Sections 7.5.5 et 7.5.6.

Ainsi, pour toute exécution ρ de l'algorithme dans un modèle $M \in \{DP, PB, DB\}$, on peut reconstruire un graphe dirigé $(\mathbf{D}, \delta')$ à partir de l'étiquetage final de ρ tel que $(Dir(\mathbf{G}), \delta_M)$ est fibré sur $(\mathbf{D}, \delta')$. Par ailleurs, dans le modèle port-à-boîte, le graphe dirigé $(Dir(\mathbf{G}), \delta_{PB})$ a un étiquetage des arcs déterministe et d'après la Proposition 7.9, $(Dir(\mathbf{G}), \delta_{PB})$ est un revêtement dirigé de $(\mathbf{D}, \delta')$. Grâce aux conditions nécessaires présentées dans la Proposition 7.46, on a donc le théorème suivant qui caractérise les graphes où on peut nommer dans chacun des trois modèles. On note que dans le modèle port-à-boîte, cette caractérisation est aussi valable pour les graphes où on peut résoudre l'élection.

Théorème 7.47 *Pour tout graphe simple étiqueté $\mathbf{G}$ avec un étiquetage des ports δ, on a les caractérisations suivantes.*

1. *Il existe un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $(\mathbf{G}, \delta)$ dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement si $(Dir(\mathbf{G}), \delta_{DP})$ (resp. $(Dir(\mathbf{G}), \delta_{DB})$) est minimal pour les fibrations.*
2. *Il existe un algorithme d'élection et un algorithme de nommage (ou d'énumération) avec détection de la terminaison pour $(\mathbf{G}, \delta)$ dans le modèle port-à-boîte si et seulement si $(Dir(\mathbf{G}), \delta_{PB})$ est minimal pour les revêtements dirigés.*

On remarque que l'Algorithme 2 peut aussi être étendu aux modèles port-à-boîte, diffusion-à-port et diffusion-à-boîte. Ici aussi, il faut faire attention à utiliser un multi-ensemble $A(v)$ pour stocker les rayons de confiance de chacun des voisins de v et lorsqu'un sommet envoie un message à l'un de ses voisins, il lui signifie aussi quel était son ancien rayon de confiance, afin que celui-ci puisse avoir une information correcte à propos du rayon de confiance de ses voisins. Par ailleurs, il est nécessaire que chaque sommet connaisse son degré afin de pouvoir décider à quel moment, il peut augmenter son rayon de confiance. Ainsi, dans le modèle port-à-boîte (resp. diffusion-à-port, diffusion-à-port), il est possible de résoudre le nommage sur la famille des réseaux $(\mathbf{G}, \delta)$ tels que $(Dir(\mathbf{G}), \delta_{PB})$ est minimal pour les revêtements dirigés (resp. $(Dir(\mathbf{G}), \delta_{DP})$ est minimal pour les fibrations, $(Dir(\mathbf{G}), \delta_{DB})$ est minimal pour les fibrations) dont le diamètre est borné. On a donc un résultat équivalent au Théorème 7.42 pour le problème du nommage.

Il faut noter que dans le modèle port-à-boîte, ce résultat est aussi vrai pour le problème de l'élection et qu'avec les mêmes techniques que dans la preuve du Théorème 7.43, on peut montrer que pour tout entier B il existe un algorithme effectif d'élection et de nommage sur la famille des réseaux $(\mathbf{G}, \delta)$ tels que $|V(G)| \leq B < 2|V(G)|$. On a donc aussi un résultat analogue au Théorème 7.43 dans le modèle port-à-boîte.

7.7.2 Élection

On considère maintenant le problème de l'élection dans les modèles diffusion-à-port et diffusion-à-boîte. La Proposition 7.46 décrit des conditions nécessaires pour qu'un réseau $(\mathbf{G}, \delta)$ admette un algorithme d'élection dans ces modèles. On va expliquer comment utiliser les techniques et résultats de la Section 7.6 et du Chapitre 4 pour montrer que ces conditions sont suffisantes.

Dans le modèle diffusion-à-port (resp. diffusion-à-boîte), pour tout réseau $(\mathbf{G}, \delta)$, on peut en connaissant une borne sur le diamètre de $\mathbf{G}$ reconstruire un graphe $(\mathbf{D}, \delta')$ tel que $(Dir(\mathbf{G}), \delta_{DP})$ (resp. $(Dir(\mathbf{G}), \delta_{DB})$) est fibré sur $(\mathbf{D}', \delta')$. Ainsi, en utilisant les techniques utilisés dans la Section 4.5, on peut montrer le théorème suivant qui est analogue au Théorème 4.33

Théorème 7.48 *Pour tout graphe simple étiqueté $\mathbf{G}$ avec un étiquetage des ports δ, il existe un algorithme d'élection pour $(\mathbf{G}, \delta)$ dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement si $(Dir(\mathbf{G}), \delta_{DP})$ (resp. $(Dir(\mathbf{G}), \delta_{DB})$) est minimal pour les fibrations non-triviales.*

Comme dans la Section 4.5, on peut aussi montrer que dans le modèle diffusion-à-port (resp. diffusion-à-boîte), une connaissance d'une borne sur le diamètre permet de résoudre l'élection sur la famille des réseaux $(\mathbf{G}, \delta)$ tels que $(Dir(\mathbf{G}), \delta_{DP})$ (resp. $(Dir(\mathbf{G}), \delta_{DB})$) est minimal pour les fibrations non-triviales. On a donc un résultat analogue au Théorème 7.42 pour l'élection dans les modèles diffusion-à-port et diffusion-à-boîte.

Par ailleurs, en utilisant les techniques mises en oeuvre dans la Section 4.5, on peut aussi montrer que dans ces deux modèles, pour tout entier B il existe un algorithme effectif d'élection et de nommage sur la famille des réseaux $(\mathbf{G}, \delta)$ tels que $|V(G)| \leq B < 2|V(G)|$. On a donc aussi un résultat analogue au Théorème 7.43 dans les modèle diffusion-à-port et diffusion-à-boîte.

7.7.3 Caractérisations

Comme Yamashita et Kameda, on souhaite caractériser les graphes $\mathbf{G}$ dans lesquels les problèmes d'élection et de nommage peuvent être résolus quel que soit la fonction d'étiquetage des ports choisie. Le théorème suivant est l'équivalent du Théorème 7.34 pour les modèles port-à-boîte, diffusion-à-port et diffusion-à-boîte. On rappelle qu'on dit qu'il existe un algorithme d'élection (ou de nommage) pour un graphe $\mathbf{G}$ s'il existe un algorithme qui permet de résoudre l'élection (ou le nommage) quel que soit la fonction d'étiquetage des ports de $\mathbf{G}$.

Théorème 7.49 *Pour tout graphe simple étiqueté $\mathbf{G}$, on a les caractérisations suivantes.*

1. *Il existe un algorithme d'élection et de nommage pour $\mathbf{G}$ dans le modèle port-à-boîte si et seulement si $Dir(\mathbf{G})$ est minimal pour les revêtements dirigés.*
2. *Il existe un algorithme de nommage pour $\mathbf{G}$ dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement si $Dir(\mathbf{G})$ est minimal pour les fibrations.*
3. *Il existe un algorithme d'élection pour $\mathbf{G}$ dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement si $Dir(\mathbf{G})$ est minimal pour les fibrations non-triviales.*

Preuve :

1. C'est un corollaire du Théorème 7.47 et de la Proposition 7.8.
2. C'est un corollaire du Théorème 7.47 et de la Proposition 7.5.
3. C'est un corollaire du Théorème 7.48 et de la Proposition 7.5.

□

On présente maintenant des étiquetages qui permettent de caractériser les graphes non-étiquetés admettant un algorithme d'élection ou de nommage dans les modèles considérés ici. Ces caractérisations en terme d'étiquetages sont dues à Yamashita et Kameda et vont nous être utiles au Chapitre 9.

Définition 7.50 *Un* étiquetage régulier symétrique *d'un graphe simple connexe G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un graphe régulier qui admet un couplage parfait si ses sommets ont un degré impair,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ est un graphe biparti régulier.*

On remarque qu'une coloration régulière (Definition 3.7) d'un graphe simple G est un étiquetage régulier symétrique de G.

Définition 7.51 *Un* étiquetage régulier *d'un graphe simple connexe G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un graphe régulier,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ est un graphe biparti régulier.*

On remarque qu'un étiquetage régulier symétrique d'un graphe simple G est un étiquetage régulier de G.

Définition 7.52 *Un* étiquetage semi-régulier *d'un graphe simple connexe G est un étiquetage ℓ de G tel que*

- *pour tout $i \in \ell(V(G))$, $G[i]$ est un graphe régulier,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i,j]$ est un stable, ou alors $G[i,j]$ est un graphe biparti semi-régulier.*

On remarque qu'un étiquetage régulier d'un graphe simple G est un étiquetage semi-régulier de G et qu'une coloration semi-régulière (Definition 5.6) de G est un étiquetage semi-régulier de G.

Dans [YK99], Yamashita et Kameda ont utilisé ces notions d'étiquetage afin d'obtenir une caractérisation des graphes admettant un algorithme d'élection ou de nommage dans les modèles port-à-port, port-à-boîte, diffusion-à-port et diffusion-à-boîte. On rappelle qu'un étiquetage ℓ d'un graphe simple est propre si $|\ell(V(G))| < |V(G)|$.

Théorème 7.53 ([YK99]) *Pour tout graphe simple G, on a les caractérisations suivantes.*

1. *Il existe un algorithme d'élection (ou de nommage) pour G dans le modèle port-à-port si et seulement si G n'admet pas d'étiquetage symétrique régulier propre.*

2. *Il existe un algorithme d'élection (ou de nommage) pour G dans le modèle port-à-boîte si et seulement si G n'admet pas d'étiquetage régulier propre.*
3. *Il existe un algorithme de nommage pour G dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement G n'admet pas d'étiquetage semi-régulier propre.*
4. *Il existe un algorithme d'élection pour G dans le modèle diffusion-à-port (resp. diffusion-à-boîte) si et seulement si pour tout étiquetage semi-régulier ℓ de G, il existe $v \in V(G)$ tel que $|\ell^{-1}(\ell(v))| = 1$.*

Grâce aux Théorèmes 7.34, 7.49 et 7.53, on peut caractériser les graphes simples non-étiquetés minimaux pour les revêtements dirigés symétriques (resp. revêtements dirigés, fibrations non-triviales, fibrations) comme les graphes à l'aide des étiquetages symétriques réguliers (resp. étiquetages réguliers, étiquetages semi-réguliers, étiquetages semi-réguliers). Ces caractérisations peuvent aussi être obtenues directement à partir des travaux de Boldi et Vigna [BV02a].

7.8 Conclusion et Perspectives

Dans ce chapitre, on a présenté une caractérisation des graphes admettant un algorithme de nommage ou d'élection utilisant des échange de messages quel que soit l'étiquetage des ports (Théorème 7.34). Les caractérisations présentées dans ce chapitre sont basées sur les notions de fibrations et de revêtements dirigés. On a par ailleurs étudié les connaissances initiales nécessaires pour résoudre ces problèmes. Ainsi, la connaissance initiale d'une borne sur le diamètre permet de résoudre élection et nommage sur un réseau $(\mathbf{G}, \delta)$ minimal pour les revêtements dirigés symétriques (Théorème 7.42). Par ailleurs, la connaissance initiale d'une borne serrée sur la taille permet d'obtenir un algorithme effectif de nommage et d'élection (Théorèmes 7.43). On a ensuite montré qu'on pouvait étendre ces résultats à d'autres modèles où les processus communiquent par échange de messages et qui avaient été étudiés par Yamashita et Kameda [YK99] (Théorème 7.49).

Les résultats obtenus dans ce modèle nous permettent de penser que les techniques présentées dans [GM03, GMM04] peuvent être adaptées dans les différents modèles étudiés dans ce chapitre. Il semble en particulier que lorsqu'aucune connaissance initiale n'est disponible, les classes de graphes simples reconnaissables dans ces modèles doivent vérifier les mêmes propriétés que celles présentées dans [GMM04]. Il semble aussi que l'utilisation des techniques présentés dans [God02b] permettent d'obtenir un algorithme de nommage auto-stabilisant dans un système synchrone dans lequel les processus communiquent par échange de messages.

Dans le travail réalisé avec Emmanuel Godard, Yves Métivier et Gerard Tel [CGMT07], on présente une caractérisation des familles de graphes dans lesquelles on peut résoudre un problème avec détection de la terminaison dans le modèle étudié dans ce chapitre. Les résultats présentés dans [CGMT07] sont basés sur les résultats présentés dans ce chapitre et sur des généralisations des méthodes utilisées dans [GM02, MT00].

Dans le travail réalisé avec Shantanu Das et Nicola Santoro [CDS06], on s'intéresse à un problème où on ne doit briser totalement les «symétries» initiales du réseau, comme c'est le cas pour les problèmes de l'élection ou du nommage. Dans ce travail, on cherche à former des «paires» dans le réseau : chaque sommet doit avoir un partenaire (dont il est lui-même le partenaire) et il doit connaître les numéros de ports apparaissant sur un chemin le

reliant à son partenaire. Si la taille du graphe est paire et qu'on a un algorithme d'élection, il est possible pour le sommet élu de construire une carte du réseau et d'indiquer à chaque sommet quel est son partenaire. Cependant, il existe des graphes pour lesquels il n'existe pas d'algorithme d'élection et dans lesquels on peut former des paires (on peut par exemple considérer le graphe complet à deux sommets). Dans [CDS06], on étudie les graphes dans lesquels on peut résoudre ce problème en fonction de la connaissance initiale disponible (la topologie du graphe, la taille du graphe, une borne sur la taille). Un corollaire important des résultats présentés dans [CDS06] est que selon la connaissance initiale disponible, on ne peut pas résoudre le problème dans les mêmes graphes. En effet, il existe des graphes pour lesquels on peut résoudre ce problème en connaissant la topologie (resp. la taille) du graphe mais pour lesquels on ne peut pas résoudre ce problème en connaissant seulement la taille (resp. une borne sur la taille) du graphe. Les méthodes présentées dans ce chapitre sont utilisées dans [CDS06] afin de permettre à chaque sommet de reconstruire la base minimale du réseau. Une étude des propriétés combinatoires des revêtements dirigés symétriques est ensuite nécessaire pour obtenir les résultats présentés dans [CDS06].

Chapitre 8

Agents Mobiles

Sommaire

8.1 Introduction

Le concept d'agents mobiles a été développé afin de pouvoir résoudre des problèmes dans des environnements hétérogènes et dynamiques [BR05]. Le modèle étudié dans ce chapitre est assez général : on considère des agents qui peuvent migrer de places en places dans un système de navigation qui peut être modélisé par un graphe. Un agent mobile est une entité qui exécute un algorithme : il peut se déplacer d'une place à une autre (avec des données ainsi que son algorithme) à travers un canal de communication et il peut exécuter des calculs lorsqu'il se trouve sur une place (une place fournit des moyens de calculs à l'agent ainsi que des données). Ainsi, un système à agents mobiles est défini par :

- un réseau, ou de manière équivalente, un graphe simple étiqueté (où les sommets correspondent aux places) avec un étiquetage des ports,
- un ensemble d'agents,

– un placement initial des agents sur le graphe.

Le sytème est asynchrone : il n'y a pas d'horloge globale, les agents migrent d'une place à une autre en un temps fini mais arbitraire et les agents exécutent les instructions à des vitesses arbitraires. Un cas particulier de notre modèle est lorsque le réseau et les agents mobiles sont anonymes. Un algorithme pour un système à agents mobiles est défini par un algorithme que chaque agent exécute indépendamment des autres agents.

Des problèmes classiques étudiés dans ce cadre sont le rendez-vous, l'élection, l'exploration, la découverte de la topologie, etc. De nombreux résultats existent dans différents modèles [ABRS95, BFFS03b, BFFS06, BS94, DFNS05, DFNS06, DKP98, DFP03] [GKKZ06, KKR06].

Dans ce chapitre, on s'intéresse à la puissance de calcul d'un système à agents mobiles et on souhaite plus particulièrement la comparer avec celle d'un système distribué où les processus communiquent en échangeant des messages. Il n'est pas difficile de voir que tout algorithme pour agents mobiles peut être simulé par un algorithme distribué où les processus communiquent par échange de messages : cette simulation est présentée dans la Section 8.3 et a été pour la première fois mise en évidence par Barrière et al. dans [BFFS03a]. Dans la Section 8.4, on montre qu'on peut simuler un algorithme distribué où les processus communiquent par échange de messages dans un système à agents mobiles. On obtient ainsi un théorème montrant que les deux modèles ont la même puissance de calcul (Théorème 8.14).

La preuve d'équivalence présentée dans ce chapitre n'est pas très difficile à obtenir, mais ce résultat permet d'obtenir des corollaires intéressants dans un modèle à partir de résultats existants dans l'autre. Dans la Section 8.5, on montre que grâce à notre résultat d'équivalence et aux résultats présentés dans le Chapitre 7, on peut caractériser quels sont les systèmes à agents mobiles dans lesquels on peut résoudre le problème du rendez-vous. Les résultats obtenus par Das et al. [DFNS05, DFNS06] et par Barrière et al. [BFFS03b, BFFS06] deviennent ainsi des corollaires de résultats existants pour les systèmes distribués où les processus communiquent par échange de messages.

Les résultats présentés dans ce chapitre ont été obtenus en collaboration avec Emmanuel Godard, Yves Métiver et Rodrigue Ossamy ; ils ont été publiés dans [CGMO06]. On a obtenu d'autres résultats dans le cadre des systèmes à agents mobiles [Cha06] dans un modèle légèrement différent. Ils ne sont pas détaillés ici mais ils sont présentés dans la conclusion de ce chapitre.

8.2 Le Modèle

On présente dans cette section une présentation formelle d'un système à agents mobiles. Cette définition correspond à la présentation de tels systèmes donnés dans l'introduction et nous sera utile pour prouver de manière formelle notre résultat d'équivalence.

8.2.1 Systèmes à Agents Mobiles

Un système à agents mobiles est défini par :
– un ensemble $\mathbb{P}$ de *places d'exécution* (que l'on appellera *places* par la suite),
– un système de navigation $\mathbb{S}$,
– un ensemble $\mathbb{A}$ d'*agents mobiles* (que l'on appellera *agents* par la suite),

– une injection $\pi_0 : \mathbb{A} \to \mathbb{P}$ décrivant la position initiale de chaque agent,
– un étiquetage initial λ des places et des agents.

Remarque 8.1 *L'étiquetage λ des places (resp. des agents) permet de coder un ensemble de places anonymes (resp. d'agents anonymes), d'attribuer des identités ou des identités partielles aux différentes places (resp. aux différents agents), de distinguer certaines places (resp. certains agents), etc.*

Le système de navigation $\mathbb{S}$ est décrit par un graphe simple connexe G dont les sommets représentent les places d'exécution et les arêtes représentent des canaux de navigation bidirectionnels entre les places. Par la suite, on identifie les places et les sommets correspondants ainsi que les arêtes et les canaux de communication. De plus, les agents peuvent distinguer les différents canaux de communication incident à un sommet. Ainsi, le graphe G correspondant au système de navigation est muni d'un étiquetage des ports δ.

Le système est asynchrone : il n'y a pas d'horloge globale, les agents exécutent leurs instructions à une vitesse arbitraire, le temps que met un agent à traverser une arête est fini mais non-borné.

8.2.2 Algorithmes pour Systèmes à Agents Mobiles

On associe à chaque agent mobile un système de transitions qui interagit avec les places d'exécution et le système de navigation.

À tout moment de l'exécution, chaque place a un état appartenant à un ensemble récursif $Q_{\mathbb{P}}$ d'états (commun à toutes les places) et chaque agent a un état appartenant à un ensemble récursif d'états $Q_{\mathbb{A}}$ d'états (commun à tous les agents). L'état initial de chaque place $\mathbf{p}$ est $\lambda(\mathbf{p}) \in Q_{\mathbb{P}}$ et l'état initial de chaque agent $\mathbf{a}$ est $\lambda(\mathbf{a}) \in Q_{\mathbb{A}}$. Pour toute place $\mathbf{p}$, on note $\texttt{state}(\mathbf{p})$ l'état de $\mathbf{p}$ et pour tout agent $\mathbf{a}$, on note $\texttt{state}(\mathbf{a})$ l'état de $\mathbf{a}$.

Pour toute arête $\{v, v'\} \in E(G)$, on note $\mathbb{M}(v, v')$ (resp. $\mathbb{M}(v', v)$) l'ensemble des agents en train de migrer entre la place $\mathbf{p}$ (resp. $\mathbf{p}'$) correspondant au sommet v (resp. v') et la place $\mathbf{p}'$ (resp. $\mathbf{p}$) correspondant au sommet v' (resp. v). Initialement, tous les ensembles $\mathbb{M}(v, v')$ sont vides. On note $\mathbb{M}$ l'ensemble des ensembles $\mathbb{M}(v, v')$ d'agents en transit. Chaque agent $\mathbf{a}$ qui n'est pas en train de migrer entre deux places se trouve sur une place $\mathbf{p}$ et on note π la fonction qui associe à chaque agent qui n'est pas en train de migrer la place sur laquelle il se trouve. Excepté dans la configuration initiale, plusieurs agents peuvent se trouver sur la même place au cours de l'exécution.

Une transition associée à un agent $\mathbf{a}$ et une place $\mathbf{p}$ correspondant à un sommet $v \in V(G)$ est noté

$$(s, q, in) \vdash_{\mathbf{p}}^{\mathbf{a}} (s', q', out),$$

où $s, s' \in S_{\mathbb{A}}$, $q, q' \in Q_{\mathbb{P}}$ et $in, out \in [0, \deg_G(v)]$. Une telle transition a la signification suivante.

– si $in = 0$ et $out = 0$, alors π et $\mathbb{M}$ ne sont pas modifiés. Cette transition ne peut être effectuée que si $\pi(\mathbf{a}) = \mathbf{p}$, si l'état de $\mathbf{p}$ est q et si l'état de $\mathbf{a}$ est s. Lorsque cette transition est effectuée, l'état de $\mathbf{p}$ devient q' et l'état de $\mathbf{a}$ devient s'.
– si $in \neq 0$, alors $out = 0$ (cela représente une transition où l'agent $\mathbf{a}$ arrive sur la place $\mathbf{p}$). Cette règle ne peut être appliquée que si l'état de $\mathbf{p}$ est q, si l'état de $\mathbf{a}$ est s et si $\mathbf{a} \in \mathbb{M}(v', v)$ où v' est le voisin de v tel que $\delta_v(v') = in$. Lorsque cette

transition est appliquée, l'état de $\mathbf{p}$ devient q', l'état de $\mathbf{a}$ devient s', $\mathbf{a}$ est effacé de $\mathbb{M}(v', v)$ et $\pi(\mathbf{a}) = \mathbf{p}$.
- Si $out = 0$, alors $in = 0$ (cela représente une transition où l'agent $\mathbf{a}$ quitte la place $\mathbf{p}$). Cette règle ne peut être appliquée que si $\pi(\mathbf{a}) = \mathbf{p}$, si l'état de $\mathbf{p}$ est q et si l'état de $\mathbf{a}$ est s. Lorsque cette transition est appliquée, l'état de $\mathbf{p}$ devient q', l'état de $\mathbf{a}$ devient s', $\pi(\mathbf{a})$ n'est plus défini et $\mathbf{a}$ est ajouté à $\mathbb{M}(v, v')$ où v' est le voisin de v tel que $\delta_v(v') = in$.

8.2.3 Exécution

Une exécution d'un algorithme pour agents mobiles est défini par une suite $(\mathtt{state}_0, \mathbb{M}_0, \pi_0), (\mathtt{state}_1, \mathbb{M}_1, \pi_1), \ldots, (\mathtt{state}_i, \mathbb{M}_i, \pi_i), \ldots$ telle que :
- $\mathbb{M}_0 = \emptyset$,
- pour chaque agent $\mathbf{a}$, $\mathtt{state}_0(\mathbf{a}) = \lambda(\mathbf{a})$ est l'état initial de a,
- pour chaque place $\mathbf{p}$, $\mathtt{state}_0(\mathbf{p}) = \lambda(\mathbf{p})$ est l'état initial de $\mathbf{p}$,
- pour chaque étape i, il existe une unique place $\mathbf{p}$ et un unique agent $\mathbf{a}$ tel que
  - si $\mathbf{p}' \neq \mathbf{p}$, alors $\mathtt{state}_{i+1}(\mathbf{p}') = \mathtt{state}_i(\mathbf{p}')$,
  - si $\mathbf{a}' \neq \mathbf{a}$, alors $\mathtt{state}_{i+1}(\mathbf{a}') = \mathtt{state}_i(\mathbf{a}')$,
  - $(\mathtt{state}_{i+1}(\mathbf{a}), \mathtt{state}_{i+1}(\mathbf{p}), \mathbb{M}_{i+1}, \pi_{i+1})$ est obtenu à partir de $(\mathtt{state}_i(\mathbf{a}), \mathtt{state}_i(\mathbf{p}), \mathbb{M}_i, \pi_i)$ par une transition de la forme $(s, q, in) \vdash_{\mathbf{p}}^{\mathbf{a}} (s', q', out)$.

La *configuration* du système à l'étape i est $(\mathtt{state}_i, \mathbb{M}_i)$. Une configuration est *finale* si aucune transition ne peut être effectuée à partir de cette configuration. On suppose que dans une configuration finale, aucun agent n'est en train de migrer. Autrement dit, pour tout agent $\mathbf{a}$ pour toute place $\mathbf{p}$ correspondant à un sommet v, pour tout état $s \in Q_{\mathbb{A}}$, pour tout état $q \in Q_{\mathbb{P}}$ et pour tout entier $in \in [1, \deg(v)]$, il existe une transition $(s, q, in) \vdash_{\mathbf{p}}^{\mathbf{a}} (s', q', 0)$.

Étant donnée une exécution qui atteint une configuration finale, la *longueur* de l'exécution est la longueur de la séquence $(\mathtt{state}_0, \mathbb{M}_0, \pi_0), (\mathtt{state}_1, \mathbb{M}_1, \pi_1), \ldots, (\mathtt{state}_i, \mathbb{M}_i, \pi_i), \ldots$

Si une place $\mathbf{p}$ est la place initiale d'un agent $\mathbf{a}$ (i.e., $\pi_0(\mathbf{a}) = \mathbf{p}$), on dit que $\mathbf{p}$ est la *base* de $\mathbf{a}$. On suppose que le placement initial des agents est codé dans l'état des places, i.e., à partir de l'état d'une place, un agent qui est sur la place savoir si cette place est une base ou non. Cependant, en général, un agent ne peut pas savoir si une base est la sienne ou celle d'un autre agent.

Ainsi, un système à agents mobiles est défini par :

$$(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda),$$

un algorithme pour un tel système est défini par

$$\mathcal{A} = (\vdash_{\mathbf{p}}^{\mathbf{a}})_{\mathbf{a} \in \mathbb{A}, \mathbf{p} \in \mathbb{P}},$$

et une exécution ρ est défini par :

$$\rho = (\mathtt{state}_i, \mathbb{M}_i, \pi_i)_{i \geq 0}.$$

Remarque 8.2 *On considère généralement des algorithmes pour agents mobiles tels que deux agents $\mathbf{a}, \mathbf{a}'$ dans le même état qui se trouvent respectivement sur deux places $\mathbf{p}, \mathbf{p}'$*

dans le même état peuvent appliquer les mêmes transitions. Cependant, si $\mathbf{p}$ *est relié à plus de places que* $\mathbf{p}'$*, alors si l'agent* $\mathbf{a}$ *sort de* $\mathbf{p}$ *par le port* $\deg_G(v)$ *(où* v *est le sommet correspondant à* $\mathbf{p}$*), l'agent* $\mathbf{a}'$ *ne peut pas effectuer la même transition.*

Ainsi, pour tous agents $\mathbf{a}, \mathbf{a}'$ *et pour toutes places* $\mathbf{p}, \mathbf{p}'$ *correspondant respectivement à des sommets* v, v' *de même degré,* $\vdash_{\mathbf{p}}^{\mathbf{a}} = \vdash_{\mathbf{p}'}^{\mathbf{a}'}$*. Pour distinguer des agents ou des places correspondant à des sommets de même degré, on utilise l'étiquetage initial* λ *des agents ou des places.*

De cette manière, un algorithme $\mathcal{A}$ *n'a pas besoin d'être défini pour un système particulier, mais il suffit de décrire pour tout entier* d*, les transitions que peuvent effectuer les agents avec les places de degré* d*.*

Par la suite, sauf si c'est explicitement indiqué, on ne considère que des algorithmes respectant cette propriété.

Remarque 8.3 *Si on considère des systèmes à agents mobiles dont les liens de navigation préservent l'ordre de départ des agents (i.e., si deux agents partent successivement d'un même sommet à travers le même numéro de port, alors ils arriveront dans le même ordre), il faut voir chaque ensemble* $\mathbb{M}_{(v,v')}$ *comme une file, comme dans le cadre des systèmes distribués avec communication par échanges de messages.*

8.2.4 Détection de la Terminaison

Un agent $\mathbf{a}$ dans un état s qui se trouve sur une place $\mathbf{p}$ dont l'état est q est dit *passif* s'il ne peut exécuter aucune transition à partir de cette configuration, i.e., il n'existe aucune transition de la forme $(s, q, 0) \vdash_{\mathbf{p}}^{\mathbf{a}} (s', q', out)$. Dans une configuration finale, tous les agents sont passifs. Comme dans les chapitres précédents, on parle alors de terminaison *implicite*, puisque les agents ne savent pas nécessairement que tous les autres agents sont passifs.

Comme dans les chapitres précédents, on dit qu'un algorithme $\mathcal{A}$ pour agents mobiles résout un problème $\mathcal{P}$ avec détection de la terminaison sur une famille $\mathcal{F}$ de systèmes à agents mobiles s'il existe une fonction $res : Q_{\mathbb{P}} \to S$ et un ensemble $L_f \subseteq Q_{\mathbb{A}}$ tels que les propriétés suivantes sont toujours vérifiées pour tout système $S = (\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda) \in \mathcal{F}$.

- Toute exécution ρ de $\mathcal{A}$ sur S termine et l'étiquetage final $\mathtt{state}_\rho$ de $\mathbb{P}$ est tel que l'étiquetage $\mathbf{res} \circ \mathtt{state}_\rho$ est une solution de $\mathcal{P}$ sur S.
- Pour toute exécution ρ de $\mathcal{A}$ sur S, il existe un agent $\mathbf{a} \in \mathbb{A}$ et une étape i tels que $\mathtt{state}_i(\mathbf{a}) \in L_f$.
- Pour toute exécution ρ de $\mathcal{A}$ sur S, s'il existe un agent $\mathbf{a} \in \mathbb{A}$ et une étape i telle que $\mathtt{state}_i(\mathbf{a}) \in L_f$, alors pour tout $i' > i$, pour tout $p \in \mathbb{P}$ $\mathbf{res} \circ \mathtt{state}_{i'}(p) = \mathbf{res} \circ \mathtt{state}_i(p)$.

On remarque qu'une fois que la terminaison a été détectée par un agent, alors les valeurs correspondant aux étiquettes des places ne sont plus modifiées, mais les agents peuvent encore se déplacer. Ainsi, une fois qu'au moins un agent a détecté la terminaison, ces agents peuvent inscrire dans l'état de toutes les places du graphe que le résultat de l'exécution est calculé et ainsi, tous les agents peuvent savoir que l'algorithme est terminé.

8.3 Simuler un Algorithme pour Agents Mobiles par un Algorithme utilisant des Échanges de Messages

On montre ici comment implémenter un algorithme pour agents mobiles dans un système où les processus communiquent par échanges de messages. Pour cela, on montre que chaque pas de calcul effectué par un agent peut être simulé dans un système où les processus communiquent par échanges de messages. Les idées principales du résultat de cette section apparaissent dans les travaux de Barrière et al. [BFFS03a].

On considère un système à agents mobiles $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ et un algorithme $\mathcal{A} = (\vdash_{\mathbf{p}}^{\mathbf{a}})_{\mathbf{a}\in\mathbb{A},\mathbf{p}\in\mathbb{P}}$ implémenté sur ce système.

On considère le graphe (G, δ) correspondant au système de navigation $\mathbb{S}$ et on suppose que $|\mathbb{A}| = k$. On définit un étiquetage supplémentaire χ_{π_0} des sommets de G tel que pour tout $v \in V(G)$, $\chi_{\pi_0}(v) = 1$ s'il existe un agent $\mathbf{a} \in \mathbb{A}$ tel que $\pi_0(\mathbf{a}) = v$ et $\chi_{\pi_0}(v) = 0$ sinon.

À partir de $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$, on construit un système distribué où les processus communiquent par échanges de messages (P, C, λ') de la manière suivante. Le graphe correspondant au système de communication C est le graphe (G, δ) et à chaque sommet $v \in V(G)$ correspond un processus $p \in P$. On considère un nouveau symbole $\sharp$ et on définit l'étiquetage λ' des processus de P de la manière suivante. Pour tout sommet $v \in V(G)$ correspondant à une place $\mathbf{p} \in \mathbb{P}$ et à un processus $p \in P$, l'étiquette $\lambda'(p)$ contient l'étiquette $\lambda(\mathbf{p})$, la valeur de $\chi_{\pi_0}(v)$ et dans le cas où $\mathbf{p}$ est une base (i.e., $\chi_{\pi_0}(v) = 1$), l'étiquette de l'agent correspondant à $\mathbf{p}$. Autrement dit,$\lambda'(v) = \lambda'(p) = (\lambda(\mathbf{p}), 1, \lambda(\mathbf{a}))$ si $\mathbf{p}$ est une base et $\lambda'(v) = \lambda'(p) = (\lambda(\mathbf{p}), 0, \sharp))$ sinon.

On construit un algorithme $\mathcal{D}$ utilisant des échanges de messages tel que chaque exécution ρ de $\mathcal{D}$ simule une exécution ρ' de $\mathcal{A}$.

Dans l'état d'un processus, on code l'état de la place correspondante, la présence éventuelle d'agents mobiles et l'état des agents présents. Pour coder la présence d'agents mobiles, on utilise des jetons qui contiennent les états des agents correspondants.

La présence d'un agent $\mathbf{a}$ sur un sommet v est représenté par un jeton $t(\mathbf{a})$ situé sur le sommet v. Chaque jeton a une *base* qui correspond à la position initiale de l'agent correspondant. À chaque jeton $t(\mathbf{a})$, on associe un état qui est l'état de l'agent $\mathbf{a}$ correspondant.

L'état d'un processus p est donc un couple (q, A) où q est l'état de la place correspondante à $p(a)$ et A est un ensemble de couples $(t(\mathbf{a}), s)$ tels que $(t(\mathbf{a}), s) \in A$ si et seulement si $\mathbf{a}$ se trouve sur la place $\mathbf{p}$ dans l'état s.

Une transition

$$(s, q, in) \vdash_{\mathbf{p}}^{\mathbf{a}} (s', q', out)$$

de l'algorithme pour agent mobile se traduit par une transition de l'algorithme utilisant des échanges de messages de la forme

$$((q, A), in, m) \vdash_p ((q', A'), out, m')$$

où p est le processus correspondant à $\mathbf{p}$. Cette transition respecte les règles suivantes.

- Si $in = 0$ et $out = 0$, alors $m = m' = \perp$. Cette transition ne peut être appliquée que si $\texttt{state}(p) = (q, A)$ et $(t(\mathbf{a}), s) \in A$. Lorsque cette règle est appliquée, le nouvel état de p est $\texttt{state}(p) = (q', A')$ où $A' = A \setminus \{(t(\mathbf{a}), s)\} \cup \{(t(\mathbf{a}), s')\}$.

– Si $out \neq 0$ et $in = 0$, alors $m = \bot$. Cette transition ne peut être appliquée que si $\texttt{state}(p) = (q, A)$ et $(t(\mathbf{a}), s) \in A$. Lorsque cette règle est appliquée, le nouvel état de p est $\texttt{state}(p) = (q', A')$ où $A' = A \setminus \{(t(\mathbf{a}), s)\}$ et $m' = \{(t(\mathbf{a}), s')\}$.
– Si $in \neq 0$ et $out = 0$, alors $m' = \bot$. Cette transition ne peut être appliquée que si $m = \{(t(\mathbf{a}), s)\}$. Lorsque cette règle est appliquée, le nouvel état de p est $\texttt{state}(p) = (q', A')$ où $A' = A \cup \{(t(\mathbf{a}), s')\}$.

On note $\mathcal{D}_p$ l'algorithme obtenu en prenant l'union des règles obtenus à partir de $(s, q, in) \vdash^{\mathbf{a}}_{\mathbf{p}} (s', q', out)$ pour tous les agents $\mathbf{a}$ où $\mathbf{p}$ est la place correspondante à p. L'algorithme $\mathcal{D}$ est alors $(\mathcal{D}_p)_{p \in P}$.

Remarque 8.4 *Si on considère un algorithme $\mathcal{A}$ pour agents mobiles qui respecte les propriétés de la Remarque 8.2, on observe que l'algorithme $\mathcal{D}$ obtenu respecte la propriété de la Remarque 7.19.*

Par récurrence sur la longueur d'une exécution, il est facile de montrer que pour toute exécution de $\mathcal{D}$ sur (P, C, λ'), il existe une exécution de $\mathcal{A}$ sur $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ telle que pour toute étape i, pour toute place $\mathbf{p}$ et pour tout agent $\mathbf{a}$, les propriétés suivantes sont satisfaites pour le processus p correspondant à $\mathbf{p}$.

– $\texttt{state}_i(\mathbf{p}) = q$ si et seulement si $\texttt{state}_i(p) = (q, A)$,
– $\pi_i(\mathbf{a}) = \mathbf{p}$ et $\texttt{state}_i(\mathbf{a}) = s$ si et seulement si $\texttt{state}_i(p) = (q, A)$ et $(t(\mathbf{a}), s) \in A$.
– $\mathbf{a} \in \mathbb{M}_i(v, v')$ et $\texttt{state}_i(\mathbf{a}) = s$ si et seulement si $(t(\mathbf{a}), s) \in \mathtt{M}_i(v, v')$.

Ainsi, il est facile de voir que si toute exécution de $\mathcal{A}$ termine sur $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$, alors toute exécution de $\mathcal{D}$ termine sur (P, C, λ').

De plus, pour toute exécution ρ de $\mathcal{D}$ sur (P, C, λ'), il existe une exécution ρ' de $\mathcal{A}$ sur $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ telle que l'étiquetage final des places de $\mathbb{A}$ obtenu après ρ' est exactement l'étiquetage final des processus de P correspondants obtenu après ρ à projection sur la première composante près.

Par ailleurs, il est facile de voir que si $\mathcal{A}$ résout un problème $\mathcal{P}$ avec détection de la terminaison sur une famille $\mathcal{F}$ de systèmes à agents, alors $\mathcal{D}$ résout le problème $\mathcal{P}$ avec détection de la terminaison sur la famille de réseaux $\mathcal{F}'$ qui correspond à $\mathcal{F}$.

On a donc montré que tout algorithme pour agents mobiles pouvait être simulé par un algorithme utilisant des échanges de messages ; c'est ce qui est rappelé dans la proposition suivante.

Proposition 8.5 *Pour tout algorithme pour agents mobiles $\mathcal{A}$, l'algorithme $\mathcal{D}$ obtenu par la construction précédente utilise des échanges de messages et simule $\mathcal{A}$.*

De plus, si $\mathcal{A}$ permet de résoudre un problème $\mathcal{P}$ sur une famille de systèmes à agents mobiles $\mathcal{F}$ avec détection de la terminaison, alors l'algorithme $\mathcal{D}$ permet de de résoudre $\mathcal{P}$ sur la famille de réseaux correspondants $\mathcal{F}'$.

Remarque 8.6 *Si on considère un algorithme $\mathcal{A}$ pour agents mobiles qui nécessite que les liens de navigation de $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ préservent l'ordre de départ des agents, alors il suffit de s'assurer que les liens de communication de (P, C, λ') préservent l'ordre des messages pour que l'exécution simulée soit une exécution correcte de $\mathcal{A}$.*

8.4 Simuler un algorithme utilisant des échanges de messages par un algorithme pour agents mobiles

Dans cette section, on montre comment implémenter un algorithme $\mathcal{A}$ pour agents mobiles à partir d'un algorithme utilisant des échanges de messages $\mathcal{D}$ de telle sorte que pour tout réseau $(\mathbf{G}, \delta)$ et pour tout ensemble non-vide $\mathbb{A}$ d'agents mobiles, toute exécution de $\mathcal{A}$ par les agents de $\mathbb{A}$ sur le système de navigation correspondant à $(\mathbf{G}, \delta)$ simule une exécution de $\mathcal{D}$ sur le réseau $(\mathbf{G}, \delta)$.

Le principe de la simulation est le suivant : l'état de chaque place **p** correspondant à un sommet v dans $(\mathbf{G}, \delta)$ contient l'état du processus p correspondant à v (avec la même valeur initiale). Initialement, chaque agent va acquérir un territoire, de telle sorte que tous les sommets du graphe appartiennent à exactement un agent. Ensuite, chaque agent va être en charge de simuler tous les pas de calculs effectués par les processus correspondants aux places de son territoire.

L'état de chaque place **p** correspondant à un sommet v est de la forme $(mark, N, \mathbf{state}, \mathbf{inbuffer})$ où les différents champs ont les significations suivantes.

- $mark \in \{NT, T\}$ est un marqueur indiquant si la place a été visité ou non par un agent ; $mark = T$ si elle a été visité et $mark = NT$ sinon.
- N est un ensemble de couples $(p, mark)$ où $p \in [1, \deg(u)]$ et $mark \in \{NT, T\}$. Si le couple (p, T) apparaît dans N, alors l'arête $\{v, \delta_v(p)\}$ appartient à la forêt construite par les agents ; si le couple (p, NT) apparaît dans N, alors l'arête $\{v, \delta_v(p)\}$ n'appartient pas à la forêt ; s'il n'existe pas de couple $(p, mark) \in N$, alors on ne sait pas encore si l'arête $\{v, \delta_v(p)\}$ appartiendra ou non à la forêt.
- **state** est l'état du processus p correspondant à **p**.
- **inbuffer** est un ensemble de couples (p, m) où $p \in [1, \deg(u)]$ et m est un message de l'algorithme $\mathcal{D}$. Si un couple (p, m) apparaît dans **inbuffer**, cela signifie qu'un message m est arrivé par le port p et que ce message n'a pas encore été traité.

Initialement, l'étiquette de chaque place correspondant à un sommet v de $\mathbf{G} = (G, \lambda)$ est $(NT, \emptyset, \mathbf{state}, \emptyset)$ où **state** est l'état initial de v, i.e., $\mathbf{state} = \lambda(v)$. Cette étiquette initiale signifie que la place n'a pas été ajouté à l'arbre, qu'aucune arête incidente à v n'a été exploré par un agent et que v n'a reçu aucun message.

Remarque 8.7 *Si le comportement du réseau* $(\mathbf{G}, \delta)$ *assure que les canaux de communication préserve l'ordre des messages, alors il faut voir l'ensemble* **inbuffer** *comme une file.*

Le principe général de l'algorithme de simulation est l'Algorithme 3. Puisque les agents et les places peuvent être anonymes, il faut s'assurer que chaque agent peut naviguer dans le graphe, et en particulier qu'il peut retrouver sa base. Pour cela, on utilise une technique utilisée par Das et al. dans [DFNS05]. Les agents vont utiliser les numéros des ports pour stocker l'étiquetage d'un chemin qui va du sommet où ils se trouvent à leur base.

À chaque fois qu'un agent traverse une arête $\{u, v\}$ de u vers v, il stocke dans sa mémoire le numéro $\delta_v(u)$, afin de pouvoir retourner au sommet visité précédemment. Ainsi, chaque agent **a** va disposer d'une suite ordonnée de ports qu'il doit traverser pour rejoindre sa base ; cette suite est appelée *ePath* (chemin d'exploration) et est construite de la manière suivante. À chaque fois qu'un agent **a** traverse une arête $\{u, v\}$ de u vers v, si $\delta_u(v)$ est le premier élément de la suite *ePath* de **a**, alors il est effacé, et sinon $\delta_v(u)$ est

Algorithme 3 : Principe de l'algorithme de simulation de la Section 8.4

Étape 1 : Au départ, chaque agent $\mathbf{a}$ construit un arbre $T_{\mathbf{a}}$ par une traversée partielle du graphe. Lorsque tous les agents ont fini cette étape, on obtient une forêt couvrante du graphe dont chaque arbre contient exactement une base, qui est la base de l'agent qui a construit l'arbre correspondant.

Étape 2 : Chaque agent $\mathbf{a}$ exécute l'algorithme $\mathcal{D}$ sur les sommets de $T_{\mathbf{a}}$.
Cette exécution est constituée de rondes telles que durant chaque ronde, chaque sommet v de $T_{\mathbf{a}}$ est visité par $\mathbf{a}$ et si v doit exécuter des pas de calculs de $\mathcal{D}$, alors $\mathbf{a}$ en simule au moins 1 et au plus d (d est une constante supérieure ou égale à 1). Ainsi, $\mathbf{a}$ est en charge de simuler tous les pas de calculs que doivent exécuter les processus correspondant aux places qui apparaissent dans son arbre.

ajouté en tête de la suite *ePath* de v. Ainsi, à tout moment de l'exécution, la suite *ePath* d'un agent $\mathbf{a}$ permet à $\mathbf{a}$ de rejoindre sa base.

8.4.1 Calcul d'un arbre par un agent

On va expliquer comment chaque agent calcule l'arbre sur les sommets duquel il sera en charge de simuler les instructions de l'algorithme $\mathcal{D}$. L'algorithme présenté ici est le même que l'algorithme de Das et al. présenté dans [DFNS05]. Pour chaque place, lors de l'exécution de cette première étape, seuls les champs $mark$ et N de chaque place sont modifiés.

À partir de sa base, chaque agent $\mathbf{a}$ fait une traversée partielle du graphe en faisant un parcours en profondeur. À chaque fois qu'un agent $\mathbf{a}$ arrive sur un sommet v qui n'est pas marqué (i.e., $mark(v) = NT$), il le marque avec l'étiquette T et il marque l'arête $\{u, v\}$ par laquelle il est arrivée avec l'étiquette T, (i.e., il ajoute $(\delta_u(v), T)$ dans $N(u)$ et $(\delta_v(u), T)$ dans $N(v)$). Si $\mathbf{a}$ arrive depuis un sommet u sur un sommet v qui est déjà marqué (i.e., $mark(v) = T$) ou qui est une base, il retourne immédiatement sur le sommet u et il marque l'arête $\{u, v\}$ avec l'étiquette NT, (i.e., il ajoute $(\delta_u(v), NT)$ dans $N(u)$ et $(\delta_v(u), NT)$ dans $N(v)$). Lorsque toutes les arêtes incidentes au sommet courant u où se trouve un agent $\mathbf{a}$ ont été explorées (i.e., pour chaque $p \in [1, \deg(u)]$, il existe $(p, mark) \in N(u)$), alors $\mathbf{a}$ retourne au sommet par lequel il est arrivé sur u (qu'il retrouve à l'aide de $ePath(\mathbf{a})$). Lorsqu'un agent $\mathbf{a}$ est revenu à sa base (i.e., $ePath(\mathbf{a})$ est vide), alors l'agent $\mathbf{a}$ sait qu'il a fini de calculer son arbre.

L'Algorithme 4 présente une description de ces idées où les suites *ePath* sont représentées par des listes. La liste est vide est appelée *nil* et les primitives sur les listes utilisées sont *car* (pour accéder à la tête de la liste), *cdr* (pour accéder à la liste privée de sa tête), *cons* (pour ajouter un élément en tête d'une liste).

La proposition suivante résume quelques propriétés intéressantes de l'Algorithme 4.

Proposition 8.8 *Lors de toute exécution de l'Algorithme 4 sur un graphe* $\mathbf{G}$ *avec un étiquetage des ports* δ*, les propriétés suivantes sont vérifiées :*

- *si une arête* $\{u, v\}$ *est étiqueté* T*, alors* u *et* v *ont été étiquetés* T *par un même agent,*

Algorithme 4 : Construction d'un arbre par un agent **a**

début
  $mark(u) := T$;
  $ePath(\mathbf{a}) := nil$;
  répéter
    tant que $\exists p \in [1, \deg(u)]$ *tel que* $\nexists (p, mark) \in N(u)$ *où* u *est le sommet courant* **faire**
      $N(u) := N(u) \cup \{(p, T)\}$;
      quitte u par le port p pour arriver par le port q au sommet v ;
      $ePath(\mathbf{a}) := cons(q, ePath(\mathbf{a}))$;
      si $mark(v) = T$ *ou si* v *est une base* **alors**
        $N(v) := N(v) \cup \{(p, NT)\}$;
        quitte v par le port q pour retourner au sommet u ;
        $ePath(\mathbf{a}) := cdr(ePath(\mathbf{a}))$;
        $N(u) := N(u) \setminus \{(p, T)\} \cup \{(p, NT)\}$;
      sinon
        $mark(v) := T$;
        $N(v) := N(v) \cup \{(p, T)\}$;
    si $ePath(\mathbf{a}) \neq nil$ **alors**
      quitte u par le port $car(ePath(\mathbf{a}))$;
      $ePath(\mathbf{a}) := cdr(ePath(\mathbf{a}))$;
  jusqu'à $ePath(\mathbf{a}) = nil$ *et* $\forall p \in [1, \deg(u)], \exists (p, mark) \in N(u)$;
fin

– *l'ensemble des sommets et des arêtes de G étiquetés T par un agent* **a** *forment un arbre.*

De plus, lorsque l'exécution est terminée, on sait que chaque sommet a été étiqueté T par exactement un agent et que chaque agent a étiqueté sa base.

Ainsi, lorsque l'exécution de l'Algorithme 4 est terminée, l'ensemble des arêtes et des sommets étiquetés T forment une forêt contenant k arbres, où k est le nombre d'agents mobiles.

8.4.2 Coder les Échanges de Messages

On s'intéresse maintenant à la deuxième étape de l'Algorithme 3 où les agents simulent les pas de calculs que doivent exécuter les sommets de $(\mathbf{G}, \delta)$. Pour cela, on utilise les champs **state** et **inbuffer** de chaque place : **state** représente l'état du sommet v et **inbuffer** l'ensemble des messages parvenus au sommet v mais qui n'ont pas encore été traités.

On montre maintenant comment implémenter les transitions internes ainsi que les opérations d'envoi et de réception de messages à l'aide d'agents mobiles.

Simuler une transition interne. Un agent **a** qui se trouve sur un sommet u dans l'état s peut facilement simuler l'application d'une transition interne par u, i.e., l'application d'une transition de la forme

$$(s, 0, \perp) \vdash_p (s', 0, \perp).$$

En effet, puisqu'il suffit de modifier l'état de u, l'agent **a** doit juste modifier la valeur de $\mathbf{state}(u)$ de telle sorte que $\mathbf{state}(u) = s'$.

Simuler une transition de la forme envoyer m par le port p. On considère un agent **a** qui se trouve sur un sommet u dans l'état s et qui doit simuler une transition d'envoi d'un message m par le port p, i.e., une transition de la forme

$$(s, 0, \perp) \vdash_p (s', p, m).$$

On note v le voisin de u tel que $\delta_u(v) = p$. Pour simuler cette transition, l'agent **a** commence par modifier l'état de u de telle sorte que $\mathbf{state}(u) = s'$, puis il quitte le sommet u par le port p et lorsqu'il arrive en v par un port q, il ajoute le couple (q, m) dans $\mathbf{inbuffer}(v)$. Ensuite, l'agent **a** quitte le sommet v par le port q pour retourner sur le sommet u.

Simuler une transition de la forme recevoir m par le port q. On considère un agent **a** qui se trouve sur un sommet u dans l'état s et qui doit simuler une transition de réception d'un message m par le port q, i.e., une transition de la forme

$$(s, q, m) \vdash_p (s', 0, \perp).$$

Pour pouvoir simuler cette règle, l'agent **a** doit s'assurer qu'il existe un couple $(q, m) \in \mathbf{inbuffer}(u)$ (sinon, cela signifie que le message m attendu n'est pas encore arrivé par le port q). Si un tel couple existe, alors l'agent **a** supprime une occurrence du message (q, m) de $\mathbf{inbuffer}(u)$ et modifie l'état de u de telle sorte que $\mathbf{state}(u) = s'$.

Remarque 8.9 *L'exécution des transitions d'envoi de messages permet à un agent* **a** *de modifier le champ* **inbuffer**(u) *d'un sommet* u *qui n'appartient pas à* $T_{\mathbf{a}}$ *pour y ajouter de l'information. Cependant, si un agent* **a** *modifie la valeur de* **state**(u)*, alors cela implique que* u *est un sommet de* $T_{\mathbf{a}}$*.*

Remarque 8.10 *Si l'algorithme* $\mathcal{D}$ *nécessite que l'ordre des messages soient conservés, il suffit de modifier l'algorithme de simulation de la manière suivante. Pour chaque sommet* $u \in V(G)$*, on implémente l'ensemble* **inbuffer**(u) *sous forme d'une file. Lorsqu'un couple* (p, m) *est ajouté à* **inbuffer**(u)*, il est ajouté en queue de file. Lorsqu'un agent doit simuler la réception d'un message* m *par le port* q*, il doit s'assurer que le premier couple* (q, m') *dans la file est tel que* $m = m'$ *avant d'exécuter la règle et il doit supprimer ce couple de* **inbuffer**(u)*.*

Puisque pour chaque sommet v*, un seul agent* **a** *peut simuler l'envoi d'un message par* v*, il n'est pas nécessaire d'assurer que le système de navigation préserve l'ordre de départ des agents.*

La proposition suivante permet de voir que pour toute exécution de l'algorithme $\mathcal{A}$ sur un système de navigation correspondant à un réseau $(\mathbf{G}, \delta)$, il existe une exécution correspondante de $\mathcal{D}$ sur le réseau $(\mathbf{G}, \delta)$.

Proposition 8.11 *Pour toute exécution* ρ *de l'algorithme* $\mathcal{A}$ *pour agents mobiles sur le système de navigation correspondant au réseau* $(\mathbf{G}, \delta)$*, il existe une exécution* ρ' *de l'algorithme* $\mathcal{D}$ *telle que pour toute étape* i *de* ρ*, il existe une étape* j *de* ρ' *telle que les propriétés suivantes sont vérifiées.*

- *Pour tout sommet* $v \in V(G)$*,* $\mathtt{state}_j(v) = \mathbf{state}_i(v)$*.*
- *Pour toute arête* $\{v, v'\}$*, l'ensemble des messages en transit à l'étape* j *de* ρ *de* v *vers* v' *est le multi-ensemble* $\{m \mid \exists(\delta_{v'}(v), m) \in \mathbf{inbuffer}_i(v)\}$ *si l'agent* **a** *auquel appartient le sommet* v *n'est pas en train de simuler un envoi de message de* v *vers* v'*. Dans le cas où* **a** *est en transit entre* v *et* v' *et qu'il simule un envoi de message, il faut y ajouter le message* m *dont l'agent* **a** *est en train de simuler l'envoi à partir du sommet* v *par port* $\delta_v(v')$*.*

Preuve : Il suffit de considérer l'exécution ρ' de $\mathcal{D}$ construite de la manière suivante. Initialement, les propriétés sont vérifiées. On suppose qu'à l'étape i de ρ, les propriétés sont vraies pour l'étape j de ρ'. Si à l'étape $i+1$, aucun agent n'applique une règle de simulation, alors la propriété est vraie après l'étape $i+1$ de ρ' pour l'étape j de ρ. Si à l'étape $i+1$, un agent simule une transition sur un sommet v, alors on peut étendre ρ' de telle sorte qu'à l'étape $j+1$, cette transition soit effectuée par le sommet v. □

8.4.3 Simuler un algorithme utilisant des échanges de messages qui termine par un algorithme pour agents mobiles qui termine

Étant donné un algorithme $\mathcal{D}$ et un réseau $(\mathbf{G}, \delta)$ tel que toute exécution de $\mathcal{D}$ sur $(\mathbf{G}, \delta)$ termine, on sait d'après la Proposition 8.11 que pour toute exécution de $\mathcal{A}$ que $(\mathbf{G}, \delta)$, il existe une étape i à partir de laquelle les étiquettes des places ne sont plus modifiées. Par ailleurs, on sait que la configuration atteinte à l'étape i est une configuration finale de $\mathcal{D}$ sur $(\mathbf{G}, \delta)$ (puisque sinon, au moins un agent pourrait appliquer une transition sur l'un des sommets de son arbre).

Cependant, l'algorithme de simulation $\mathcal{A}$ présenté précédemment repose sur le fait que les agents parcourent leurs arbres indéfiniment. On va maintenant montrer qu'on peut modifier $\mathcal{A}$ pour s'assurer que toute exécution de $\mathcal{A}$ sur $(\mathbf{G}, \delta)$ termine dans une configuration finale où les étiquettes des places correspondent aux états des sommets correspondants dans une configuration finale de $\mathcal{D}$ sur $(\mathbf{G}, \delta)$.

Cas particulier où $\mathcal{D}$ permet de détecter la terminaison

On considère d'abord le cas où l'algorithme $\mathcal{D}$ résout un problème $\mathcal{P}$ sur une famille $\mathcal{F}$ de réseaux avec détection de la terminaison. On va montrer que les agents peuvent alors eux aussi détecter la terminaison de l'algorithme.

On note L l'ensemble d'étiquette utilisée par $\mathcal{D}$ et on sait qu'il existe une fonction **res**, un ensemble d'étiquette $L_f \subseteq L$ tels que pour tout réseau $(\mathbf{G}, \delta) \in \mathcal{F}$, toute exécution ρ de $\mathcal{D}$ sur $(\mathbf{G}, \delta)$ termine et l'étiquetage $\mathbf{res} \circ \mathtt{state}$ de $\mathbf{G}$ est une solution de $\mathcal{P}$ sur $(\mathbf{G}, \delta)$. Par ailleurs, il existe une étape i de ρ et un sommet $v \in V(G)$ telle que $\mathtt{state}_i(v) \in L_f$ et pour tous $i' > i$ et $v' \in V(G)$, $\mathbf{res} \circ \mathtt{state}_{i'} = \mathbf{res} \circ \mathtt{state}_i$.

On modifie $\mathcal{A}$ de la manière suivante afin d'assurer que $\mathcal{A}$ permet de résoudre $\mathcal{P}$ sur $\mathcal{F}$ avec détection de la terminaison. Pour cela, à chaque fois qu'un agent **a** modifie l'état d'un sommet v, il vérifie si le nouvel état de v appartient à L_f. Dans un tel cas, l'agent **a** marque le sommet v pour indiquer que l'algorithme est terminé. Il utilise ensuite un algorithme similaire à l'algorithme de construction d'arbre présenté précédemment pour marquer tous les sommets du graphe afin d'indiquer que l'algorithme est terminé. Lorsqu'un agent ne peut plus marquer de sommet (il a fini son parcours en profondeur du graphe), il retourne à sa base et s'endort.

Ainsi, pour toute exécution ρ de $\mathcal{A}$ sur $(\mathbf{G}, \delta)$, il existe une étape i de ρ lors de laquelle l'état d'un sommet v appartient à L_f et par conséquent, il existe une étape $j > i$ lors de laquelle tous les sommets portent une marque indiquant que l'algorithme a calculé le résultat souhaité et où tous les agents sont endormis dans leurs bases et savent que le résultat est calculé. Ainsi, l'algorithme de simulation $\mathcal{A}$ permet de résoudre $\mathcal{P}$ sur $\mathcal{F}$ avec détection de la terminaison ; c'est ce qui est rappelé dans la proposition suivante.

Proposition 8.12 *Pour tout algorithme $\mathcal{D}$ utilisant des échanges de message, si l'algorithme $\mathcal{D}$ résout un problème $\mathcal{P}$ sur une famille de réseaux $\mathcal{F}$ avec détection de la terminaison, alors l'algorithme $\mathcal{A}$ pour agents mobiles décrit précédemment résout $\mathcal{P}$ sur $\mathcal{F}$ avec détection de la terminaison.*

Cas Général

Dans le cas où les sommets de $(\mathbf{G}, \delta)$ ne peuvent pas savoir si l'exécution de $\mathcal{D}$ est terminée, on ne peut pas utiliser la méthode précédente qui repose sur l'apparition sur un sommet d'une étiquette indiquant que l'algorithme $\mathcal{D}$ a calculé son résultat final.

Afin d'assurer l'arrêt des agents même s'ils ne peuvent pas détecter la terminaison, on considère que l'arbre de chaque agent **a** est un arbre enraciné dont la racine est **a**. On ajoute des champs dans l'étiquette des places de la manière suivante.

- Dans l'étiquette de chaque base v, on ajoute un booléen **finished** qui indique à l'agent **a** dont v est la base s'il doit encore simuler des pas de calculs sur les sommets de son arbre.

– Dans l'étiquette de chaque place v, on ajoute un entier **father** qui est le numéro du port permettant d'atteindre le père de v dans l'arbre. L'entier **father** de chaque base v est égal à 0 puisque v est la racine de l'arbre.

Initialement, pour chaque base v, **finished**(v) = FAUX et pour tout sommet v, **father**$(v) = 0$, puisque les arbres n'ont pas été construits et qu'aucun pas de l'algorithme n'a été simulé.

L'Algorithme 3 est alors modifiée de telle sorte que lors de la construction des arbres dans l'étape 1, les valeurs de **father**(v) est initialisée correctement et contient le numéro du port par lequel l'agent **a** qui a marqué v est arrivé sur ce sommet.

L'étape 2 de l'Algorithme 3 est aussi modifiée de façon à ce qu'un agent qui n'a plus aucun pas de calculs à simuler sur les sommets de son arbre s'arrête.

Pour cela, chaque agent **a** exécute l'étape 2 de l'algorithme $\mathcal{A}$ en rondes, où chaque ronde correspond à un parcours par **a** de son arbre. Au début de chaque ronde, l'agent **a** est toujours sur sa base v et il assigne la valeur VRAI à **finished**(v). À la fin de chaque ronde, l'agent **a** retourne à sa base v. Si lors d'une ronde, un agent **a** a simulé tous les pas de calculs possibles sur tous les sommets de son arbre et si la valeur de **finished**(v) est VRAI, alors **a** s'endort sur sa base. Dans le cas contraire, il a encore des pas de calculs à simuler et il passe à la ronde suivante.

Il se peut qu'un agent **a** soit endormi sur sa base et qu'un autre agent $\mathbf{a}'$ modifie l'étiquette d'un sommet v de l'arbre $T_{\mathbf{a}}$ de l'agent **a** pour ajouter un couple (p, m) à **inbuffer**(v). Dans ce cas là, il faut aller réveiller l'agent **a** pour lui signifier qu'il a encore des pas de calculs à effectuer. Il se peut aussi que lorsqu'un agent **a** revient sur sa base à la fin d'une ronde, il a simulé tous les pas de calculs possibles sur chacun des sommets de son arbre, mais qu'entre-temps un couple (p, m) a été ajouté à **inbuffer**(v) pour un sommet $v \in V(T_{\mathbf{a}})$. Le rôle du champ **finished** de la base de chaque agent si un couple (p, m) a été ajouté à **inbuffer**(v) pour un sommet v de l'arbre de **a**.

Ainsi, à chaque fois qu'un agent **a** simule une transition d'envoi de message m d'un sommet v vers un sommet v', **a** modifie l'état de **inbuffer**(v) comme précédemment, puis à l'aide des champs **father** de chaque sommet il va à la racine r de l'arbre contenant v' et il assigne la valeur FAUX à **finished**(r) (si un agent est endormi en r, il le réveille). Ensuite, l'agent **a** retourne sur le sommet v et continue d'exécuter l'algorithme de simulation.

Puisque les champs de `state`(v) et **inbuffer**(v) sont modifiées exactement de la même façon que dans l'Algorithme 3, la Proposition 8.11 est toujours vraie pour cette adaptation de l'algorithme et il suffit donc de prouver que toute exécution de l'algorithme $\mathcal{A}$ termine et que la configuration finale de chaque exécution de $\mathcal{A}$ est une configuration finale de $\mathcal{D}$.

Proposition 8.13 *Pour tout algorithme $\mathcal{D}$ utilisant des échanges de messages, il existe un algorithme $\mathcal{A}$ pour agents mobiles tels que $\mathcal{A}$ simule $\mathcal{D}$.*

Preuve : Étant donné un réseau $(\mathbf{G}, \delta)$ sur lequel tout exécution de $\mathcal{D}$ termine, on considère une exécution ρ de $\mathcal{A}$ sur $(\mathbf{G}, \delta)$ par un ensemble non-vides d'agents.

Puisque toute exécution de $\mathcal{D}$ termine sur $(\mathbf{G}, \delta)$, il existe d'après la Proposition 8.11 une étape i_1 de ρ telle que pour toute étape $i \geq i_1$ et pour tout sommet $v \in V(G)$, $\mathbf{state}_{i+1}(v) = \mathbf{state}_i(v)$ et $\mathbf{inbuffer}_i(v) = \emptyset$. Par conséquent, il existe une étape $i_2 \geq i_1$ telle que pour toute étape $i \geq i_1$ et pour toute base v, **finished**(v) = VRAI. Ainsi, il existe une étape $i_3 \geq i_2$ lors de laquelle tous les agents sont endormis sur leurs bases et l'exécution est donc terminée.

On remarque que pour toute ronde i de ρ et pour tout agent **a**, si **a** est sur sa base

v à la fin de la ronde i et que **finished**(v) = Vrai, alors ou bien pour tout $v' \in V(T_{\mathbf{a}})$, **inbuffer**$(v') = \emptyset$, ou bien il existe un agent $\mathbf{a}'$ qui se trouve sur un sommet de $T_{\mathbf{a}}$. Ainsi, puisque dans la configuration finale de ρ, chaque agent est sur sa base, on sait que pour tout sommet $v \in V(G)$, **inbuffer**$(v) = \emptyset$ et puisque tous les agents se sont endormis sur leurs bases, aucune transition interne ou d'envoi de messages ne peut être simulée sur un sommet $v \in V(G)$. La configuration finale de ρ correspond donc d'après la Proposition 8.11 à la configuration finale d'une exécution de $\mathcal{D}$ sur $\mathbf{G}$. □

8.5 Résultat d'Équivalence et Applications

On présente dans cette section le théorème principal du chapitre obtenu grâce aux propositions prouvées dans les Sections 8.3 et 8.4. On présente ensuite les corollaires obtenus à partir des résultats du Chapitre 7 pour les problèmes de rendez-vous et d'élection dans les systèmes à agents mobiles.

8.5.1 Résultat Principal

On rappelle qu'on associe à tout système à agents mobiles $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ dont le système de navigation est un graphe G avec un étiquetage des ports δ, un réseau $\mathbf{G}' = (G, \lambda')$ avec un étiquetage des ports δ où λ' est défini de la manière suivante. Pour tout sommet $v \in V(G)$ qui correspond à une place $\mathbf{p}$, $\lambda'(v) = (\lambda(\mathbf{p}), 1, \lambda(\mathbf{a}))$ si $\mathbf{p}$ est la base de l'agent $\mathbf{a}$ et $\lambda'(v) = (\lambda(\mathbf{p}), 0, \sharp))$ sinon.

Grâce aux Propositions 8.5, 8.12 et 8.13, on a donc montré les théorèmes suivants.

Théorème 8.14 *Il existe un algorithme $\mathcal{A}$ pour agents mobiles qui permet de résoudre un problème $\mathcal{P}$ sur un système à agents mobiles $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ dont le système de navigation est un graphe G avec un étiquetage des ports δ si et seulement s'il existe un algorithme $\mathcal{D}$ utilisant des échanges de messages qui permet de résoudre $\mathcal{P}$ sur le réseau $(\mathbf{G}', \delta)$ où $\mathbf{G}' = (G, \lambda')$ et λ' est l'étiquetage de G obtenu à partir de λ et de π_0.*

Théorème 8.15 *Il existe un algorithme $\mathcal{A}$ pour agents mobiles qui permet de résoudre un problème $\mathcal{P}$ (avec détection de la terminaison) sur une famille $\mathcal{F}$ de systèmes à agents mobiles si et seulement s'il existe un algorithme $\mathcal{D}$ utilisant des échanges de messages (avec détection de la terminaison) qui permet de résoudre $\mathcal{P}$ sur la famille $\mathcal{F}'$ de réseaux correspondants aux systèmes de $\mathcal{F}$.*

8.5.2 Élection et Rendez-vous pour Agents Mobiles

Dans le cadre des systèmes à agents mobiles, le but d'un algorithme d'élection est d'élire un agent qui prend l'étiquette élu alors que les autres sommets prennent l'étiquette non-élu. Ces étiquettes sont terminales et un agent qui a une de ces étiquettes ne peut donc pas en changer.

Le problème du *rendez-vous* a aussi été étudié dans ce cadre et le but d'un algorithme de rendez-vous est d'arriver dans une configuration où tous les agents sont réunis en un même sommet du graphe correspondant au système de navigation. On demande de plus que lors de l'exécution, chaque agent détecte qu'il se trouve sur le sommet de rendez-vous et qu'il ne bougera plus.

Ces deux problèmes sont équivalents puisqu'une fois qu'un agent a été élu, tous les agents peuvent se retrouver sur la base de l'agent élu. Réciproquement, une fois qu'un agent est arrivé sur le sommet où aura lieu le rendez-vous, il essaie de marquer la place où il se trouve : s'il est le premier à le faire, il prend l'étiquette ÉLU et sinon, il prend l'étiquette NON-ÉLU.

Ces deux problèmes et quelques autres problèmes équivalents ont été étudiés dans le modèle considéré dans ce chapitre [BFFS03b, BFFS06, DFNS05, DFNS06] ainsi que dans des modèles plus faibles [DFP03, GKKZ06, KKR06].

On remarque que le problème du rendez-vous revient à attribuer une étiquette particulière à un unique sommet du graphe (qui est le sommet où se fait le rendez-vous), i.e., à élire un sommet du graphe. Ainsi, d'après les Théorèmes 7.32 et 8.14, on a la caractérisation suivante des systèmes à agents mobiles dans lesquels on peut résoudre les problèmes de l'élection et du rendez-vous.

Corollaire 8.16 *Étant donné un système à agents mobiles* $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ *correspondant à un graphe* $\mathbf{G}' = (G, \lambda')$ *avec un étiquetage des ports* δ*, il existe un algorithme d'élection ou de rendez-vous pour* $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ *si et seulement si le graphe* $(Dir(\mathbf{G}'), \delta)$ *est minimal pour les revêtements dirigés symétrique.*

De même, on peut montrer grâce au Théorèmes 7.42 et 8.15 qu'il suffit de connaître une borne sur le diamètre du graphe pour pouvoir résoudre élection et nommage dans un système à agents mobiles qui vérifie les conditions du Corollaire 8.16.

Par ailleurs, grâce aux Théorèmes 7.43 et 8.15, on sait que la connaissance d'une borne serrée sur la taille du graphe permet ou bien d'élire un agent (ou de résoudre le rendez-vous) dans un système à agents mobiles $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$ ou de détecter qu'il n'existe pas d'algorithme d'élection ou de rendez-vous pour $(\mathbb{A}, \mathbb{P}, \mathbb{S}, \pi_0, \lambda)$.

Ainsi, les caractérisations obtenues par Das et al. [DFNS05, DFNS06] deviennent des corollaires des travaux présentés dans ce chapitre et dans le chapitre précédent. De la même manière, en utilisant les résultats de Flocchini et al. [FRS03], on peut caractériser les systèmes à agents mobiles où on peut résoudre élection et rendez-vous lorsque le système de navigation est muni d'un sens de la direction. Les résultats de Barrière et al. [BFFS03b, BFFS06] deviennent alors des corollaires de ces caractérisations.

8.6 Conclusion et Perspectives

Dans ce chapitre, on a montré que les systèmes à agents mobiles ont la même puissance de calcul que les systèmes distribués où les processus communiquent par échange de messages (Théorème 8.14). On a montré que ce résultat permettait de caractériser les systèmes à agents mobiles dans lesquels le problème du rendez-vous pouvait être résolu.

Dans [BFFS03a], Barrière et al. considère des systèmes à agents mobiles où chaque agent à une étiquette unique mais il n'existe pas d'ordre global sur ces étiquettes. Autrement dit, chaque agent à une couleur et son propre ordre sur l'ensemble des couleurs de tous les agents, mais les ordres utilisés par les différents agents peuvent être différents. De même, chaque port est étiqueté avec une couleur afin de permettre à chaque agent de distinguer les arêtes incidentes à chaque sommet, mais il n'existe pas d'ordre global sur les couleurs utilisées pour étiqueter les ports. Barrière et al. étudient les problèmes de l'élection et du rendez-vous dans ce modèle et ils exhibent des algorithmes effectifs

d'élection pour certaines classes de graphes. Un problème ouvert présenté dans [BFFS03a] est de déterminer s'il existe un algorithme d'élection pour la classe de tous les graphes.

Dans [Cha06], on considère un modèle a priori plus faible que celui de Barrière et al. En effet, on suppose de plus que lorsqu'un agent laisse un message sur un sommet, les autres agents ne peuvent pas le comprendre mais peuvent juste déterminer quel est l'auteur du message (i.e., sa couleur). On présente dans ce modèle un algorithme effectif d'élection pour la classe de tous les graphes ainsi qu'une caractérisation des systèmes à agents mobiles dans lesquels on peut résoudre l'élection. Cette caractérisation s'exprime à l'aide de la notion d'automorphismes «bien équilibrés» qui ont été introduits par Bougé dans [Bou88] pour étudier le problème de l'élection symétrique dans CSP. Il est important de noter que les conditions nécessaires présentées dans [Cha06] restent vraies dans le modèle étudié dans [BFFS03a] : on répond donc positivement à la question posée par Barrière et al.

Chapitre 9

Complexité

Sommaire

9.1 Introduction

Dans ce chapitre, on s'intéresse à la complexité de décider si un graphe donné admet un algorithme de nommage ou d'élection dans les différents modèles considérés dans les chapitres précédents.

Des résultats existent déjà pour les modèles étudiés dans le Chapitre 7 ; ils ont été obtenus par Boldi et Vigna [BV02a] et par Yamashita et Kameda [YK96c]. On rappelle qu'un graphe G admet un algorithme d'élection (ou le nommage) dans un des modèles utilisant des messages si cet algorithme permet de résoudre l'élection (ou le nommage) sur G quel que soit l'étiquetage des ports. Dans [BV02a], Boldi et Vigna montrent qu'on peut décider si un graphe est minimal pour les fibrations en temps polynomial en calculant la base minimale de ce graphe. De plus, une fois que cette base minimale est calculée, on peut vérifier qu'un graphe est minimal pour les fibrations non-triviales en s'assurant qu'il existe un sommet de la base minimale dont la fibre est triviale. Ainsi, d'après les Théorèmes 7.47 et 7.48, on sait qu'on peut vérifier en temps polynomial si on peut nommer ou élire dans les modèles diffusion-à-port et diffusion-à-boîte. Par ailleurs, Yamashita et Kameda ont montré dans [YK96c] qu'il est co-NP-complet de décider si un graphe G donné admet un algorithme d'élection dans le modèle port-à-port.

Il est naturel de se demander quelle est la complexité de décider si un graphe G donné admet un algorithme de nommage ou d'élection dans les autres modèles considérés dans ce mémoire. On montre dans ce chapitre que pour tous les modèles, à l'exception des modèles diffusion-à-port et diffusion-à-boîte, ces problèmes sont co-NP-complets.

Afin d'obtenir une preuve générale pour le plus grand nombre de modèles possibles, on introduit la notion d'étiquetage pseudo-régulier (Définition 9.1) et on montre dans la

Section 9.2 qu'il est co-NP-complet de décider si un graphe G donné admet un algorithme d'élection ou de nommage dans chacun des sept modèles (incluant le modèle port-à-port pour lequel ce résultat était déjà connu [YK96c]) étudiés dans les chapitres précédents où élection et nommage peuvent être résolus sur les mêmes graphes (Corollaire 9.8). Il faut noter que notre réduction pour montrer qu'il est NP-difficile de décider si un graphe G admet un algorithme d'élection (ou de nommage) dans chacun de ces modèles est totalement différente de la preuve de Yamashita et Kameda pour le modèle port-à-port.

Dans les Sections 9.3 et 9.4, on montre qu'il est co-NP-complet de décider si un graphe G admet un algorithme de nommage (ou d'élection) dans les deux modèles restants : les calculs locaux cellulaires sur les étoiles (Corollaire 9.19) et les calculs locaux cellulaires sur les arêtes non-étiquetées (Corollaire 9.23).

Les résultats présentés dans ce chapitre ont été obtenus en collaboration avec Daniël Paulusma et ont été publiés dans [CP06].

9.2 Étiquetages Pseudo-Réguliers

Afin de pouvoir obtenir des résultats de NP-complétude pour une classe de modèles la plus large possible, on introduit la notion d'*étiquetage pseudo-régulier*. Il faut noter que les étiquetages réguliers (Définition 7.51) et les colorations pseudo-régulières (Définition 5.6) sont des cas particuliers d'étiquetages réguliers.

Définition 9.1 *Un* étiquetage pseudo-régulier *d'un graphe simple non-étiqueté connexe G est un étiquetage ℓ de G tel que*
- *pour tout $i \in \ell(V(G))$, $G[i]$ est un graphe régulier,*
- *pour tout $i, j \in \ell(V(G))$ avec $i \neq j$, $G[i, j]$ est un stable, ou alors $G[i, j]$ admet un couplage parfait.*

Grâce à cette notion, on va montrer qu'il est co-NP-complet de décider si un graphe simple G admet un algorithme de nommage ou d'élection dans tous les modèles considérés dans les chapitres précédents où nommage et élection sont équivalents.

Dans la proposition suivante, on montre que toutes les couleurs d'un étiquetage pseudo-régulier d'un graphe simple connexe G apparaissent avec la même cardinalité dans G.

Proposition 9.2 *Étant donné un graphe simple connexe G et un étiquetage pseudo-régulier ℓ de G, il existe un entier constante q tel que pour tout $i \in \ell(V(G))$, $|\ell^{-1}(i)| = q$.*

Preuve : Étant donné un graphe simple connexe G, un étiquetage pseudo-régulier ℓ de G, et deux couleurs $i, j \in \ell(V(G))$. S'il existe deux sommets voisins $u, v \in V(G)$ tels que $\ell(u) = i$ et $\ell(v) = j$, $G[i, j]$ admet un couplage parfait et alors $|\ell^{-1}(i)| = |\ell^{-1}(j)|$. Puisque G est connexe, pour toutes couleurs $i, j \in \ell(V(G))$, $|\ell^{-1}(i)| = |\ell^{-1}(j)| = \frac{|V(G)|}{|\ell(V(G))|}$. □

Le problème du H-REVÊTEMENT est de décider si un graphe simple G est un revêtement simple du graphe simple H qui est fixé.

H-REVÊTEMENT
Instance : Un graphe simple G.
Question : Est-ce que G est un revêtement simple de H ?

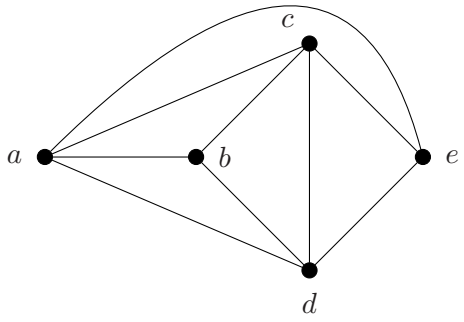


Fig. 29 – Le graphe K.

Pour prouver qu'il est NP-difficile de décider si un graphe simple G admet un étiquetage pseudo-régulier propre, on utilise la NP-complétude du problème du K-Revêtement, où K est le graphe simple obtenu en supprimant une arête du graphe complet à 5 sommets. Kratochvíl et al. ont montré dans [KPT98] que le problème du K-Revêtement est NP-complet. Le graphe K est représenté sur la Figure 29 : il contient deux sommets de degré 3 non-adjacents, notés b et e, et trois sommets de degrés 4 adjacents à tous les autres sommets, notés a, c et d.

La proposition suivante présente quelques propriétés que vérifie tout graphe simple qui est un revêtement simple de K.

Proposition 9.3 *Pour tout graphe simple G qui est un revêtement simple de K, on peut partitionner $V(G)$ en deux blocs B_1 et B_2 tels que les conditions suivantes sont vérifiées :*
- *il existe un entier k tel que $|B_1| = 2k$ et $|B_2| = 3k$,*
- *pour tout $u \in B_1$, $|N_G(u) \cap B_1| = 0$ et $|N_G(u) \cap B_2| = 3$,*
- *pour tout $u \in B_2$, $|N_G(u) \cap B_1| = 2$ et $|N_G(u) \cap B_2| = 2$.*

Preuve : On considère un graphe simple G qui est un revêtement simple de K à travers un homomorphisme φ. On sait d'après la Proposition 2.6 qu'il existe un entier k tel que pour tout $v \in V(K)$, $|\varphi^{-1}(v)| = k$.

Puisque K contient deux sommets de degré 3 non-adjacents et 3 sommets de degré 4, il suffit de considérer $B_1 = \varphi^{-1}(\{v \mid \deg_K(v) = 3\})$ et $B_2 = \varphi^{-1}(\{v \mid \deg_K(v) = 4\})$. □

Un graphe simple G qui vérifie les conditions de la Proposition 9.3 est appelé un *K-candidat*. Puisque ces conditions peuvent être vérifiées en temps polynomial, on peut supposer que toute instance du problème du K-Revêtement est un K-candidat.

Pour notre preuve de NP-complétude, on modifie tout K-candidat G de la manière suivante. Étant donné deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$, on remplace l'arête $\{u, v\}$ de G par une chaîne de $q \geq 1$ *diamants* comme décrit sur la Figure 30. Le graphe simple G' ainsi obtenu est le *graphe-diamant* de G *respectivement à* l'arête $\{u, v\}$. Pour tout $i \in [1, q]$, le sous-graphe $D_i = G'[\{a_i, b_i, c_i, d_i, e_i\}]$ est un *diamant* de G'.

Dans le lemme suivant, on montre que tout étiquetage pseudo-régulier d'un graphe-diamant est localement injectif sur tous les sommets de chaque diamant.

Lemme 9.4 *Étant donné un K-candidat G de taille $5k$ et deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$, on considère un graphe simple G' qui est*

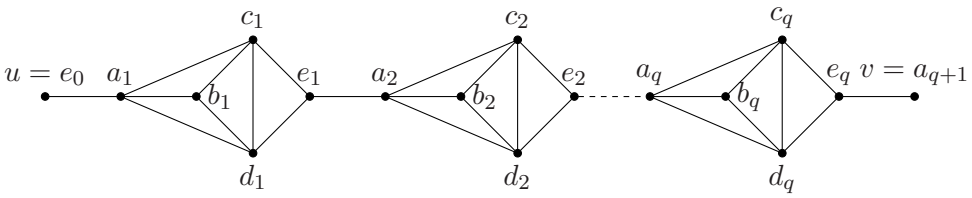


FIG. 30 – La chaîne de q diamants qui remplace l'arête $\{u, v\}$.

le graphe-diamant de G respectivement à $\{u, v\}$ dont les diamants sont $D_1, \ldots, D_q$ où $q > k + 2$ et $q + k$ n'est pas divisible par 2 ou 3.

Si ℓ est un étiquetage pseudo-régulier de G', alors pour tout $i \in [1, q]$, $|\ell(V(D_i))| = 5$, $\ell(e_{i-1}) \notin \ell(V(D_i) \setminus \{e_i\})$ et $\ell(a_{i+1}) \notin \ell(V(D_i) \setminus \{a_i\})$.

Preuve : On note $p = q + k$; on a alors $|V(G')| = 5p$ et p n'est divisible ni par 2, ni par 3. On considère un diamant D_i avec $i \in [1, q]$ et on rappelle que $u = e_0$ et $v = a_{q+1}$.

On suppose sans perte de généralité que $\ell(a_i) = 1$.

Propriété 1 : $\ell(b_i) \neq 1$

On prouve cette propriété par l'absurde. On suppose que $\ell(b_i) = 1$.

Si $\ell(c_i) = \ell(d_i) = 1$, alors $|\ell(V(G'))| = 1$, ce qui est impossible, puisque G' n'est pas un graphe régulier.

On suppose que $\ell(c_i) = 1$ et alors $\ell(d_i) \neq 1$; on note $\ell(d_i) = 2$. Puisque $G'[1, 2]$ admet un couplage parfait, il faut que $\ell(e_{i-1}) = \ell(e_i) = 2$. Par conséquent, $|\ell(V(G'))| = 2$ et d'après la Proposition 9.2, cela implique que 2 divise p, ce qui est impossible en raison de nos hypothèses sur p. Ainsi, $\ell(c_i) \neq 1$ et on suppose sans perte de généralité que $\ell(c_i) = 2$.

Par symétrie, on sait que $\ell(d_i) \neq 1$. Si $\ell(d_i) = 2$, alors puisque $G'[2, \ell(e_i)]$ admet un couplage parfait, cela implique que $\ell(e_i) = 1$ ou $\ell(e_i) = 2$. Si $\ell(e_i) = 1$ (resp. $\ell(e_i) = 2$), $\ell(a_{i-1}) = 2$ ou $\ell(a_{i+1}) = 2$ (resp. $\ell(a_{i+1}) = 1$) puisque $G'[1, 2]$ admet un couplage parfait. Dans tous les cas, $|\ell(V(G'))| = 2$, ce qui est impossible. Sans perte de généralité, on suppose donc que $\ell(d_i) = 3$. Puisque $G'[1]$ est régulier, $\ell(e_{i-1}) \neq 1$. Si $\ell(e_{i-1}) = i$ où $i \in \{2, 3\}$, alors $G'[1, 5 - i]$ n'admet pas de couplage parfait. Si $\ell(e_{i-1}) \notin \{1, 2, 3\}$, alors $G'[1, \ell(e_{i-1})]$ n'admet pas de couplage parfait.

Par conséquent, $\ell(b_i) \neq 1$ et sans perte de généralité, on suppose maintenant que $\ell(b_i) = 2$.

Propriété 2 : $\ell(c_i) \notin \{1, 2\}$

On prouve cette propriété par l'absurde. On suppose que $\ell(c_i) = 1$. Si $\ell(d_i) = 1$, alors $\ell(e_{i-1}) = \ell(e_i) = 2$ puisque $G'[1, 2]$ admet un couplage parfait ; mais alors $|\ell(V(G'))| = 2$, ce qui est impossible. Si $\ell(d_i) = 2$, alors $G'[2]$ est 1-régulier, et $\ell(e_i) = 1$, mais cela implique que $|\ell(V(G'))| = 2$, ce qui est impossible. Sans perte de généralité, on suppose donc que $\ell(d_i) = 3$. Si $\ell(e_i) \in \{1, 2, 3\}$, alors $|\ell(V(G'))| = 3$, ce qui est impossible en raison de nos hypothèses sur p. Si $\ell(e_i) \notin \{1, 2, 3\}$, il faut que $\ell(e_{i-1}) = \ell(e_i)$ puisque $G'[1, \ell(e_i)]$ admet un couplage parfait ; mais alors $G'[1, 2]$ n'admet pas de couplage parfait.

On suppose $\ell(c_i) = 2$. Par symétrie, on sait que $\ell(d_i) \neq 1$. Si $\ell(d_i) = 2$, $G'[1, 2]$ n'admet pas de couplage parfait. Si $\ell(d_i) \notin \{1, 2\}$, $\ell(e_i) = \ell(d_i)$ puisque $G'[2, \ell(d_i)]$ admet un couplage parfait ; mais alors $G'[1, 2]$ n'admet pas de couplage parfait.

Par conséquent, $\ell(c_i) \notin \{1,2\}$ et sans perte de généralité, on suppose maintenant que $\ell(c_i) = 3$.

Propriété 3 : $\ell(d_i) \notin \{1,2,3\}$

On prouve cette propriété par l'absurde. Par symétrie, on sait que $\ell(d_i) \neq 1$ et $\ell(d_i) \neq 2$.

Si $\ell(d_i) = 3$, alors $\ell(e_i) = 2$ puisque $G'[2,3]$ admet un couplage parfait ; mais alors $G'[1,3]$ n'admet pas de couplage parfait.

Par conséquent, $\ell(d_i) \notin \{1,2,3\}$ et sans perte de généralité, on suppose maintenant que $\ell(d_i) = 4$.

Propriété 4 : $\ell(e_i) \notin \{1,2,3,4\}$

On prouve cette propriété par l'absurde. Si $\ell(e_i) = 1$, alors $\ell(a_{i+1}) = 2$ puisque $G'[1,2]$ admet un couplage parfait ; mais alors $|\ell(V(G'))| = 4$, ce qui est impossible en raison de nos hypothèses sur p.

Si $\ell(e_i) = 2$, alors $\ell(a_{i+1}) = 1$ puisque $G'[1,2]$ admet un couplage parfait ; mais alors $G'[2,3]$ n'admet pas de couplage parfait.

Si $\ell(e_i) = 3$, alors $\ell(a_{i+1}) = 4$ puisque $G'[3,4]$ admet un couplage parfait ; mais alors $G'[2,3]$ n'admet pas de couplage parfait. Par symétrie, on sait que $\ell(e_i)$ ne peut pas non plus être égal à 4.

Par conséquent, $\ell(e_i) \notin \{1,2,3,4\}$ et sans perte de généralité, on suppose maintenant que $\ell(e_i) = 5$.

Propriété 5 : $\ell(e_{i-1}) \notin \{1,2,3,4\}$

On prouve cette propriété par l'absurde. Si $\ell(e_{i-1}) = 1$ (resp. $\ell(e_{i-1}) = 3$, $\ell(e_{i-1}) = 4$), alors $\ell(N_{G'}(e_{i-1})) = \{1,2,3,4\}$ (resp. $\ell(N_{G'}(e_{i-1})) = \{1,2,4,5\}$, $\ell(N_{G'}(e_{i-1})) = \{1,2,3,5\}$) ; ce qui est impossible puisque $\deg_{G'}(e_{i-1}) = 3$.

Si $\ell(e_{i-1}) = 2$, alors les deux voisins de e_{i-1} qui n'appartiennent pas à D_i doivent être étiquetés par 3 et 4 ; mais alors $G'[1,2]$ n'admet pas de couplage parfait.

Par conséquent, $\ell(e_i) \notin \{1,2,3,4\}$.

Propriété 6 : $\ell(a_{i+1}) \notin \{2,3,4,5\}$

On prouve cette propriété par l'absurde. Puisque $5 \notin \ell(N_{G'}(b_i))$ et que $\ell(b_i) = 2$, on sait que $\ell(a_{i+1}) \neq 2$.

Si $\ell(a_{i+1}) = 3$, alors les trois voisins de a_{i+1} qui n'appartiennent pas à D_i doivent être étiquetés par 1, 2 et 4 ; mais alors $G'[1,5]$ n'admet pas de couplage parfait. Par symétrie, on sait que $\ell(a_{i+1}) \neq 4$.

Si $\ell(a_{i+1}) = 5$, alors les trois voisins de a_{i+1} qui n'appartiennent pas à D_i doivent être étiquetés par 3 et 4, puisque $G'[5]$ est 1-régulier. On sait que a_{i+1} a un voisin v de degré 3 et par conséquent, $\ell(v) \in \{3,4\}$; sans perte de généralité, on suppose que $\ell(v) = 3$. Cependant, cela implique que $\ell(N_{G'}(v)) = \ell(N_{G'}(c_i)) = \{1,2,4,5\}$; ce qui est impossible, puisque $\deg_{G'}(v) = 3$.

Par conséquent, $\ell(a_{i+1}) \notin \{2,3,4,5\}$ □

Le lemme suivant décrit la propriété fondamentale qu'on utilise pour prouver qu'il est NP-difficile de décider si un graphe simple $\mathbf{G}$ admet un étiquetage pseudo-régulier.

Lemme 9.5 *Étant donné un K-candidat G de taille $5k$ et deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$, on considère un graphe simple G' qui est le graphe-diamant de G respectivement à $\{u, v\}$ dont les diamants sont $D_1, \ldots, D_q$ où $q > k + 2$ et $q + k$ n'est pas divisible par 2 ou 3.*

Si G' admet un étiquetage pseudo-régulier propre, alors G' est un revêtement simple de K.

Preuve : On note $p = q + k$; on a alors $|V(G')| = 5p$ et p n'est divisible ni par 2, ni par 3. On considère un étiquetage pseudo-régulier propre ℓ de G'.

On note H le graphe construit à partir des couleurs apparaissant dans G' de la manière suivante : $V(H) = \ell(V(G'))$ et $E(H) = \{\{i, j\} \mid \exists \{u, v\} \in E(G'), \ell(u) = i \text{ et } \ell(v) = j\}$. Puisque ℓ est un étiquetage pseudo-régulier propre de G', on sait que H n'est pas isomorphe à G'. On va montrer que ℓ induit un revêtement simple de G dans H et que H est un revêtement simple de K.

D'après la Proposition 9.3 et la construction de G', tout sommet de G' a un degré égal à 3 ou 4. De plus, pour tout $i \in [1, q]$, on sait d'après le Lemme 9.4, que $|\ell(V(D_i))| = 5$ et $\ell(e_{i-1}) \notin \ell(V(D_i) \setminus \{e_i\})$. Ainsi, pour tout sommet $v \in V(G')$ tel que $\ell(v) = \ell(a_i)$ (resp. $\ell(v) = \ell(c_i)$, $\ell(v) = \ell(d_i)$), on sait que $|\ell(v)| = |\ell(a_i)| = 4$ (resp. $|\ell(v)| = |\ell(c_i)| = 4$, $|\ell(v)| = |\ell(d_i)| = 4$) et par conséquent, $\deg_{G'}(v) = 4$.

Par ailleurs, s'il existe un sommet $v \in V(G')$ tel que $\ell(v) = \ell(b_i)$ (resp. $\ell(v) = \ell(e_i)$), alors $\ell(N_{G'}(v)) = \{\ell(a_i), \ell(c_i), \ell(d_i)\}$ (resp. $\ell(N_{G'}(v)) = \{\ell(a_{i+1}), \ell(c_i), \ell(d_i)\}$) et v doit donc avoir trois voisins de degré 4. D'après la Proposition 9.3 et la construction de G', on sait alors que $\deg_{G'}(v) = 3$.

Ainsi, pour tout $u \in V(D_i)$ et pour tout $v \in V(G')$ tel que $\ell(v) = \ell(u)$, ℓ est localement bijectif en v, i.e., $N_H(\ell(v)) = \ell(N_{G'}(v))$ et $|N_H(\ell(v))| = |N_{G'}(v)|$.

Puisque $q > k + 2$, il existe deux entiers $i, j \in [1, q - 1]$ et deux sommets $u \in V(D_i)$ et $v \in V(D_j)$ tels que $\ell(u) = \ell(v)$. D'après le Lemme 9.4, on sait que les diamants D_i et D_j sont différents. On choisit u et v de telle sorte que $i < j$ et qu'il n'existe pas deux sommets $u', v' \in V(D_i) \cup \ldots \cup V(D_{j-1})$ tels que $\ell(u') = \ell(v')$.

D'après le Lemme 9.4, on peut supposer sans perte de généralité que $\ell(a_i) = 1$, $\ell(b_i) = 2$, $\ell(c_i) = 3$, $\ell(d_i) = 4$, $\ell(e_i) = 5$ et on sait que $\ell(e_{i-1}) \notin \{1, 2, 3, 4\}$.

On suppose que $\ell(a_j) = 1$. En raison du choix de D_i et D_j, on sait que $\ell(e_{j-1}) \notin \{2, 3, 4\}$ et par conséquent, $\ell(e_{j-1}) = \ell(e_{i-1})$. De plus, on sait que pour tous $u, v \in V(D_i), \ldots, V(D_{j-1})$, $\ell(u) \neq \ell(v)$. Ainsi, $V(H) = \ell(V(G)) = \{\ell(u) \mid u \in V(D_k), k \in [i, j - 1]\}$. Par conséquent, pour tout sommet $v \in V(G')$, ℓ est localement bijectif en v. On a donc montré que G' est un revêtement simple de H.

De plus, on sait que $E(H)$ est l'ensemble constitué de l'arête $\{1, \ell(e_{j-1})\}$ et des arêtes $\{\ell(u), \ell(v)\}$ telles qu'il existe une arête $\{u, v\} \in V(G')$ dont les extrémités u et v appartiennent à $V(D_i) \cup \ldots \cup V(D_{j-1})$. Puisque tous les sommets de $V(D_i) \cup \ldots \cup V(D_{j-1})$ ont des couleurs différentes, H est isomorphe au graphe simple $G[V(D_i) \cup \ldots \cup V(D_{j-1})]$ auquel l'arête $\{a_i, e_{j-1}\}$ a été ajoutée. Si on considère l'homomorphisme φ de H dans K qui envoie chaque sommet a_i (resp. b_i, c_i, d_i, e_i) sur le sommet a (resp. b, c, d, e) de K, il est clair que H est un revêtement simple de K.

Par conséquent, G' est un revêtement simple de H et H est un revêtement simple de K : on sait d'après la Proposition 2.4 que G' est un revêtement de K.

On suppose maintenant que $\ell(a_j) \neq 1$. Si $\ell(e_{i-1}) = 5$, alors $\ell(a_{i+1}) = 1$ et par conséquent $j = i+1$ et on sait déjà que G' est un revêtement de K. On suppose donc sans perte de généralité que $\ell(e_{i-1}) = 6$.

On montre que $1 \notin \ell(V(D_j))$. On sait déjà que $\ell(a_j) \neq 1$ et que $1 \notin \ell(\{b_j, e_j\})$, puisque $\deg_{G'}(b_j) = \deg_{G'}(e_j) = 3$. Si $\ell(c_j) = 1$, alors $\ell(d_j) \in \{3, 4\}$ et par conséquent, $\ell(\{b_j, e_j\}) = \{2, 6\}$. Mais cela implique qu'il existe un sommet avec une étiquette 3 ou 4 qui est voisin d'un sommet étiqueté 6, ce qui est impossible. Par symétrie, on sait que $\ell(d_j) \neq 1$.

On montre que $2 \notin \ell(V(D_j))$. On sait déjà que $2 \notin \ell(\{a_j, c_j, d_j\})$ puisque $\deg_{G'}(a_j) = \deg_{G'}(c_j) = \deg_{G'}(d_j) = 4$. Si $\ell(b_j) = 2$, alors $1 \in \ell(\{a_j, c_j, d_j\})$, ce qui est impossible. Si $\ell(e_j) = 2$, alors ou bien $1 \in \ell(\{c_j, d_j\})$, ou bien $\ell(\{c_j, d_j\}) = \{3, 4\}$ et $\ell(a_j) = 1$.

On montre que $3 \notin \ell(V(D_j))$. On sait déjà que $3 \notin \ell(\{b_j, e_j\})$. Si $\ell(a_j) = 3$, alors ou bien $1 \in \ell(\{c_j, d_j\})$, ou bien $\ell(e_{j-1}) = 1$, ce qui est impossible en raison de notre choix de D_i et de D_j. Si $3 \in \ell(\{c_j, d_j\})$, alors $1 \in \ell(V(D_j))$.

Par symétrie, on sait que $4 \notin \ell(V(D_j))$.

On montre que $5 \notin \ell(V(D_j))$. On sait déjà que seuls les sommets b_j et e_j peuvent être étiquetés 5 et dans les deux cas, au moins une des étiquettes 3 ou 4 est l'étiquette d'un sommet de D_j. □

Dans le lemme suivant, on montre qu'un graphe simple G est un revêtement simple de K si et seulement si tout graphe-diamant de G est un revêtement simple de K.

Lemme 9.6 *Étant donné un graphe simple G qui contient deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$, on considère un graphe simple G' qui est le graphe-diamant de G respectivement à $\{u, v\}$.*

Le graphe simple G est un revêtement simple de K si et seulement si G' est un revêtement simple de K.

Preuve : On considère un graphe simple G contenant deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$ et un graphe diamant de G respectivement à $\{u, v\}$ dont les diamants sont $D_1, \ldots, D_q$.

On rappelle que $V(K) = \{a, b, c, d, e\}$ et que $E(K)$ est l'ensemble constitué des arêtes $\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{d, e\}$.

Si G est un revêtement simple de K à travers un homomorphisme φ, on sait que $\deg_K(\varphi(u)) = 3$ et $\deg_K(\varphi(v)) = 4$. Sans perte de généralité, on suppose que $\varphi(u) = e$ et $\varphi(v) = a$. On considère l'homomorphisme φ' de G' dans K, tel que pour tout $w \in V(G)$, $\varphi'(w) = \varphi(w)$ et pour tout diamant D_i de G', $\varphi(a_i) = a$, $\varphi(b_i) = b$, $\varphi(c_i) = c$, $\varphi(d_i) = d$, $\varphi(e_i) = e$. Puisque G est un revêtement simple de K à travers φ, il est clair que G' est un revêtement de K à travers φ.

Réciproquement, si G' est un revêtement de K à travers φ', alors il facile de voir que pour tout $i \in [1, q]$, $\varphi'(a_i) = \varphi'(a_{i+1})$ et $\varphi'(e_i) = \varphi'(e_{i+1})$. Par conséquent, $\varphi(a_1) = \varphi(a_{q+1}) = \varphi(v)$ et $\varphi(e_q) = \varphi(e_0) = \varphi(u)$. Ainsi, la restriction φ de φ' aux sommets de G est un homomorphisme tel que G est un revêtement de K à travers φ. □

On peut maintenant montrer le résultat principal de cette section. On montre qu'il est NP-complet de décider si un graphe simple G admet un étiquetage pseudo-régulier propre, un étiquetage régulier propre, un étiquetage régulier symétrique propre, une coloration

pseudo-régulière propre, une coloration régulière propre ou une coloration régulière parfaite propre.

Théorème 9.7 *Le problème de décider si un graphe simple G donné admet un étiquetage pseudo-régulier propre, (resp. un étiquetage régulier propre, un étiquetage régulier symétrique propre, une coloration pseudo-régulière propre, une coloration régulière propre, une coloration régulière parfaite propre) est* NP*-complet.*

Preuve : Puisqu'il suffit de deviner un étiquetage ℓ (et un ensemble de couplages parfaits dans certains cas) d'un graphe simple G donné tel que $|\ell(V(G))| < |(V(G)|$, puis de vérifier des conditions locales, tous ces problèmes sont clairement dans NP.

Pour montrer que ces problèmes sont NP-difficiles, on fait une réduction du problème du K-REVÊTEMENT qui est NP-complet [KPT98].

On considère un graphe simple G et on veut décider si G est un revêtement simple de K. Puisque les conditions de la Proposition 9.2 peuvent être vérifiées en temps polynomial, on peut supposer que G est un K-candidat et ainsi G contient deux sommets voisins $u, v \in V(G)$ tels que $\deg_G(u) = 3$ et $\deg_G(v) = 4$. On considère un graphe simple G' qui est le graphe-diamant de G respectivement à $\{u, v\}$ dont les diamants sont $D_1, \ldots, D_q$ où $q > k + 2$ et $q + k$ n'est pas divisible par 2 ou 3. On remarque que la taille de G' est bien polynomiale en la taille de G.

On montre que les assertions suivantes sont équivalentes :

(1) G est un revêtement simple de K,

(2) G' est un revêtement simple de K,

(3) G' admet une coloration régulière parfaite propre,

(4) G' admet une coloration régulière propre,

(5) G' admet une coloration pseudo-régulière propre,

(6) G' admet un étiquetage régulier symétrique propre,

(7) G' admet un étiquetage régulier propre,

(8) G' admet un étiquetage pseudo-régulier propre.

Les assertions (1) et (2) sont équivalentes d'après le Lemme 9.6. D'après la Proposition 2.11, on sait que (2) implique (3). De par la définition des étiquetages et colorations considérés, (2) implique (3), (3) implique (4), (4) implique (5), (4) implique (6), (5) implique (8), (6) implique (7) et (7) implique (8). De plus, d'après le Lemme 9.5, on sait que (8) implique (2).

On a donc montré qu'il est NP-difficile de décider si un graphe simple G admet un étiquetage pseudo-régulier propre (resp. un étiquetage régulier propre, un étiquetage régulier symétrique propre, une coloration pseudo-régulière propre, une coloration régulière propre, une coloration régulière parfaite propre). □

Le théorème précédent permet d'obtenir un corollaire sur la complexité de décider si un graphe simple G admet un algorithme d'élection ou de nommage dans les modèles considérés dans les Chapitres 2, 3, 5 et 7. Ces modèles sont exactement les modèles ou nommage et élection sont équivalents.

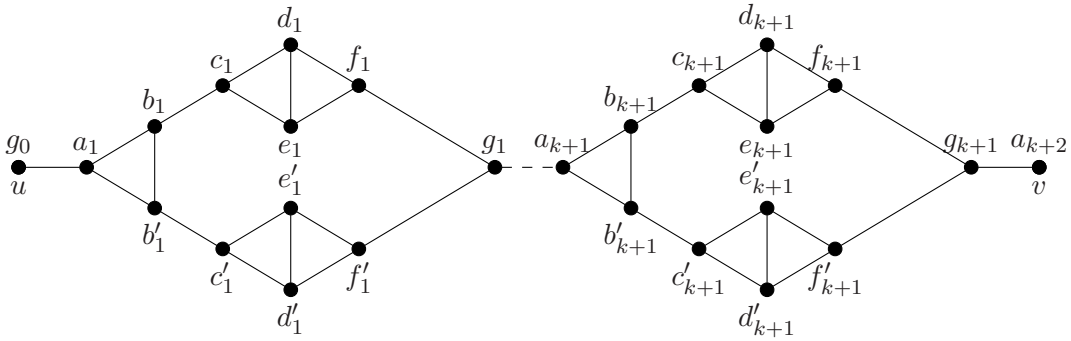


Fig. 31 – La chaîne de $k+1$ multi-diamants qui remplace l'arête $e_k = \{u, v\}$.

Corollaire 9.8 *Le problème de décider si on peut élire (ou nommer) dans un graphe simple G donné en utilisant des calculs locaux sur les étoiles fermées (resp. sur les arêtes étiquetées, sur les arêtes non-étiquetées) est co-*NP*-complet.*

*Le problème de décider si on peut élire (ou nommer) dans un graphe simple G en utilisant des messages dans le modèle port-à-port (resp. port-à-boîte) est co-*NP*-complet.*

Preuve : C'est un corollaire des Propositions 2.11, 3.8 et 5.7, des Théorèmes 2.21, 3.29 5.19 et 7.53 et du Théorème 9.7. □

9.3 Colorations Semi-Régulières

On s'intéresse maintenant au problème de décider si un graphe simple G admet une coloration semi-régulière propre, i.e., décider si un graphe simple G n'admet pas d'algorithme de nommage utilisant des calculs locaux cellulaires sur les étoiles. On montre aussi que notre réduction permet aussi de prouver qu'il est co-NP-complet de décider si un graphe simple G admet un algorithme d'élection utilisant des calculs locaux cellulaires sur les étoiles.

Pour montrer qu'il est NP-difficile de décider si un graphe simple G admet une coloration semi-régulière propre, on fait une réduction du problème du K_4-Revêtement où K_4 est le graphe complet à 4 sommets. Kratochvíl a montré que ce problème était NP-complet dans [Kra91].

Remarque 9.9 *Étant donné un graphe simple G, on sait que si G est un revêtement de K_4 à travers un homomorphisme φ, alors G est un graphe 3-régulier, puisque φ préserve le degré.*

Puisque cette propriété peut être vérifiée en temps polynomial, on sait que le problème du K_4-Revêtement est NP*-complet même si on se restreint aux graphes simples 3-réguliers.*

Étant donné un graphe G, on note $e_1, \ldots, e_m$ les arêtes de G. On va modifier le graphe G de la manière suivante. Pour chaque $k \in [1, m]$, on remplace l'arête e_k par une chaîne de $k+1$ *multi-diamants* $D_1(k), \ldots, D_{k+1}(k)$, comme représenté sur la Figure 31. Le graphe G' ainsi obtenu est un *graphe-multi-diamants* de G et est noté G'. Les sommets du $i$ème multi-diamant de la chaîne qui remplace l'arête e_k sont notés

$a_i(k), b_i(k), b'_i(k), c_i(k), c'_i(k), d_i(k), d'_i(k), e_i(k), e'_i(k), f_i(k), f'_i(k), g_i(k)$. Lorsqu'aucune confusion n'est possible, on note a_i à la place de $a_i(k)$, b_i à la place de $b_i(k)$, etc.

Pour toute arête $e_k \in E(G)$, on note u l'extrémité de e_k qui est un voisin de a_1 dans G' et v l'extrémité de e_k incidente au sommet g_{k+1} dans G'. On note alors $g_0 = u$ et $a_{k+2} = v$.

On va montrer que G est un revêtement de K_4 si et seulement si G' n'admet pas de coloration semi-régulière propre.

Le lemme suivant permet de montrer que pour tous diamants, si les mêmes couleurs apparaissent à une extrémité de chaque diamant, alors les autres extrémités de chaque diamant ont les mêmes couleurs.

Lemme 9.10 *Pour toute coloration semi-régulière ℓ d'un graphe-multi-diamants G' et pour tout multi-diamants $D_i(k), D_j(k')$, $\ell(g_i(k)) = \ell(g_j(k'))$ et $\ell(a_{i+1}(k)) = \ell(a_{j+1}(k'))$ si et seulement si $\ell(g_{i-1}(k)) = \ell(g_{j-1}(k'))$ et $\ell(a_i(k)) = \ell(a_j(k'))$.*

Preuve : On considère une coloration semi-régulière ℓ de G' et quatre sommets $g_i(k)$, $g_j(k'), a_{i+1}(k), a_{j+1}(k')$ tels que $\ell(g_i(k)) = \ell(g_j(k'))$ et $\ell(a_{i+1}(k)) = \ell(a_{j+1}(k'))$.

Puisque ℓ est une coloration semi-régulière, $\ell(\{f_i(k), f'_i(k)\}) = \ell(\{f_j(k'), f'_j(k')\})$. Sans perte de généralité, on suppose que $\ell(f_i(k)) = \ell(f_j(k'))$ et que $\ell(f'_i(k)) = \ell(f'_j(k'))$. Par conséquent, $\ell(\{d_i(k), e_i(k)\}) = \ell(\{d_j(k'), e_j(k')\})$ et $\ell(c_i(k)) = \ell(c_j(k'))$. Ainsi, $\ell(b_i(k)) = \ell(b_j(k'))$ et par symétrie, $\ell(b'_i(k)) = \ell(b'_j(k'))$. Par conséquent, $\ell(a_i(k)) = \ell(a_j(k'))$ et $\ell(g_{i-1}(k)) = \ell(g_{j-1}(k'))$.

De la même manière, on obtient la réciproque. □

On montre dans la proposition suivante qu'un graphe simple G est un revêtement de K_4 si et seulement si tout graphe-multi-diamants de G est un revêtement de K_4.

Proposition 9.11 *Pour tout graphe simple G et pour tout graphe-multi-diamants G' de G, G est un revêtement de K_4 si et seulement si G' est un revêtement de K_4.*

Preuve : On considère un graphe simple G et un graphe-multi-diamants G' de G. On suppose que G est un revêtement de K_4 à travers un homomorphisme φ. Étant donnée une arête $e_k = \{u, v\} \in E(G)$ telle que $u = g_0$ et $v = a_{k+2}$, on étend φ en un homomorphisme φ' de G' de la manière suivante. On note $a = \varphi(u)$ et $b = \varphi(v)$ et on note c, d les deux autres sommets de K_4. Pour tout $i \in [1, k+1]$, on définit $\varphi'(a_i) = b$, $\varphi'(b_i) = c$, $\varphi'(b'_i) = d$, $\varphi'(c_i) = a$, $\varphi'(c'_i) = a$, $\varphi'(d_i) = d$, $\varphi'(d'_i) = c$, $\varphi'(e_i) = b$, $\varphi'(e'_i) = b$, $\varphi'(f_i) = c$, $\varphi'(f'_i) = d$ et $\varphi'(g_i) = a$. Cette extension est représentée sur la Figure 32. L'homomorphisme φ' ainsi construit est un homomorphisme localement bijectif de G' dans K_4.

Réciproquement, on suppose que G' est un revêtement de K_4 à travers un homomorphisme φ'. On considère l'homomorphisme φ de G dans K_4 tel que pour tout $v \in V(G)$, $\varphi(v) = \varphi'(v)$. D'après le Lemme 9.10, on sait que $\varphi(N_G(v)) = \varphi'(N_G(v'))$ et que $|\varphi(N_G(v))| = |\varphi'(N_G(v'))|$. Ainsi, puisque φ' est un homomorphisme localement bijectif, φ est un homomorphisme localement bijectif et G est un revêtement de K_4. □

Par conséquent, si un graphe G est un revêtement de K_4, alors tout graphe-multi-diamants G' de G est un revêtement de K_4 et il admet donc une coloration régulière parfaite propre qui est une coloration semi-régulière. On suppose maintenant que le graphe G n'est pas un revêtement de K_4 et on veut montrer que tout graphe-multi-diamants G' de G n'admet pas de coloration semi-régulière propre.

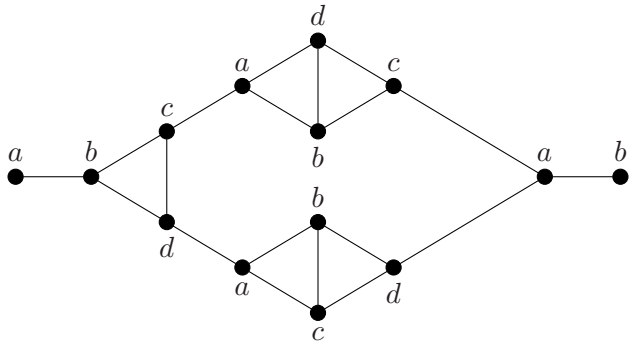


Fig. 32 – Étendre un homomorphisme localement bijectif de G dans K_4 en un homomorphisme de G' dans K_4.

Dans le lemme suivant, on montre que si G' n'est pas un revêtement de K_4, alors tous les sommets de chaque multi-diamant de G' ont des couleurs différentes.

Lemme 9.12 *Si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors pour toute coloration semi-régulière ℓ de G', pour tout k et pour tout $i \in [1, k+1]$, $|\ell(D_i(k))| = 12$.*

Preuve : On considère un graphe-multi-diamants G' qui n'est pas un revêtement de K_4 et une coloration semi-régulière ℓ de G.

On considère un multi-diamant D_i de G'. Puisque deux sommets voisins ont des couleurs différentes, on sait que c_i, d_i et e_i ont des couleurs différentes. Sans perte de généralité, on suppose que $\ell(d_i) = 1$, $\ell(e_i) = 2$ et $\ell(c_i) = 3$.

Propriété 1 : $\ell(b_i) \notin \{1,2,3\}$ et $\ell(b'_i) \notin \{\ell(c'_i), \ell(d'_i), \ell(e'_i)\}$.

On sait que $\ell(b_i) \neq 3$. On note $x = \ell(f_i)$ et $y = \ell(g_i)$. Si $\ell(b_i) = 1$, alors ou bien $\ell(a_i) = 2$ et $\ell(b'_i) = x$, ou bien $\ell(a_i) = x$ et $\ell(b'_i) = 2$. Dans le premier cas, $\ell(c'_i) = y$ et alors $\ell(f'_i) \in \{\ell(d'_i), \ell(e'_i)\}$, ce qui est impossible. Dans le second cas, $\ell(c'_i) = 3$, mais alors $\ell(d'_i) = \ell(e'_i) = 1$; ce qui est impossible. Par symétrie, on sait que $\ell(b_i) \neq 2$.

On a donc montré que $\ell(b_i) \notin \{1,2,3\}$ et par symétrie, on a $\ell(b'_i) \notin \{\ell(c'_i), \ell(d'_i), \ell(e'_i)\}$.

On suppose donc sans perte de généralité que $\ell(b_i) = 4$.

Propriété 2 : $\ell(f_i) \notin \{1,2,3,4\}$ et $\ell(c'_i) \notin \{\ell(b'_i), \ell(c'_i), \ell(d'_i), \ell(e'_i)\}$.

On sait que $\ell(f_i) \notin \{1,2\}$. Si $\ell(f_i) = \ell(b_i) = 4$, alors $\ell(V(G')) = \{1,2,3,4\}$ et ℓ définit un homomorphisme localement bijectif de G' dans K_4 ; ce qui est impossible. On suppose que $\ell(f_i) = \ell(c_i) = 3$. Puisque $4 \in \ell(N_{G'}(f_i)) = \ell(N_{G'}(c_i))$, on a $\ell(g_i) = 4$. Par conséquent, ou bien $\ell(f'_i) = \ell(a_i)$, ou bien $\ell(f'_i) = \ell(b'_i)$. Dans le premier cas, $\ell(b'_i) \in \ell(\{d'_i, e'_i\})$; ce qui est impossible d'après la Propriété 1. Dans le second cas, $\ell(c'_i) \in \ell(\{d'_i, e'_i\})$; ce qui est impossible.

On a donc montré que $\ell(f_i) \notin \{1,2,3,4\}$ et par symétrie, on a $\ell(f'_i) \notin \{\ell(b'_i), \ell(c'_i), \ell(d'_i), \ell(e'_i)\}$.

On suppose donc sans perte de généralité que $\ell(f_i) = 5$.

Propriété 3 : $\ell(a_i) \notin \{1,2,3,4,5, \ell(b'_i), \ell(c'_i), \ell(d'_i), \ell(e'_i), \ell(f'_i)\}$.

On sait que $\ell(a_i) \neq 4$. Puisque $4 \in \ell(N_{G'}(a_i))$ et $4 \notin \ell(N_{G'}(c_i) \cup N_{G'}(d_i))$, on sait que $\ell(a_i) \notin \{1,2\}$. On suppose que $\ell(a_i) \in \{3,5\}$. Par conséquent, $\ell(b'_i) \in \{1,2\}$ et sans perte de généralité, on suppose que $\ell(b'_i) = 1$. Mais alors, $\ell(b_i) = 4$ apparaît dans $\ell(N_{G'}(d_i))$; ce qui est impossible. Par symétrie, $\ell(a_i) \notin \{\ell(b'_i), \ell(c'_i), \ell(d'_i), \ell(e'_i), \ell(f'_i)\}$.

On suppose donc sans perte de généralité que $\ell(a_i) = 6$.

Propriété 4 : $\ell(b'_i) \notin \{1,2,3,4,5,6\}$.

On sait que $\ell(b'_i) \notin \{1,2,3,4,6\}$ puisque ℓ est une coloration semi-régulière. Si $\ell(b'_i) = 5$, alors les couleurs 1, 2, 4 et 6 appartiennent à $\ell(N_{G'}(b'_i)) = \ell(N_{G'}(f_i))$; ce qui est impossible puisque $\deg_{G'}(b'_i) = 3$.

On suppose donc sans perte de généralité que $\ell(b'_i) = 7$.

Propriété 5 : $\ell(c'_i) \notin \{1,2,3,4,5,6,7\}$.

On sait que $\ell(c'_i) \notin \{1,2,3,4,7\}$ puisque ℓ est une coloration semi-régulière et que $\ell(c'_i) \neq 6$ d'après la Propriété 3. Si $\ell(c'_i) = 5$, alors $\ell(g_i) = 7$ et $\ell(\{d'_i, e'_i\}) = \{1,2\}$. Mais alors, $\ell(f'_i) = 3$; ce qui est impossible puisque $7 \notin \ell(N_{G'}(c_i))$.

On suppose donc sans perte de généralité que $\ell(c'_i) = 8$.

Propriété 6 : $\ell(d'_i), \ell(e'_i) \notin \{1,2,3,4,5,6,7,8\}$.

On sait que $\ell(d'_i) \notin \{1,2,3,4,8\}$ puisque ℓ est une coloration semi-régulière, que $\ell(d'_i) \neq 7$ d'après la Propriété 2 et que $\ell(d'_i) \neq 6$ d'après la Propriété 3. Si $\ell(d'_i) = 5$, alors $\ell(e'_i) \in \{1,2\}$; ce qui est impossible puisque $\ell(c'_i) = 8 \notin \ell(N_{G'}(d_i)) \cup \ell(N_{G'}(e_i))$. Par symétrie, on sait que $\ell(e'_i) \notin \{1,2,3,4,5,6,7,8\}$.

Puisque e'_i et d'_i sont voisins, il sont des couleurs différentes, et on suppose donc sans perte de généralité que $\ell(d'_i) = 9$ et $\ell(e'_i) = 10$.

Propriété 7 : $\ell(f'_i) \notin \{1,2,3,4,5,6,7,8,9,10\}$.

On sait que $\ell(f'_i) \notin \{1,2,3,4,9,10\}$ puisque ℓ est une coloration semi-régulière, que $\ell(f'_i) \neq 8$ d'après la Propriété 1, que $\ell(f'_i) \neq 7$ d'après la Propriété 2 et que $\ell(f'_i) \neq 6$ d'après la Propriété 3. Si $\ell(f'_i) = 5$, alors les couleurs 1, 2, 9 et 10 appartiennent à $\ell(N_{G'}(f'_i)) = \ell(N_{G'}(f_i))$; ce qui est impossible puisque $\deg_{G'}(f'_i) = 3$.

On suppose donc sans perte de généralité que $\ell(f'_i) = 11$.

Propriété 8 : $\ell(g_i) \notin \{1,2,3,4,5,6,7,8,9,10,11\}$.

S'il existe un sommet $u \in V(D_i)$ tel que $\ell(u) = \ell(g_i)$, alors $\{5,11\} \subseteq \ell(N_{G'}(u))$; ainsi, $\ell(g_i) \notin \{1,2,3,4,5,7,8,9,10,11\}$. Si $\ell(g_i) = \ell(a_i) = 6$, alors $\{4,5,7,11\} \subseteq \ell(N_{G'}(g_i))$; ce qui est impossible puisque $\deg_{G'}(g_i) = 3$.

On a donc montré que pour tout multi-diamant D_i de G', $|\ell(V(D_i))| = 12$. □

Dans le lemme suivant, on montre que si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors une couleur apparaissant sur un sommet $g_i(k)$ ne peut apparaître dans un autre multi-diamant $D_j(k')$ que sur $g_j(k')$.

Lemme 9.13 *Si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors pour toute coloration semi-régulière ℓ de G', pour tous k, k' et pour tous $i \in [1, k+1]$, $j \in [1, k'+1]$, pour tout $u \in V(D_i(k)) \setminus \{g_i(k)\}$, $\ell(u) \neq \ell(g_j(k'))$.*

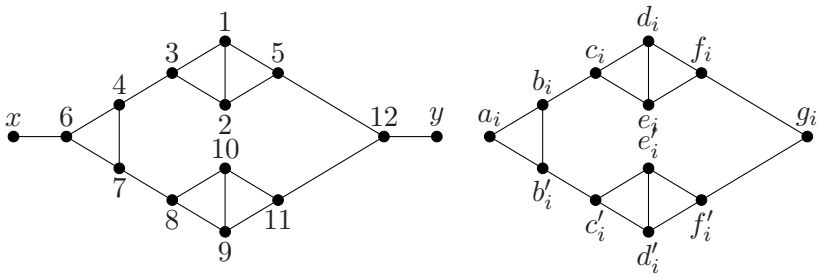


Fig. 33 – Les deux multi-diamants considérés dans la preuve du Lemme 9.14.

Preuve : On suppose qu'il existe un sommet $u \in V(D_i(k)) \setminus \{g_i(k)\}$ tel que $\ell(u) = \ell(g_j(k'))$. Puisque $u \in V(D_i(k)) \setminus \{g_i(k)\}$, u a deux voisins $v, w \in V(D_i(k))$ tels que $\{v, w\} \in E(G')$.

Par conséquent, $\{\ell(v), \ell(w)\} \cap \{\ell(f_j(k')), \ell(f'_j(k'))\} \neq \emptyset$; sans perte de généralité, on suppose que $\ell(v) = \ell(f_j(k'))$. Ainsi, $\ell(w) \in \{\ell(d_j(k')), \ell(e_j(k'))\}$; sans perte de généralité, on suppose que $\ell(w) = \ell(d_j(k'))$.

Par conséquent, $\ell(g_j(k')) = \ell(u) \in \ell(N_{G'}(w)) = \ell(N_{G'}(d_j(k'))) = \{\ell(c_j(k')), \ell(e_j(k')), \ell(f_j(k'))\}$; ce qui est impossible d'après le Lemme 9.12. □

Le lemme suivant est l'équivalent du lemme précédent pour les sommets $a_i(k)$: si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors une couleur apparaissant sur un sommet $a_i(k)$ ne peut apparaître dans un autre multi-diamant $D_j(k')$ que sur $a_j(k')$.

Lemme 9.14 *Si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors pour toute coloration semi-régulière ℓ de G', pour tous k, k' et pour tous $i \in [1, k+1]$, $j \in [1, k'+1]$, pour tout $u \in V(D_i(k)) \setminus \{a_i(k)\}$, $\ell(u) \neq \ell(a_j(k'))$.*

Preuve : On considère un graphe-multi-diamants G', une coloration semi-régulière ℓ de G' et un multi-diamant $D_j(k')$ de G'. D'après le Lemme 9.12, on peut supposer sans perte de généralité que $\ell(d_j(k')) = 1, \ell(e_j(k')) = 2, \ell(c_j(k')) = 3, \ell(b_j(k')) = 4, \ell(f_j(k')) = 5, \ell(a_j(k')) = 6, \ell(b'_j(k')) = 7, \ell(c'_j(k')) = 8, \ell(d'_j(k')) = 9, \ell(e'_j(k')) = 10, \ell(f'_j(k')) = 11, \ell(g_j(k')) = 12$, comme représenté à gauche sur la Figure 33. On note par ailleurs $x = \ell(g_{i-1})$ et $y = \ell(a_{i+1})$.

Par la suite, on note a_i le sommet $a_i(k)$, b_i le sommet $b_i(k)$, etc. Il suffit de montrer que pour tout sommet $v \in \{b_i, c_i, d_i, f_i, g_i\}$, $\ell(v) \neq 6$. D'après le Lemme 9.13, on sait déjà que $\ell(g_i) \neq 6$.

On suppose que $\ell(c_i) = 6$ (resp. $\ell(f_i) = 6$). Dans ce cas là, $\{\ell(d_i), \ell(e_i)\} \cap \{4, 7\} \neq \emptyset$. Sans perte de généralité, on suppose donc que $\ell(d_i) = 4$. Dans ce cas là, c_i (resp. f_i) et d_i doivent tous les deux avoir un voisin étiqueté par 7. D'après le Lemme 9.12, on a donc $\ell(e_i) = 7$. Puisque e_i doit alors avoir un voisin étiqueté 8, cela implique que $\ell(f_i) = 8$ (resp. $\ell(c_i) = 8$) ; ce qui est impossible, puisque le sommet f_i (resp. c_i), étiqueté 8 ne peut pas être voisin du sommet d_i étiqueté 4.

On suppose que $\ell(d_i) = 6$. Dans ce cas là, $\{\ell(c_i), \ell(f_i)\} \cap \{4, 7\} \neq \emptyset$. Sans perte de généralité, on suppose donc que $\ell(c_i) = 4$ (resp. $\ell(f_i) = 4$). Dans ce cas là, c_i (resp. f_i) et d_i doivent tous les deux avoir un voisin étiqueté par 7. D'après le Lemme 9.12, on a alors $\ell(e_i) = 7$. Puisque e_i doit alors avoir un voisin étiqueté 8, cela implique que $\ell(f_i) = 8$

(resp. $\ell(c_i) = 8$) ; ce qui est impossible, puisque le sommet f_i (resp. c_i), étiqueté 8 ne peut pas être voisin du sommet d_i étiqueté 6.

On suppose que $\ell(b_i) = 6$. Si $j \geq 2$, b_i doit avoir un voisin étiqueté $x = \ell(g_{j-1}(k'))$; ce qui est impossible d'après le Lemme 9.13. On suppose maintenant que $j = 1$. On sait que b_i a deux voisins étiquetés 4 et 7, et d'après le Lemme 9.12, ces deux sommets sont adjacents. Ainsi, $\ell(\{a_i, b'_i\}) = \{4, 7\}$ et sans perte de généralité, on suppose que $\ell(a_i) = 7$ et $\ell(b'_i) = 4$. Dans ce cas là, $\ell(c'_i) = 3$, $\ell(\{d'_i, e'_i\}) = \{1, 2\}$, $\ell(f'_i) = 5$, $\ell(g_i) = 12$ et $\ell(f_i) \in \{11, y\}$. On sait déjà qu'il est impossible que $\ell(f_i) = \ell(a_2(k')) = y$, puisque $k = 1$ et que $a_2(k')$ appartient à un diamant. Si $\ell(f_i) = 11$, alors $\ell(\{d_i, e_i\}) = \{9, 10\}$ et $\ell(c_i) = 8$; ce qui est impossible puisque c_i étiqueté 8 ne peut pas avoir être voisin de b_i étiqueté 6. □

Dans le lemme suivant, on montre que si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors un sommet qui n'est pas dans un multi-diamant de G' ne peut pas avoir la même couleur qu'un sommet d'un multi-diamant de G'.

Lemme 9.15 *Si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors pour toute coloration semi-régulière ℓ de G', pour tout sommet $v \in V(D_i(k))$ pour $k \in [1, m]$ et $i \in [1, k+1]$ et pour tout sommet $u \in V(G')$ tel que $\forall k, \forall i, u \notin V(D_i(k))$, $\ell(u) \neq \ell(v)$.*

Preuve : On considère un graphe-multi-diamants G', une coloration semi-régulière ℓ de G' et un sommet u n'apparaissant pas dans un multi-diamant de G'.

Pour tout sommet $u' \in N_{G'}(u)$, il existe k' tel que ou bien $u' = a_1(k')$, ou bien $u' = g_{k'+1}(k')$. Étant donné un sommet v d'un multi-diamant $D_i(k)$, il existe un sommet $v' \in V(D_i(k)) \setminus \{a_i(k), g_i(k)\}$. Si $\ell(u) = \ell(v)$, alors il existe un sommet $u' \in N_{G'}(u)$ tel que $\ell(u') = \ell(v')$; ce qui est impossible d'après les Lemmes 9.13 et 9.14. □

Dans le lemme suivant, on montre que pour tout graphe-multi-diamants G' qui n'est pas un revêtement de K_4, toute coloration semi-régulière de G' est une coloration régulière parfaite de G'.

Lemme 9.16 *Si un graphe-multi-diamants G' n'est pas un revêtement de K_4, alors pour toute coloration semi-régulière ℓ de G' et pour tout sommet $v \in V(G')$, $|\ell(N_{G'}(v))| = |N_{G'}(v)|$.*

Preuve : On considère un graphe-multi-diamants G' qui n'est pas un revêtement de K_4 et une coloration semi-régulière ℓ de G'.

On considère un sommet v apparaissant dans un multi-diamant $D_i(k)$. D'après le Lemme 9.12, on sait que $|\ell(N_{G'}(v))| = |N_{G'}(v)|$ si $v \notin \{a_i(k), g_i(k)\}$. D'après les Lemmes 9.13, 9.14 et 9.15, on a aussi $|\ell(N_{G'}(v))| = |N_{G'}(v)|$ si $v \in \{a_i(k), g_i(k)\}$.

On considère maintenant un sommet $u \in V(G')$ qui n'appartient pas à un multi-diamant de G'. On suppose qu'il existe deux sommets $v, v' \in N_{G'}(u)$ tels que $\ell(v) = \ell(v')$. D'après les Lemmes 9.13 et 9.14, il existe deux entiers k, k' tels que ou bien, $v = a_1(k)$ et $v' = a_1(k')$, ou bien $v = g_{k+1}(k)$ et $v' = g_{k'+1}(k')$. Puisque G' est le graphe-multi-diamants d'un graphe simple G, on sait que $k \neq k'$ et sans perte de généralité, on suppose que $k < k'$. Si on applique $k+1$ fois le Lemme 9.10, on a $\ell(a_{k+2}(k)) = \ell(a_{k+2}(k')$ (resp. $\ell(g_0(k)) = \ell(g_{k'-k}(k'))$). Dans le premier cas, on sait que $a_{k+2}(k)$ (resp. $g_0(k)$) n'appartient pas à un multi-diamant alors que $a_{k+2}(k')$ (resp. $g_{k'-k}(k')$) est un sommet d'un multi-diamant ; on a donc une contradiction d'après le Lemme 9.15. □

Dans le lemme suivant, on montre que si un graphe multi-diamant de G' n'est pas un revêtement de K_4, alors G' n'admet pas de coloration semi-régulière propre.

Lemme 9.17 *On considère un graphe-multi-diamants G' sans sommets isolés qui n'est pas un revêtement de K_4. Pour toute coloration semi-régulière ℓ de G', $|\ell(V(G'))| = |V(G')|$.*

Preuve : On considère un graphe-multi-diamants G' qui n'est pas un revêtement de K_4 et une coloration semi-régulière ℓ de G'.

On considère un sommet $u \in V(G')$ qui n'appartient pas à un multi-diamant de G'. On suppose qu'il existe un sommet $u' \in V(G')$ tel que $\ell(u) = \ell(u')$. D'après le Lemme 9.15, on sait que u' n'appartient pas non plus à un multi-diamant de G'.

Puisque G' ne contient pas de sommets isolés, il existe $v \in N_{G'}(u)$ et $v' \in N_{G'}(u')$ tels que $\ell(v) = \ell(v')$. D'après les Lemmes 9.13 et 9.14, i existe k, k' tels que $v = a_1(k)$ et $v' = a_1(k')$, ou $v = g_{k+1}(k)$ et $v' = g_{k'+1}(k')$. Puisque G' est le graphe-multi-diamants d'un graphe simple G, on sait que $k \neq k'$ et avec le même raisonnement que dans la preuve du Lemme 9.16, on aboutit à une contradiction.

Par conséquent, $|\ell^{-1}(\ell(u))| = 1$ et d'après le Lemme 9.16, on sait que ℓ est une coloration régulière parfaite de G'. Par conséquent, pour tout sommet $v \in V(G')$, $|\ell^{-1}(\ell(v))| = 1$. Ainsi, $|\ell(V(G'))| = |V(G')|$ et ℓ n'est pas une coloration semi-régulière propre de G'. □

On peut maintenant montrer le résultat principal de cette section. On montre qu'il est NP-complet de décider si un graphe simple G admet une coloration semi-régulière propre.

Théorème 9.18 *Le problème de décider si un graphe simple G donné admet une coloration semi-régulière propre est* NP*-complet.*

Preuve : Puisqu'il suffit de deviner un étiquetage ℓ d'un graphe simple G donné tel que $|\ell(V(G))| < |(V(G)|$, puis de vérifier des conditions locales pour s'assurer que ℓ est une coloration semi-régulière, le problème est clairement dans NP.

Pour montrer que ce problème est NP-difficile, on fait une réduction du problème du K_4-Revêtement qui est NP-complet [Kra91].

On considère un graphe simple G et on veut décider si G est un revêtement simple de K_4. D'après la Remarque 9.9, on peut supposer que G est 3-régulier. On considère un graphe simple G' qui est un graphe-multi-diamants de G. On remarque que la taille de G' est bien polynomiale en la taille de G.

D'après la Proposition 9.11, on sait que G est un revêtement de K_4 si et seulement si G' est un revêtement de K_4. De plus, si un graphe-multi-diamants G' est un revêtement de K_4, alors il admet une coloration régulière parfaite propre qui est une coloration semi-régulière. Réciproquement, si G' n'est pas un revêtement de K_4, on sait d'après le Lemme 9.17 que G' n'admet pas de coloration semi-régulière propre.

On a donc montré qu'il est NP-difficile de décider si un graphe simple G admet une coloration semi-régulière propre. □

Le théorème précédent permet d'obtenir un corollaire sur la complexité de décider si un graphe simple G admet un algorithme de nommage ou d'élection utilisant des calculs locaux cellulaires sur les étoiles.

Corollaire 9.19 *Le problème de décider si on peut élire (resp. nommer) dans un graphe simple G connexe donné en utilisant des calculs locaux sur les étoiles est co-*NP*-complet.*

Preuve : Pour le nommage, c'est un corollaire de la Proposition 4.9, du Théorème 4.22 et du Théorème 9.18.

On considère le problème de l'élection. On rappelle qu'on ne peut pas élire dans un graphe simple G en utilisant des calculs locaux cellulaires sur les étoiles si et seulement si G admet une coloration semi-régulière ℓ telle que pour tout $v \in V(G)$, $|\ell^{-1}(\ell(v))| > 1$.

Puisqu'il suffit de deviner un étiquetage ℓ d'un graphe simple G donné, puis de vérifier que pour tout $v \in V(G)$, $|\ell^{-1}(\ell(v))| > 1$ et de vérifier des conditions locales pour s'assurer que ℓ est une coloration semi-régulière, le problème de décider si un graphe simple G admet un algorithme d'élection utilisant des calculs locaux cellulaires sur les étoiles est clairement dans co-NP.

Pour montrer que ce problème est NP-difficile, comme dans la preuve du Théorème 9.18, on fait une réduction du problème du K_4-REVÊTEMENT qui est NP-complet [Kra91].

Étant donné un graphe simple G qui est 3-régulier, on considère un graphe-multi-diamants G' de G, dont la taille est polynomiale en la taille de G. Si G est un revêtement de K_4, alors G' est un revêtement de K_4 et admet une coloration régulière parfaite propre ℓ. Puisque ℓ est une coloration régulière parfaite propre, il existe un entier $q > 1$ tel que pour tout $v \in V(G')$, $|\ell^{-1}(\ell(v))| = q > 1$. Ainsi, si G est un revêtement de K_4, alors on ne peut pas résoudre le problème de l'élection sur G' en utilisant des calculs locaux cellulaires sur les étoiles. Réciproquement, si G n'est pas un revêtement de K_4, alors G' n'admet pas de coloration semi-régulière propre d'après le Lemme 9.17. On peut alors résoudre le problème de l'élection sur G' en utilisant des calculs locaux cellulaires sur les étoiles.

On a donc montré qu'il est NP-difficile de décider si un graphe simple G admet un algorithme d'élection utilisant des calculs locaux cellulaires sur les étoiles. □

9.4 Colorations Connexes

On s'intéresse maintenant au problème de décider si un graphe simple G admet une coloration connexe propre, i.e., décider si un graphe simple G n'admet pas d'algorithme de nommage utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées. On montre aussi que notre réduction permet aussi de prouver qu'il est co-NP-complet de décider si un graphe simple G admet un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

Pour prouver qu'il est NP-difficile de décider si un graphe simple G admet une coloration connexe propre, on utilise la NP-complétude du problème de la 2-COLORABILITÉ D'HYPERGRAPHE.

Un hypergraphe $(Q, \mathcal{S})$ est défini par un ensemble $Q = \{q_1, \ldots, q_m\}$ et un ensemble $\mathcal{S} = \{S_1, \ldots, S_n\}$ de sous-ensembles de Q. Une 2-coloration d'un hypergraphe $(Q, \mathcal{S})$ est une partition de Q en deux ensembles Q_1 et Q_2 tels que pour tout ensemble $S_j \in \mathcal{S}$, $Q_1 \cap S_j \neq \emptyset$ et $Q_2 \cap S_j \neq \emptyset$. Pour notre preuve de NP-complétude, on fait une réduction du problème de la 2-COLORABILITÉ D'HYPERGRAPHE qui est NP-complet (cf. [GJ79]).

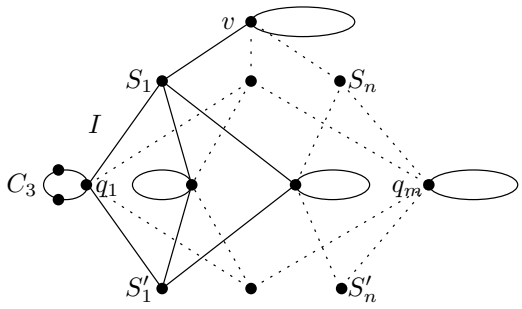


Fig. 34 – La construction du graphe $I'(Q, \mathcal{S})$.

2-Colorabilité d'Hypergraphe
Instance : Un hypergraphe $(Q, \mathcal{S})$.
Question : Est-ce qu'il existe une 2-coloration de $(Q, \mathcal{S})$?

Étant donné un hypergraphe $(Q, \mathcal{S})$, on lui associe son *graphe d'incidence* I qui est un graphe simple biparti défini par $V(I) = Q \cup \mathcal{S}$ et $E(I) = \{\{q, S\} \mid q \in S\}$.

Remarque 9.20 *On sait que le problème de la* 2-Colorabilité d'Hypergraphe *est* NP*-complet, même lorsqu'on considère seulement les hypergraphes dont le graphe d'incidence est connexe. Si le graphe d'incidence d'un hypergraphe $(Q, \mathcal{S})$ n'est pas connexe, décider si $(Q, \mathcal{S})$ est 2-coloriable revient à décider si deux hypergraphes $(Q', \mathcal{S}')$ et $(Q'', \mathcal{S}'')$ sont 2-coloriables où $Q = Q' \cup Q''$ et $\mathcal{S} = \mathcal{S}' \cup \mathcal{S}''$.*

Par ailleurs, le problème de la 2-Colorabilité d'Hypergraphe *est* NP*-complet, même si on suppose que pour tout hypergraphe $(Q, \mathcal{S})$, pour tous $S, S' \in \mathcal{S}$, $S \neq S'$. En effet, s'il existe deux ensembles égaux $S, S' \in \mathcal{S}$, une 2-coloration de $(Q, \mathcal{S})$ est une 2-coloration de $(Q, \mathcal{S} \setminus \{S'\})$ et réciproquement.*

Pour notre preuve de NP-complétude, on modifie le graphe d'incidence de tout hypergraphe $(Q, \mathcal{S})$ de la manière suivante.

On considère un ensemble $\mathcal{S}' = \{S'_1, \ldots, S'_n\}$ de sommets tels que pour tout $i \in [1, m]$ et pour tout $j \in [1, n]$, l'arête $\{q_i, S'_j\}$ existe si et seulement si l'arête $\{q_i, S_j\}$ existe. On ajoute ensuite un sommet v qui est voisin de tous les sommets de $\mathcal{S}$.

Puis, pour tout $i \in [1, m]$, on considère un cycle C_{q_i} de longueur $6i-3$ dont on identifie un sommet avec q_i. Enfin, on ajoute un cycle C_v de longueur $6m + 3$ dont on identifie un sommet avec v. Le graphe simple ainsi obtenu est noté $I'(Q, \mathcal{S})$. Un exemple de cette construction est présenté sur la Figure 34.

Dans le lemme suivant, on montre que si le graphe simple $I'(Q, \mathcal{S})$ admet une coloration connexe propre, alors cette coloration utilise 3 couleurs. C'est la propriété fondamentale qui va nous être utile dans notre preuve de NP-complétude.

Lemme 9.21 *On considère un hypergraphe $(Q, \mathcal{S})$ où $\mathcal{S} = \{S_1, \ldots, S_n\}$ dont le graphe d'incidence est connexe et tel que pour tous entiers distincts $j, k \in [1, n]$, $S_j \neq S_k$. Si le graphe $I'(Q, \mathcal{S})$ admet une coloration connexe propre ℓ, alors $|\ell(V(I'(Q, \mathcal{S})))| = 3$.*

Preuve : On considère un hypergraphe $(Q, \mathcal{S})$ dont le graphe d'incidence est connexe où $Q = \{q_1, \ldots, q_m\}$ et $\mathcal{S} = \{S_1, \ldots, S_n\}$ De plus, on suppose que pour tous entiers distincts $j, k \in [1, n]$, $S_j \neq S_k$. On considère une coloration connexe ℓ de $I'(Q, \mathcal{S})$.

Par définition, on sait que deux sommets voisins ont des couleurs différentes. On note $\ell(q_1) = 1$ et on note 2 et 3 les deux autres couleurs apparaissant dans Q_{q_1} (qui est de longueur 3). Si la couleur de chaque voisin de q_1 est 2 ou 3, alors $\ell(V(I'(Q,\mathcal{S}))) = \{1,2,3\}$ puisque $I'(Q,\mathcal{S})$ est connexe.

On suppose maintenant que q_1 a un voisin dont la couleur n'appartient pas à $\{2,3\}$. Par conséquent, tous les sommets dont la couleur est 1 ont un degré supérieur ou égal à 3. On va montrer que dans ce cas là, $|\ell(V(I'(Q,\mathcal{S})))| = |V(I'(Q,\mathcal{S}))|$, et alors ℓ n'est pas une coloration propre de $I'(Q,\mathcal{S})$.

Propriété 1 : Pour tout $w \in V(I'(Q,\mathcal{S})) \setminus V(C_{q_1}), \ell(w) \notin \{2,3\}$.

On montre cette propriété par l'absurde. On considère un sommet $w \in V(I'(Q,\mathcal{S})) \setminus V(C_{q_1})$ tel que $\ell(w) = 2$. On sait que $\ell(N_{I'(Q,\mathcal{S})}) = \{1,3\}$.

Si w est le sommet d'un cycle C_p distinct de p, où $p \in Q \setminus \{q_1\} \cup \{v\}$, alors w a un voisin u étiqueté 1. Puisqu'un sommet étiqueté 1 a un degré supérieur ou égal à 3, $u = p$. Cependant, w a aussi un voisin étiqueté 3 qui a un voisin u' étiqueté 1 ; ce qui est impossible car C_p contient au moins 6 sommets puisque $p \neq q_1$.

Si $w = p \in Q \setminus \{q_1\} \cup \{v\}$, alors w a un voisin $u \in V(C_p)$ de degré 2. Par conséquent, $\ell(u) = 3$; ce qui est impossible pour les mêmes raisons que précédemment.

Si $w = S \in \mathcal{S} \cup \mathcal{S}'$, alors w doit avoir un voisin p' étiqueté 3. De par la construction de $I'(Q,\mathcal{S})$, on sait que p' appartient à $Q \cup \{v\}$; ce qui est impossible pour les mêmes raisons que précédemment.

On a donc montré que les couleurs 2 et 3 apparaissent seulement sur le cycle C_{q_1}.

Propriété 2 : Pour tout $p \in Q \cup \{v\}, |\ell(V(C_p))| = |V(C_p)|$.

La condition est vérifiée pour $p = q_1$. Dans les autres cas, on montre la propriété par l'absurde. Pour cela, on utilise l'observation suivante qui peut être facilement montrée par récurrence.

Observation : On considère k couleurs distinctes $a_1, a_2, \ldots, a_k \in \ell(V(I'(Q,\mathcal{S})))$ telles que pour tout $j \in [1,k]$, $I'(Q,\mathcal{S})[a_j, a_{j+1}]$ n'est pas un stable. Pour tout sommet r_1 tel que $\ell(r_1) = a_1$, il existe un chemin $(r_1, r_2, \ldots, r_k)$ dans $I'(Q,\mathcal{S})$ tel que pour tout $j \in [1,k]$, $\ell(r_j) = a_j$.

On considère maintenant un sommet $u \in V(C_p) \setminus \{p\}$ tel que $\ell(u) = \ell(p)$. D'après la Propriété 1, aucun sommet de C_p n'est étiqueté 2 et par conséquent, tout chemin entre u et un sommet u' tel que $\ell(u') = 2$ contient le sommet p ; ce qui est impossible d'après l'observation précédente.

Pour tous sommets $u_1, u_2 \in V(C_p) \setminus \{p\}$ tels que $\ell(u_1) = \ell(u_2)$, il est facile de montrer par récurrence sur $\text{dist}_{C_p}(u_1, p)$ que $\text{dist}_{C_p}(u_1, p) = \text{dist}_{C_p}(u_2, p)$. Parmi tous les couples de sommets distincts (u_1, u_2) tels que $\ell(u_1) = \ell(u_2)$, on choisit le couple (u_1, u_2) tel que $\text{dist}_{C_p}(u_1) = k$ est maximum. Puisque u_1 et u_2 ne peuvent pas être voisins, u_1 a un voisin u_1' tel que $\text{dist}_{C_p}(u_1') = k+1$ et u_2 a un voisin u_2' tel que $\ell(u_2') = \ell(u_1')$. De par notre choix de (u_1, u_2), on sait que $u_1' = u_2'$ et cela implique que $|C_p| = 2\,\text{dist}_{u_1,p} + 2$, ce qui est impossible, puisque C_p est un cycle de longueur impaire.

On a donc montré que pour tout $p \in Q \cup \{v\}, |\ell(V(C_p))| = |V(C_p)|$.

Propriété 3 : Pour tout $p, q \in Q \cup \{v\}$ tels que $p \neq q$, $\ell(V(C_p)) \cap \ell(V(C_q)) = \emptyset$.

On montre cette propriété par l'absurde. On considère deux cycles distincts C_p, C_q tels que $\ell(V(C_p)) \cap \ell(V(C_q)) \neq \emptyset$. On sait que C_p et C_q ont des longueurs différentes et on

suppose donc que $|V(C_p)| < |V(C_q)|$. On sait d'après la Propriété 2 que les deux voisins de p qui appartiennent à C_p ont des couleurs différentes a et b. Si pour tout $u \in N_{I'(Q,\mathcal{S})}(p)$, $\ell(u) \in \{a, b\}$, alors $|\ell(V(I(Q,\mathcal{S})))| = |\ell(V(C_p))|$, ce qui est impossible puisque d'après la Propriété 2, $|\ell(V(I(Q,\mathcal{S})))| \geq |V(C_v)| = 6m + 3 > |V(C_p)|$. Par conséquent, pour tout sommet $u \in V(I(Q,\mathcal{S}))$ tel que $\ell(u) = \ell(p)$, le degré de u est supérieur ou égal à 3.

On considère une couleur $a \in \ell(C_p) \cap \ell(C_q)$. Si $a \neq \ell(p)$, on considère le sommet $u \in V(C_q)$ tel que $\ell(u) = a$. Il existe deux chemins disjoints dans C_p entre u et p. Puisque C_q contient au moins trois sommets de plus que C_p, on sait d'après l'observation précédente qu'il existe un sommet $u' \in V(C_q)\backslash\{q\}$ tel que $\ell(u) = \ell(p)$; ce qui est impossible puisque u est un sommet de degré 2. Par conséquent, on sait que $a = \ell(p) = \ell(q)$ et que $\ell(V(C_p)) \cap \ell(V(C_q)) = \{a\}$.

On considère un sommet $u_1 \in N_{C_p}(p)$ qui est un voisin de p dans C_p et on note u_2 le voisin de u_1 différent de p. Ainsi, $\ell(u_1)$ est la couleur d'un voisin de q qui appartient à $\mathcal{S} \cup \mathcal{S}'$ et il existe donc un sommet $q' \in Q \cup \{v\}$ tel que $\ell(q') = \ell(u_2)$. D'après la Propriété 2, on sait que $\ell(u_2) \neq \ell(p)$; mais en considérant les cycles C_p et $C_{q'}$ à la place de C_p et C_q, on obtient une contradiction.

Par conséquent, pour tout $p, q \in Q \cup \{v\}$ tels que $p \neq q$, $\ell(V(C_p)) \cap \ell(V(C_q)) = \emptyset$.

D'après les Propriétés 2 et 3, on sait que pour tous sommets $u, u' \in \bigcup_{p \in Q \cup \{v\}} V(C_p)$, $\ell(u) \neq \ell(u')$. De plus, si un sommet $S \in \mathcal{S} \cup \mathcal{S}'$ a la même couleur qu'un sommet $u \in V(C_p)$ où $p \in Q \cup \{v\}$, alors les couleurs des voisins de u appartenant à C_p apparaissent dans le voisinage de S qui est constitué de sommets de $Q \cup \{v\}$, ce qui est impossible. Par conséquent les couleurs des sommets de $\mathcal{S} \cup \mathcal{S}'$ sont disjointes des couleurs apparaissant sur tout cycle C_p où $p \in Q \cup \{v\}$.

Par ailleurs, puisque pour tous entiers différents $j, k \in [1, n]$, les ensembles S_j et S_k sont différents, on sait que pour tous sommets distincts $S_j, S_k \in \mathcal{S}$, $\ell(N_{I'(Q,\mathcal{S})}(S_j)) \neq \ell(N_{I'(Q,\mathcal{S})}(S_k))$. De même, on sait que pour tous sommets distincts $S'_j, S'_k \in \mathcal{S}'$, $\ell(N_{I'(Q,\mathcal{S})}(S'_j)) \neq \ell(N_{I'(Q,\mathcal{S})}(S'_k))$. Par ailleurs, pour tous sommets $S_j \in \mathcal{S}$ et $S'_k \in \mathcal{S}'$, $\ell(v) \in \ell(N_{I'(Q,\mathcal{S})}(S_j))$ et $\ell(v) \notin \ell(N_{I'(Q,\mathcal{S})}(S'_k))$. Ainsi, tous les sommets de $\mathcal{S} \cup \mathcal{S}'$ ont des couleurs différentes.

On a donc montré que tous les sommets de $I'(Q,\mathcal{S})$ ont des couleurs différentes et par conséquent, ℓ n'est pas une coloration connexe propre de $I'(Q,\mathcal{S})$. □

On peut maintenant montrer le résultat principal de cette section. On montre qu'il est NP-complet de décider si un graphe simple G admet une coloration connexe propre.

Théorème 9.22 *Le problème de décider si un graphe simple connexe G donné admet une coloration connexe propre est* NP*-complet.*

Preuve : Puisqu'il suffit de deviner un étiquetage ℓ d'un graphe simple G donné, puis de vérifier que des conditions locales pour s'assurer que ℓ est une coloration connexe, le problème est clairement dans NP.

Pour montrer que ce problème est NP-difficile, on fait une réduction du problème de la 2-Colorabilité d'Hypergraphe qui est NP-complet [GJ79].

Étant donné un hypergraphe $(Q,\mathcal{S})$ où $Q = \{q_1, \ldots, q_m\}$ et $\mathcal{S} = \{S_1, \ldots, S_n\}$, on peut supposer d'après la Remarque 9.20 que le graphe d'incidence de $(Q,\mathcal{S})$ est connexe et que

pour tous entiers distincts $j, k \in [1, n]$, $S_j \neq S_k$. On va montrer que $(Q, \mathcal{S})$ admet une 2-coloration si et seulement si $I'(Q, \mathcal{S})$ admet une coloration connexe propre.

On considère un hypergraphe $(Q, \mathcal{S})$ qui admet une 2-coloration $Q_1 \cup Q_2$. On définit un étiquetage ℓ de la manière suivante.

- $\ell(v) = 1$,
- pour tout $q \in Q_1$, $\ell(q) = 1$,
- pour tout $q \in Q_2$, $\ell(q) = 2$
- pour tout $S \in \mathcal{S} \cup \mathcal{S}'$, $\ell(S) = 3$,
- pour tout $p \in Q \cup \{v\}$, on étiquette C_p avec $1, 2, 3$ en respectant la couleur de p et de telle sorte que pour tout sommet u étiqueté 1 (resp. 2, 3), les couleurs 2 et 3 (resp. 1 et 3, 1 et 2) apparaissent dans $\ell(N_{C_p}(u)$; ce qui est possible puisque pour tout $p \in Q \cup \{v\}$, $|V(C_p)|$ est divisible par 3.

L'étiquetage ainsi obtenu assure bien que deux sommets voisins ont des couleurs différentes et de plus, c'est une coloration connexe. En effet, tous les sommets apparaissant dans un cycle C_p ont deux couleurs différentes dans leurs voisinages et tous les sommets $S \in \mathcal{S} \cup \mathcal{S}'$ sont étiquetés 3 et ont un voisin $q_1 \in Q_1$ étiqueté 1 et un voisin $q_2 \in Q_2$ étiqueté 2.

On considère maintenant un hypergraphe $(Q, \mathcal{S})$ où $Q = \{q_1, \ldots, q_m\}$ et $\mathcal{S} = \{S_1, \ldots, S_n\}$ dont le graphe d'incidence est connexe et tel que pour tous entiers distincts $j, k \in [1, n]$, $S_j \neq S_k$. On suppose que $I'(Q, \mathcal{S})$ admet une coloration connexe propre ℓ. D'après le Lemme 9.21, on sait que $|\ell(V(I'(Q, \mathcal{S})))| = 3$ et on note $1, 2$ et 3 les couleurs utilisées. Si on considère le cycle C_{q_1} de longueur 3, dont tous les sommets ont des couleurs distinctes, on sait que tout sommet étiqueté 1 (resp. 2, 3) a des voisins étiquetés 2 et 3 (resp. 1 et 3, 1 et 2).

On suppose que $\ell(v) = 1$ et ainsi, on peut supposer sans perte de généralité que pour tout $j \in [1, n]$, $\ell(S_j) \in \{2, 3\}$. S'il existe un entier j tel que $\ell(S'_j) = 1$, alors S'_j a un voisin q_2 étiqueté 2 et un voisin q_3 étiqueté 3 qui sont tous les deux adjacents à S_j, ce qui est impossible. Par conséquent, pour tout $j \in [1, n]$, $\ell(S'_j) \in \{2, 3\}$. On définit alors $Q_1 = \{q \in Q \mid \ell(q) = 1\}$ et $Q_2 = Q \setminus Q_1$. Puisque tout sommet S'_j a au moins deux voisins avec des couleurs différentes et au moins un voisin étiqueté 1, la partition $Q_1 \cup Q_2$ est bien une 2-coloration de $(Q, \mathcal{S})$.

On a donc montré qu'il est NP-difficile de décider si un graphe simple G admet une coloration connexe propre. □

Le théorème précédent permet d'obtenir un corollaire sur la complexité de décider si un graphe simple G admet un algorithme de nommage ou d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

Corollaire 9.23 *Le problème de décider si on peut élire (resp. nommer) dans un graphe simple G connexe donné en utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées est co-NP-complet.*

Preuve : Pour le nommage, c'est un corollaire de la Proposition 6.4, du Théorème 6.16 et du Théorème 9.22.

On considère le problème de l'élection. On rappelle qu'on ne peut pas élire dans un graphe simple G en utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées si et seulement si l'une des deux conditions suivantes est satisfaite.

- il existe un graphe simple H tel que G est une submersion de H et tel que pour

tout sommet $v \in V(H)$, il existe un sous-graphe G' de G et un homomorphisme localement surjectif φ' de G' dans G tel que $|\varphi'^{-1}(v)| > 1$.
– il existe deux graphes simples H_1, H_2 tels que G est une submersion de H_1 et de H_2 et tels qu'il existe deux sous-graphes disjoints G_1, G_2 de G qui sont respectivement des submersions de H_1 et H_2.

On montre que ce problème est dans NP. On devine d'abord laquelle de ces deux conditions n'est pas satisfaite.

Dans le premier cas, on devine ensuite un graphe simple H et une fonction de $V(G)$ dans $V(H)$. Puisque G doit être une submersion de H, on vérifie que $|V(H)| \leq |V(G)|$. Pour tout sommet $v \in V(H)$, on devine ensuite un sous graphe $G'(v)$ de G et une fonction de $V(G'(v))$ dans $V(H)$. Ainsi, il faut deviner au plus $|V(G)|$ sous-graphes de G et $|V(G)|+1$ fonctions : on devine donc un nombre polynomial de graphes et de fonctions qui sont toutes de tailles polynomiales en $|V(G)|$. Il suffit ensuite de vérifier des conditions locales sur G et sur chaque $G'(v)$ pour s'assurer que les fonctions devinées sont bien des homomorphismes localement surjectifs ; ce qui se fait en temps polynomial.

Dans le second cas, on devine deux graphes simples H_1, H_2, une fonction de $V(G)$ dans $V(H_1)$ et une fonction de $V(G)$ dans $V(H_2)$. On devine ensuite deux sous-graphe G_1 et G_2 de G ainsi qu'une fonction de $V(G_1)$ dans $V(H_1)$ et une fonction de $V(G)$ dans $V(H_2)$. Il suffit ensuite de vérifier des conditions locales sur G, G_1 et G_2 pour s'assurer que les fonctions devinées sont bien des homomorphismes localement surjectifs ; ce qui se fait en temps polynomial.

Pour montrer qu'il est NP-difficile de décider si un graphe simple G admet un algorithme d'élection, on considère la même réduction que dans le Théorème 9.22.

On considère un hypergraphe $(Q, \mathcal{S})$ où $Q = \{q_1, \ldots, q_m\}$ et $\mathcal{S} = \{S_1, \ldots, S_n\}$. On suppose de plus que le graphe d'incidence de $(Q, \mathcal{S})$ est connexe et que pour tout entiers distincts $j, k \in [1, n]$, $S_j \neq S_k$. On remarque que si le graphe $I'(Q, \mathcal{S})$ associé à $(Q, \mathcal{S})$ admet une coloration connexe propre ℓ, alors $|\ell(V(I'(Q, \mathcal{S})))| = 3$. De plus, puisque le graphe d'incidence est connexe, Q est non-vide et le cycle C_v contient donc au moins 9 sommets et chacune des trois couleurs de $\ell(V(I'(Q, \mathcal{S})))$ apparaît au moins trois fois sur C_v.

Par conséquent, si $I'(Q, \mathcal{S})$ admet une coloration connexe propre, alors $I'(Q, \mathcal{S})$ est une submersion du cycle à 3 sommets à travers l'homomorphisme induit par ℓ et pour tout $u \in V(C_3)$, $|\ell^{-1}(u)| > 1$. Ainsi, si $I'(Q, \mathcal{S})$ admet une coloration connexe propre, on ne peut pas résoudre l'élection sur $I'(Q, \mathcal{S})$ en utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées et si $I'(Q, \mathcal{S})$ n'admet pas de coloration connexe propre, on peut résoudre le nommage et donc l'élection sur $I'(Q, \mathcal{S})$ en utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

Ainsi, en utilisant la même technique que dans la preuve du Théorème 9.22, on peut montrer que $(Q, \mathcal{S})$ admet une 2-coloration si et seulement s'il n'existe pas d'algorithme d'élection pour le graphe $I'(Q, \mathcal{S})$ utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées.

On a donc montré qu'il est co-NP-complet de décider si un graphe simple G admet un algorithme d'élection utilisant des calculs locaux cellulaires sur les arêtes non-étiquetées. □

9.5 Conclusion et Perspectives

Dans ce chapitre, on a montré que pour tous les modèles étudiés dans ce mémoire à l'exception du modèle diffusion-à-port et diffusion-à-boîte, il était co-NP-complet de décider si un graphe G donné admettait un algorithme d'élection ou de nommage.

Il est intéressant de remarquer que les algorithmes de nommage présentés dans les Chapitres 2, 3 et 4 sont des algorithmes distribués déterministes qui nécessitent un nombre polynomial de pas de réétiquetages et où chaque sommet dispose d'une mémoire polynomiale. Cela met en évidence le non-déterminisme qui est du à l'exécution d'un algorithme distribué dans ces modèles. Par ailleurs, cela permet aussi d'expliquer pourquoi dans ces modèles, il existe des exécutions de nos algorithmes qui permettent de résoudre élection et nommage dans des graphes qui ne sont pas minimaux (pour les revêtements simples, les revêtements ou les fibrations discrètes).

De même, les algorithmes proposés pour les modèles port-à-port et port-à-boîte dans le Chapitre 7 sont des algorithmes qui nécessitent un nombre polynomial de messages de taille polynomiale. Cependant, le non-déterminisme ne vient pas que de l'exécution, puisqu'on sait d'après la Proposition 7.36 que pour tout graphe $(\mathbf{G}, \delta)$, il existe une exécution «canonique» (l'exécution synchrone) qui permet d'élire un sommet de $\mathbf{G}$ si et seulement si $(\mathbf{G}, \delta)$ est minimal pour les revêtements dirigés symétriques. En fait, dans ces modèles, le non-déterminisme provient de l'étiquetage des ports du réseau, puisqu'un algorithme d'élection pour un graphe $\mathbf{G}$ doit permettre d'élire un sommet de $\mathbf{G}$ quel que soit l'étiquetage des ports. Cela permet de mettre en évidence le rôle que peut avoir l'étiquetage des ports d'un graphe pour briser les «symétries» du graphe.

Conclusion et Perspectives

Dans ce mémoire, on a considéré différents modèles d'algorithmique distribué. On a étudié les relations entre ces modèles à travers l'étude des problèmes du nommage et de l'élection. Pour chacun de ces modèles, on a caractérisé quels étaient les graphes dans lesquels on pouvait résoudre le nommage et l'élection. Les caractérisations obtenues sont constructives : pour chaque modèle, on a exhibé des algorithmes qui permettaient de résoudre le nommage ou l'élection pour tous les graphes pour lesquels on n'avait pas obtenu de résultat d'impossibilité.

Une des motivations de cette thèse était de déterminer si les résultats existants qui sont basés sur l'algorithme de Mazurkiewicz étaient propre au modèle considéré par Mazurkiewicz, ou si on pouvait obtenir des résultats similaires dans d'autres modèles. On a montré qu'on pouvait obtenir des algorithmes «à la Mazurkiewicz» dans tous les modèles considérés dans ce mémoire. Cela permet de mettre en évidence la robustesse du principe de l'algorithme de Mazurkiewicz, puisque même si un sommet ne parvient pas à distinguer ses voisins (comme c'est le cas dans les modèles utilisant des calculs locaux sur les arêtes non-étiquetées), chaque sommet arrive à obtenir le maximum d'informations possibles à propos du graphe auquel il appartient. Cela permet de penser que les résultats obtenus par Godard, Métivier et Muscholl [GM03, GMM04] sur les familles de graphes reconnaissables de manière distribuée peuvent eux aussi être étendus aux différents modèles considérés dans ce mémoire.

On a par ailleurs montré que dans tous ces modèles, si les sommets connaissaient initialement leur degré, il existait des algorithmes de nommage universels sur les familles de graphes minimaux de diamètre borné. À l'aide des outils utilisés dans [GM02], on pense pouvoir caractériser, dans chaque modèle, les familles de graphes qui admettent des algorithmes universels de nommage, ou de manière équivalente, déterminer quelles sont les connaissances initiales dont il faut disposer à propos d'un graphe pour pouvoir résoudre le problème du nommage sur ce graphe.

De plus, pour tous les modèles considérés à l'exception des calculs locaux (cellulaires) sur les arêtes non-étiquetées étudiés aux Chapitres 5 et 6, les algorithmes de nommage présentés sont polynomiaux, comme l'est l'algorithme de Mazurkiewicz. Dans ces mêmes modèles, on a adapté les outils utilisés par Godard, Métivier et Tel [GM02, MT00] pour montrer que si les sommets connaissaient initialement leurs degrés, il était possible d'obtenir des algorithmes effectifs d'élection en connaissant une borne serrée sur la taille du graphe. Ainsi, il est raisonnable de penser qu'on puisse étendre les techniques utilisées dans [MT00] pour caractériser dans ces modèles les familles de graphes sur lesquelles on puisse transformer tout algorithme distribué à terminaison implicite en algorithme à terminaison explicite comme cela a été fait dans [CGMT07].

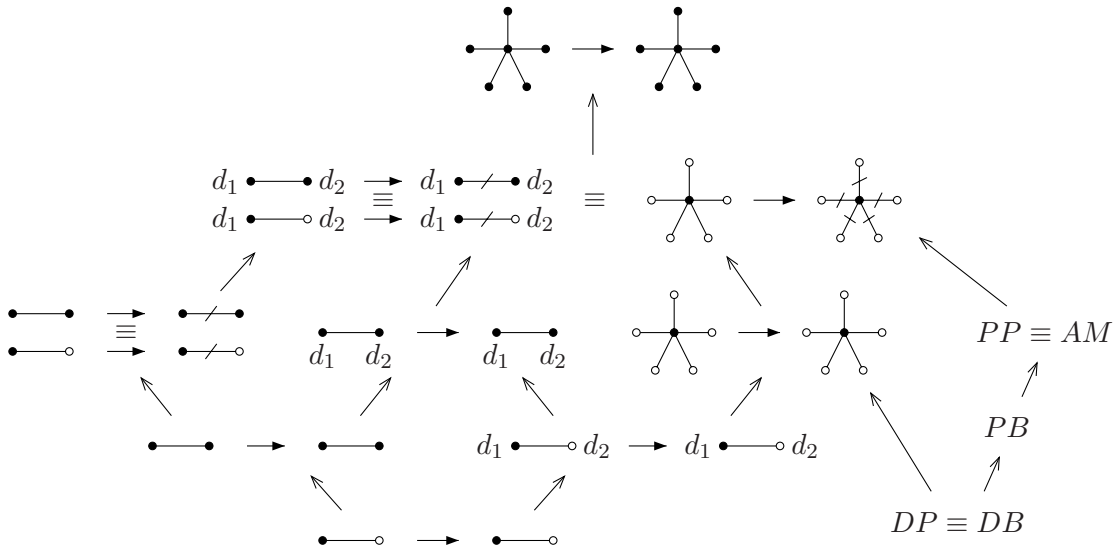


FIG. 35 – Une hiérarchie des différents modèles étudiés dans ce mémoire.

Un autre objectif de cette thèse était de présenter un nouvel algorithme d'élection pour le modèle où les processus communiquent par échange de messages. Il faut noter que l'algorithme d'élection présenté au Chapitre 7 est le premier algorithme polynomial d'élection pour des réseaux de topologie arbitraire dans ce modèle. Cette amélioration par rapport aux algorithmes existants repose sur une bonne compréhension de la notion de revêtements dirigés. Plus généralement, au vu des résultats obtenus, on pense que les différents types d'homomorphismes utilisés dans ce mémoire sont les bons outils pour déterminer ce qui est calculable de manière distribuée dans les différents modèles considérés.

Par ailleurs, on a montré au Chapitre 9 que pour chaque modèle considéré dans ce mémoire, à l'exception des modèles diffusion-à-port et diffusion-à-boîte, il est co-NP-complet de décider si un graphe G admet un algorithme d'élection ou de nommage dans ce modèle. Ce résultat et le fait que la plupart des algorithmes de nommage et d'élection présentés sont polynomiaux permet de mettre en évidence l'aspect non-déterministe d'une exécution d'un algorithme distribué.

Schéma Récapitulatif

Sur la Figure 35, on a représenté les différents modèles étudiés dans ce mémoire.

Les modèles étudiés au Chapitre 7 utilisant des messages sont notés PP, PB, DP, DB et correspondent respectivement aux modèles port-à-port, port-à-boîte, diffusion-à-port et diffusion-à-boîte. Le modèle de système à agents mobiles étudié au Chapitre 8 utilisant des agents mobiles est noté AM.

Les modèles utilisant des calculs locaux étudiés aux Chapitres 2, 3, 4, 5 et 6 sont représentés par la forme des règles utilisées. Les sommets noirs sont les sommets actifs dont les étiquettes peuvent être modifiées lors de l'application de la règle. Les sommets blancs sont les sommets passifs qui permettent l'application d'une règle, mais dont les étiquettes ne sont pas modifiées. Lorsque les arêtes sont marquées, cela signifie que dans le modèle correspondant, les étiquettes des arêtes peuvent être modifiées par l'application d'une règle. Lorsque les sommets sont étiquetés d_1 et d_2, cela signifie que dans le modèle

correspondant, chaque sommet connaît initialement son degré.

S'il existe une flèche $m \to m'$ entre les représentations de deux modèles m et m' sur la Figure 35, cela signifie qu'on peut simuler le modèle m dans m' mais que la réciproque est fausse. Cette relation est évidemment transitive. Lorsque $m \equiv m'$ sur la Figure 35, cela signifie que les modèles m et m' ont la même puissance de calcul. Lorsqu'il n'existe pas de chemin dirigé entre les représentations de deux modèles m et m' sur la Figure 35, cela signifie que les puissances de calcul de m et de m' sont incomparables.

Perspectives

Dans le cadre des calculs locaux, il peut être intéressant de caractériser quels sont les graphes dans lesquels on peut élire en utilisant des algorithmes qui nécessitent une mémoire finie pour chaque sommet. L'algorithme d'élection pour la famille des arbres présenté dans le Chapitre 1 est un exemple d'algorithme de ce type, puisque les étiquettes des sommets du graphe peuvent prendre seulement trois valeurs (A,Non-élu, Élu). D'après les travaux d'Angluin et al. [AAER05], on sait qu'il existe un algorithme utilisant une mémoire finie qui permet de distinguer un sommet dans un graphe complet dans le modèle étudié au Chapitre 6. Cependant, dans les modèles considérés par Angluin et al., les processus ne peuvent pas détecter que l'algorithme est terminé et on ne peut donc pas obtenir un véritable algorithme d'élection à partir de ces résultats. Dans [Maz06], Mazurkiewicz considère un modèle, où en un pas de calcul, un sommet peut observer son voisinage «fermé» (i.e., ses voisins et les éventuelles arêtes entre ses voisins), et modifier son état en conséquence. Dans ce modèle, Mazurkiewicz définit la famille des graphes *fermés* qui contient en particulier les graphes triangulés et qui contient donc les graphes complets et les arbres. Il présente ensuite un algorithme universel d'élection pour cette famille. Cet algorithme est un algorithme de «démontage» du graphe, comme l'est l'algorithme d'élection pour la famille des arbres présenté au Chapitre 1 : un sommet peut prendre l'étiquette Non-élu si lorsque ce sommet est supprimé du graphe, le graphe obtenu est toujours connexe et reste dans la famille des graphes fermés. Dans l'algorithme présenté dans [Maz06], chaque sommet peut avoir seulement trois étiquettes différentes : son étiquette initiale, l'étiquette Élu et l'étiquette Non-élu.

Dans le cadre des algorithmes utilisant des échanges de messages, il semble être possible d'obtenir de nouveaux algorithmes en exploitant le résultat d'équivalence entre les systèmes à agents mobiles et les systèmes où les processus communiquent par échange de messages. En particulier, Das et al. présentent dans [DFNS06] un algorithme effectif d'élection pour les systèmes à agents mobiles de taille donnée. Même si les preuves des résultats présentés dans ce travail reposent sur la notion de vues de Yamashita et Kameda, l'algorithme présenté est polynomial. Cet algorithme est très différent de l'algorithme présenté au Chapitre 7 et n'a pas du tout les mêmes propriétés. Cependant, il est possible d'exhiber des liens entre l'algorithme de Das et al. et la construction de Reidemeister présentée au Chapitre 2 : il semble donc possible de décrire l'algorithme de Das et al. en utilisant les notions de revêtements dirigés présentées au Chapitre 7.

Bibliographie

[AAD+04] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Proc. of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 290–299. ACM press, 2004.

[AAD+06] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4) :235–253, 2006.

[AAE06a] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. In *Proc. of Distributed Computing, 20th International Conference (DISC 2006)*, 2006.

[AAE06b] D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. of the 25th annual ACM symposium on Principles of distributed computing (PODC 2006)*, pages 292–299. ACM Press, 2006.

[AAER05] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. On the power of anonymous one-way communication. In *Proc. of the 9th International Conference on Principles of Distributed Systems (OPODIS 2005)*, pages 307–318, 2005.

[ABRS95] B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot (extended abstract). In *Proc. of the 8th annual conference on Computational learning theory (COLT 1995)*, pages 321–328. ACM Press, 1995.

[AFS91] J. Abello, M.R. Fellows, and J.C. Stillwell. On the complexity and combinatorics of covering finite complexes. *Australasian Journal of Combinatorics*, 4 :103–112, 1991.

[AG81] D. Angluin and A. Gardiner. Finite common coverings of pairs of regular graphs. *J. Combin. Theory, Ser. B*, 30(2) :183–187, 1981.

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proc. of the 12th Symposium on Theory of Computing (STOC 1980)*, pages 82–93, 1980.

[AW04] H. Attiya and J. Welch. *Distributed computing : fundamentals, simulations, and advanced topics.* John Wiley and Sons, 2004.

[BCG+96] P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks : characterizations. In *Proc. of the 4th Israeli Symposium on Theory of Computing and Systems (ISTCS 1996)*, pages 16–26. IEEE Press, 1996.

[BFFS03a] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Can we elect if we cannot compare ? In *Proc. of the 15th annual ACM Symposium on Parallel Algorithms and Architectures, (SPAA 2003)*, pages 324–332. ACM Press, 2003.

[BFFS03b] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Election and rendezvous in fully anonymous systems with sense of direction. In *Proc. of the 10th International Colloquium on Structural Information Complexity (SIROCCO 2003)*, volume 17, pages 17–32. Carleton Scientific, 2003.

[BFFS06] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents : impact of sense of direction. *Theory of Computing Systems*, to appear, 2006.

[Bod89] H.L. Bodlaender. The classification of coverings of processor networks. *Journal of parallel and distributed computing*, 6(1) :166–182, 1989.

[Bou88] L. Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25(2) :179–201, 1988.

[BR05] P. Braun and W. Rossak. *Mobile agents : basic concepts, mobility models and the tracy toolkit*. Morgan Kaufman, 2005.

[BS94] M. Bender and D. Slonim. The power of team exploration : two robots can learn unlabeled directed graphs. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*, pages 75–85, 1994.

[BV99] P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proc. of the 18th ACM Symposium on principles of distributed computing (PODC 1999)*, pages 181–188. ACM Press, 1999.

[BV01] P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In *Proc. of Distributed Computing, 15th International Conference (DISC 2001)*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 2001.

[BV02a] P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Mathematics*, 243(1-3) :21–66, 2002.

[BV02b] P. Boldi and S. Vigna. Universal dynamic synchronous self-stabilization. *Distributed Computing*, 15(3) :137–153, 2002.

[CBMT96] B. Charron-Bost, F. Mattern, and G. Tel. Synchronous, asynchronous and causally ordered communication. *Distributed Computing*, 9(4) :173–191, 1996.

[CDS06] J. Chalopin, S. Das, and N. Santoro. Groupings and pairings in anonymous networks. In *Proc. of Distributed Computing, 20th International Conference (DISC 2006)*, volume 4167 of *Lecture Notes in Computer Science*, pages 105–119. Springer-Verlag, 2006.

[CGMO06] J. Chalopin, E. Godard, Y. Métivier, and R. Ossamy. Mobile agent algorithms versus message passing algorithms. In *Proc. of the 10th International Conference on Principles of Distributed Systems (OPODIS 2006)*, volume 4305 of *Lecture Notes in Computer Science*, pages 187–201. Springer-Verlag, 2006.

[CGMT07] J. Chalopin, E. Godard, Y. Métivier, and G. Tel. About the termination detection in the asynchronous message passing model. In *Proc. of the 33rd Conference on Current Trends in Theory and Practice of Informatics (SOFSEM 2007)*, volume 4362 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, 2007.

[Cha05] J. Chalopin. Local computations on closed unlabelled edges : the election problem and the naming problem. In *Proc. of the 31st Conference on Current Trends in Theory and Practice of Informatics (SOFSEM 2005)*, volume 3381 of *Lecture Notes in Computer Science*, pages 81–90. Springer-Verlag, 2005.

[Cha06] J. Chalopin. Election in the qualitative world. In *Proc. of Structural Information and Communication Complexity, 13th International Colloquium (SIROCCO 2006)*, volume 4056 of *Lecture Notes in Computer Science*, pages 85–99. Springer-Verlag, 2006.

[CM04] J. Chalopin and Y. Métivier. Election and local computations on edges. In *Proc. of Foundations of Software Science and Computation Structures, 7th International Conference (FOSSACS 2004)*, volume 2987 of *Lecture Notes in Computer Science*, pages 90–104. Springer-Verlag, 2004.

[CM05] J. Chalopin and Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs. In *Proc. of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005)*, volume 3618 of *Lecture Notes in Computer Science*, pages 212–223. Springer-Verlag, 2005.

[CMZ04] J. Chalopin, Y. Métivier, and W. Zielonka. Election, naming and cellular edge local computations. In *Proc. of the 2nd International Conference on Graph Transformations (ICGT 2004)*, volume 3256 of *Lecture Notes in Computer Science*, pages 242–256. Springer-Verlag, 2004.

[CMZ06] J. Chalopin, Y. Métivier, and W. Zielonka. Local computations in graphs : the case of cellular edge local computations. *Fundamenta Informaticae*, 74(1) :85–114, 2006.

[CP06] J. Chalopin and D. Paulusma. Graphs labelings derived from models in distributed computing. In *Proc. of Graph-Theoretic Concepts in Computer Science, 32nd International Workshop (WG 2006)*, volume 4271 of *Lecture Notes in Computer Science*, pages 301–312. Springer-Verlag, 2006.

[Der06] B. Derbel. *Local aspects in distributed algorithms.* PhD thesis, Université Bordeaux 1, 2006.

[DFNS05] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed exploration of an unknown graph. In *Proc. of Structural Information and Communication Complexity, 12th International Colloquium (SIROCCO 2005)*, volume 3499 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2005.

[DFNS06] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Effective elections for anonymous mobile agents. In *Proc. of Algorithms and Computation, 17th International Symposium (ISAAC 2006)*, Lecture Notes in Computer Science. Springer, 2006.

[DFP03] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Proc. of the 11th Annual European Symposium (ESA 2003)*, volume 2832 of *Lecture Notes in Computer Science*, pages 184–195, 2003.

[DKP98] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. i : the rectilinear case. *J. ACM*, 45(2) :215–245, 1998.

[EB91] M. Everett and S. Borgatti. Role colouring a graph. *Mathematical Social Sciences*, 21(2) :183–188, 1991.

[FP03] J. Fiala and D. Paulusma. The computational complexity of the role assignment problem. In *Proc. of the 30th International Colloquium on Automata, Languages and Programming (ICALP 2003)*, volume 2719 of *Lecture Notes in Computer Science*, pages 817–828. Springer-Verlag, 2003.

[FP05] J. Fiala and D. Paulusma. A complete complexity classification of the role assignment problem. *Theoretical Computer Science*, 349(1) :67–81, 2005.

[FRS03] P. Flocchini, A. Roncato, and N. Santoro. Computing on anonymous networks with sense of direction. *Theoretical Computer Science*, 301(1-3) :355–379, 2003.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Co., New York, 1979.

[GKKZ06] L. Gasieniec, E. Kranakis, D. Krizanc, and X. Zhang. Optimal memory rendezvous of anonymous mobile agents in a unidirectional ring. In *Proc. of Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2006)*, volume 3831 of *Lecture Notes in Computer Science*, pages 282–292. Springer, 2006.

[GM02] E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible (*ext. abstract*). In *Proc. of Foundations of Software Science and Computation Structures, 5th International Conference (FOSSACS 2002)*, volume 2303 of *Lecture Notes in Computer Science*, pages 159–172. Springer-Verlag, 2002.

[GM03] E. Godard and Y. Métivier. Deducible and equivalent structural knowledges in distributed algorithms. *Theory of Computing Systems*, 36(2) :631–654, 2003.

[GMM04] E. Godard, Y. Métivier, and A. Muscholl. Characterization of classes of graphs recognizable by local computations. *Theory of Computing Systems*, 37(2) :249–293, 2004.

[God02a] E. Godard. *Réécritures de Graphes et Algorithmique Distribuée.* PhD thesis, Université Bordeaux 1, 2002.

[God02b] E. Godard. A self-stabilizing enumeration algorithm. *Information Processing Letters*, 82(6) :299–305, 2002.

[Hal35] P. Hall. On representation of subsets. *J. London Math. Soc.*, 10 :26–30, 1935.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8) :666–677, 1978.

[HU79] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley, 1979.

[KKR06] E. Kranakis, D. Krizanc, and S. Rajsbaum. Mobile agent rendezvous : A survey. In *Proc. of Structural Information and Communication Complexity, 13th International Colloquium (SIROCCO 2006)*, volume 4056 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006.

[KPT94] J. Kratochvíl, A. Proskurowski, and J.A. Telle. Complexity of graph covering problems. In *Proc. of Graph-Theoretic Concepts in Computer Science, 20th International Workshop (WG 1994)*, volume 903 of *Lecture Notes in Computer Science*, pages 93–105. Springer, 1994.

[KPT97] J. Kratochvíl, A. Proskurowski, and J.A. Telle. Covering regular graphs. *J. Comb. Theory, Ser. B*, 71(1) :1–16, 1997.

[KPT98] J. Kratochvíl, A. Proskurowski, and J.A. Telle. Complexity of graph covering problems. *Nordic Journal of Computing*, 5(3) :173–195, 1998.

[Kra91] J. Kratochvíl. *Perfect codes in general graphs*. Academia Praha, 1991.

[KT00] P. Kristiansen and J.A. Telle. Generalized H-coloring of graphs. In *Proc. of the 11th International Conference on Algorithms and Computation (ISAAC 2000)*, volume 1969 of *Lecture Notes in Computer Science*, pages 456–466. Springer-Verlag, 2000.

[Lei82] F.T. Leighton. Finite common coverings of graphs. *J. Combin. Theory, Ser. B*, 33(3) :231–238, 1982.

[LeL77] G. LeLann. Distributed systems, towards a formal approach. In *Information processing 1977*, pages 155–160. North-Holland, 1977.

[Lyn96] N. Lynch. *Distributed algorithms*. Morgan Kaufman, 1996.

[Maz87] A. Mazurkiewicz. Trace theory. In *Advances in Petri Nets*, volume 255 of *Lecture notes in computer science*, pages 279–324. Spinger, 1987.

[Maz97] A. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61(5) :233–239, 1997.

[Maz06] A. Mazurkiewicz. Locally derivable graphs. *manuscript*, 2006.

[Mil99] R. Milner. *Communicating and mobile systems : the π-calculus*. Cambridge University Press, 1999.

[MMW97] Y. Métivier, A. Muscholl, and P.-A. Wacrenier. About the local detection of termination of local computations in graphs. In *Proc. of the 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO 1997)*, Proceedings in Informatics, pages 188–200. Carleton Scientific, 1997.

[MS01] M. Mosbah and A. Sellami. Implementation of an enumeration protocol algorithm using local graph computations. In *Proc. of Second International Workshop on Graph Transformation and Visual Modeling Techniques*, 2001.

[MSZ02] Y. Métivier, N. Saheb, and A. Zemmari. Randomized local elections. *Information Processing Letters*, 82(6) :313–320, 2002.

[MSZ03] Y. Métivier, N. Saheb, and A. Zemmari. Analysis of a randomized rendezvous algorithm. *Inf. Comput.*, 184(1) :109–128, 2003.

[MT00] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *Proc. of 7th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2000)*, Proceedings in Informatics, pages 237–251. Carleton Scientific, 2000.

[NMR01] C. Nichitiu, J. Mazoyer, and E. Rémila. Algorithms for leader election by cellular automata. *J. Algorithms*, 41(2) :302–329, 2001.

[Nor95] N. Norris. Universal covers of graphs : isomorphism to depth $n-1$ implies isomorphism to all depths. *Discrete Applied Math.*, 56(1) :61–74, 1995.

[NPR04] C. Nichitiu, C. Papazian, and E. Rémila. Leader election in plane cellular automata, only with left-right global convention. *Theoretical Computer Science*, 319(1-3) :367–384, 2004.

[Pal03] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculus. *Mathematical Structures in Computer Science*, 13(5) :685–719, 2003.

[Rei32] K. Reidemeister. *Einführung in die kombinatorische topologie*. Vieweg, Brunswick, 1932.

[RFH72] P. Rosenstiehl, J.-R. Fiksel, and A. Holliger. Intelligent graphs. In R. Read, editor, *Graph theory and computing*, pages 219–265. Academic Press (New York), 1972.

[Sak99] N. Sakamoto. Comparison of initial conditions for distributed algorithms on anonymous networks. In *Proc. of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC 1999)*, pages 173–179. ACM Press, 1999.

[Sel04] A. Sellami. *Des calculs locaux aux algorithmes distribués*. PhD thesis, Université Bordeaux 1, 2004.

[SSP85] B. Szymanski, Y. Shy, and N. Prywes. Terminating iterative solutions of simultaneous equations in distributed message passing systems. In *Proc. of the 4th Annual ACM Symposium on Principles of Distributed Computing (PODC 1985)*, pages 287–292. ACM Press, 1985.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.

[YK96a] M. Yamashita and T. Kameda. Computing functions on asynchronous anonymous networks. *Math. Systems Theory*, 29(4) :331–356, 1996.

[YK96b] M. Yamashita and T. Kameda. Computing on anonymous networks : Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1) :69–89, 1996.

[YK96c] M. Yamashita and T. Kameda. Computing on anonymous networks : Part ii - decision and membership problems. *IEEE Transactions on parallel and distributed systems*, 7(1) :90–96, 1996.

[YK98] M. Yamashita and T. Kameda. Erratum to computing functions on anonymous asynchronous networks. *Theory of Computing Systems*, 31(1) :109, 1998.

[YK99] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on parallel and distributed systems*, 10(9) :878–887, 1999.

Algorithmique distribuée, calculs locaux et homomorphismes de graphes

Résumé : Dans cette thèse, on étudie ce qui est calculable dans différents modèles d'algorithmique distribuée. Les modèles considérés correspondent à différents niveaux d'abstraction et à différents niveaux de synchronisation entre les processus d'un système distribué.

On s'intéresse en particulier au problèmes de l'élection et du nommage dans ces différents modèles. Pour chaque modèle, on caractérise les systèmes distribués dans lesquels on peut résoudre ces problèmes et on étudie la complexité des problèmes de décision correspondants. Nos caractérisations utilisent des homomorphismes de graphes qui préservent certaines propriétés locales.

Nos preuves sont constructives : quand on peut résoudre l'élection (ou le nommage) dans un réseau, on présente un algorithme d'élection (ou de nommage) pour ce réseau. Ces problèmes permettent de mettre en évidence les différences entre les puissances de calculs des différents modèles considérés. De plus, l'étude de ces problèmes permet de mettre à jour les bons outils qui permettent d'étudier ce qui est calculable de manière distribuée dans les différents modèles.

Mots clés : algorithmique distribuée, calculs locaux, échange de messages, agents mobiles, réseaux anonymes, élection, nommage, homomorphismes de graphes, revêtements, complexité

Distributed algorithms, local computations and graph homomorphisms

Abstract: In this thesis, we consider different models of distributed computations. These models correspond to different levels of abstraction and they encode different levels of synchronization between processes in a distributed system.

In these different models, we particularly focus on two classical problems in distributed computing : election and naming. For each model, we present a characterization of distributed systems where these problems can be solved and we study the complexity of the corresponding decision problems. Our characterizations are expressed in terms of graph homomorphisms that preserve some local properties.

Our proofs are constructive : when a network admits an election (or a naming) algorithm, we present such an algorithm for this network. These problems enable to highlight the differences between the computation powers of the different models we consider. Moreover, studying these problems enable to introduce some combinatorial and algorithmic tools that can be used to study what can be computed in a distributed way in these different models.

Keywords: distributed algorithms, local computations, message passing systems, mobile agents, anonymous networks, election, naming, graph homomorphisms, coverings, complexity