$N°$d'ordre : 2518

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Emmanuel GODARD

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : Informatique

****

REECRITURES DE GRAPHES ET ALGORITHMIQUE DISTRIBUEE

****

Soutenue le : 27 juin 2002

Après avis de :

MM.	Antoni Mazurkiewicz	Professeur	**Rapporteurs**
	Igor Litovsky	Professeur	

Devant la commission d'examen formée de :

MM.	Joffroy Beauquier	Professeur	**Examinateur**
	Bruno Courcelle	Professeur	**Président**
	Igor Litovsky	Professeur	**Rapporteur**
	Yves Métivier	Professeur	**Directeur**
	Mohamed Mosbah	Maître de Conférences	**Examinateur**
	Anca Muscholl	Professeur	**Examinatrice**

- 2002 -

Remerciements

Je remercie Antoni Mazurkiwiecz de m'avoir fait l'honneur d'être rapporteur de ma thèse. Il sait combien je regrette qu'il n'est pu être présent à ma soutenance. Je remercie également Igor Litovsky d'avoir accepté d'être rapporteur. Je suis très heureux de le retrouver à l'issue de cette thèse. Je remercie Bruno Courcelle d'avoir accepté d'être le président du jury. Ses interventions et questions lors de mes exposés ont toujours été très instructives. Je remercie également les autres examinateurs, Joffroy Beauquier, Mohamed Mosbah et Anca Muscholl pour l'interêt porté à mon travail. Je remercie particulièrement M. Mosbah pour ses relectures et pour sa disponibilité. Je remercie A. Muscholl pour l'agrément que j'ai eu à travailler avec elle.

Je remercie finalement et sincèrement Yves Métivier pour l'encadrement de mon travail de thèse. Le sujet proposé était extrêmement intéressant et s'est révélé, au cours de ces trois ans, très riche, énormément d'idées avaient été préalablement rassemblées autour des réécritures de graphes et j'ai eu la chance de me trouver au bon moment pour pouvoir rassembler ce qui était en circulation. Je le remercie également pour sa compréhension lors de mes deux ans d'"interruption involontaire d'activité" pour cause de service national. Je le remercie surtout pour son aide lors de ces deux dernières années. Je ne pense pas que ce travail aurait pu être achevé dans les conditions d'éloignement qui étaient les miennes sans son soutien sans faille et sa bonne volonté. Qu'il en soit encore remercié ici.

Concernant mes collègues doctorants du LaBRI, je remercie en premier lieu Cédric pour sa gentillesse et son énergie, en particulier lors de mes déplacements de "squatteur du jeudi". Je le remercie très sincèrement pour son accueil et son amitié. Mon parcours ayant été assez atypique et éloigné par éclipses du laboratoire, j'ai assisté à de nombreux changements de génération... Parmi les derniers arrivés, je remercie Davy pour sa gentillesse permanente et surtout pour les coups de main cruciaux de dernières minutes. Je remercie également les autres membres du bureau A (Valère, Nicolas, Pascal, Guillaume, Ireneusz) pour leur coopération bienveillante à s'absenter au moins un à la fois lors de mes passages éclairs afin de libérer un poste. Ceci me permit de travailler dans des conditions très raisonnables. Je remercie également les "anciens" pour leur accueil lors de mon arrivée au LaBRI, Hélène, Anne, tous les Nico, Philippe, Augustin... Ils n'étaient déjà plus doctorants lorsque j'ai fait leur connaissance mais ils méritent des remerciements quand même. Merci à Isabelle pour la férule bienveillante. Merci à Gwenola et Olivier pour le coup d'œil *très* candide et le coup de main *très* apprécié.

Mes derniers remerciements, mais pas les moindres, vont bien sûr à Marie-Laure et Mélisse. Elles ont dû supporter les atermoiements inhérents, apparemment, à la fin de rédaction mais elles méritent tant de remerciements pour tant d'autres choses ....

Ce mémoire a été réalisé avec le paquet LaTeX2e `thloria` de D. Roegel sur un système GNU / Linux.

Table des matières

Introduction

L'algorithmique est l'étude des méthodes que l'on peut utiliser de façon systématique pour réaliser une tâche : faire des confitures (peser les fruits, ajouter la proportion de sucre indiquée, cuire une minute à gros bouillons pour 100g de fruits), trouver un taxi pour demain matin en feuilletant les pages jaunes (tant que personne ne répond ou que le taxi n'est pas libre, téléphoner au numéro suivant), savoir si un entier n est premier ou non (essayer de diviser n par tous les entiers inférieurs). Si la méthode est automatisable, on pourra éventuellement la traduire sous forme de logiciel, sur un ordinateur. Ce logiciel pourra interagir avec un utilisateur (traitement de texte) ou encore avec d'autres logiciels sur d'autres ordinateurs (envoi / transmission / réception de courrier électronique).

Un système distribué est un environnement où différents processus (exécution particulière d'un logiciel) se partagent les ressources (accès à une imprimante), collaborent pour réaliser un objectif commun (mise à jour d'une base de données distribuée). Un réseau est un système distribué où les processus ne peuvent communiquer directement qu'avec un nombre restreint d'autres processus, leurs "voisins". L'algorithmique distribuée a pour objet l'étude des réseaux et s'attache à décrire ce que l'on peut, et de quelle manière, réaliser sur un réseau de processus exécutant le même logiciel. Un élément du réseau s'appelle un nœud et ce qui permet de distinguer un nœud d'un autre peut être un identifiant (numéro IP sur Internet), mais dans le cas le plus général, c'est uniquement sa situation sur le réseau (sans qu'elle soit connue explicitement) *par rapport* aux autres nœuds qui le distingue. Ce qui est réellement particulier aux systèmes distribués, c'est le fait qu'il n'existe *a priori* aucun système de centralisation permettant de coordonner globalement les comportements des processus. Quels comportements globaux peut-on faire émerger dans ce cadre ? A quelles conditions ? Comment les obtenir ? Voici quelques unes des questions auxquelles l'algorithmique des réseaux de processus tente de répondre.

Quelques Problématiques de l'Algorithmique Distribuée

Il apparaît que, et nous y reviendrons plus loin, certains problèmes naturels sur les réseaux distribués sont difficiles à résoudre et n'admettent parfois aucune solution. On va en fournir quelques exemples et voir comment leur "difficulté" peut être modulée. Il y a en particulier les problèmes de type "rupture de symétrie" dont le problème de l'élection est le plus significatif : étant donné une certaine situation initiale, est-il pos-

sible de distinguer un nœud unique. Précisément est-il possible qu'un seul nœud soit dans l'état ÉLU et tous les autres étant dans l'état PAS ÉLU. Si les nœuds possèdent un numéro d'identification, une solution peut être d"'élire" celui possédant le numéro le plus, ou le moins, élevé. Si l'on ne peut utiliser de tels identifiants, le réseau est ce que l'on appelle anonyme. Alors, il apparaît que dans ce cas, si le réseau a de plus une topologie "symétrique" le problème n'a pas de solution. Nous y reviendrons dans la section suivante. La difficulté de ces problèmes peut être modulée en fonction de connaissances supplémentaires sur la structure du réseau que peuvent avoir les processus. Par exemple, un certain nombre de problèmes sont plus "faciles" à résoudre si l'on connaît le nombre de nœuds dans le réseau.

Un autre degré d'exigence possible concerne la détection de la terminaison. Dans le cas séquentiel, lorsqu'un logiciel s'exécute, il lui est facile de constater s'il a terminé sa tâche ou pas. Dans le cas distribué c'est plus délicat. En effet, même si, localement, un processus peut constater qu'il n'a aucune instruction à effectuer, du fait qu'il fonctionne en interaction avec le reste du réseau, il ne peut prédire si, *dans le futur*, il va recevoir, ou non, un message qui devra provoquer une réaction de sa part. Il ne peut *a priori* pas cesser d'attendre des messages éventuels provenant de ses voisins. Un observateur extérieur peut constater que, globalement, la tâche à réaliser est achevée, le "calcul distribué" est terminé : tous les nœuds sont dans un état où il n'ont et n'auront plus rien à faire. Une telle terminaison est définie comme *implicite* : le calcul est effectivement terminé mais aucun nœud ne le "sait". On peut également prescrire, cela paraît raisonnable et est même indispensable dans certain cas, que la terminaison soit détectée localement par un, ou tous les nœuds. On peut exiger la détection de la terminaison locale (le nœud sait qu'il a atteint son état final, qu'il n'effectuera plus aucune opération) ou la terminaison globale (un noeud sait que *tous* les sommets ont atteint leur état final ; il s'agit en général du sommet effectuant le dernier pas de calcul). Une autre exigence qu'il ne faut pas perdre de vue est le fait que les solutions doivent être effectives et que l'on peut souhaiter concevoir un algorithme qui fournit, en fonction d'une connaissance particulière (par exemple le nombre exact de sommets), l'algorithme distribué à appliquer pour résoudre le problème.

Symétries et Impossibilté de l'Election

Depuis le travail original d'Angluin [Ang80], il est connu que certains problèmes comme l'élection n'ont parfois aucune solution pour des raisons de symétrie "interne" au réseau. Les définitions formelles seront données ultérieurement mais présentons intuitivement l'idée fondamentale du raisonnement. Un réseau $\mathbf{G}$ est "symétrique" s'il existe un autre graphe $\mathbf{H}$, de taille inférieure, tel que $\mathbf{G}$ est en quelque sorte constitué de copies de $\mathbf{H}$ entrelacées de telle manière qu'un processus ne puisse, à l'aide des informations *locales* auxquelles il a accès, savoir si le réseau sous-jacent sur lequel il s'exécute est $\mathbf{G}$ ou $\mathbf{H}$. Dans cette configuration, on dit que $\mathbf{G}$ est un revêtement de $\mathbf{H}$. Un algorithme distribué étant constitué essentiellement d'échanges d'informations entre un nœud et ses voisins, il ne peut rassembler que des informations locales, et par conséquent son exécution sur $\mathbf{H}$ peut être similaire et indistinguable en un certain

sens d'une exécution sur $\mathbf{G}$. Si l'on cherche à élire, on peut éventuellement obtenir un seul nœud ÉLU sur $\mathbf{H}$, mais $\mathbf{G}$ étant constitué de copies de $\mathbf{H}$, il y a une exécution de l'algorithme sur $\mathbf{G}$, "similaire" à une exécution sur $\mathbf{H}$, qui s'achèvera avec de multiples nœuds ÉLU s. L'algorithme ne pourra donc élire systématiquement sur $\mathbf{G}$. Le problème de l'élection n'a pas de solution sur $\mathbf{G}$.

Si le modèle utilisé par Angluin pouvait paraître assez peu orthodoxe, le fait est que la preuve – esquissée ici – c'est à dire l'idée sous-jacente indépendante d'un modèle spécifique, s'adapte aisément dans tous les modèles raisonnables connus. Elle s'adapte en particulier au cadre de notre modèle, le réétiquetage de graphes. La transposition consistera en le lemme 0.31 dit lemme de Relèvement.

Étudier les conditions de réalisabilité d'une tâche distribuée éclaire particulièrement le domaine, cependant il ne s'agit pas de préoccupations strictement théoriques, comme on pourrait le penser. D'un point de vue pratique en effet, on peut se demander si se préoccuper des cas d'impossibilité peut avoir de l'intérêt. Après tout, si l'on ne peut élire dans un réseau anonyme, il suffit de numéroter les nœuds et le tour est joué. Cependant, il y a des implications de maintenabilité (attribuer *de manière distribuée* des numéros n'est pas si aisé) et de tolérance aux pannes (les numéros peuvent être corrompus). Il faut sans doute voir ces études comme la recherche de la réponse à la question : qu'est-ce qui est *absolument nécessaire*, et non simplement *commode* pour pouvoir résoudre un problème distribué donné ? La version duale est : que peut-on exactement faire si l'on a telle connaissance sur la structure sous-jacente du graphe ? On s'aperçoit alors que ce qui est fondamental ce sont certaines propriétés combinatoires, topologiques des réseaux. Celles-ci sont directement liées au caractère distribué des problèmes considérés. Une autre manière d'éluder les théorèmes d'impossibilité peut être d'utiliser des algorithmes probabilistes. Nous n'en parlerons pas ici.

Pour étudier les systèmes distribués, comme pour tout système, on est amené à effectuer certaines simplifications, à privilégier certaines hypothèses afin de définir un modèle formel. En l'état actuel, il n'y a pas de modèle universellement admis, et comme le regrette N. Lynch dans son article "Cent preuves d'impossibilité" [Lyn89], même si les preuves sont en général fondées sur des techniques assez semblables de simulation (d'un état qui ne convient pas, je peux toujours passer à un autre état qui ne convient pas, et ainsi ne jamais résoudre le problème) les modèles sont toujours différents. Cela requiert à chaque article une longue partie de présentation et de définition formelle du modèle. L'une des motivations de cette thèse est de montrer que le modèle que nous allons utiliser est un modèle pertinent pour l'étude des problèmes d'algorithmique distribuée.

La manière dont les échanges entre nœuds s'effectuent détermine le type de réseau. Ici, nous considérons des communications locales et simultanées entre un nœud et tous ses voisins. Le modèle présenté ici est en quelque sorte intermédiaire entre les classiques réseaux synchrones et asynchrones, au sens où le système est globalement asynchrone et les changements d'états s'effectuent par *synchronisation locale* des processeurs. Le modèle que nous allons présenter ici, poursuivant de nombreux travaux initiés par Y. Métivier, est le suivant :

Le Modèle

Le réseau de communication peut avoir une topologie arbitraire, il est **fiable** : il ne se produit aucune défaillance ni sur les nœuds, ni sur les liens de communications. Le réseau est modélisé, classiquement, par un **graphe étiqueté** avec la correspondance usuelle : les nœuds correspondent aux sommets du graphe, les liens de communications aux arêtes, et les états des processeurs aux étiquettes. Un algorithme correspond à un **système de règles** et chaque nœud suit les instructions de l'algorithme qui prescrit comment s'effectuent ses changements d'états. Le fonctionnement est **asynchrone**, il n'y a pas d'horloge globale et chaque nœud peut appliquer ou ne pas appliquer une règle. Plus précisément, les règles de changement d'état ont la forme de **lecture et d'écriture des états d'un nœud et de tous ses voisins**. Cela s'interprète de la façon suivante : si les états d'un nœud et de ses voisins sont préalablement dans une certaine configuration, les états de ce nœud et de tous ses voisins changent en fonction de la configuration préalable. Un tel changement local d'états (réétiquetage) est **atomique**. Un pas d'éxécution de l'algorithme correspondra à l'application d'une règle en un nœud où elle peut s'y appliquer. Une exécution de l'algorithme correspond alors à une affectation initiale, qui initialise les nœuds dans un certain état, pas nécessairement uniforme, et une suite de tels réétiquetages. Les réétiquetages qui se produisent en des régions ne se recouvrant pas peuvent s'effectuer en parallèle. Il est important de noter que ce qui distingue particulièrement le modèle réétiquetage de graphe, c'est le fait que dans ce modèle la méthode de communication entre nœuds est complètement implicite et est en fait abstraite au niveau des règles de lecture et d'écriture. En outre, l'utilisation de règles - plutôt qu'une méthode impérative - pour décrire un algorithme distribué paraît assez légitime puisqu'un calcul distribué asynchrone est intrinsèquement de forme événementielle : chaque nœud réagit au comportement de ses voisins.

Ce modèle a de fait de nombreux intérêts : il fournit un modèle abstrait pour décrire et résoudre, comme nous allons le voir, certains problèmes dans le domaine de l'algorithme distribué indépendamment de la large variété de modes de communication entre processus utilisées dans la littérature. En outre, les résultats d'impossibilité restent vrais dans des modèles plus faibles, et toute solution positive peut indiquer une direction de recherche pour un modèle plus faible ou bien être implémentée en utilisant des algorithmes probabilistes. Ce modèle fournit des propriétés et des exemples intéressants en utilisant de la combinatoire des graphes très classique.

L'étude d'un modèle passe en particulier par l'étude de son pouvoir d'expression, c'est-à-dire l'étude précise de ce qui ou n'est pas réalisable dans ce modèle. On va s'intéresser au pouvoir d'expression de ce modèle (étude formelle) tout en essayant d'en retirer des enseignements originaux sur l'algorithmique distribuée ainsi que de critiquer et enrichir ce modèle, en particulier par rapport aux autres modèles.

Techniques et Résultats

Le but de notre travail est de caractériser exactement ce qui est réalisable par calcul distribué *en fonction* de la connaissance - arbitraire - de la structure du réseau sous-jacent (connaissance structurelle). Les preuves de caractérisation possèdent deux fa-

cettes : les preuves d'impossibilité ou conditions nécessaires de réalisabilité, basées sur des techniques à base de revêtements et quasi-revêtements (applications se comportant comme des revêtements uniquement sur une boule d'un certain rayon) ; et les preuves de possibilités ou conditions suffisantes, basées sur l'algorithme d'énumération d'A. Mazurkiewicz. Cet algorithme numérote de manière distribuée les nœuds dans un réseau anonyme et sera le socle de tous les algorithmes que nous exposerons. Notre présentation est organisée comme suit.

Nous présenterons tout d'abord le formalisme du réétiquetage de graphes (chapitre 0). Nous exposerons certains résultats de la combinatoire des graphes nécessaires à notre étude. Nous prouverons certains résultats originaux en particulier l'équivalence avec le formalisme utilisé par A. Mazurkiewicz dans [Maz97] (lemme 0.48), et l'extension aux quasi-revêtements du résultat classique de simulation d'Angluin (lemme 0.35).

La première partie abordera l'algorithme d'énumération de Mazurkiewicz. Nous exposerons et prouverons les propriétés de cet algorithme dans notre formalisme (chapitre 1). Nous étudierons sa complexité (chapitre 2) et nous étendrons celui-ci au cadre autostabilisant, au cadre des réseaux non anonymes et au cadre des réétiquetages généralisés (chapitre 3).

En se basant sur les différentes généralisations de l'algorithme et sur les lemmes combinatoires du chapitre 0, nous exposerons un certain nombre de résultats. Dans la deuxième partie, nous étudierons, dans un but exploratoire et d'illustration, l'une des tâches distribuées les plus simples : déterminer, c'est-à-dire reconnaître, avec terminaison implicite, si la structure du réseau a, ou n'a pas, certaines propriétés. Les familles de graphes dont les propriétés peuvent être ainsi reconnues de manière distribuée sont dites reconnaissables et nous présenterons une caractérisation des familles reconnaissables sans connaissance structurelle (chapitre 4), et avec connaissance structurelle arbitraire (chapitre 5). Les caractérisations seront exprimées en terme de fermeture des familles reconnaissables par la relation "être revêtement de" (Th. 4.6 et 5.10). Nous verrons plusieurs exemples d'application pour des connaissances structurelles particulières (chapitre 6). Nous verrons en particulier (Th. 6.4) que les familles de graphes fermées par mineurs ne sont en général pas reconnaissables même en connaissant la taille.

Au cours de la troisième partie, nous étudierons de manière plus générale certaines tâches réalisables avec terminaison explicite. Nous commencerons par le problème de l'élection (chapitre 7), et nous présenterons à cet occasion une façon d'exploiter l'algorithme de Mazurkiewicz afin de réaliser un algorithme de cartographie partielle du réseau sous-jacent, qui s'avérera être optimal au sens où tout ce qui pourra espérer être obtenu par calculs distribués, le sera par cet algorithme. Une caractérisation complète et effective du problème de l'élection sera proposée en terme de borne sur les rayons des quasi-revêtements admissibles (Th. 7.27).

Nous caractériserons, par des techniques similaires et donc en terme de fermeture

pour des relations à base de quasi-revêtements, les fonctions scalaires, c'est-à-dire les tâches dont le résultat est un étiquetage uniforme du graphe par une étiquette fonction du graphe sous-jacent (chapitre 8). Un exemple simple et important de problème de ce type est le problème de la Majorité où il s'agit pour tous les sommets de déterminer l'étiquette majoritaire parmi les étiquettes initialement présentes sur le réseau. Nous prouverons l'équivalence avec les fonctions scalaires autostabilisantes lorsque l'on n'a aucune connaissance structurelle (Th. 8.4). Deux applications de la caractérisation obtenue dans le cas d'une connaissance structurelle arbitraire (Th. 8.6) seront exposées : l'étude de l'équivalence entre connaissances structurelles, c'est-à-dire comment déduire une caractéristique particulière du graphe à partir d'une autre caractéristique ; l'étude des problèmes statistiques, c'est-à-dire ceux qui, comme le problème de la Majorité, dépendent de la densité des étiquettes présentes sur le réseau.

Ces résultats ont fait l'objet des articles suivants : [God02] en ce qui concerne l'autostabilisation de l'algorithme de Mazurkiewicz et une étude succincte de sa complexité ; [GMM00] et [GM01] en ce qui concerne la deuxième partie, une version complète [GMM02] est actuellement en cours de publication ; [GM02a] présentait la caractérisation de l'élection ; [GM02b] étudiait l'équivalence des connaissances structurelles. Les résultats combinatoires du chapitre 0 et les extensions de l'algorithme de Mazurkiewicz ont été publiés conjointement dans ces articles.

L'une des conclusions principales de notre étude est, nous semble-t-il, de montrer, en mettant en avant des résultats d'impossibilité, l'intérêt non seulement théorique mais aussi pratique, comme on le verra dans la conclusion du chapitre 7, qu'il y a à expliciter, lors de la description d'algorithmes distribués, les hypothèses effectuées sur la structure du réseau et sur la connaissance supposée de cette structure par les nœuds.

Travaux précédents

Ce travail approfondit en particulier les travaux précédemment réalisés dans le cadre des réétiquetages de graphes. Autour d'Y. Métivier, un grand nombre de résultats d'impossibilités ont été publiés [CM94, MMW97] ainsi que l'étude de la puissance d'expression avec la comparaison entre différentes manières d'exprimer les règles (par contexte interdit ou par priorités) [LMS95a, LMS99]. La reconnaissance de familles de graphes par calculs locaux a été initialement abordée dans [LMZ95]. On trouve également des études de possibilités, comme l'étude du problème de la Majorité (en tant que problème de décision) dans [LMS95b]. On trouvera ici une réciproque des principaux théorèmes d'impossibilité. Une étude des revêtements et de la manière d'en construire certains est présentée dans [Bot97].

Les idées présentées ici ont pour origine, outre [Ang80] et les articles précités de réécritures de graphes, principalement les articles d'A. Mazurkiewicz [Maz97] pour son algorithme d'énumération, véritable algorithme "à tout faire" qui se révélera capable de réaliser tout ce que l'on n'aura pas prouvé irréalisable, de Métivier-Muscholl-Wacrenier [MMW97] et Métivier-Tel [MT00] pour l'étude de la détection de la terminaison à l'aide des quasi-revêtements. Ce qui est original est leur adaptation et intégra-

tion en une même étude. Les idées et preuves de [MMW97] et [MT00] ont également été simplifiées.

Concernant d'autres modèles, il faut noter en particulier le travail fondateur d'Angluin [Ang80]. Celui-ci a été poursuivi par Johnson et Schneider [JS85], où le concept de "vue locale" d'un nœud a été introduit. Approfondissant particulièrement cette idée, Yamashita et Kameda ont étudié ce qui pouvait être calculé [YK96a, YK98] ou réalisé [YK96b, YK96c] sur des réseaux anonymes. Leur modèle est synchrone et leur étude porte sur la caractérisation de la réalisabilité de certains problèmes particuliers (élection, construction d'arbres couvrants, problèmes de décisions) en fonction de certaines connaissances structurelles particulières (sans aucune connaissance, borne sur la taille, la taille, la topologie). Poursuivant cette étude de façon plus systématique, Boldi et Vigna ont caractérisé les fonctions calculables sur un réseau anonyme [BV01] dans le cas synchrone, asynchrone et entrelacé. Le cas entrelacé est celui qui est le plus proche de notre étude, et dans ce cas, selon le propre aveu des auteurs, la caractérisation est assez "redoutable". Leur but principal est la caractérisation des fonctions autostabilisantes [BV02b], qui n'est pas encore complète dans le cas asynchrone. Ils ont également, avec S. Shammah et P. Gemmel, B. Condenotti et J. Simon caractérisé dans [BCG$^+$96], poursuivant un travail entamé par ces trois derniers auteurs dans [CGS95], les cas où l'élection était réalisable. Cependant, leur algorithme nécessite la connaissance de la taille, et lorsqu'aucune connaissance structurelle n'est disponible, l'algorithme est à terminaison implicite, c'est-à-dire qu'au cours de l'exécution, plusieurs nœuds peuvent simultanément être dans l'état ÉLU, ce n'est qu' "à la limite" qu'il n'y a qu'un seul sommet ÉLU. On trouvera dans [BCG$^+$96] une introduction particulièrement instructive dont je me suis en partie inspiré pour établir cet historique. Concernant un panorama plus général sur l'algorithmique distribuée, on pourra se reporter à [Tel00, Lyn96, Pel00] et en particulier à [Lav95] pour une revue assez détaillée des différentes hypothèses et différents modèles que l'on peut utiliser pour décrire un algorithme distribué.

Chapitre 0

Préliminaires

Nous commencerons par rappeler certaines définitions élémentaires sur les graphes et ainsi en profiter pour poser les notations que l'on retrouvera fréquemment. Nous définirons ensuite ce que nous entendons par algorithme distribué et comment cela peut être modélisé au moyen de réétiquetages de graphes. Finalement, nous présenterons et illustrerons des outils de la théorie des graphes spécifiques à notre étude, en particulier les revêtements et quasi-revêtements.

0.1 Graphes

Les définitions et notations utilisées sont standard [Ber83, Gib85, Ros00].

0.1.1 Définitions

Définition 0.1. *Un* graphe simple non orienté G *est défini par un ensemble de* sommets, *noté* $V(G)$, *et un ensemble de paires de sommets distincts, noté* $E(G)$. *Les éléments de* $E(G)$ *sont des* arêtes.

Une arête ayant deux sommets u *et* v *pour extrémités sera notée* $\{u, v\}$. *Les sommets* u *et* v *sont dits* adjacents *ou* voisins.

Un graphe sera dit fini *si l'ensemble de ses sommets et de ses arêtes est fini.*

Dans toute la suite, le terme *graphe* désigne, sauf précision contraire, un graphe fini simple (*i.e.* non orienté, sans boucle et sans arête multiple).

Étant donné deux graphes G et H, on définit les notions suivantes.

Définition 0.2. *Le graphe* H *est un* sous-graphe partiel *de* G *si* $V(H) \subseteq V(G)$ *et* $E(H) \subseteq E(G)$.

Le graphe H *est un* [*sous-graphe !induit]sous-graphe induit de* G *si* H *est sous-graphe partiel de* G *et si* H *contient toutes les arêtes de* G *dont les extrémités sont dans* $V(H)$, *i.e.*

$$\forall x, y \in V(H), \{x, y\} \in E(H) \Leftrightarrow \{x, y\} \in E(G).$$

Définition 0.3. *Un* chemin *dans un graphe est une suite de sommets voisins, c'est-à-dire une suite* $\Gamma = (u_0, u_1, \ldots, u_n)$ *telle que pour tout* i,

0.3.i $\{u_i, u_{i+1}\} \in E(G)$,

Les sommets u_0 *et* u_n *sont les extrémités de ce chemin. On dit que* Γ *est un chemin de* u_0 *à* u_n*. Si les sommets sont distincts, excepté peut-être pour les extrémités, alors le chemin sera dit* simple.

Si, pour tout i,

0.3.ii $u_{i-1} \neq u_{i+1}$,

le chemin sera dit non bègue *[BV02a].*

La longueur d'un chemin est égale au nombre d'arêtes parcourues, ici n*. On note* $\Gamma_G(u)$ *l'ensemble des chemins sur* G *issus du sommet* u*.*

Lorsque les extrémités d'un chemin sont confondues, ce chemin est appelé cycle.

Définition 0.4. *Un graphe* G *est* connexe *si quels que soient* u *et* v*, deux sommets de* G*, il existe un chemin entre* u *et* v*.*

Un *arbre* est un graphe connexe et sans cycle. Un *anneau* est un graphe constitué d'un cycle simple.

Définition 0.5. *Soient* u *et* v *deux sommets d'un graphe connexe* G*. On appelle distance de* u *à* v*, notée* $d_G(u, v)$*, la longueur du plus court chemin de* u *à* v*. Le* diamètre *de* G *est* $\Delta(G) = \max_{u,v \in V(G)}(d(u, v))$.

Définition 0.6. *Le* voisinage d'un sommet u *dans un graphe* G*, noté* $N_G(u)$*, est l'ensemble des sommets adjacents à* u *dans* G *:*

$$N_G(u) = \{v \in V(G) | \{u, v\} \in E(G)\}.$$

On notera le cardinal d'un ensemble S *indifféremment par* $\mathrm{card}(S)$ *ou par* $|S|$*. Le* degré *de* u *dans* G*, noté* $deg_G(u)$*, est le nombre de voisins de* u *dans* G *i.e. le cardinal du voisinage de* u *:* $deg_G(u) = |N_G(u)|$*. Un graphe dont tous les sommets ont le même degré* d *sera dit* $d-$régulier.

Définition 0.7. *On appellera* boule *de centre* u *et on notera* $B_G(u)$*, le graphe constitué des sommets* $\{u\} \cup N_G(u)$ *et des arêtes issues de* u*.*

Soit un graphe G *et* u *un de ses sommets. Soit* $k \in \mathbb{N}$*, la boule de centre* v *et de rayon* k *est le sous-graphe constitué des chemins issus de* v *de longueur inférieure ou égale à* k*. Elle sera également appelée* $k-$boule *de centre* u *et notée* $B_G(u, k)$*. Elle contient les sommets et arêtes suivants :*

$$\begin{aligned} V\left(B_G(u,k)\right) &= \{v \in V(G) | d_G(u,v) \leq k\} \\ E\left(B_G(u,k)\right) &= \{\{v,w\} \in E(G) \mid d(u,v) \leq k-1, d(u,w) \leq k\} \end{aligned}$$

Si l'on ajoute les arêtes reliant les sommets à distance exactement k *on obtient la* boule fermée

de centre u *et de rayon* k*, notée* $\overline{B_G}(u, k)$ *:*

$$\begin{aligned} V\left(\overline{B_G}(u,k)\right) &= \{v \in V(G) | d_G(u,v) \leq k\} \\ E\left(\overline{B_G}(u,k)\right) &= \left\{\{v,w\} \in E(G) \mid v,w \in V\left(\overline{B_G}(u,k)\right)\right\}. \end{aligned}$$

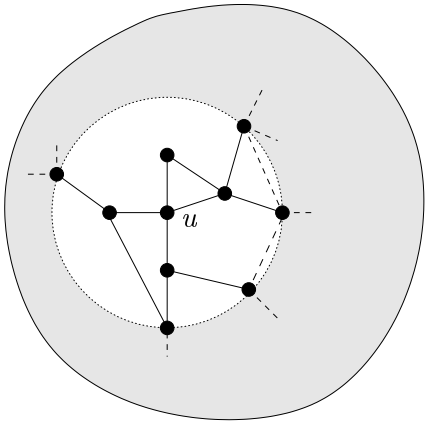


FIG. 0.1 – 2-boule (ouverte) de centre u

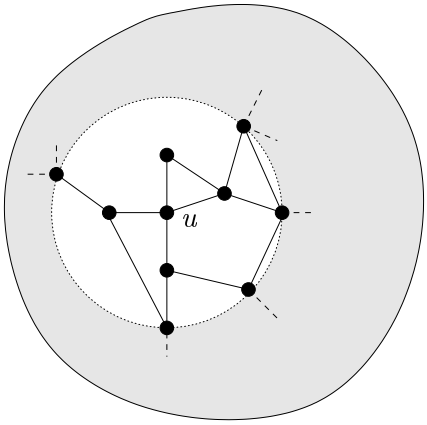


FIG. 0.2 – 2-boule fermée de centre u

On pourra se reporter aux figures 0.1 et 0.2.

Remarques 0.8. On a, bien sûr, $B_G(u) = B_G(u, 1)$.

Lorsque le graphe G sera clairement précisé d'après le contexte, on n'indiquera pas l'indice et l'on abrégera les notations précédentes en $d(u, v)$, $B(u, k)$ ...

Définition 0.9. *Un* homomorphisme *entre G et H est une application φ de $V(G)$ dans $V(H)$ telle que si $\{x, y\} \in E(G)$ alors $\{\varphi(x), \varphi(y)\} \in E(H)$.*

Définition 0.10. *De plus, si φ est une application bijective et si φ^{-1} est un homomorphisme de H vers G, alors φ définit un* isomorphisme *entre les deux graphes G et H. On dit alors que G et H sont isomorphes.*

Définition 0.11. *On appellera* classe ou famille de graphes*, toute famille $\mathcal{F}$, au sens de la théorie des ensembles, close par isomorphisme.*

On considérera essentiellement des graphes finis et connexes.

Exemple 0.12. Un anneau de taille $n \in \mathbb{N}$, noté A_n est le graphe suivant, où $k \mod n$ dénote le reste de la division euclidienne de k par n :

$$\begin{aligned} V(A_n) &= \{1, \ldots, n\} \\ E(A_n) &= \{\{i, j\} \mid i - j = \pm 1 \mod n\} \end{aligned}$$

0.1.2 Graphes étiquetés

On travaille principalement sur des graphes dont les sommets et les arêtes sont étiquetés par un ensemble d'étiquettes L, qui peut être infini.

Un graphe étiqueté sera noté de la manière suivante (G, λ), où G est un graphe (dit *graphe sous-jacent*) et λ une application (dite *étiquetage*) de $V(G) \cup E(G)$ dans L. Lorsqu'il ne sera pas nécessaire d'expliciter la fonction d'étiquetage, nous adopterons la notation $\mathbf{G}$, $\mathbf{H}$, ... pour désigner des graphes étiquetés.

On note $\lambda(G)$ l'ensemble des étiquettes présentes sur G. Un exemple simple de tels graphes et que nous utiliserons fréquemment est le suivant. Soit G un graphe et

α une étiquette de L. Alors Λ_α^G est l'étiquetage de G tel que chaque arête et sommet sont étiquetés par α. Dans ce cas on dit que l'étiquetage est *uniforme* en α. On abrégera fréquemment en (G, Λ_α).

Soient (G, λ) et (G', λ') deux graphes étiquetés. Alors (G, λ) est un sous-graphe de (G', λ'), ce qui sera noté $(G, \lambda) \subseteq (G', \lambda')$, si G est un sous-graphe de G' et λ est la restriction de l'étiquetage λ' à $V(G) \cup E(G)$.

Définition 0.13. *Une application $\varphi : V(G) \cup E(G) \to V(G') \cup E(G')$ est un homomorphisme de (G, λ) sur (G', λ') si φ est un homomorphisme de graphe de G sur G' qui conserve l'étiquetage, i.e. tel que $\lambda'(\varphi(x)) = \lambda(x)$ pour tout $x \in V(G) \cup E(G)$. L'application φ est un isomorphisme, si elle définit, sur les graphes sous-jacents, un isomorphisme de graphes. On écrira alors $(G, \lambda) \simeq (G', \lambda')$.*

Une *occurrence* de (G, λ) dans (G', λ') est un isomorphisme φ entre (G, λ) et un sous-graphe (H, η) de (G', λ').

Définition 0.14. *On appellera* classe ou famille de graphes étiquetés, *toute famille $\mathcal{F}$, au sens de la théorie des ensembles, close par isomorphisme de graphes étiquetés.*

On notera $\mathcal{G}_L$ la classe des graphes L-étiquetés.

On sera amené à utiliser des sur-étiquetages de graphes étiquetés, ou encore étiquetage produit :

Définition 0.15. *Soit $\mathbf{G} = (G, \lambda)$ un graphe étiqueté. On appelle* sur-étiquetage $\mu : V(\mathbf{G}) \longrightarrow L$ *de $\mathbf{G}$ le graphe étiqueté $(\mathbf{G}, \mu) = (G, \lambda \times \mu)$. Dans ce cas on appelle* registre, *les composantes λ ou μ des étiquettes.*

Par la suite, en assimilant tout graphe non étiqueté G à (G, Λ_ε), où ε dénote le mot vide, on se ramènera à ne manipuler que des graphes étiquetés. Par conséquent, sauf mention contraire, *le terme "graphe" désignera indistinctement un graphe non-étiqueté ou étiqueté.*

0.2 Réétiquetages

Les calculs locaux considérés ici peuvent être décrits de la manière suivante. Lors d'un calcul distribué, les nœuds changent d'état en fonction de leur état et de ceux de leurs voisins.

0.2.1 Définition

Soit $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ une relation binaire sur $\mathcal{G}$. Alors $\mathcal{R}$ est un système de réécriture de graphes. On imposera que $\mathcal{R}$ soit fermé par isomorphisme, *i.e.* si $(G, \lambda)\mathcal{R}(G', \lambda')$, et si $(G_1, \lambda_1) \simeq (G, \lambda)$ alors $(G_1, \lambda_1)\mathcal{R}(G_1', \lambda_1')$ pour un certain graphe étiqueté $(G_1', \lambda_1') \simeq (G', \lambda')$. Par la suite $\mathcal{R}^*$ sera la fermeture transitive de $\mathcal{R}$. Le graphe étiqueté (G, λ) est $\mathcal{R}-$*irréductible* (ou irréductible si $\mathcal{R}$ est fixé) s'il n'existe aucun étiquetage λ' de G tel que $(G, \lambda)\mathcal{R}(G, \lambda')$.

Soit $(G, \lambda) \in \mathcal{G}_L$, alors $\text{Irred}_{\mathcal{R}}((G, \lambda))$ est l'ensemble des graphes $\mathcal{R}-$irréductibles en relation $\mathcal{R}^*$ avec (G, λ). La relation $\mathcal{R}$ est dite noethérienne s'il n'existe pas de chaîne infinie $(G, \lambda_1)\mathcal{R}(G, \lambda_2)\mathcal{R}\ldots$.

Définition 0.16. *Soit $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ un système de réécriture de graphes.*

1. *$\mathcal{R}$ est un* système de réétiquetage *de graphes si pour tout couple de graphes en relation alors les graphes sous-jacents sont égaux (égaux, et pas seulement isomorphes), i.e. :*

$$(G, \lambda)\mathcal{R}(H, \lambda') \quad \Longrightarrow \quad G = H.$$

2. *$\mathcal{R}$ est* local *si $\mathcal{R}$ ne modifie les étiquettes que dans une boule de rayon 1, i.e. $(G, \lambda)\mathcal{R}(G, \lambda')$ implique qu'il existe un sommet $v \in V(G)$ tel que*

$$\lambda(x) = \lambda'(x) \text{ pour tout } x \notin V(B_G(v)) \cup E(B_G(v)).$$

La prochaine définition exprimera qu'un système de réétiquetage de graphes local $\mathcal{R}$ est *localement engendré* si sa restriction aux boules de rayon 1 détermine son comportement sur tout graphe.

Définition 0.17. *Soit $\mathcal{R}$ un système de réétiquetage de graphes. Alors $\mathcal{R}$ est* localement engendré *si les conditions suivantes sont satisfaites :*

Pour tous graphes étiquetés (G, λ), (G, λ'), (H, η), (H, η') pour tous sommets $v \in V(G)$, $w \in V(H)$ tels que les boules $B_G(v)$ et $B_H(w)$ sont isomorphes via $\varphi : V(B_G(v)) \longrightarrow V(B_H(w))$, avec $\varphi(v) = w$,

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

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

Dans la suite de ce mémoire, nous n'utiliseront que des systèmes de réétiquetage de graphes localement engendrés. Le comportement de tels systèmes étant déterminé par leur comportement sur les boules, un système de réétiquetage sera donc décrit par la donnée d'un ensemble de couples de boules étiquetées, de graphe non étiqueté sous-jacent identique. La première composante indique la *condition préalable* et la seconde composante le *réétiquetage*. On imposera que cet ensemble de boules étiquetées soit "raisonnable", c'est-à-dire qu'il définisse une fonction des conditions préalables récursive au sens de la théorie de la calculabilité [CL93], sans que cela ne soit nécessairement redit expressément.

Étant donné un système de réétiquetage de graphes $\mathcal{R}$, un *pas de calcul* sur le graphe $\mathbf{G}$ est l'application en une boule de $\mathbf{G}$ d'une règle $\mathcal{R}_i = (C_i, R_i)$ de $\mathcal{R}$, c'est-à-dire la substitution, dans $\mathbf{G}$, d'une boule de rayon 1 isomorphe à la condition préalable C_i par le réétiquetage R_i. Une exécution de $\mathcal{R}$ sur le graphe $\mathbf{G}$ est alors une suite $\mathbf{G} = \mathbf{G}_0\mathcal{R}\mathbf{G}_1\mathcal{R}\mathbf{G}_2\mathcal{R}\cdots\mathcal{R}\mathbf{G}_i\mathcal{R}\cdots$ de pas de calcul tels que pour tout i, $\mathbf{G}_i\mathcal{R}\mathbf{G}_{i+1}$.

On utilisera indistinctement les termes réécritures et réétiquetages, étant entendu qu'ici il s'agit toujours, à strictement parler, de réétiquetages.

Remarques 0.18. Il faut souligner que les définitions précédentes ne précisent pas *a priori* les conditions de terminaison, un calcul se termine lorsque plus aucune règle ne peut s'appliquer, il s'agit d'une *terminaison implicite*. Un sommet v peut également rechercher à détecter localement la terminaison *locale* (son étiquette $\lambda(v)$ ne sera plus modifiée) ou *globale* (le réseau a atteint un état irréductible). Nous reviendrons en détail sur ces terminaisons au Chapitre 8.

Il faut également noter que puisque nous raisonnerons sur les chaînes de réétiquetages, et en particulier sur leur dernier terme - irréductible -, les exécutions seront (implicitement) supposées "équitables" (fair), c'est à dire que le système atteindra toujours un état irréductible.

Notations

Une exécution de l'algorithme est ainsi déterminée par une succession de règles de $\mathcal{R}$ appliquée en une boule particulière. On adoptera donc la notation suivante pour une exécution ρ de $\mathcal{R} = \{\mathcal{R}_j \mid j \in J\}$:

$$\rho \quad = \quad \mathcal{R}_{i_1} B_{i_1} \mathcal{R}_{i_2} B_{i_2} \cdots$$

où $\mathcal{R}_{i_j} \in \mathcal{R}$ et B_{i_j} est la boule de rayon 1 où le pas de calcul est effectué.

Lorsqu'il n'y a pas d'ambiguïté dans le réétiquetage des voisins, on indiquera uniquement le centre de la boule et on notera :

$$\rho \quad = \quad \mathcal{R}_{i_1} u_{i_1} \mathcal{R}_{i_2} u_{i_2} \cdots$$

où $\mathcal{R}_{i_j} \in \mathcal{R}$ et u_{i_j} est le centre de la boule de rayon 1 où le pas de calcul est effectué. L'ensemble des exécutions partielles possibles sur un graphe $\mathbf{G}$ est notée $\Xi_{\mathbf{G}}(\mathcal{R})$.

Si l'on s'intéresse à la complexité, c'est-à-dire ici, en première approximation, au nombre de pas de calcul, on notera la chaîne $\mathcal{R}_{i_1}\mathcal{R}_{i_2}\cdots$ et $|\mathcal{R}_{i_1}\mathcal{R}_{i_2}\cdots|$ correspondra à sa longueur, c'est-à-dire au nombre de termes $\mathcal{R}_{i_j}$.

0.2.2 Sérialisation

Les notions et notations de suites de réétiquetages définies ci-dessus correspondent de manière évidente à une notion de calcul *séquentiel*. Il est important de noter qu'un système de réétiquetage de graphes permet également des réécritures parallèles, puisque des boules qui ne se recouvrent pas peuvent être réétiquetées indépendamment. On peut donc définir une exécution distribuée en disant que deux pas de réétiquetages consécutifs appliqués à des boules disjointes peuvent être effectués dans n'importe quel ordre. On dit que de tels pas commutent et peuvent être exécutés de manière concurrente.

Plus généralement, deux suites de réétiquetages partant du même graphe étiqueté et telles que l'on peut passer de l'une à l'autre par de telles commutations aboutiront au même résultat. Donc notre notion de suite de réécritures peut être vue comme une *sérialisation* [Maz87] d'une exécution distribuée donnée.

Ce modèle est clairement asynchrone : plusieurs pas de réétiquetage *peuvent* être effectués en même temps(mais ne doivent pas nécessairement l'être). Par la suite, on présentera essentiellement les réécritures comme étant séquentielles, mais il sera important de se rappeler qu'elles pourrait être effectuées de manière distribuée.

0.2.3 Un premier exemple : $(d+1)$coloration de graphes $d-$réguliers

Cet exemple nous permet de donner les conventions de définitions de système de réétiquetage de graphes que nous adopterons au cours de notre exposé. Si le nombre de règles est fini nous les décrirons toutes sous la forme *Condition préalable / Réétiquetage*. Nous pourrons également décrire une famille de règles de manière générique sous la forme *Condition préalable générique / Réétiquetage générique*. Dans ce cas, on considérera une boule $B(v_0)$ de centre générique v_0, et l'on fera référence aux sommets voisins et / ou au centre sous la forme $\exists v \in B(v_0)$, ou $\forall v \in B(v_0)$. Si $\lambda(v)$ est la valeur préalable du registre, alors $\lambda'(v)$ décrira la valeur de ce registre après réétiquetage. On omettra de mentionner en général, pour simplifier la lecture des règles, les registres qui ne sont pas réécrits. Ce qui signifie que si $\lambda'(v)$ n'est pas spécifié dans la règle pour un certain v, alors $\lambda'(v) := \lambda(v)$. Tout ceci doit être considéré comme une convention de description de systèmes de réétiquetage de graphes, et non pas comme la spécification formelle d'un langage de description des systèmes de réétiquetage de graphes. Ceci se trouve en dehors du champ de notre étude actuelle.

Pour être plus concret, considérons le système de réétiquetage suivant, COLO_d, qui construit une $(d+1)-$coloration d'un graphe $d-$régulier. L'étiquette de chaque sommet v est noté $c(v)$. Les "couleurs" utilisées sont les entiers de $[1, d+1]$, chaque sommet est initialement étiqueté 0. La règle suivante s'interprète comme "si v_0 est étiqueté 0 alors v_0 prend comme étiquette la plus petite étiquette n'apparaissant pas sur l'un de ses voisins".

COLO_d–1 : $(d+1)$**Coloration**
Condition préalable :
$- \ c(v_0) = 0$
Réétiquetage :
$- \ c'(v_0) := \min\left([1, d+1] \setminus \{c(v) \mid v \in B(v_0), c(v) \neq 0\}\right)$

La Figure 0.3 présente, sous forme graphique, quelques règles de COLO_3.

Exemple 0.19. Les figures suivantes présentent un exemple d'éxécution commentée du système de réétiquetage de graphes COLO_3 sur le cube.

L'étiquetage initiale est le suivant :

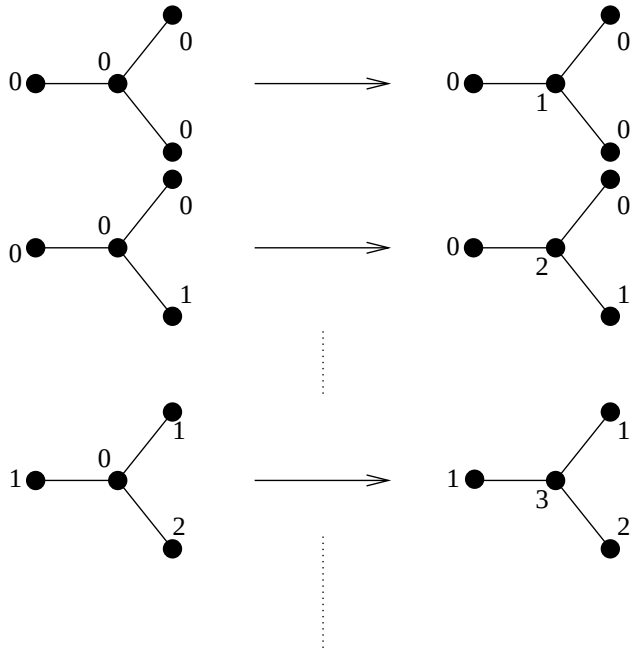
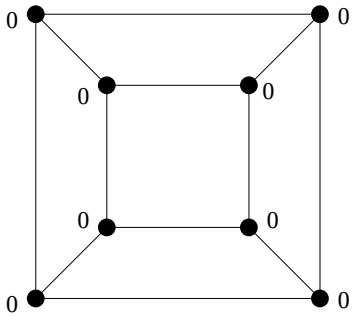
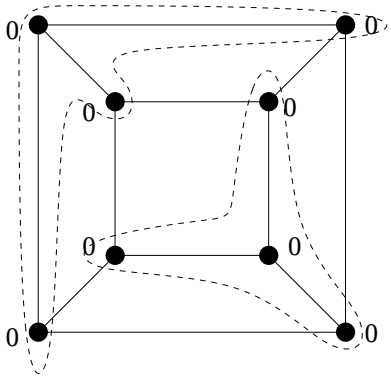


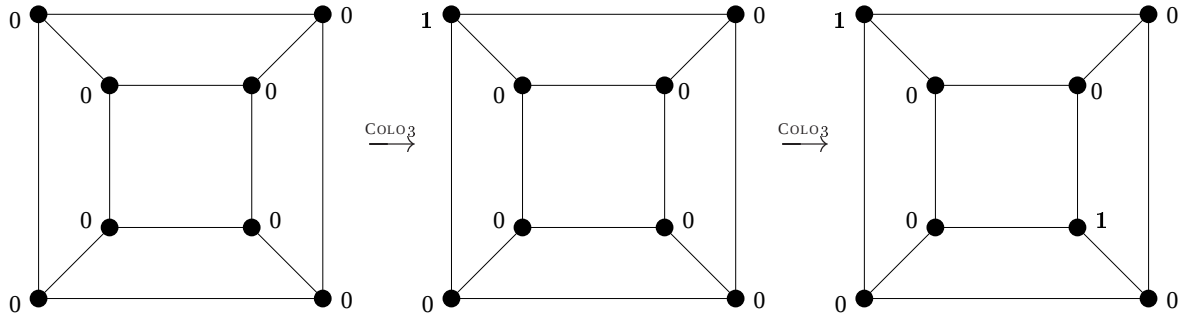
FIG. 0.3 – Représentation graphique de quelques règles. On notera qu'ici seul le centre de la boule est réécrit.



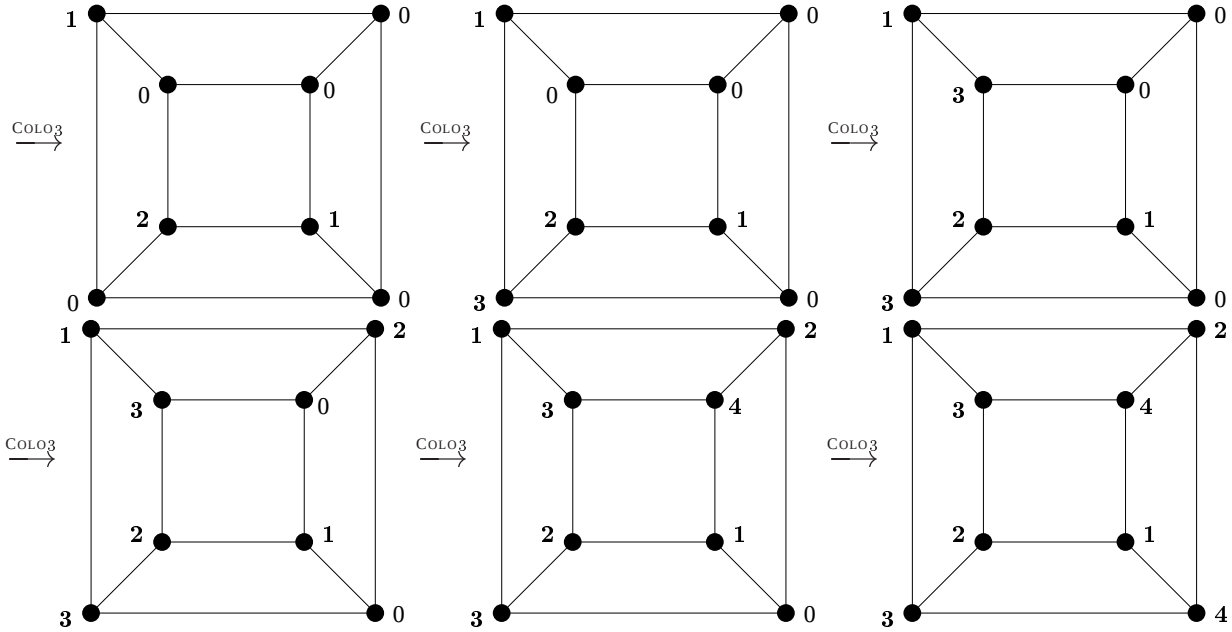
Deux boules où une règle peut être appliquée sont indiquées sur le schéma suivant.



Le réétiquetage correspondant est alors le suivant.



La suite de l'exécution est, par exemple,



Ici la correction de l'algorithme est bien entendu garantie par le fait que l'ensemble sur lequel le minimum est pris n'est jamais vide.

0.2.4 Implémenter les systèmes de réétiquetages : le projet ViSiDiA

Au LaBRI, sous la direction de M. Mosbah, un projet logiciel pour la simulation et la visualisation d'algorithmes distribués à l'aide de réétiquetage de graphes est actuellement en cours (ViSiDiA, Visualization and Simulation of Distributed Algorithms – `http ://www.labri.fr/Recherche/LLA/visidia/`). On pourra se reporter à [BGM+01] pour une description complète de l'outil.

Cet outil implémente les règles des réétiquetages locaux dans un modèle à passage de messages en se basant pour réaliser les synchronisations locales sur différents algorithmes probabilistes de rendez-vous [MSZ00]. L'algorithme de Mazurkiewicz décrit et intensément utilisé ici y a été implémenté par A. Sellami et M. Mosbah [MS01]. Sur la page web, on trouvera également un exemple animé très didactique de construction d'arbre couvrant par un système de réétiquetage :
`http ://www.labri.fr/Recherche/LLA/visidia/Examples/examples.html`

0.3 Connaissances Structurelles

Lors de l'élaboration d'algorithmes distribués, il est fréquent de poser certaines hypothèses simplificatrices sur la topologie du réseau sous-jacent comme le fait qu'il possède une topologie particulière (anneau, grille, ...), que chaque nœud dispose d'une identité distincte, ou encore que chaque nœud ait connaissance de certaines propriétés structurelles du réseau (sa topologie, sa taille, une borne sur sa taille,...). Souvent, en terme de calculabilité, le fait de poser certaines hypothèses simplifie de manière conséquente le problème. Par exemple, on sait que dans un réseau anonyme (*i.e.* sans identité), on ne peut pas toujours élire, alors que si l'on suppose l'existence d'identités distinctes, on peut élire quelle que soit la topologie du graphe, la difficulté devenant alors de concevoir les algorithmes les plus rapides. Le thème principal de cette thèse est l'étude de ce qui est ou n'est pas réalisable par calculs locaux et, par conséquent, il semble judicieux de pouvoir définir rigoureusement quelles sont exactement les hypothèses nécessaires. Par exemple, en ce qui concerne l'élection, entre aucune hypothèse et l'existence d'identités distinctes (qui est une spécification exigeante et coûteuse à maintenir sur un réseau), y a-t-il une hypothèse *intermédiaire* qui permet de pouvoir élire. Nous reviendrons en détail sur ces questions au chapitre 7, en particulier dans sa conclusion.

Une connaissance structurelle est simplement une fonction qui associe à tout graphe une étiquette.

Définition 0.20. *On appelle connaissance structurelle toute fonction récursive* $\iota : \mathcal{G}_L \longrightarrow L$

Une connaissance structurelle peut "encoder" toute connaissance dépendant de la structure globale du graphe. On étiquettera initialement et uniformément $\mathbf{G}$ par $\Lambda_{\iota(\mathbf{G})}$ pour exprimer formellement cette connaissance. On s'intéressera plus particulièrement ici à certaines connaissances de la liste, non exhaustive, ci-dessous.

- Connaissances "Métriques"
  - Une *borne supérieure* b *sur la taille* du graphe : il s'agit d'une fonction b telle que $b(G) \geq |V(G)|$ pour tout G.
  - Une *bonne borne* : il s'agit d'une borne b telle que pour tout graphe G, $|V(G)| \leq b(G) < 2|V(G)|$.
  - La taille, *i.e.* le nombre de sommets $\mathrm{card}(V(G))$.
  - Le diamètre $\Delta(G)$ du graphe G. Une *borne sur le diamètre*.
- "Caractéristique" ou connaissance booléenne : savoir si le graphe est ou n'est pas dans une classe donnée $\mathcal{C}$. On note $\chi_{\mathcal{C}}$ la connaissance structurelle associant chaque graphe de $\mathcal{C}$ à l'étiquette VRAI et chaque graphe de $\mathcal{G}\backslash\mathcal{C}$ à l'étiquette FAUX. En utilisant de telles fonctions, l'on peut également formaliser, avec nos définitions, le problème de, par exemple, calculer la taille dans une classe particulière.
- Connaissance de la topologie (par exemple, la matrice d'adjacence du graphe sous-jacent est donnée).

Si il n'y a aucune connaissance disponible, comme, par exemple au chapitre 4, on utilisera alors l'étiquette ε. Dans tous les cas, ι devra être récursive au sens de la théorie de la calculabilité.

Remarque 0.21. Formellement, les connaissances structurelles auxquelles nous nous intéresserons sont *scalaires*, au contraire de connaissances, ou informations, telles que la présence d'identifiants uniques, ou encore la présence d'un sommet unique étiqueté "initiateur". Le lecteur devra garder à l'esprit cette distinction entre connaissance structurelle (une propriété structurelle du graphe sous-jacent qui est uniformément encodé sur tout le réseau) et information structurelle (informations distributivement encodées, telles que identités distinctes ou sens de la direction, ...).

On pourra traiter indirectement le second cas en considérant des familles de graphes étiquetés particulières, par exemple, la famille des graphes avec identités.

0.4 Revêtements

L'un des buts de cette thèse est de fournir une approche permettant de caractériser ce qui est calculable de manière distribuée et ce qui ne l'est pas . Et comme le souligne N. Lynch dans [Lyn89], ce qui est impossible en algorithmique distribuée l'est en règle générale pour des raisons de "similarités", la définition des similarités étant à chaque fois spécifique au problème considéré. Ici, les réseaux "similaires" se comportent de manière similaires car l'on peut simuler ce que l'on fait avec l'un sur l'autre. Les propriétés que l'on cherchera à prouver dans le modèle de réétiquetage précédemment défini utiliseront, implicitement, la notion de simulation. Les outils permettant de manipuler ces similarités et simulations sont des morphismes de graphes conservant les propriétés locales sur lesquelles s'appuient les règles de réétiquetages, *i.e.* conservant les boules : les revêtements.

Nous commencerons par donner les définitions et quelques propriétés immédiates. Puis nous donnerons le lien avec les systèmes de réétiquetages. Nous détaillerons ensuite davantage les propriétés combinatoires de ces morphismes.

0.4.1 Définitions et premières propriétés

Définition 0.22. *Soit γ un homomorphisme surjectif de G sur H. Le morphisme γ est un* revêtement *si pour tout sommet $v \in V(G)$, γ induit un isomorphisme de $B_G(u)$ sur $B_H(\gamma(u))$. Un revêtement de G sur H est dit* propre *si G et H ne sont pas isomorphes.*

Remarque 0.24. Un revêtement est exactement un revêtement au sens classique de la topologie, voir [Mas91]. A noter que pour parler rigoureusement, le revêtement est le morphisme. Par extension et abus de langage, on parlera d'un graphe comme étant un revêtement d'un graphe donné.

Définition 0.25. *Un graphe est* minimal *s'il n'est revêtement propre d'aucun autre graphe. La famille des graphes minimaux sera noté $\mathcal{G}_{\min}$.*

On étend la notion de revêtement aux graphes étiquetés de la manière suivante. Le morphisme γ est un revêtement de (G, λ) sur (H, λ'), si γ est un homomorphisme de (G, λ) sur (H, λ') dont la restriction à $B_G(v)$ est un isomorphisme de $(B_G(v), \lambda_{|B_G(v)})$ sur $(B_H(\gamma(v)), \lambda'_{|B_H(\gamma(v))})$.

Exemple 0.23. Un premier exemple simple de revêtement est donné par la figure 0.4. L'image sur H, par le revêtement, de chacun des sommets du graphe G est donnée par la lettre romane correspondante. En outre, on remarquera que l'image de chaque sommet de G est également donné par sa position sur le motif de H. Par la suite, les exemples de revêtements seront implicitement décrits par cette règle de placement géométrique, qui recevra toute sa justification section 0.4.4.

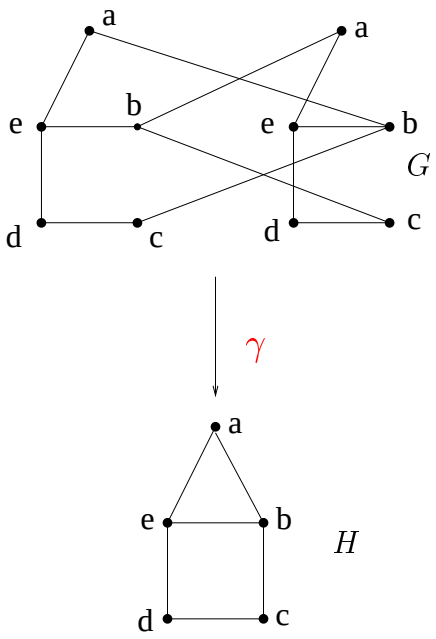


FIG. 0.4 – G est un revêtement de H.

Précisons maintenant l'intuition que l'on peut avoir d'un revêtement. Le lemme suivant découle directement de la définition.

Lemme 0.26. *Soit γ un revêtement de $\mathbf{G}$ sur $\mathbf{H}$. Soit u un sommet de $\mathbf{H}$. Alors $\gamma^{-1}(B_{\mathbf{H}}(u))$ est une collection de boules disjointes isomorphes à $B_{\mathbf{H}}(u)$.*

Démonstration. Par l'absurde, si deux boules s'intersectent, alors en un de leur point d'intersection u, la condition d'injectivité de γ sur $B_{\mathbf{H}}(u)$ n'est pas vérifiée. □

Étant donnée une boule B de $\mathbf{H}$, un certain nombre de "copies conformes" de celle-ci sont présentes sur le graphe $\mathbf{G}$, on appellera ces copies des *relèvements* de B. Voir Fig. 0.5. Un argument de connexité permet de prouver que chaque boule de $\mathbf{H}$ est copiée avec la même multiplicité.

Corollaire 0.27. *Soit $\mathbf{H}$ un graphe connexe et $\mathbf{G}$ un revêtement fini de $\mathbf{H}$ via γ. Alors il existe un entier q tel que, pour tout sommet u de $V(\mathbf{H})$, on a $\mathrm{card}(\gamma^{-1}(u)) = q$*

Démonstration. Par l'absurde, supposons que le cardinal des antécédents ne soit pas constant, alors, par connexité, il existe deux sommets adjacents u et v tel que $\mathrm{card}(\gamma^{-1}(u)) < \mathrm{card}(\gamma^{-1}(v))$, par exemple. Or, d'après le lemme précédent, les antécédents de $B_{\mathbf{H}}(v)$ sont disjointes et isomorphes à $B_{\mathbf{H}}(v)$. Il va manquer un antécédent du voisin u dans au moins une de ces boules préimages. □

Définition 0.28. *On dit alors que γ est un revêtement à q* feuillets.

L'exemple 0.23 est un revêtement à 2 feuillets. On peut décrire *tous* les revêtements d'un graphe d'un nombre de feuillets q donné. Ce sera l'objet de la section 0.4.4.

Les propositions suivantes sont des illustrations de ce qui précède

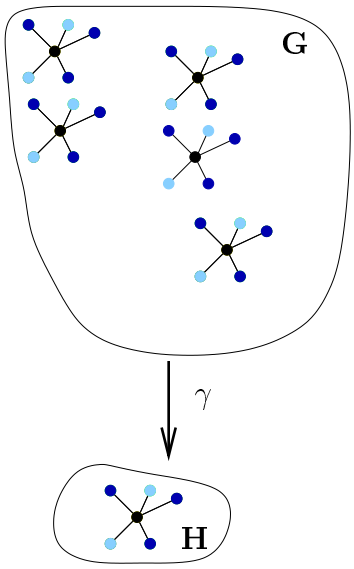


FIG. 0.5 – les antécédents d'une boule de $\mathbf{H}$ forment une collection de boules disjointes de $\mathbf{G}$.

Proposition 0.29. *Soit n, m deux entiers. Alors l'anneau A_n est un revêtement de l'anneau A_m si et seulement si $m \mid n$. Dans ce cas, $\frac{n}{m}$ est le nombre de feuillets.*

Proposition 0.30. *Soit $\mathbf{G}$ un graphe connexe, si $|V(\mathbf{G})|$ et $|E(\mathbf{G})|$ sont premiers entre eux, alors $\mathbf{G}$ est minimal.*

En particulier, les graphes de taille premières sont minimaux, et d'après la Prop. 0.29, un anneau A_n est minimal si et seulement si n est premier.

0.4.2 Revêtements et réétiquetages locaux

Nous allons voir immédiatement comment les revêtements permettent d'étudier les systèmes de réétiquetages. Le lemme suivant fait le lien entre l'existence d'un revêtement et le réétiquetage de deux graphes. Il est d'une importance fondamentale pour notre étude.

Lemme 0.31 (de Relèvement [Ang80]). *Soit $\mathcal{R}$ un système de réétiquetage de graphes et soit $\gamma : \mathbf{G} \longrightarrow \mathbf{H}$ un revêtement. Supposons qu'il existe un graphe $\mathbf{H}'$ tel que $\mathbf{H}\mathcal{R}^*\mathbf{H}'$. Alors il existe un graphe étiqueté $\mathbf{G}'$ (de même graphe sous-jacent que $\mathbf{G}$) tel que*

- *$\mathbf{G}\mathcal{R}^*\mathbf{G}'$,*
- *$\mathbf{G}'$ soit un revêtement de $\mathbf{H}'$.*

Démonstration. Il suffit de montrer le résultat dans le cas $\mathbf{H}\mathcal{R}\mathbf{H}'$. Supposons que le pas de réétiquetage modifie les étiquettes de $B_{\mathbf{H}}(v)$, pour un certain sommet $v \in V(\mathbf{H})$. On peut appliquer la règle de réétiquetage associée à ce pas de réétiquetage à chacune des boules de $\gamma^{-1}(B_{\mathbf{H}}(v))$, puisqu'elles sont disjointes et isomorphes à $B_H(v)$, par lemme 0.26. On obtient ainsi un graphe étiqueté $\mathbf{G}'$ ayant les propriétés voulues. □

La figure suivante résume ce lemme qui signifie intuitivement que pour tout système de réétiquetage de graphes $\mathcal{R}$, on peut simuler sur $\mathbf{G}$ toute exécution de $\mathcal{R}$ sur $\mathbf{H}$.

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

0.4.3 Quasi-revêtements

Les quasi-revêtements ont été introduits dans [MMW97] pour étudier la détection de la terminaison des systèmes de réétiquetage. Ce sont des applications qui se comportent partiellement comme des revêtements.

Définition 0.32. *Soit* $\mathbf{K}$ *et* $\mathbf{H}$ *deux graphes, et soit* δ *une fonction (partielle) de* $V(\mathbf{K})$ *dans* $V(\mathbf{H})$. δ *est un* quasi-revêtement de rayon r *s'il existe un graphe* $\mathbf{G}$ *(éventuellement infini) qui soit revêtement de* $\mathbf{H}$ *via* γ*, et s'il existe* $z_{\mathbf{K}} \in V(\mathbf{K})$, $z_{\mathbf{G}} \in V(\mathbf{G})$ *tels que*

0.32.i il existe un isomorphisme $\varphi : B_{\mathbf{K}}(z_{\mathbf{K}}, r) \longrightarrow B_{\mathbf{G}}(z_{\mathbf{G}}, r)$

0.32.ii Le domaine de définition de δ *contient* $B_{\mathbf{K}}(z_{\mathbf{K}}, r)$*,*

0.32.iii $\delta_{|V(B_{\mathbf{K}}(z_{\mathbf{K}},r))} = \gamma \circ \varphi_{|V(B_{\mathbf{K}}(z_{\mathbf{K}},r))}$

L'entier $s = |B_{\mathbf{K}}(z_{\mathbf{K}}, r)|$ *sera appelée l'*étendue *du quasi-revêtement et le graphe* $\mathbf{G}$ *le* revêtement associé.

On pourra se reporter à la figure 0.6. Il est à noter que cette définition est légèrement différente de celle de [MMW97] au sens où dans notre définition, "le" paramètre du quasi-revêtement est le rayon et non pas l'étendue. Le lemme 0.35 à suivre est par conséquent une adaptation originale du lemme de relèvement.

On effectuera le même abus de langage que pour les revêtements, et l'on pourra dire que $\mathbf{K}$ est un quasi-revêtement de $\mathbf{H}$ de rayon r. On notera qu'en particulier, tout revêtement de $\mathbf{H}$ est également un quasi-revêtement pour n'importe quel rayon $r \in \mathbb{N}$. Et que tout quasi-revêtement de rayon r est aussi quasi-revêtement de rayon $r' \leq r$.

En utilisant les notations précédentes :

Définition 0.33. *On définit le* nombre de feuillets q *d'un quasi-revêtement comme étant la cardinalité minimale des ensembles d'antécédents des sommets de* $\mathbf{H}$ *:*

$$q = \min_{w \in \mathbf{H}} \{\mathrm{card}\left(\{v \in B_{\mathbf{K}}(z_{\mathbf{K}}, r) \mid \delta(v) = w\}\right)\}$$

Définition 0.34. *Un quasi-revêtement est dit* strict *si* $B_{\mathbf{K}}(z_{\mathbf{K}}, r-1) \subsetneq \mathbf{K}$.

Un quasi-revêtement non strict est un revêtement, et dans ce cas les deux définitions du nombre de feuillets coïncident.

Le lemme qui suit est une extension du lemme de Relèvement, si un graphe est un quasi-revêtement, on pourra simuler mais le rayon de simulation diminuera à chaque simulation.

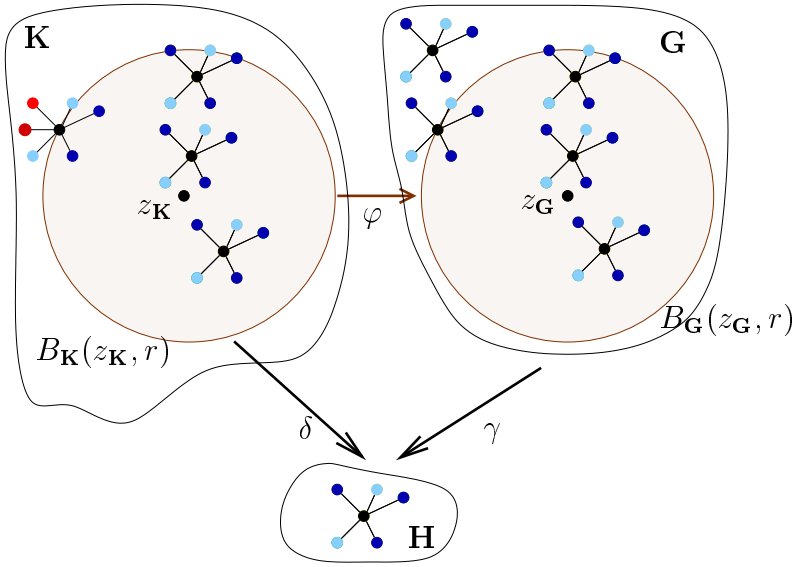


FIG. 0.6 – δ : $\mathbf{K} \longrightarrow \mathbf{H}$ est un quasi-revêtement de rayon r et revêtement associé γ : $\mathbf{G} \longrightarrow \mathbf{H}$

Lemme 0.35 (de Quasi-Relèvement). *Soit $\mathcal{R}$ un système de réétiquetage de graphes et soit $\mathbf{K}$ un quasi-revêtement de $\mathbf{H}$ via γ de rayon $r \geq 2$. Supposons qu'il existe un graphe $\mathbf{H}'$ tel que $\mathbf{H}\mathcal{R}\mathbf{H}'$.*

Alors il existe un graphe $\mathbf{K}'$ tel que
- $\mathbf{K}\mathcal{R}^*\mathbf{K}'$,
- *$\mathbf{K}'$ est un quasi-revêtement de $\mathbf{H}'$ de rayon $r-2$.*

Démonstration. Notons $z_{\mathbf{K}}$ le centre de la boule de rayon r du quasi-revêtement et $\mathbf{G}$ le revêtement associé. Supposons maintenant que le pas de réétiquetage modifie les étiquettes de $B_{\mathbf{H}}(v)$, pour un sommet donné $v \in V(\mathbf{H})$. On peut alors appliquer la règle de réétiquetage correspondante à toutes les boules de $\delta^{-1}(B_{\mathbf{H}}(v))$, on obtient alors $\mathbf{G}'$ et γ'. De même, la règle de réétiquetage s'applique à toutes les boules de $\delta^{-1}(B_{\mathbf{H}}(v))$ qui sont incluses dans $B_{\mathbf{K}}(z_{\mathbf{K}}, r)$ puisqu'elles sont isomorphes à $B_{\mathbf{H}}(v)$. Ceci nous donne $\mathbf{K}'$ et δ' satisfaisant aux propriétés de quasi-revêtement pour un rayon $r-2$: considérons w dans la boule $B_{\mathbf{K}'}(z_{\mathbf{K}}, r-2)$, puisque toute boule de rayon 1 contenant w est elle-même incluse dans $B_{\mathbf{K}}(z_{\mathbf{K}}, r)$, w a la même étiquette que $\gamma'(w)$ et donc φ, qui définit toujours un isomorphisme sur les deux $r-$boules non étiquetées sous-jacentes, est également un isomorphisme des $(r-2)-$boules étiquetées. □

Ainsi, dans le cas d'une relation de quasi-revêtement, on peut, comme dans le cas des revêtements, effectuer des simulations mais seulement un nombre fini de fois.

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{G}' \\ \text{quasi-revêtement de rayon } r \;\downarrow & & \downarrow\; \text{quasi-revêtement de rayon } r\text{-}2 \\ \mathbf{H} & \xrightarrow[\mathcal{R}]{} & \mathbf{H}' \end{array}$$

0.4.4 Description des revêtements à q feuillets : le théorème de Reidemeister

K. Reidemeister a présenté dans [Rei32] une méthode de construction de tous les revêtements d'un graphe donné. Il a en effet montré que tout revêtement d'un graphe G donné se construit à l'aide d'un arbre recouvrant de G dont on fait plusieurs copies et que l'on relie à l'aide des arêtes n'appartenant à l'arbre couvrant. La façon de connecter les copies de l'arbre couvrant est décrite par un ensemble de permutations.

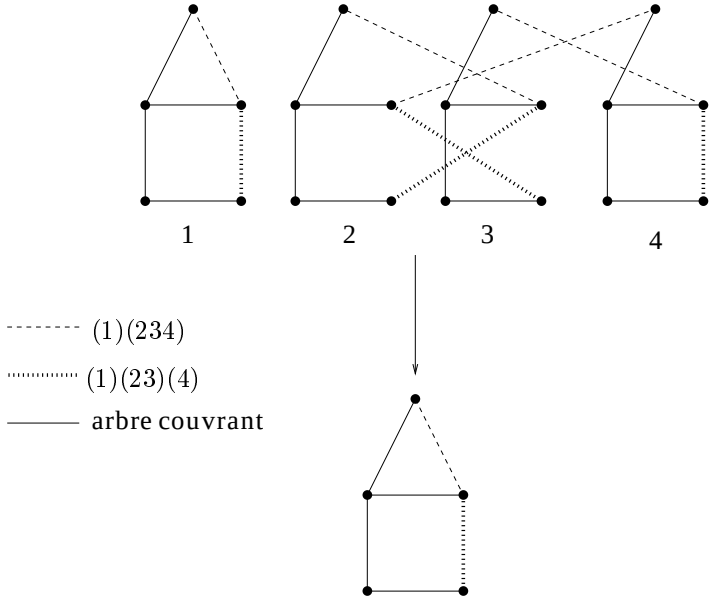


FIG. 0.7 – La construction de Reidemeister pour 4 feuillets. Les permutations sont données comme produit de permutations circulaires.

Soit G un graphe connexe, A un arbre recouvrant de G et q un entier. On note F l'ensemble des arêtes de G qui ne sont pas dans A et on associe à chaque arête de F une permutation de l'ensemble $\{1, \cdots, q\}$.

Nous allons relier q copies de l'arbre recouvrant entre elles grâce aux q copies de chaque arête de F obtenues en permutant leurs extrémités selon la permutation associée à l'arête. Le graphe que l'on obtient de cette façon est un revêtement de G à q feuillets. La figure 0.7 présente une telle construction, les permutations sont décrites par leur décomposition en cycles.

Théorème 0.36 ([Rei32]). *Soit H un graphe et T un arbre couvrant de H. Un graphe G est un revêtement de H si et seulement si il existe un entier q et un ensemble $\Sigma = \{\sigma_{(x,y)},\ \{x,y\} \in E(H) \setminus E(T)\}$ de permutations sur $[1,q]$ (avec la convention que $\sigma_{(y,x)} = \sigma^{-1}_{(x,y)}$) tels que G est isomorphe au graphe $H_{T,\Sigma}$ défini par :*

$$\begin{aligned} V(H_{T,\Sigma}) &= \{\ (x,i) \mid\ x \in V(H),\ i \in [1,q]\ \}, \\ E(H_{T,\Sigma}) &= \{\ \{(x,i),(y,i)\} \mid\ \{x,y\} \in E(T),\ i \in [1,q]\ \} \cup \\ &\quad \{\ \{(x,i),(y,\sigma_{\{x,y\}}(i))\} \mid\ \{x,y\} \in E(H) \setminus E(T),\ i \in [1,q]\ \}. \end{aligned}$$

Remarque 0.37. On obtient *tous* les revêtements à q-feuillets, *i.e.* $(q!)^{m-n+1}$ revêtements pour un graphe à n sommets et m arêtes. Cependant certains des revêtements obtenus ne sont pas connexes et il y a également des répétitions, certaines permutations produisant des graphes isomorphes. Par exemple, pour un anneau A_n, si on ne garde que les revêtement connexes et si on enlève les copies isomorphes, il ne subsiste qu'*un* seul revêtement à q feuillets, A_{qn}, au lieu de $q!$.

0.4.5 Propriétés Combinatoires des Revêtements et Quasi-revêtements

Revêtement Universel

Pour tout chemin $\Gamma = (u_0, \ldots, u_n)$ et tout sommet v, on note Γv le chemin $(u_0, \ldots, u_n, v)$

Définition 0.38. *Soit* $\mathbf{G}$ *un graphe (éventuellement étiqueté). Soit* u *un sommet de* $\mathbf{G}$*. On note* $\widehat{\mathbf{G}}(u)$ *le graphe des chemins non bègues issus de* u *:*

$$\begin{aligned} V(\widehat{\mathbf{G}}(u)) &= \{\Gamma \in \Gamma_{\mathbf{G}}(u) \mid \Gamma \textit{ est non bègue}\} \\ E(\widehat{\mathbf{G}}(u)) &= \{\{\Gamma, \Gamma'\} \mid \Gamma, \Gamma' \in V(\widehat{\mathbf{G}}(u)), \textit{ et il existe un sommet } v \textit{ de } \mathbf{G} \textit{ tel que } \Gamma' = \Gamma v\} \end{aligned}$$

On note $\widehat{\pi}$ *la projection de* $\widehat{\mathbf{G}}(u)$ *sur* $\mathbf{G}$ *qui associe à tout chemin son extrémité finale.*

Proposition 0.39. *La projection* $\widehat{\pi}$ *est un revêtement de* $\widehat{\mathbf{G}}(u)$ *sur* $\mathbf{G}$*.*

Démonstration. Soit v un sommet de $\mathbf{G}$. Soit un chemin $\Gamma = (u_0, \ldots, u_n)$ avec $u_0 = u$ et $u_n = v$. Supposons que Γ n'est pas le chemin vide. Par construction, Γ a pour voisin $(u_0, \ldots, u_{n-1})$. Étant non bègue, il a également pour voisins les chemins de l'ensemble $\{\Gamma w \mid w \in N(v), w \neq u_{n-1}\}$. Par conséquent $\widehat{\pi}$ définit un isomorphisme de $B_{\widehat{\mathbf{G}}(u)}(\Gamma)$ sur $B_{\mathbf{G}}(v)$.

Si Γ est le chemin vide, la preuve est immédiate □

Pour tous sommets u,v, $\widehat{\mathbf{G}}(u)$ est isomorphe à $\widehat{\mathbf{G}}(v)$ [BV02a]. On notera $\widehat{\mathbf{G}}$ ce graphe défini à isomorphisme près. On appelle *revêtement universel* de $\mathbf{G}$ ce graphe $\widehat{\mathbf{G}}$. Les revêtements universels sont des arbres, et on a, par construction de $\widehat{\mathbf{G}}$:

Lemme 0.40. *Soit* $\mathbf{G}$ *un graphe. Son revêtement universel est fini si et seulement si* $\mathbf{G}$ *est un arbre.*

Remarque 0.41. Pour un graphe $\mathbf{G}$ donné, certains de ses quasi-revêtements s'obtiennent en tronquant le revêtement universel $\widehat{\mathbf{G}}$. Voir l'exemple sur la Figure 0.8.

Quasi-revêtements

Les deux propositions suivantes caractérisent les quasi-revêtements de rayon 0 et 1, et sont surtout énoncés à titre d'exemple :

Proposition 0.42. *Soient* $\mathbf{K}$ *et* $\mathbf{H}$ *deux graphes étiquetés.* $\mathbf{K}$ *est un quasi-revêtement de* $\mathbf{H}$ *de rayon* 0 *si et seulement si il existe une étiquette présente sur* $\mathbf{K}$ *et* $\mathbf{H}$ *simultanément.*

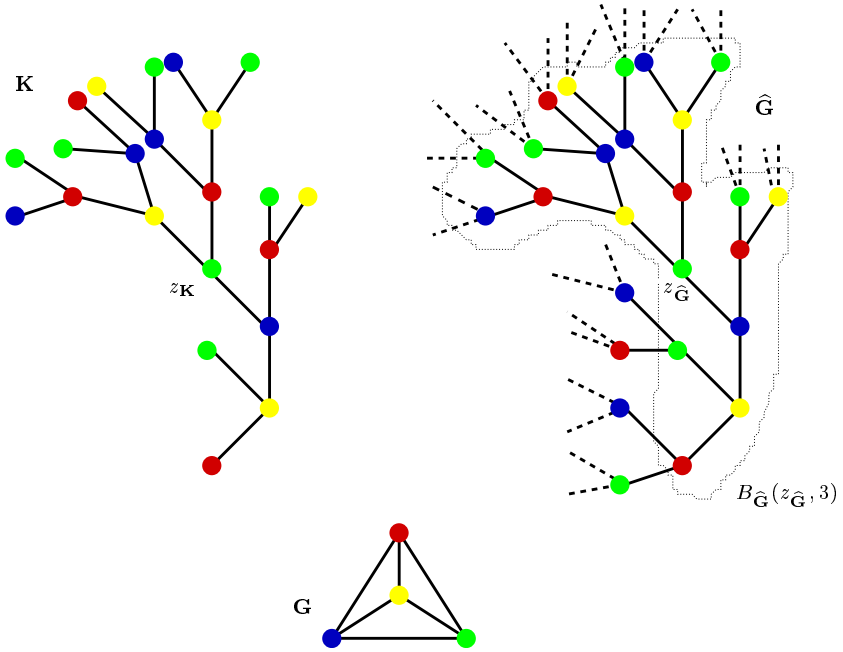


FIG. 0.8 – $\mathbf{K}$ est un quasi-revêtement de rayon 3 de $\mathbf{G}$ obtenu par troncature de $\widehat{\mathbf{G}}$

Proposition 0.43. *Soient* $\mathbf{K}$ *et* $\mathbf{H}$ *deux graphes uniformément étiquetés par* ε. $\mathbf{K}$ *est un quasi-revêtement de* $\mathbf{H}$ *de rayon* 1 *si et seulement si il existe deux sommets de* $\mathbf{K}$ *et* $\mathbf{H}$ *de degré identique.*

On a les lemmes techniques suivants

Lemme 0.44. *Soit* $\mathbf{K}$ *un quasi-revêtement strict de* $\mathbf{H}$ *de rayon* r *via* γ. *Alors, pour tout* $q \in \mathbb{N}$, *si* $r \geq q|V(\mathbf{H})|$ *alors* γ *a au moins* q *feuillets.*

Démonstration. Notons $\mathbf{G}$ le revêtement associé. Le quasi-revêtement étant strict, on a $|B_{\mathbf{K}}(z_{\mathbf{K}}, r)| \geq r \geq q|V(\mathbf{H})|$, *i.e.* $|V(\mathbf{G})| \geq q|V(\mathbf{H})|$. On déduit donc du Lem. 0.27 que $\mathbf{G}$ a au moins q feuillets.

Maintenant, considérons un arbre couvrant T de $\mathbf{H}$ enraciné en $\gamma(z_{\mathbf{K}})$. Notons T_1 le relèvement de T enraciné en $z_{\mathbf{G}}$. D'après le Th. 0.36, il existe $q-1$ relèvement de T distincts $T_2, \ldots, T_q$ tel que le sous -graphe induit par $T_1 \cup \cdots \cup T_q$ soit connexe. Comme T a un diamètre au plus égal à $|V(\mathbf{H})|-1$, on obtient que $T_1 \cup \cdots \cup T_q \subset B_{\mathbf{G}}(z_{\mathbf{G}}, q|V(\mathbf{H})|)$. Ce qui signifie que chaque sommet de $\mathbf{H}$ a au moins q antécédents dans $B_{\mathbf{G}}(z_{\mathbf{G}}, q|V(\mathbf{H})|)$, donc dans $\mathbf{K}$. □

Le lemme suivant permet de relier le rayon et l'étendue d'un quasi-revêtement d'un graphe donné.

Lemme 0.45. *Soit* $\mathbf{H}$ *un graphe de degré maximal* d. *Alors pour tout quasi-revêtement de* $\mathbf{H}$ *d'étendue* s *et de rayon* r, *on a*

$$s \leq (d+1)^r.$$

Démonstration. Soit $\mathbf{G}$ un quasi-revêtement de $\mathbf{H}$. Soit z le centre et r le rayon. $B_{\mathbf{K}}(z,r)$ est alors un sous-graphe de $\mathbf{G}$ de degré maximal d. Par induction, en remarquant que $|B(z,i+1)\backslash B(z,i)| \leq d|B(z,i)|$, on obtient que tout boule de rayon r et de degré maximal d a une taille d'au plus $(d+1)^r$ □

Cette borne n'est évidemment pas optimale mais suffisante pour ce qui nous intéresse : en remarquant qu'un quasi-revêtement à $q-$feuillets d'un graphe donné $\mathbf{H}$ a une taille supérieure à $q|V(\mathbf{H})|$, on obtient, d'après ces deux lemmes, une relation complète entre le rayon, l'étendue et le nombre de feuillet d'un quasi-revêtement de $\mathbf{H}$.

0.4.6 Étiquetages localement bijectifs

Une présentation d'un certain type d'étiquetage a été donnée par A. Mazurkiewicz, les étiquetages localement bijectifs de graphes. Ici, nous prouvons l'équivalence de cette notion avec la notion de revêtement.

Un étiquetage est dit *localement bijectif* si les sommets de même étiquette ont des voisinages étiquetés isomorphes. Formellement, on a

Définition 0.46 ([Maz97]). *Soit (G,λ) un graphe étiqueté. L'étiquetage λ est dit* localement bijectif *s'il vérifie les deux contions suivantes :*

0.46.i Pour tout $v \in V$ et pour tous $v', v'' \in B_G(v)$ on a $\lambda(v') = \lambda(v'')$ si et seulement si $v' = v''$.

0.46.ii Pour tout $v', v'' \in V$ tel que $\lambda(v') = \lambda(v'')$, les boules étiquetées $(B_G(v'),\lambda)$ et $(B_G(v''),\lambda)$ sont isomorphes.

G est un graphe ambigu *si il existe un étiquetage de G qui soit à la fois non bijectif et localement bijectif.*

Les étiquetages localement bijectifs et les revêtements sont étroitement liés au travers des quotients de graphes. Étant donné un graphe étiqueté, on "identifie" les sommets de même étiquette :

Définition 0.47. *Soit λ un étiquetage d'un graphe G. on définit le* graphe quotient *G/λ par*
- $V(G/\lambda) = \lambda(V(G))$
- $E(G/\lambda) = \{\{\alpha,\alpha'\} \mid \exists v, v' \in V(G), \{v,v'\} \in E(G), \alpha = \lambda(v), \alpha' = \lambda(v')\}$

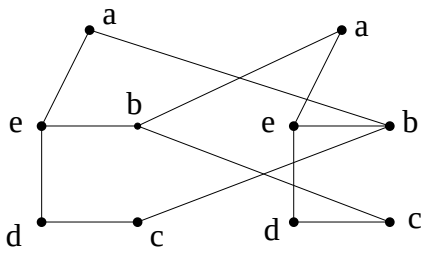


FIG. 0.9 – L'exemple 0.23 vu comme étiquetage localement bijectif

Lemme 0.48.

1. *Soit λ un étiquetage localement bijectif de G alors la projection $G \longrightarrow G/\lambda$ définit un revêtement.*
2. *Soit un revêtement $\gamma : G \longrightarrow H$, alors γ définit un étiquetage localement bijectif de G.*

Démonstration.

1. Notons $\pi : G \longrightarrow G/\lambda$. Du fait de la condition (0.46.i), G/λ ne contient aucune boucle. Maintenant, pour chaque $v \in V(G)$, les conditions 0.46.i et 0.46.ii impliquent respectivement que γ est une injection et une surjection de $B_G(v)$ sur $B_G(\pi(v))$. Par conséquent $B_G(v)$ et $B_{G/\lambda}(\pi(v))$ sont isomorphes.
2. Réciproquement, en étiquetant les sommets de G par leur image sur H via γ, on obtient un étiquetage qui satisfait les deux conditions 0.46.i et 0.46.ii. γ est donc un étiquetage localement bijectif.

□

Corollaire 0.49. *Un graphe est minimal si et seulement si il est non-ambigu.*

Ces lemmes d'équivalence entre le formalisme utilisé par A. Mazurkiewicz et le nôtre nous permettra d'utiliser directement les résultats de [Maz97].

0.4.7 Configurations standards

On cherche à comparer le déroulement d'un système de réétiquetage de graphes donné sur deux graphes G et G' en mettant ceux-ci en liaison via des revêtements. La configuration la plus simple est celle d'un revêtement, on peut ensuite utiliser le lemme 0.31, ce qui est réalisable sur le "petit" graphe peut être relevé sur le "grand". En combinant plusieurs revêtements, on obtient deux autres configurations également intéressantes.

Configuration en V

Elle correspond à la situation suivante. Soit trois graphes $\mathbf{G}, \mathbf{G}'$ et $\mathbf{H}$ tels que $\mathbf{G}$ et $\mathbf{G}'$ soient tous les deux revêtements de $\mathbf{H}$. Dans ce cas on note $\mathbf{G}\sigma\mathbf{G}'$ et on note $\sim$ la fermeture transitive de la relation σ.

Configuration en Λ

C'est en quelque sorte la situation symétrique de la précédente. On a trois graphes $\mathbf{G}, \mathbf{G}'$ et $\mathbf{F}$ tels que $\mathbf{F}$ est revêtement de $\mathbf{G}$ et de $\mathbf{G}'$. On note alors $\mathbf{G}\tau\mathbf{G}'$.

Propriétés

Nous allons voir quelques propriétés de ces relations.

Proposition 0.50 (V implique Λ). *Soient deux graphes $\mathbf{G}$ et $\mathbf{G}'$. Si $\mathbf{G}\sigma\mathbf{G}'$ alors $\mathbf{G}\tau\mathbf{G}'$.*

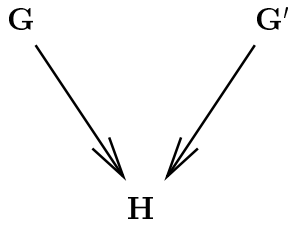


FIG. 0.10 – configuration en V : $\mathbf{G}\sigma\mathbf{G}'$

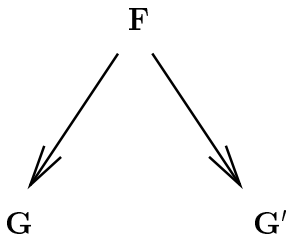


FIG. 0.11 – configuration en Λ : $\mathbf{G}\tau\mathbf{G}'$

Démonstration. Soit deux graphes $\mathbf{G}$ et $\mathbf{G}'$ tels que $\mathbf{G}\sigma\mathbf{G}'$. Soit $\mathbf{H}$ le graphe dont $\mathbf{G}$ et $\mathbf{G}'$ sont tous deux revêtements. Et notons γ (resp. γ') le revêtement de $\mathbf{G}$ (resp. $\mathbf{G}'$) sur $\mathbf{H}$.

On pose (produit de Kronecker [Wei62])

$$V_{\tilde{\mathbf{H}}} = \{(u,u') \in V_{\mathbf{G}} \times V_{\mathbf{G}'} \mid \gamma(u) = \gamma'(u')\} \quad (0.1)$$

$$E_{\tilde{\mathbf{H}}} = \{((u,u'),(v,v')) \in V_{\tilde{\mathbf{H}}} \times V_{\tilde{\mathbf{H}}} \mid (u,v) \in E_{\mathbf{G}}, (u',v') \in E_{\mathbf{G}'}\} \quad (0.2)$$

Montrons qu'alors les projections π, π' de $\tilde{\mathbf{H}}$ sur $\mathbf{G}$, $\mathbf{G}'$ respectivement sont des revêtements.

Considérons la fonction ψ de $\widetilde{\mathbf{H}}$ dans $\mathbf{H}$ qui à $w = (u,u') \in V_{\tilde{\mathbf{H}}}$ associe $\gamma(u)$ $(= \gamma(u'))$. ψ est un morphisme de graphe par définition de $E_{\tilde{\mathbf{H}}}$. C'est également un revêtement.

Soit $w = (u,u') \in V_{\tilde{\mathbf{H}}}$ et $u_0 = \psi(w)$. Montrons que $B_{\tilde{\mathbf{H}}}(w)$ est isomorphe à $B_{\mathbf{H}}(u_0)$.

Par définition du revêtement, les boules $B_{\mathbf{G}}(u)$, $B_{\mathbf{G}'}(u')$ et $B_{\mathbf{H}}(u_0)$ sont isomorphes et on note $v_1, \ldots, v_r$ et $v'_1, \ldots, v'_r$ leurs éléments respectifs de telle manière que $\forall i, \gamma(v_i) = \gamma'(v'_i)$. Alors si on note $B = \{(v_i, v'_i),\ 0 \leq i \leq r\}$ on a $B = V(B_{\tilde{\mathbf{H}}}(w))$.

Par conséquent, ψ est bien un revêtement et $B_{\mathbf{G}}(u)$ (resp. $B_{\mathbf{G}'}(u')$) étant isomorphe à $B_{\mathbf{H}}(\psi(w))$ via γ (resp. γ'), on en déduit que π et π' sont des revêtements. □

Corollaire 0.51. *τ est une relation d'équivalence, et $\sim = \tau$.*

La relation τ est également la fermeture réflexive, symétrique et transitive de la relation "être revêtement de".

Suite à l'introduction par Angluin des revêtements pour étudier les systèmes distribués des recherches ont été effectuée sur les propriétés de ces morphismes en particulier, T. Leighton a introduit la notion de raffinement de degré. Pour notre propos, nous allons généraliser cette définition aux graphes étiquetés

Définition 0.52. *Soit $\mathbf{G} = (G, \lambda)$ un graphe étiqueté. On appelle* raffinement de degré étiqueté *la partition de nombre de classes minimal de $V(\mathbf{G})$, $V_0, V_1, \cdots, V_{t-1}$, telle que*

0.52.i pour tout $i < t$, pour tout u et v appartenant à V_i, $\lambda(u) = \lambda(v)$,

0.52.ii il existe des constantes $r_{i,j}$, $0 \leq i,j < t$, telles que chaque sommet v de V_i est adjacent à $r_{i,j}$ sommets de V_j.

Il a ensuite prouvé

Théorème 0.53 (Leighton, [Lei82]). *Étant donné deux graphes étiquetés finis connexes* $\mathbf{G}$ *et* $\mathbf{H}$, $\mathbf{G}$ *et* $\mathbf{H}$ *ont un revêtement commun si et seulement si ils ont le même raffinement de degré étiqueté.*

Démonstration. La preuve de [Lei82] s'étend directement au cas étiqueté. □

Avec nos notations cela revient à dire que la relation $\sim$ signifie "avoir le même raffinement de degré".

0.4.8 Contre-exemples

Voici un petit bestiaire qui peut contredire certaines "conjectures intuitives" que l'on pourrait avoir sur les revêtements et double-revêtements.

Cycles et graphes non-minimaux

Contrairement à l'intuition, il peut ne pas y avoir que des longs cycles dans un graphe revêtement propre d'un autre graphe. Sur la figure 0.12, chaque arête appartient à un cycle de longueur 3.

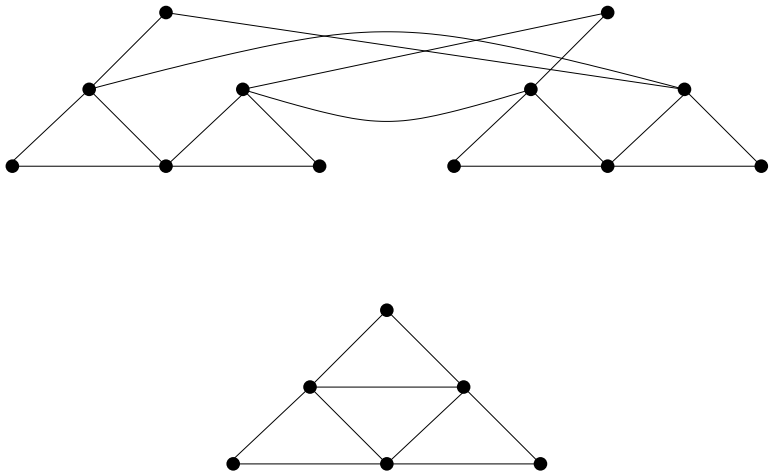


FIG. 0.12 – Graphe non-minimal dont toutes les arêtes appartiennent à un triangle

Correspondance des boules dans la configuration en Λ

Lorsqu'on se trouve en configuration Λ on pourrait croire que l'on a une relation simple entre les boules de $\mathbf{G}$ et celles de $\mathbf{G}'$, y compris au niveau de leurs préimages dans $\mathbf{F}$. La figure 0.13 montre qu'il n'en est rien (les sommets du graphe supérieur sont étiquetés par leurs images) par .

En considérant par exemple les boules de centre b et b' (entourées sur le schéma) on s'aperçoit qu'elles peuvent, de gauche à droite, coïncider, juste se recouvrir partiellement, être disjointes.

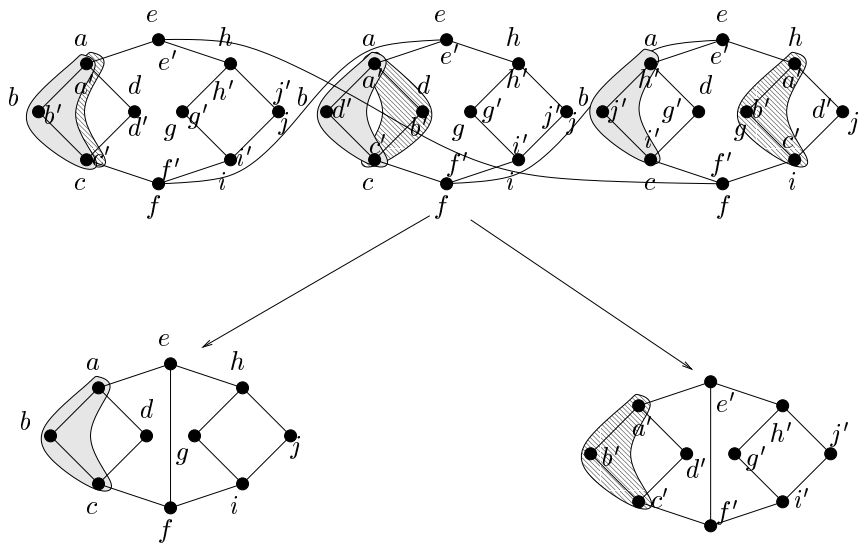


FIG. 0.13 – configuration en chapeau

D'autre part, G étant ici isomorphe à G', on voit qu'aucune propriété supplémentaire (comme le même nombre de sommets par exemple) ne peut assurer que ça va bien se passer, ce qui est peu surprenant, puisque si il y avait un lien, cela voudrait dire, en prenant le lemme de Relèvement, que toutes les exécutions d'un algorithme sur un réseau donné seraient en quelque sorte équivalentes (en prenant la configuration en chapeau $\mathbf{G = G' = F}$).

0.5 Réétiquetages de Rayon Supérieur à 1

Étant fixé un entier non nul k, il est possible d'établir les mêmes définitions et résultats que précédemment en faisant cette fois opérer les réétiquetages sur des boules de rayon k. Dans les chapitres suivants nous nous restreignons au cas $k = 1$ mais le lecteur devra garder à l'esprit le fait que la plupart des résultats énoncés et leurs preuves s'étendent directement aux réécritures de rayons supérieurs. Nous présentons les grandes lignes de cette extension.

0.5.1 Définition

On donne les premières définitions et quelques résultats. Pour les autres, il suffit de prendre toutes nos définitions précédentes en remplaçant boule par $k-$boule.

Définition 0.54. *Soit γ un homomorphisme de G sur H. γ est un* $k-$revêtement *si pour tout sommet $v \in V(G)$, γ induit un isomorphisme de $B_G(u,k)$ dans $B_H(\gamma(u,k))$.*

Remarque 0.55. Un revêtement est un $1-$revêtement.

Nota Bene. Les définitions précédentes de k-revêtement de [LMZ95, DZ96] utilisent des k-boules *fermées*. On pourra parler dans ce cas de $k-$*revêtements fermés* pour les distinguer.

0.5.2 Construction

Cette section présente quelques caractéristiques des k-revêtements vus comme cas particuliers de revêtements. Cela peut permettre d'isoler les k-revêtements parmi les revêtements donnés par le Th. 0.36.

Lemme 0.56. *Soient deux entiers k, k', tels que $k \leq k'$. Alors tout $k'-$revêtement est aussi un k-revêtement. En particulier, tout k-revêtement est un revêtement.*

A l'aide de ce théorème et de celui de Reideimester, nous en déduisons que tout k-revêtement peut être obtenu à l'aide d'arbre couvrant : on calcule tous les revêtements et on ne retient que ceux qui sont effectivement des k-revêtements.

On a également la caractérisation suivante :

Lemme 0.57 (Th. 3, p43[Bot97]). *Soit $k > 0$ et G, G' deux graphes tel que G est un revêtement de G' via γ. Alors μ est un k-revêtement (resp. un k-revêtement fermé) si et seulement si pour tout cycle C' dans G' de longueur au plus $2k$ (resp. $2k+1$), l'image inverse $\gamma^{-1}(C')$ est une union disjointe de cycles isomorphe à C'.*

On va maintenant décrire une construction permettant d'obtenir des k-revêtements à partir de revêtements, ce qui pourra permettre de porter des résultats (en particulier existentiels) des revêtements aux k-revêtements.

Diviser une arête consiste en l'insertion d'un sommet de degré 2 (*i.e.* l'arête est remplacée par un chemin de longueur 2) Un graphe G' est une division élémentaire de G si G' peut être obtenu à partir de G en divisant une arête de G.

On dira que H est une division de G si H peut-être obtenu de G par une suite de divisions élémentaires. On dit de deux graphes qu'ils sont homéomorphes s'ils admettent des divisions isomorphes. Une classe de graphes sera fermé par homéomorphisme si dès qu'un graphe G appartient à celle-ci alors tout graphe homéomorphe à G y appartient également.

Soit k un entier. On va définir une division de graphe particulière de la manière suivante. Si G est un graphe, alors $\nu_k(G)$ est le graphe obtenu en divisant chaque arête de G en $k+1$ arêtes successives par insertion de k sommets de degré 2. Soit $v, w \in V(G)$ deux sommets adjacents. On note $(v, v_1, \ldots, v_k, w)$ la suite de sommets de $\nu_k(G)$ obtenu en subdivisant l'arête $\{v, w\}$. Avec ces notations :

Lemme 0.58. *Si G est un revêtement de G' via γ, alors $\nu_k(G)$ est un k-revêtement de $\nu_k(G')$ via γ_k défini par :*

1. $\gamma_k(v) = \gamma(v),\ v \in V(G)$
2. *if $\{v, w\} \in E(G)$, $(v, v_1, \ldots, v_k, w)$ et $(\gamma(v), v'_1, \ldots, v'_k, \gamma(w))$ étant les suites correspondante dans $\nu_k(G)$ et $\nu_k(G')$, alors $\gamma_k(v_i) = v'_i$, pour $1 \leq i \leq k$.*

Il est à noter que, modulo des $k-$subdivisions, les exemples d'impossibilités s'appliquent également aux k-revêtements.

0.5.3 Étiquetages $k-$localement bijectifs

On peut étendre les équivalences de la section 0.4.6 en utilisant des $k-$boules. On obtient la même équivalence entre étiquetages $k-$localement bijectifs et k-revêtements.

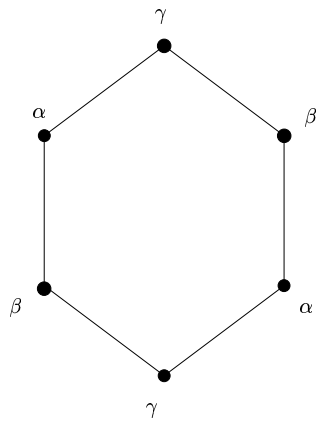


FIG. 0.14 – Un étiquetage localement bijectif de A_6 dont le quotient n'est pas un revêtement fermé

En revanche, ces propriétés d'équivalence ne sont pas vraies si l'on utilise comme voisinage les $k-$boules fermés, car les arêtes induites ne peuvent pas être prises en compte localement (cf Fig. 0.14). C'est l'une des grandes différences entre les k-revêtements et les k-revêtements fermés, qui semble donner de meilleures propriétés aux k-revêtements (ouverts). En particulier ce qui est fait à l'aide de l'algorithme de Mazurkiewicz peut être porté aux réétiquetages de rayon k, voir section 3.3, mais cela semble plus délicat pour les réétiquetages sur des boules fermées de rayon k.

Première partie

Autour de l'Algorithme de Mazurkiewicz

Chapitre 1

Présentation

Nous allons, dans ce premier chapitre, présenter un algorithme proposé par A. Mazurkiewicz dans [Maz97] pour résoudre le problème de l'énumération des nœuds dans un réseau anonyme.

1.1 Motivations

1.1.1 Les réseaux anonymes

Un réseau anonyme est un réseau dans lequel on suppose que tous les nœuds utilisent le même algorithme au sens strict : en particulier ils ne possèdent pas de numéro d'identification, unique, qui permettrait de distinguer un nœud d'un autre. Le seul élément les différenciant est leur position dans le réseau de communication.

Un réseau anonyme est par conséquent le système distribué le plus rudimentaire. Et il a été prouvé qu'un certain nombre de problème n'avait pas de solution pour ces réseaux, alors qu'ils en admettaient lorsque l'on avait à disposition des identifiants. Un système où certaines identités pourraient être dupliquées, avec une multiplicité bornée, est un système intermédiaire entre les réseaux anonymes et les réseaux avec identités. L'étude de ces réseaux est donc très intéressante en ce qu'elle permet de distinguer très précisément ce qui est nécessaire, ou bien commode, pour résoudre un problème distribué. Nous allons voir que les problèmes dans le cas anonyme se rapporte souvent à essayer de recréer des identifiants en partant de rien, c'est à dire de résoudre le problème de l'énumération.

1.1.2 Énumération

Chaque nœud n'ayant aucun moyen de se distinguer a priori des autres, la résolution des conflits globaux de manière locale apparaît difficile, par exemple attribuer l'accès à une ressource qui ne peut être partagée. Cependant un certain nombre de problèmes peuvent être résolus dans ce cadre contraignant. Le problème de l'énumération apparaît alors être le paradigme en ce domaine. En effet, une fois une identité

distincte attribuée à chaque nœud, les méthodes classiques de gestion de conflit avec identifiants peuvent être utilisées.

On peut énoncer formellement le problème de l'énumération de la manière suivante [Maz97], soit $\mathcal{F}$ une famille de graphes :

Définition 1.1. *Un système de réétiquetage de graphes $\mathcal{R}$ résout le* Problème de l'énumération *si il existe une étiquette initiale s telle que, pour tout graphe G de $\mathcal{F}$, les deux conditions suivantes sont vérifiées :*

1.1.i Toute exécution de $\mathcal{R}$ sur (G, Λ_s) se termine.

1.1.ii Tout étiquetage final est bijectif.

1.1.3 Numérotation forte

La définition précédente n'implique aucune considération sur la détection locale de la terminaison, à savoir que le calcul peut être terminé mais que, localement, si chaque nœud a conscience du fait qu'il ne peut appliquer aucune règle, aucun nœud n'est en mesure de déterminer si c'est le cas de tous les autres nœuds. D'autre part, la condition 1.1.ii ne permet pas non plus à un noeud de savoir quel est l'ensemble des identités existant sur le réseau. On définit donc le problème de l'énumération Forte :

Définition 1.2. *Un système de réétiquetage de graphes $\mathcal{R}$ résout le* Problème de l'énumération Forte *si il existe une étiquette initiale s tels que, pour tout graphe G de $\mathcal{F}$, les trois conditions suivantes sont vérifiées :*

1.2.i Toute exécution de $\mathcal{R}$ sur (G, Λ_s) se termine.

1.2.ii Tout étiquetage final est une bijection de $V(G)$ sur $\Lambda_f = \{1, \ldots, \mid V(G) \mid\}$.

1.2.iii Pour tout sommet u, $\lambda_i(u) \in \Lambda_f \Rightarrow \lambda_j(u) = \lambda_i(u)$ pour tout $j > i$.

1.1.4 Impossibilité

La résolution du problème de l'énumération n'admet pas de solution sur l'ensemble des graphes. En particulier, le lemme de relèvement 0.31 a pour corollaire immédiat que pour un graphe non-minimal, la condition 1.1.ii ci-dessus ne peut-être satisfaite pour toute exécution.

Théorème 1.3. *Soit $\mathcal{F}$ contenant un graphe non minimal. Alors il n'existe pas de système de réétiquetage de graphes résolvant le problème de l'énumération pour $\mathcal{F}$.*

A fortiori, il n'y pas non plus de solution pour l'énumération forte. La section suivante expose l'algorithme proposé par A. Mazurkiewicz pour résoudre le problème de l'énumération sur les graphes minimaux.

1.2 L'Algorithme

Dans cette section, après une description informelle, nous présenterons le système de réécriture décrivant l'algorithme de Mazurkiewicz. A noter que, dans [Maz97], le formalisme utilisé était différent, mais il est en fait équivalent, il s'agit de réécriture locale. Se reporter aux propositions et remarques de la section 0.4.6.

Nous donnons ensuite quelques exemples d'éxécution de cet algorithme. Enfin, nous décrirons et prouverons quelques unes de ses propriétés les plus remarquables.

Ce qu'il est très important de noter c'est, si cet algorithme a été présenté pour résoudre le problème de l'énumération sur les graphes minimaux, nous le présentons ici dans un cadre complètement général, *i.e.* sauf mention explicite du contraire, les résultats, en particulier le Th. 1.10, seront valables pour tous les graphes. Nous montrerons également au chapitre suivant comment cela s'étend aisément aux graphes étiquetés.

1.2.1 Présentation

Nous allons commencer par donner une description informelle de l'algorithme. Rappelons quels sont les buts à atteindre : la condition 1.1.ii impose que l'étiquetage final soit bijectif, sachant que l'on démarre avec un étiquetage uniforme. Chaque sommet va donc tenter de prendre une identité et ensuite la communiquer aux autres nœuds afin qu'aucun autre ne prenne la sienne. S'il s'aperçoit qu'il y a déjà un nœud qui porte son identité, il va en changer. Il y aura un soin particulier à apporter lors de ses changements de noms afin d'éviter un nombre infini de tentatives.

Chaque nœud essaie d'obtenir un numéro qui lui soit propre, qui devra être un entier entre 1 et $|V(G)|$. Un nœud choisit un numéro et le diffuse sur tout le réseau accompagné de sa *vue locale, i.e.* la liste des numéros de ses voisins. Si un nœud u, recevant des messages de diffusion, découvre l'existence d'un autre nœud v ayant le même numéro, alors il compare leur vue locale respective. Si la vue locale de son rival v est "plus forte", en un certain sens déterminé ultérieurement, alors u essaie un autre numéro. Ce nouveau numéro est diffusé lui-aussi, accompagné de la vue locale. A la fin d'un calcul, il n'est pas garanti, à moins que le graphe soit minimal, que chaque nœud ait un identifiant unique. Cependant, la façon de procéder assure que des nœuds ayant le même numéro auront des vues locales identiques.

1.2.2 Notations

Le point crucial de cet algorithme est la définition d'un ordre total sur les vues locales permettant d'arbitrer les conflits entre sommets de même numéro, en s'assurant que la "force" des vues locales d'un sommet ne puisse diminuer au cours des calculs. Ceci afin d'éviter qu'un nœud ne se trouve battu par un de ses messages passés, ce qui pourrait provoquer la non terminaison de l'algorithme.

On utilise, pour décrire la vue locale de v, les notations suivantes : si v a pour degré d et ses voisins ont pour numéros $n_1, \ldots, n_d$, avec $n_1 \geq \cdots \geq n_d$, alors $N(v)$, la vue locale, est le d-uple $(n_1, \ldots, n_d)$. Soit $\mathcal{N}$ l'ensemble de tels multiensembles ordonnés de $\mathbb{N}$. L'ordre alphabétique définit un ordre total, $\preccurlyeq$, sur $\mathcal{N}$. Si $N \neq N'$, on notera $N \prec N'$.

Un sommet v est étiqueté par un triplet $(n(v), N(v), M(v))$ ayant la signification suivante lors du calcul :

- $n(v) \in \mathbb{N}$ est le *numéro* du sommet v,
- $N(v) \in \mathcal{N}$ est sa *vue locale* [1],
- $M(v) \subset \mathbb{N} \times \mathcal{N}$ est la *boîte aux lettres* de v et contient tous les messages diffusés et reçus à ce stade du calcul.

On définit l'opération suivante qui servira à mettre à jour les vues locales des voisins après un changement de numéro : $\mathrm{sub}(N, n, n')$ est la liste ordonnée obtenue en supprimant n de N et en rajoutant n'. On a, par définition de l'ordre alphabétique, le lemme clé suivant

Lemme 1.4. *Soit $N \in \mathcal{N}$ et deux entiers n, n'. Alors*

$$n < n' \implies N \prec \mathrm{sub}(N, n, n').$$

1.2.3 Le système de réétiquetage

L'étiquette initiale pour tout sommet est $s = (0, \emptyset, \emptyset)$.

Les règles sont maintenant décrites ci-dessous pour une boule $B(v_0)$ de centre v_0. Nous rappelons nos conventions de description. Les sommets v de $B(v_0)$ ont pour étiquettes $(n(v), N(v), M(v))$, les étiquettes obtenues après application de la règle de réétiquetage sont $(n'(v), N'(v), M'(v))$. De plus, et afin de rendre les règles plus faciles à lire et de ne pas surcharger, nous rappelons que les étiquettes qui ne sont pas mentionnées demeurent inchangées.

$\mathcal{D}$: **Diffusion**

Condition préalable :

– $\exists v \in B(v_0), M(v) \neq M(v_0)$

Réétiquetage :

– Pour tout $v \in B(v_0)$, $M'(v) := \bigcup_{w \in B(v_0)} M(w)$

$\mathcal{C}$: **Changement de numéro**

Condition préalable :

– $\forall v \in B(v_0), M(v) = M(v_0)$

– 1. $n(v_0) = 0$

ou

2. $n(v_0) > 0$ et $(\exists N_1\ (n(v_0), N_1) \in M(v_0)$ et $N(v_0) \prec N_1)$.

Réétiquetage :

– $n'(v_0) := 1 + \max\{n \in \mathbb{N} \mid (n, N) \in M(v_0)$ pour un certain $N \in \mathcal{N}\}$

– Pour tout $v \in B(v_0) \backslash \{v_0\}$, $N'(v) :=$ sub $(N(v), n(v_0), n'(v_0))$

[1] La notion de vue locale, définie ici, est différente de celle utilisée dans le modèle de Yamashita et Kameda.

– $M'(v) := M(v_0) \bigcup_{w \in B(v_0)} (n'(w), N'(w))$.

Ce système de réétiquetage $\{\mathcal{D}, \mathcal{C}\}$ sera noté $\mathcal{M}$.

1.3 Exemples d'Exécution

L'exemple que l'on va donner montre qu'il existe au moins une exécution de $\mathcal{M}$ qui s'achève sur une numérotation donnant une énumération. On donnera ensuite un contre-exemple sur l'anneau de taille 6. La section suivante montrera exactement quelles sont tous les étiquetages finaux possibles.

1.3.1 Énumération

Soit G un graphe. Soit φ une bijection de $V(G)$ sur $\{1, \ldots, |V(G)|\}$. On va exhiber une exécution de l'algorithme qui attribuera à chaque sommet v de G, l'identité $\varphi(v)$.

L'algorithme se déroule sous la forme de l'alternance de 2 phases distincts. L'une consiste à numéroter un sommet et ensuite, la règle de diffusion est appliqué jusqu'à ce que chaque sommet connaisse le nouveau numéro. Une telle application maximale de la règle de diffusion sera notée $\mathcal{D}_{\text{max}}$.

Formellement, on définit la chaîne de réécriture suivante :

$$\rho = \mathcal{C}\varphi^{-1}(1)\mathcal{D}_{\text{max}}\mathcal{C}\varphi^{-1}(2)\mathcal{D}_{\text{max}} \cdots$$

Pour tout $i \in [1, .., n]$

1. Phase de nouveau numéro $\mathcal{C}$: le sommet tel que $\varphi(v) = i$, applique la règle de changement de numéro et se numérote ... $\varphi(v)$.
2. Phase de diffusion maximale $\mathcal{D}_{\text{max}}$, application de la règle de diffusion, ainsi chaque nœud connaît le nouveau numéro i.

Par récurrence sur i, on peut montrer que cela définit bien une chaîne de réétiquetage de $\mathcal{M}$.

1.3.2 Pas d'énumération sur l'anneau à 6 éléments

L'exemple suivant est en fait un contre-exemple : la chaîne de réécriture est

$$\mathcal{C}u_1\mathcal{C}u_4\mathcal{C}u_2\mathcal{C}u_5\mathcal{C}u_3\mathcal{C}u_6$$

Se reporter à la figure 1.1.

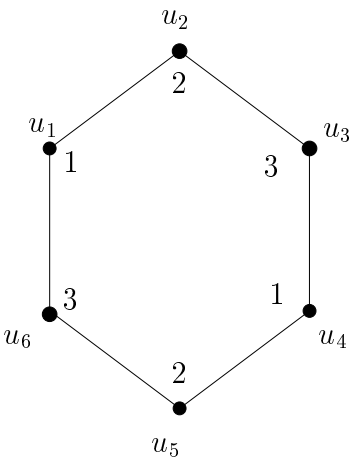


FIG. 1.1 – Exécution sans énumération sur A_6. Les entiers sont les numéros finaux.

1.4 Propriétés de l'Algorithme de Mazurkiewicz

Soit G un graphe connexe quelconque, on prouvera tout d'abord la terminaison de l'algorithme. Ensuite, on énoncera diverses propriétés intéressantes qui découlent directement du fait que, par définition, dans une configuration finale, on ne peut appliquer aucune règle.

1.4.1 Terminaison

Théorème 1.5 (Mazurkiewicz, [Maz97]). *Soit G un graphe. Toute exécution de $\mathcal{M}$ sur G se termine.*

Démonstration. La preuve est la même que [Maz97] même si présentée de manière peut-être un peu différente.

Soit $I = \{0, \ldots, |V(G)|\}$ et soit $\mathcal{N}_I$ l'ensemble des parties ordonnées de I. $\mathcal{N}_I$ est ordonné par $\preccurlyeq$. On ordonne $\mathcal{P}(I \times \mathcal{N}_I)$ par inclusion ($\subset$).

On note $X = (I \times \mathcal{N}_I \times \mathcal{P}(I \times \mathcal{N}_I))^{V(G)}$. L'ordre produit $\leqq$ définit un ordre partiel sur X.

Notons $(n_i(v), N_i(v),\ M_i(v))$ l'étiquette v après la $i$ème étape de calcul. On note $x^i = ((n_i(v), N_i(v), M_i(v)))_{v \in V(G)}$ et on va montrer que $(x^i)_i$ est une suite *strictement* croissante de X.

Tout d'abord, on a le lemme de croissance suivant

Lemme 1.6. *Pour tout i et tout sommet v*
- $n_i(v) \leq n_{i+1}(v)$
- $N_i(v) \preccurlyeq N_{i+1}(v)$
- $M_i(v) \subseteq M_{i+1}(v)$

Démonstration. Le résultat est trivialement vrai pour les sommets qui ne sont pas concernés par l'application de la règle à l'étape i. Pour les autres sommets, on notera que $\mathcal{D}$ se contente d'ajouter des éléments aux boîtes aux lettres. En ce qui concerne $\mathcal{C}$, $n_i(v_0)$

ne peut qu'augmenter et, par conséquent la "force" des vues locales des voisins augmentent également par Lem. 1.4. □

De plus, l'une de ces inégalités est stricte pour au moins un sommet, en particulier pour celui sur lequel l'application de la règle était centrée. Par conséquent $x^i \lneqq x^{i+1}$ pour tout i.

On dira qu'un numéro m est *connu* par v si $(m, N) \in M(v)$ pour un certain N. Afin de prouver la terminaison, le lemme suivant affirme qu'à tout moment, dès lors qu'un numéro est connu, il y a un sommet qui porte ce numéro.

Lemme 1.7. $\forall i, \forall v \in V(G)\ \forall (m, N) \in M_i(v)$, *il existe* $w \in V(G)$ *tel que* $n_i(w) = m$.

Démonstration. Supposons que le numéro m soit connu par v_0 et soit $U = \{u \in V(G) \mid \exists j < i, n_j(u) = m\}$, alors U n'est pas vide puisque le numéro m a été diffusé au moins une fois. Soit w le sommet de U avec la plus forte vue locale dans $\{N_j(u) \mid u \in U, \exists j < i, n_j(u) = m\}$. Par maximalité, la règle de changement de numéro $\mathcal{C}$ n'a pas pu être appliquée à w. Par conséquent $n_i(w) = m$. □

La dernière étape consiste à montrer que lorsqu'un numéro est connu, alors tous les numéros plus petits sont connus, et par conséquent attribués à au moins un sommet.

Lemme 1.8 (Connexité des numéros connus). $\forall i, \forall v \in V(G)$ *tel que* $n_i(v) \neq 0$,

$$\forall m, m', m \leq m', v \textit{ connaît } m' \Longrightarrow v \textit{ connaît } m.$$

Démonstration. La preuve de cette assertion s'effectue par récurrence sur i. A l'étape initiale $i = 0$ et l'assertion est vraie.

Supposons que l'assertion soit vraie pour $i \geq 0$. Montrons qu'elle est vraie après application de l'une ou l'autre règle. S'il s'agit d'une diffusion, alors il n'y a pas de changement de numéro et l'assertion est dans ce cas facilement vérifiée : les nouveaux noms connus proviennent de sommets voisins pour lesquels le lemme est vérifié.

Si la règle suivante est la règle de changement de numéro appliquée à v_0 alors, il suffit de vérifier l'assertion pour v_0, c'est-à-dire de prouver que les numéros de $[1, \ldots, n_{i+1}(v_0) - 1]$ sont connus par v_0. Par définition du réétiquetage mettant à jour le numéro dans les vues voisines de $\mathcal{C}$, $n_{i+1}(v_0) - 1$ est connu à l'étape i par v_0. Par hypothèse de récurrence, tous les numéros de $[1, \ldots, n_{i+1}(v_0) - 1]$ sont donc connus par v_0, et $n_{i+1}(v_0)$ est connu de v_0 par le réétiquetage mettant à jour les boîtes à lettres de $\mathcal{C}$. □

Comme il ne peut y avoir qu'au plus $|V(G)|$ numéros différents assignés à un moment donné, on en déduit que, pour tout i, pour tout sommet v, $n_i(v) \leq |V(G)|$ et par conséquent $x^i \in X$. X est un espace ordonné fini et donc la suite strictement croissante (x^i) est finie. Le système de réétiquetage $\mathcal{M}$ est nœthérien. □

Lemme 1.9. *Soit* v_0 *un sommet. A tout moment, on a que pour tous voisins* $v, v' \in B(v_0)$,

$$n(v) = n(v') \neq 0 \Rightarrow v = v'$$

Démonstration. Supposons sans perte de généralité que v a atteint $n(v)$ avant v'. Alors $n(v)$ est connu de v_0 et par conséquent, quand v' applique la règle de changement de numéro, v' connaît $n(v)$ et par conséquent $n(v') > n(v)$. Contradiction. □

1.4.2 Propriétés de l'étiquetage final

On notera Π_ρ l'étiquetage final obtenu après une exécution ρ. Si v est un sommet de v, alors $\Pi_\rho(v) = (n_\rho(v), N_\rho(v), M_\rho(v))$.

Théorème 1.10 (Mazurkiewicz [Maz97]). *Soit ρ une exécution de $\mathcal{M}$. Alors, pour tout u, v*

1.10.i $M_\rho(u) = M_\rho(v)$

1.10.ii $(n_\rho(u), N_\rho(u)) \in M_\rho(v)$

1.10.iii Pour tout $(n_\rho(u), N) \in M_\rho(v)$, $N \preccurlyeq N_\rho(u)$

1.10.iv $n_\rho(u) = n_\rho(v) \implies N_\rho(u) = N_\rho(v)$

1.10.v Π_ρ *et* n_ρ *induisent un étiquetage localement bijectif*

1.10.vi la projection $\pi : G \longrightarrow G/\Pi_\rho \simeq G/n_\rho$ *est un revêtement.*

Démonstration.

1.10.i Sinon on pourrait appliquer la règle de Diffusion.

1.10.ii D'après le point précédent et en remarquant que $(n(u), N(u)) \in M(u)$ à tout moment du calcul.

1.10.iii Sinon on pourrait appliquer la règle de Changement de numéro.

1.10.iv Cela découle du point précédent appliqué à u et v simultanément, et du fait que $\preccurlyeq$ est un ordre total.

1.10.v Conséquence directe du point précédent.

1.10.vi d'après le Lem. 0.48 et le point précédent. L'isomorphisme provient du point 1.10.i et du lemme 1.9.

□

Corollaire 1.11 ([Maz97]). *$\mathcal{M}$ résout le problème de l'énumération sur la famille des graphes minimaux.*

Démonstration. Soit G un graphe minimal. Par minimalité et Th.1.10.vi, G/Π_ρ est nécessairement isomorphe à G, et donc l'étiquetage final Π_ρ est une bijection. □

Remarque 1.12. En revanche, $\mathcal{M}$ ne permet pas de déterminer la terminaison pour un graphe minimal si l'on n'a pas d'information supplémentaire comme on le verra au chapitre 7.

La connaissance de la taille permet de détecter la terminaison, une fois que le numéro $|V(G)|$ apparaît, n est une bijection et le calcul va s'arrêter après toutes les applications possible de la règle de diffusion. On pourrait par conséquent ajouter une troisième règle de détection de la terminaison sous ces hypothèses.

On sait donc numéroter si l'on connaît la taille du réseau.

Proposition 1.13 ([Maz97]). *Si G est minimal, $\mathcal{M}$ résout le problème de l'énumération forte sur G.*

Le Th. 1.10 montre que l'on peut interpréter l'étiquetage final comme un graphe que chaque sommet calcule. Étant donné une boîte aux lettres M, on définit le prédicat $\underline{\text{CHAMPION}}(n, N, M)$ qui est vérifié si $N' \preccurlyeq N$ pour tout $(n, N') \in M$, en d'autres termes si N est la vue locale maximale associée à n dans la boîte M.

Pour une boîte aux lettres donnée M, on définit le graphe H_M de la manière suivante

$$\begin{aligned} V(H_M) &= \{n \mid \exists N, (n, N) \in M\} \\ E(H_M) &= \{\{n, n'\} \mid \exists N \text{ tel que } n' \in N \text{ et } \underline{\text{CHAMPION}}(n, N, M)\} \end{aligned}$$

Comme pour l'étiquetage final, chaque numéro est associé à une vue locale maximale, on a, d'après le Th. 1.10.vi

Proposition 1.14. *Pour une exécution ρ de l'algorithme de Mazurkiewicz sur le graphe G, pour tout u, on a*

$$\begin{aligned} V(H_{M_{\rho(u)}}) &= \{n_\rho(v) \mid v \in V(G)\} \\ E(H_{M_{\rho(u)}}) &= \{\{n_\rho(v), n_\rho(w)\} \mid \{v, w\} \in E(G)\} \end{aligned}$$

Par conséquent, étant donné une exécution ρ de $\mathcal{M}$, $H_{M_\rho(u)}$ est constant et ne dépend pas de u. On définit $H_\rho = H_{M_{\rho(u)}}$.

De plus H_ρ est le graphe quotient de G par n_ρ. De manière équivalente, par Th. 1.10.vi, H_ρ est le graphe quotient de G par Π_ρ.

Remarque 1.15. Avant que nous ne mettions en avant le rôle possible de H_ρ, il convient de noter que celui-ci peut-être calculé localement par chaque sommet v, et que ce calcul ne dépend que du contenu de la boîte à lettres $M(v)$.

La proposition suivante établit que l'on peut voir $\mathcal{M}$ comme le calcul d'un graphe H dont G est un revêtement. Réciproquement, tout graphe H dont G est revêtement peut être obtenu par une exécution de l'algorithme.

Proposition 1.16 ([Maz97]). *Soit G un graphe.*

1. *Pour toute exécution ρ de $\mathcal{M}$, G est un revêtement de H_ρ.*
2. *(Équité) Pour tout graphe H tel que G est un revêtement de H, il existe une exécution de ρ tel que $H \simeq H_\rho$.*

Démonstration.

1. C'est le Th. 1.10.vi.
2. La preuve est une généralisation de l'exemple d'éxécution présenté section 1.3, page 33.
   Supposons que l'on ait une bijection des sommets de H vers $\{1, \ldots, |V(H)|\}$. Soit λ l'étiquetage de G obtenu par relèvement des numéros de H. Il existe une exécution de $\mathcal{M}$ telle que, chaque sommet de G obtient comme étiquetage final $n_\rho(v) = \lambda(v)$.
   Ceci est réalisé de la manière suivante. D'abord, on applique $\mathcal{C}$ à tous les sommets de $\lambda^{-1}(1)$. Cela est possible car les boules centrées en ces sommets ne se

recouvrent pas, G étant un revêtement de H. Ensuite, on applique la règle de diffusion aussi longtemps que possible. Une fois cela fait, le numéro 1 est connu par tous les sommets, et on recommence à appliquer la règle de changement de numéro, cette fois aux sommets de $\lambda^{-1}(2)$. Et ainsi de suite, jusqu'à que chaque sommet soit numéroté par $\lambda(v)$.

□

Chapitre 2

Complexité

La notion de complexité si elle est bien établie, avec de quelques variantes, pour les algorithmes séquentiels, est beaucoup moins claire pour les algorithmes distribués ([Lav95], chap. 2). Ici on s'attachera au nombre d'applications de règles, en supposant qu'elles ont toutes un coût égal. Ceci peut être assuré par le mécanisme de synchronisation sous-jacent [Zem00, MSZ00]. Ce calcul n'est d'ailleurs pas abordé dans le papier original.

Nous prouverons tout d'abord une borne supérieure grossière de complexité. Nous exposerons ensuite divers exemples et montrerons que cette borne peut être atteinte (au coefficient près). Une étude plus exhaustive de la complexité, en particulier en moyenne, reste à faire.

2.1 Borne Supérieure

Ce premier résultat établit que la complexité totale de $\mathcal{M}$ est au plus cubique.

Théorème 2.1. *Soit G un graphe de taille n. Alors toute exécution de $\mathcal{M}$ sur G utilise au plus $\frac{1}{2}n^3 - \frac{1}{2}n^2 + n$ applications de la règle de Diffusion $\mathcal{D}$ et $\frac{1}{2}n^2 + \frac{1}{2}n$ applications de la règle de Changement de numéro $\mathcal{C}$.*

Au total, on utilise au plus $O(n^3)$ règles.

Démonstration. Notons $n_{\mathcal{C}}$ (resp. $n_{\mathcal{D}}$) le nombre d'applications de la règle de Changement de numéro (resp. de Diffusion). On commence par remarquer que

$$n_{\mathcal{D}} \leq (n-2) \times n_{\mathcal{C}}.$$

En effet, si on considère une exécution ρ en rassemblant les diffusions,

$$\rho = \mathcal{C}\mathcal{D}_1\mathcal{C}\mathcal{D}_2\mathcal{C}\cdots\mathcal{C}\mathcal{D}_{n_{\mathcal{C}}}$$

On a, pour tout i, $\sum_{1\leq j\leq i} |\mathcal{D}_j| \leq i \times (n-2)$. Car $\mathcal{D}_1 \cdots \mathcal{D}_i$ correspond en fait à la diffusion de i valeurs et chaque application de $\mathcal{D}$ augmentant le nombre de sommets contactés, et il y en a au moins deux au départ, on a au plus $n-2$ diffusions associées à chaque valeur.

Le nombre de changement de numéros possible est limité par le nombre de numéro que peut prendre chaque sommet. Pour tout i, il y a au moins $i-1$ sommets qui ont pour numéro $j < i$ et qui ne peuvent en changer. Donc il y a au plus $n-i+1$ sommets qui peuvent prendre le numéro i. Par conséquent, il y a au plus

$$\sum_{i=1}^{n} n - i + 1 = \frac{n(n+1)}{2}$$

applications de la règle de Changement de numéro. □

2.2 Complexité en Espace

On remarque à la lecture de la règle $\mathcal{D}$ que celle-ci est très gourmande en espace puisque l'on diffuse des informations inutiles. En fait, la partie de M utile est exactement l'ensemble $\{(n, N)\}$ des numéros et vues locales telle que $\underline{\text{CHAMPION}}(n, N, M)$ est vérifié. L'algorithme fonctionne de manière identique si l'on remplace M par cet ensemble. Cette considération n'était pas une priorité et l'exposition et la preuve de l'algorithme est plus simple ainsi.

On retrouve cette même remarque et des simplifications supplémentaires dans [Dem98], où un "protocole économique", dans le cadre du passage de messages, est donné par P. Dembinski, pour implémenter l'algorithme de Mazurkiewicz. La taille d'un message y est alors bornée par la taille du graphe.

2.3 Une Exécution en $\Omega(n^3)$

La preuve pour la borne supérieure est issue de majorations assez grossières, et l'on peut se demander si il est possible que cela se passe assez mal pour qu'il y ait effectivement un nombre quadratique de changements de numéros. L'exemple générique suivant démontre que c'est effectivement et malheureusement le cas. Soit G_m le graphe suivant (voir la fig. 2.1) :

$$\begin{aligned}
V(G_m) &= \{a_1, \ldots, a_m\} \cup \{b_1, \ldots, b_m\} \cup \{c_1, \ldots, c_m\} \cup \{d\} \cup \{e_1, \ldots, e_m\} \cup \{f_1, \ldots, f_m\} \\
E(G_m) &= \{\{a_i, b_i\} \mid i \in [1, \ldots, m]\} \\
&\quad \cup \{\{b_i, c_i\} \mid i \in [1, \ldots, m]\} \\
&\quad \cup \{\{c_i, d\} \mid i \in [1, \ldots, m]\} \\
&\quad \cup \{\{d, e_i\} \mid i \in [1, \ldots, m]\} \\
&\quad \cup \{\{e_i, f_i\} \mid i \in [1, \ldots, m]\}
\end{aligned}$$

La chaîne de réétiquetage présentera deux grande phases, la première de mise en place, la seconde à la complexité de laquelle nous nous intéresserons consistera à répéter sur les différentes branches de G_m les même motifs de réécriture. La preuve que cette chaîne est valide se fait par induction et est omise.

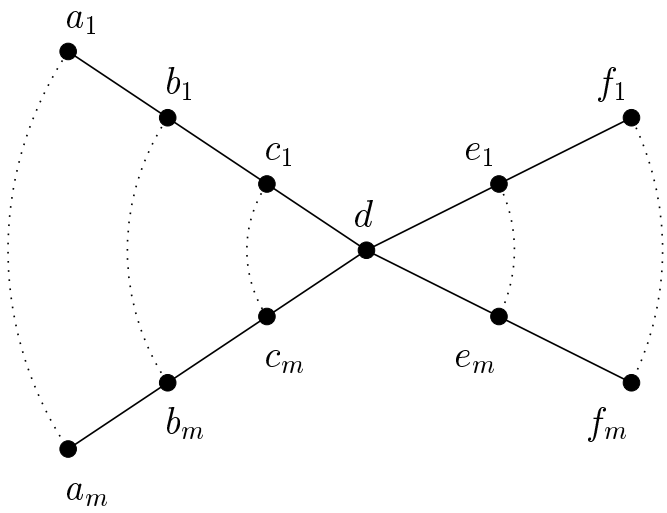


FIG. 2.1 – Le graphe "étoilé" G_m.

1. Phase de mise en place (de chaîne $\prod_{1\leq i\leq m}(\mathcal{C}a_i\mathcal{D}_{\max})\left(\prod_{1\leq i\leq m}\mathcal{C}f_i\right)\mathcal{D}_{\max}\prod_{1\leq i\leq m}\mathcal{C}b_i$) :
   (a) Pour $i \in [1,\ldots,m]$
      – a_i applique la règle de changement de numéro.
      – Diffusion $\mathcal{D}_{\max}$
   (b) Pour $i \in [1,\ldots,m]$, f_i applique la règle de Changement de numéro.
   (c) Diffusion $\mathcal{D}_{\max}$.
   (d) Pour $i \in [1,\ldots,m]$, b_i applique la règle de Changement de numéro.

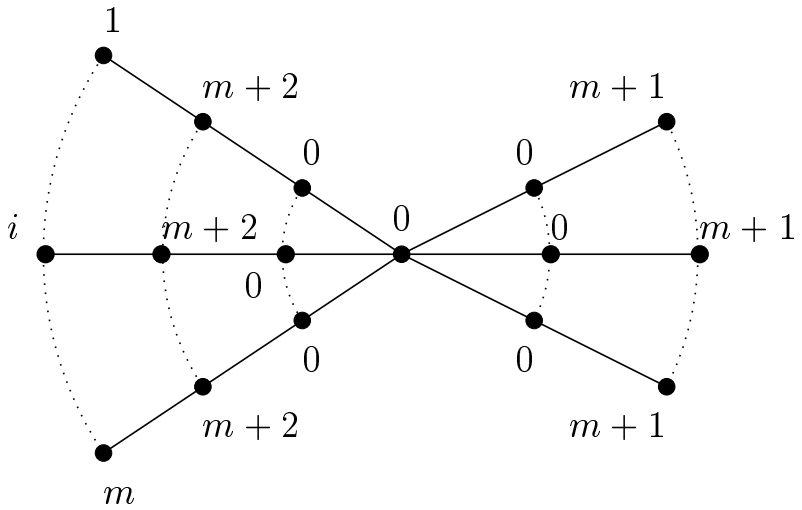


FIG. 2.2 – Numéros des sommets de G_m après la phase de "mise en place"

2. Phase "complexe", de chaîne $\prod_{1\leq j\leq m} j\left(\mathcal{C}e_j\mathcal{D}d\prod_{1\leq i\leq m}\left(\mathcal{D}b_i\mathcal{C}b_i\mathcal{D}c_i\prod_{1\leq k\leq m}\mathcal{D}e_k\right)\right)$:

- Pour $j \in [1, \ldots, m]$
  - e_j se renumérote
  - d diffuse
  - Pour $i \in [1, \ldots, m]$
    (a) b_i diffuse
    (b) b_i se renumérote.
    (c) c_i diffuse
    (d) Pour $k \in [1, \ldots, m]$, e_k diffuse

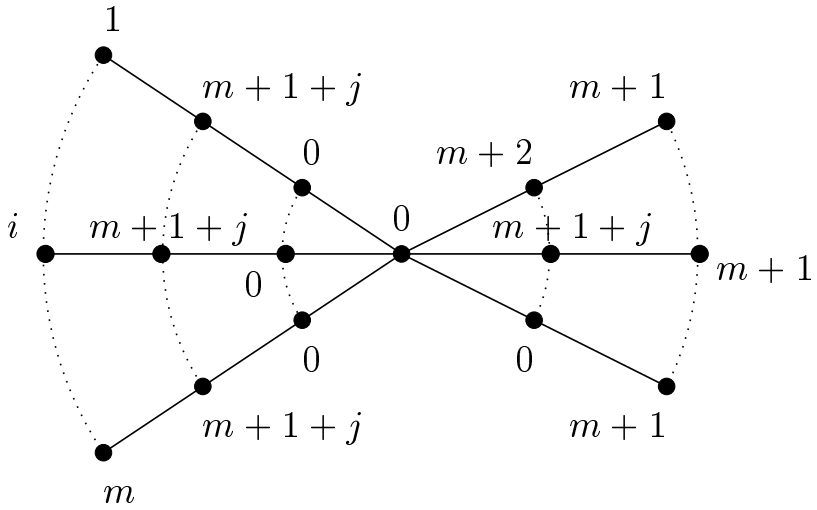


FIG. 2.3 – Numéros des sommets de G_m au cours de la phase "complexe" au pas j : le sommet e_j l'emporte systématiquement sur ses rivaux b et ceux-ci doivent changer de numéro.

Une analyse simple montre alors que la phase complexe met en œuvre $m(1 + 1 + m(1 + 1 + 1 + m)) \sim m^3$ applications des règles de $\mathcal{M}$, car à chaque tour, l'information connue de d est "collectée" par un seul e_i à la fois. Si on note $n = |V(G_m)| = 5m + 1$, on a une complexité d'au moins $\frac{1}{125}n^3$, *i.e.* en $\Omega(n^3)$.

Remarque 2.2. On remarquera que sur les graphes complets K_n, l'exécution est linéaire, il n'y a en fait aucune diffusion. Plus généralement, une exécution du type de la Prop. 1.16 section 1.3 est quadratique. Les expérimentations effectuées au moyen de ViSiDiA donnent également une complexité quadratique en moyenne [Mos].

2.4 Exécution parallèle en $\Omega(\Delta n^2)$

Ce qui pénalise l'exécution précédente, c'est que la boucle la plus intérieure (diffusion des e_k) est comptée séquentiellement alors qu'elle peut s'exécuter en parallèle. Dans ce cas, une borne pour la complexité parallèle est donné en allongeant les bras droits de l'étoile et on obtient une exécution en $m(1 + 1 + m(2 + 1 + \Delta - 4)) \sim \Delta n^2$ où Δ est le diamètre.

Chapitre 3

Extensions de l'Algorithme de Mazurkiewicz

La pleine exploitation de l'algorithme de Mazurkiewicz apparaîtra dans les parties suivantes. Et afin d'atteindre toute la généralité possible pour les résultats obtenus, on va dès à présent étendre l'algorithme de Mazurkiewicz à différents domaines. dans un premier temps, nous présenterons une variante autostabilisante de l'algorithme puis nous l'étendrons aux graphes étiquetés et enfin aux réécritures opérant sur des boules de rayon arbitraire.

3.1 Version Autostabilisante

3.1.1 Autostabilisation

Le concept d'autostabilisation fut introduit par [Dij74] dans l'optique de concevoir des algorithmes d'une fiabilité exigeante : commençant dans un état arbitraire et corrompu, ces algorithmes atteignent le but recherché en temps fini sans aucune intervention extérieure ou centralisée. La théorie des réseaux anonymes décrit l'existence ou non d'algorithme résolvant des conflits globaux par des calculs locaux sans que chaque nœud n'ait accès à un identifiant *unique* Il ne s'agit donc pas seulement d'un intérêt théorique de déterminer ce qui fonctionne dans ce pire des pire des cas (non-disponibilité d'identités bien utiles et valeurs initiales arbitraire), il y a également des considérations de résistance aux pannes très importantes puisque la famille où le problème peut être résolu, la famille $\mathcal{G}_{\min}$ des graphes minimaux , est assez vaste. Ce travail a été publié dans [God02].

On commencera par donner la définition d'un système auto-stabilisant [Dol00]. Étant donné un réseau, une exécution est une séquence (finie ou infinie) de configurations. Une *exécution correcte* d'un algorithme est une exécution pour laquelle on passe de chaque configuration à la suivante par une règle de l'algorithme. Une tâche distribuée est alors un ensemble d'exécutions admissibles et une configuration c est *sûre* si toute exécution commençant en c est admissible. Une tâche admet une solution autostabilisante si il existe un algorithme pour lequel, de toute configuration initiale, on

peut atteindre, par une exécution correcte, une configuration sûre.

Ici, la tâche d'énumération est définie par l'ensemble des configurations où un registre particulier sur chaque processeur, le registre ID, définit une bijection de l'ensemble des nœuds vers $\{1, \ldots, |V|\}$. Ce ne sera en fait qu'un simple problème technique de calculer les ID à partir des numéros et boîtes aux lettres obtenus par l'algorithme de Mazurkiewicz.

3.1.2 Variante autostabilisante

La description générale de l'algorithme est assez similaire à celle de $\mathcal{M}$, il faudra juste être un peu plus précautionneux du fait du démarrage dans un état corrompu. Comme précédemment, chaque sommet tente d'obtenir un numéro qui lui soit propre. Chaque sommet se choisit donc un numéro et le diffuse accompagné de sa vue locale. Si un sommet u s'aperçoit qu'il y a un autre sommet qui a le même numéro mais avec une vue locale plus forte, u change de numéro. Il change également de numéro s'il s'aperçoit que son numéro apparaît deux fois dans une vue locale (comme résultat d'un état initial corrompu). Chaque numéro est de nouveau diffusé jusqu'à stabilisation. A stabilisation, on a juste l'assurance que les sommets de même numéros ont les même vues locales. Ces numéros ne peuvent être immédiatement utilisés comme identités car ils peuvent dépasser $|V(G)|$. Cependant on peut calculer le rang des numéros parmi les numéros effectivement attribués et ainsi donner une valeur correcte aux registres ID.

Les vues locales seront des multiensembles ordonnés de $\mathbb{N}$. L'ordre total sur l'ensemble des vues locales, noté $\mathcal{P}_<(\mathbb{N})$, sera l'ordre alphabétique, et sera noté, comme précédemment $\preccurlyeq$.

Les sommets seront étiquetés par des couples de la forme (n, M) qui représenteront, pendant le calcul :

- $n(v) \in \mathbb{N}$ est le numéro du sommet v,
- $M(v) \subset \mathbb{N} \times \mathcal{P}_<(\mathbb{N})$ est la boîte aux lettres de v et contiendra tous les messages (n, N) reçus au cours du déroulement de l'algorithme ou présents dès le départ.

Remarque 3.1. On remarquera l'absence de l'étiquette $N(v)$ représentant la vue locale. En fait celle-ci peut être recalculée à tout moment, et, du fait de sa corruption initiale possible, il est plus simple de ne pas la mémoriser dans l'étiquette du sommet.

On pourrait en fait procéder de même pour la version initiale de $\mathcal{M}$. Mais comme, originellement, A. Mazurkiewicz ne l'a pas fait, et que c'est, nous semble-t-il aussi, plus clair ainsi, nous avons conservé cette présentation au chapitre précédent.

Afin de déterminer les numéros corrompus, on veut également pouvoir compter le nombre de fois qu'un numéro donné apparaît dans une vue locale. Si N est une vue locale, et $n \in \mathbb{N}$, on définit $\delta_N(n)$ comme étant le nombre d'occurrences de n dans le tuple N.

On rappelle que dans le contexte de l'autostabilisation, les étiquettes initiales de tous les sommets sont des éléments arbitraires de $\mathbb{N} \times \mathcal{P}_<(\mathbb{N})$.

Avec les conventions habituelles, les règles sont maintenant décrites ci-dessous pour une boule $B = B(v_0)$ de centre v_0. Les sommets v de B ont pour étiquettes

$(n(v), M(v))$. Les étiquettes obtenues après application de la règle de réétiquetage sont $(n'(v), M'(v))$.

$\mathcal{D}_{\text{a.s.}}$: **Diffusion et Mise à jour éventuelle**

Condition préalable :

– $\exists v \in B(v_0), M(v) \neq M(v_0)$

ou

– $(n(v_0), N(v_0)) \notin M(v_0)$.

Réétiquetage :

– Pour tout $v \in B(v_0)$, $M'(v) := \bigcup_{w \in B} M(w) \ \cup (n(v_0), N(v_0))$

$\mathcal{C}_{\text{a.s.}}$: **Changement de numéro**

Condition préalable :

– $(n(v_0), N(v_0)) \in M(v_0)$

– $\forall v \in B(v_0), M(v) = M(v_0)$

– 1. $n(v_0) = 0$

ou

2. $(n(v_0) > 0$ et $\exists N_1\ (n(v_0), N_1) \in M(v_0))$ et $N(v_0) \prec N_1$

ou

3. $(n(v_0) > 0$ et $\exists (n_1, N_1) \in M(v_0)$ tel que $\delta_{N_1}\ (n(v_0)) \geq 2$

Réétiquetage :

– $n'(v_0) := 1 + \max\{n \in \mathbb{N} \mid (n, N) \in M(v_0)$ pour un certain $N \in \mathcal{P}_<(\mathbb{N})\}$

– Pour tout $v \in B(v_0), M'(v) := M(v_0) \cup \{(n'(v_0), N(v_0))\}$.

3.1.3 Correction

Soit G un graphe quelconque.

Théorème 3.2. *Toute exécution de $\mathcal{M}_{\text{a.s.}}$ sur G étiqueté arbitrairement au départ se termine*[2].

Démonstration. Soit t le plus grand numéro apparaissant dans l'étiquetage initiale (sur les étiquettes n et dans les étiquettes M). La preuve est très similaire à celle du Th. 1.5, les lemmes n'étant plus vérifiés que pour des numéros supérieurs à t.

Soit $I = \{0, \ldots, t + |V(G)|\}$ et soit $\mathcal{N}_I$ l'ensemble des parties ordonnées de I. L'ensemble $\mathcal{N}_I$ est ordonné par $\preccurlyeq$. On ordonne $\mathcal{P}(I \times \mathcal{N}_I)$ par inclusion ($\subset$). Soit $X = (I \times \mathcal{N}_I \times \mathcal{P}(I \times \mathcal{N}_I))^{V(G)}$. L'ordre produit $\leqq$ définit un ordre partiel sur X.

[2]Dans le cadre de l'autostabilisation, la terminaison explicite n'a aucun sens puisqu'un état initial corrompu peut laisser croire que l'on se trouve dans un état terminal. Ici, il faudra entendre la terminaison au sens implicite, *i.e.* au sens stabilisation.

Notons $(n_i(v), N_i(v), M_i(v))$ l'étiquette v après la $i$ème étape de calcul. On note $x^i = ((n_i(v), N_i(v), M_i(v)))_{v \in V(G)}$ et on va montrer que $(x^i)_i$ est une suite *strictement* croissante de X.

Tout d'abord, on a le lemme de croissance suivant

Lemme 3.3. *Pour tout i et tout sommet v*
- $n_i(v) \leq n_{i+1}(v)$
- $N_i(v) \preccurlyeq N_{i+1}(v)$
- $M_i(v) \subseteq M_{i+1}(v)$

Démonstration. Le résultat est trivialement vrai pour les sommets qui ne sont pas concernés par l'application de la règle à l'étape i. Pour les autres sommets, on notera que $\mathcal{D}_{\text{a.s.}}$ se contente d'ajouter des éléments aux boîtes aux lettres. En ce qui concerne $\mathcal{C}_{\text{a.s.}}$, $n_i(v_0)$ ne peut qu'augmenter et, par conséquent la "force" des vues locales augmentent également par Lem. 1.4. □

De plus, l'une des inégalités est stricte pour au moins un sommet, en particulier pour celui sur lequel l'application de la règle était centrée. Par conséquent $x^i \lneqq x^{i+1}$ pour tout i.

On dira qu'un numéro m est *connu* par v si $(m, N) \in M(v)$ pour un certain N. Afin de prouver la terminaison, le lemme suivant affirme qu'à tout moment, dès lors qu'un numéro est connu, il y a un sommet qui porte ce numéro.

Lemme 3.4. $\forall i, \forall v \in V(G)\ \forall (m, N) \in M_i(v)$ *avec* $m > t$*, il existe* $w \in V(G)$ *tel que* $n_i(w) = m$.

Démonstration. Supposons que le numéro m soit connu par v_0 et soit $U = \{u \in V(G) \mid \exists j < i, n_j(u) = m\}$, alors U n'est pas vide puisque, étant supérieur à t, le numéro m a été diffusé au moins une fois. Soit w le sommet de U avec la plus forte vue locale dans $\{N_j(u) \mid u \in U, \exists j < i, n_j(u) = m\}$. Par conséquent, la règle de changement de numéro $\mathcal{C}_{\text{a.s.}}$ n'a pas pu être appliquée à w. Par conséquent $n_i(w) = m$. □

Lemme 3.5. $\forall i, \forall v \in V(G)\ \forall m > t, \delta_{N(v)}(m) \leq 1$.

Démonstration. Supposons que la propriété soit fausse, alors il existe un entier m et un sommet v avec deux voisins w_1 et w_2 nommés m, comme il n' y avait aucun m dans l'étiquetage initial, m a été obtenu par des applications de la règle de changement de numéro. Considérons le sommet w_2 ayant été numéroté m en dernier. w_2 a pour voisin v et la première application de la règle de changement de numéro pour w_1 aurait dû faire connaître m à v et par conséquent à w_2 avant qu'il ne puisse appliquer la moindre règle de changement de numéro. On aboutit donc à une contradiction. □

La dernière étape consiste à montrer que lorsqu'un numéro est connu, alors tous les numéros plus petits sont connus, et par conséquent attribué à au moins un sommet.

Lemme 3.6. $\forall i, \forall v \in V(G)$ *tel que* $n_i(v) \neq 0$*,*

$$\forall m,\ t < m \leq n_i(v),\ v \textit{ connaît } m.$$

Démonstration. La preuve de cette assertion s'effectue par récurrence sur i. A l'étape initiale $i = 0$ et l'assertion est vraie, puisque vide.

Supposons que l'assertion soit vraie pour $i \geq 0$. Montrons qu'elle est vraie après application de la règle du pas $i + 1$. S'il s'agit d'une diffusion, alors il n'y a pas de changement de numéro et l'assertion est dans ce cas facilement vérifiée.

Si la règle suivante est la règle de changement de numéro appliquée à v_0 alors, il suffit de vérifier l'assertion pour v_0, c'est-à-dire de prouver que les numéros de $[t+1, \ldots, n_{i+1}(v_0)-1]$ sont connus par v_0. Si $n_{i+1}(v_0)-1 < t+1$, le résultat est immédiat, on supposera donc que $n_{i+1}(v_0) - 1 \geq t + 1$. Par définition du changement de numéro proprement dit de $\mathcal{C}_{\text{a.s.}}$, $n_{i+1}(v_0) - 1$ est connu à l'étape i par v_0. Par hypothèse de récurrence, tous les numéros de $[t + 1, \ldots, n_{i+1}(v_0) - 1]$ sont donc connus par v_0, et $n_{i+1}(v_0)$ est connu de v_0 par le réétiquetage de mise à jour de la boîte à lettres de $\mathcal{C}_{\text{a.s.}}$. □

Comme il ne peut y avoir qu'au plus $|V(G)|$ numéros différents assignés à un moment donné, on en déduit que, pour tout i, pour tout sommet v, $n_i(v) \leq t + |V(G)|$ et par conséquent $x^i \in X$. X est un espace ordonné fini et donc la suite strictement croissante (x^i) est finie. Le système de réétiquetage $\mathcal{M}_{\text{a.s.}}$ est nœthérien. □

Le Théorème 1.10 est vrai, avec une preuve identique, pour $\mathcal{M}_{\text{a.s.}}$.

3.1.4 Enumeration forte

Maintenant chaque sommet va calculer l'ensemble des numéros finaux. Pour une boîte aux lettres donné M et un entier $n \in \mathbb{N}$ donné, on note $N_M(n)$ la vue locale telle que $\underline{\text{CHAMPION}}(n, N, M)$.

On définit l'ensemble $V_M(n)$ par récurrence :

$$\begin{aligned} V_0 &= \{n\} \\ V_{i+1} &= V_i \cup \{t \mid \exists s \in V_i, \delta_{N_M(s)}(t) = 1\} \end{aligned}$$

Si i_0 est tel que $V_{i_0} = V_{i_0+1}$ alors on définit

$$V_M(n) = V_{i_0}$$

A tout moment, chaque sommet u calcule son *indice de numérotation* ID en calculant le rang de $n(u)$ dans $V_M(n(u))$. Finalement, si on note $V_\rho = \{n_\rho(v) \mid v \in V(G)\}$, on a par induction que

Lemme 3.7. *Pour tout u, $V_\rho = V_{M_\rho}(n_\rho(u))$.*

Ce qui signifie que les registres ID finissent par recevoir une attribution correcte. Et donc, comme dans le chapitre précédent, après stabilisation, ID induit une correspondance bijective entre l'ensemble des sommets de G et l'ensemble des entiers $\{1, \ldots, |V(G)|\}$.

Comme précédemment, on définit le graphe H_M calculé par $\mathcal{M}_{\text{a.s.}}$. On pose

$$\begin{aligned} V(H_M) &= V_M \\ E(H_M) &= \{s,t \in V_M \mid \delta_{N_M(s)}(t) = 1\} \end{aligned}$$

3.2 Extension aux Graphes Étiquetés

On peut étendre l'algorithme de Mazurkiewicz pour tenir compte d'éventuelles étiquettes, que l'on supposera ordonnées. Le point central est de définir un ordre sur les vues locales étiquetées qui assure la terminaison de l'algorithme.

3.2.1 Un nouvel ordre

On commencera par donner une description générale de l'algorithme $\mathcal{M}$ appliqué au graphe $\mathbf{G}$. Soit $\mathbf{G} = (G, \lambda)$, soit v_0 un sommet de G, soit $\{v_1, ..., v_d\}$ l'ensemble des voisins de v_0.

L'étiquetage du sommet v_0 utilisé par $\mathcal{M}$ est le couple $(\lambda(v_0), c(v_0))$ où $c(v_0)$ est un triplet $(n(v_0), N(v_0), M(v_0))$ qui représentera l'information suivante lors des calculs (les définitions formelles seront présentées ensuite) :

- $n(v_0) \in \mathbb{N}$ est le numéro du sommet v_0 calculé par l'algorithme
- $N(v_0) \in \mathcal{N}_L$ est la vue locale étiquetée de v_0, c'est l'ensemble des triplets :

$$\{(n(v_i), \lambda(v_i), \lambda(\{v_0, v_i\}))|1 \leq i \leq d\}$$

- $M(v_0) \subset \mathbb{N} \times L \times \mathcal{N}_L$ est la boîte aux lettres de v_0, et contient toute l'information reçue par v_0 à ce stade du calcul.

Le principe de l'algorithme est identique et la propriété importante de celui-ci se fonde sur un ordre total des vues locales étiquetées tel que celles-ci augmentent continuellement lors du calcul. On supposera que l'ensemble d'étiquettes L est muni d'un ordre total noté $>_L$. Soit v_0 un sommet et soient $\{v_1, ..., v_d\}$ les voisins de v_0 , on suppose que : $n(v_1) \geq n(v_2) \geq ... \geq n(v_d)$, si $n(v_i) = n(v_{i+1})$ alors $\lambda(v_i) \geq \lambda(v_{i+1})$, si $n(v_i) = n(v_{i+1})$ et $\lambda(v_i) = \lambda(v_{i+1})$ alors $\lambda(\{v_0, v_i\}) \geq_L \lambda(\{v_0, v_{i+1}\})$. Alors à $N(v)$, la vue locale, on associe le d-tuple $((n(v_1), \lambda(v_1), \lambda(\{v_0, v_1\})), \ldots, (n(v_d), \lambda(v_d), \lambda(\{v_0, v_d\})))$. Soit $\mathcal{N}_{>_L}$ l'ensemble des tuples ainsi ordonnés. On définit un ordre total $\prec$, sur $\mathcal{N}_{>_L}$ en comparant le numéro, puis les étiquettes des sommets, puis les étiquettes des arêtes. Formellement, soient $((n_1, l_1, e_1), ..., (n_d, l_d, e_d))$ et $((n'_1, l'_1, e'_1), ..., (n'_{d'}, l'_{d'}, e'_{d'}))$ deux éléments de $\mathcal{N}_{>_L}$ alors

$$((n'_1, l'_1, e'_1), ..., (n'_{d'}, l'_{d'}, e'_{d'})) \prec ((n_1, l_1, e_1), ..., (n_d, l_d, e_d))$$

si l'une des conditions suivantes est remplie :

1. $n_1 = n'_1, ..., n_{i-1} = n'_{i-1}$ et $n'_i < n_i$ pour un certain i
2. $d' < d$ et $n_1 = n'_1, ..., n_{d'} = n'_{d'}$
3. $d = d'$, $n_1 = n'_1, ..., n_d = n'_d$ et $l_1 = l'_1, ..., l_{i-1} = l'_{i-1}$ et $l'_i <_L l_i$ pour un certain i
4. $d = d'$ et $n_1 = n'_1, ..., n_d = n'_d$ et $l_1 = l'_1, ..., l_d = l'_d$ et $e_1 = e'_1, ..., e_{i-1} = e'_{i-1}$ et $e'_i <_L e_i$ pour un certain i.

3.2.2 Description des règles

L'étiquette initiale du sommet v_0 est $(\lambda(v_0), (0, \emptyset, \emptyset))$. Les règles sont décrites avec les conventions usuelles pour une boule centrée donnée $B = B(v_0)$, de centre v_0. Les sommets v de B ont pour étiquettes $(\lambda(v), (n(v), N(v), M(v))$. Les étiquettes obtenues après applications d'une règle sont $(\lambda(v), (n'(v), N'(v), M'(v)))$.

$\mathcal{D}$: **Règle de diffusion**

Condition préalable :

– $\exists v \in B(v_0), M(v) \neq M(v_0)$

Réétiquetage :

– Pour tout $v \in B(v_0)$, $M'(v) := \bigcup_{w \in B(v_0)} M(w)$

$\mathcal{C}$: **Règle de changement de numéro**

Condition préalable :

– $\forall v \in B(v_0), M(v) = M(v_0)$

– 1. $n(v_0) = 0$

ou

2. $(n(v_0) > 0$ et $\exists\ (n(v_0), l, N) \in M(v_0))$ tel que
   $((\lambda(v_0) < l)$ ou $((\lambda(v_0) = l)$ et $(N(v_0) \prec N)))$

Réétiquetage :

– $n'(v_0) := 1 + \max\{n \in \mathbb{N} \mid (n, l', N) \in M(v_0)\}$

– $\forall v \in B(v_0, 1)$, $N'(v) := \mathrm{sub}(N(v), n(v_0), n'(v_0))$

– $\forall v \in B(v_0, 1), M'(v) := M(v) \cup \bigcup_{w \in B} \{(n'(w), \lambda(w), N'(w))\}$.

On conservera la notation $\mathcal{M}$ pour ce système de réétiquetage de graphes, qui ne fait qu'étendre de manière évidente l'algorithme de Mazurkiewicz. Soit $\mathbf{G}$ un graphe étiqueté, si v est un sommet de $\mathbf{G}$, l'étiquette de v après une exécution ρ de l'algorithme de Mazurkiewicz est noté $(\lambda(v), n_\rho(v), N_\rho(v), M_\rho(v))$. Le théorème suivant est la transposition directe du théorème 1.10. La preuve est identique et nous ne la redonnons pas.

Théorème 3.8. *Toute exécution ρ de l'algorithme de numérotation de Mazurkiewicz $\mathcal{M}$ sur un graphe étiqueté connexe $\mathbf{G} = (G, \lambda)$ se termine et on obtient un étiquetage final $(\lambda(v), n_\rho(v), N_\rho(v), M_\rho(v))$ vérifiant la condition suivante pour tous les sommets v, v' de G :*

3.8.i $M_\rho(v) = M_\rho(v')$.

3.8.ii $(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$.

3.8.iii $n_\rho(v) = n_\rho(v')$ *implique* $(\lambda(v) = \lambda(v')$ *et* $N(v) = N(v'))$

Tout comme précédemment on interprète l'étiquetage final comme un graphe que chaque sommet peut calculer localement. On (re)définit le prédicat CHAMPION$(n, l, N,$

M) qui est vérifié si $N' \preccurlyeq N$ pour tout $(n, l, N') \in M$, en d'autres termes si N est la vue locale étiquetée maximale associée à (n, l) dans la boîte M.

De même, pour une boîte M, on définit le graphe étiqueté $\mathbf{H}_M$ par

$$\begin{aligned} V(\mathbf{H}_M) &= \{n \mid \exists N, l \text{ tels que } (n, l, N) \in M\} \\ E(\mathbf{H}_M) &= \{\{n, n'\} \mid \exists N, l, l' \text{ tels que } (n', l') \in N \text{ et } \underline{\textsc{Champion}}(n, l, N, M)\} \end{aligned}$$

L'étiquetage sur $\mathbf{H}_M$ est défini comme suit $\lambda_M(n) = (n, l, N, M)$, avec $\underline{\textsc{Champion}}(n, l, N, M)$. Il convient de noter que l'unicité de N et l provient de la définition de $\underline{\textsc{Champion}}$ et du fait que $\prec$ est un ordre total. Soit ρ une exécution de $\mathcal{M}$, alors $(\mathbf{H}_{M_\rho(u)}, \lambda_{M_\rho(u)})$ ne dépend pas de u par Th. 3.8. On peut alors, comme précédemment définir, $\mathbf{H}_\rho = (\mathbf{H}_{M_\rho(u)}, \lambda_{M_\rho(u)})$.

Le théorème sur lequel repose une partie notable de notre étude est finalement, en toute généralité, le suivant :

Théorème 3.9. *Soit $\mathbf{G}$ un graphe étiqueté. Pour toute exécution ρ de $\mathcal{M}$ sur le graphe $\mathbf{G}$ de graphe irréductible $\mathbf{G}_f$, $\mathbf{G}_f$ est un revêtement de $\mathbf{H}_\rho$.*

3.3 Extension aux Réétiquetages de Rayon $k \geq 1$

L'algorithme peut être étendu au cas où l'on autorise des réétiquetages sur des boules de rayon k. En effet, on peut supposer que la synchronisation locale s'effectue sur un sous-graphe centré sur u_0 et de rayon k, voir sect. 0.5.

3.3.1 Adaptation de l'algorithme

La propriété fondamentale de l'algorithme est ici encore basée sur un ordre total des vues locales (de rayon k) tel que celles-ci ne puissent qu'augmenter pendant le déroulement du calcul. Soit $\mathcal{N}_k$ l'ensemble des $k-$boules étiquetés par des entiers positifs, à isomorphisme près[3] Soit $\blacktriangleleft$ un ordre total sur l'ensemble des $k-$boules (non étiquetées). Celui-ci peut être quelconque il suffit qu'il soit total.

On notera $W((B, \lambda)) = \sum_{v \in V(B)} \lambda(v)$ le *poids* de la boule étiquetée (B, λ). Pour toute $k-$boule B et tout entier w on choisit un ordre total (encore arbitraire) $\blacktriangleleft_w^B$ sur l'ensemble *fini* des boules étiquetés sur $\mathbb{N}$ de graphe sous-jacent B et de poids w.

On définit maintenant l'ordre total $\preccurlyeq_k$ sur $\mathcal{N}_k$ par $(B_1, \lambda_1) \preccurlyeq (B_2, \lambda_2)$ si l'une des conditions suivantes est satisfaite :

1. $B_1 \blacktriangleleft B_2$
2. $B_1 = B_2$ et $W(B_1, \lambda_1) \leq W(B_2, \lambda_2)$.
3. $B_1 = B_2$, $W(B_1, \lambda_1) = W(B_2, \lambda_2)$ et $(B_1, \lambda_1) \blacktriangleleft_w^B (B_2, \lambda_2)$, où $B = B_1 = B_2$ et $w = W(B_1, \lambda_1) = W(B_2, \lambda_2)$.

Remarque 3.10. On notera bien qu'on ne fait aucune hypothèse particulière sur les ordres $\blacktriangleleft$ et $\blacktriangleleft_w^B$, excepté que la comparaison doit être calculable (récursive).

[3] Dans toute la suite, les $k-$boules seront considérées à isomorphisme près.

On va maintenant modifier les règles en conséquence. La principale différence proviendra du fait que l'on aura besoin d'une nouvelle **Règle de mise à jour** et d'une nouvelle étiquette, le drapeau f. La raison en est qu'une vue locale de rayon 1, comme dans le cas précédent, est un simple ensemble et donc les vues locales des voisins peuvent être remises à jour directement pendant la règle de changement de numéro, par le sommet central, ce qui n'est pas possible a priori pour les $k-$boules. En effet, tant que les numéros sont à 0, un sommet ne peut se repérer sur la vue de rayon k de son voisin. La figure 3.1 présente une telle situation où le sommet u ne peut se situer sur la vue de rayon 2 de v. D'après sa propre vue de rayon 2, il pourrait très bien se situer en u', ce qui n'est pas équivalent.

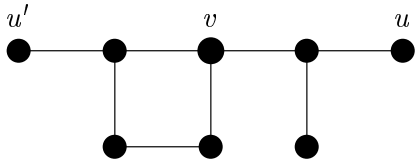


FIG. 3.1 – D'après sa $2-$vue, u ne peut pas conclure quant à sa situation dans la $2-$vue de v

Un sommet v est maintenant étiqueté par un tuple de la forme (n, N, M, f) représentant l'information suivante :

- n, N, M représente exactement la même chose que précédemment
- $f(v) \in \{\text{ÀJOUR}, \text{PÉRIMÉ}\}$ est un drapeau qui sera levé (PÉRIMÉ) quand le sommet devra remettre à jour sa vue locale, après changement de numéro d'un de ses voisins.

3.3.2 Les règles

La règle à ajouter est celle où v_0 met à jour sa vue locale :

$\mathcal{M}_k$–3 : **Règle de mise à jour**
Condition préalable :
– $f(v_0) = \text{PÉRIMÉ}$
Réétiquetage :
– r $N'(v_0) := (B(v_0, k), \lambda)$, où λ est donné par $\lambda(v) = n(v)$.
– $f'(v_0) := \text{ÀJOUR}$.

Les règles de diffusion et de changement de numéro sont adaptées pour tenir compte du drapeau f.

$\mathcal{M}_k$–1 : **Diffusion**
Condition préalable :
– $\exists v, M(v) \neq M(v_0)$
– $f(v_0) = \text{ÀJOUR}$

Réétiquetage :
– Pour tout v, $M'(v) := \bigcup_{w \in B} M(w)$

$\mathcal{M}_k$–2 : **Changement de numéro**
Condition préalable :
– $\forall v, M(v) = M(v_0)$
– $f(v_0) =$ ÀJOUR
– 1. $n(v_0) = 0$

ou

2. $(n(v_0) > 0$ et $\exists N_1\ (n(v_0), N_1) \in M(v_0))$ et $N(v_0) \prec N_1$

Réétiquetage :
– $n'(v_0) = 1 + \max\{n \in \mathbb{N} \mid (n, N) \in M(v_0)$ pour un certain $N \in \mathcal{N}\}$
– $N'(v) =$ sub $(N(v), n(v_0), n'(v_0))$
– Le contenu de la boîte à lettre change
$M'(v) = M(v_0) \bigcup_{w \in B} (n'(w), N'(w))$
– $f(v) =$ PÉRIMÉ.

En tenant compte de cette nouvelle règle et du fait qu'à tout changement de numéro d'un sommet correspondra, avant application de toute autre règle pour ses voisins, une mise à jour de la vue locale, on peut adapter directement les preuves de terminaison et de correction précédentes.

Deuxième partie

Reconnaissance de Familles de Graphes

Dans cette partie nous nous intéressons au problème que nous pensons être le plus simple pour commencer notre étude. Il s'agit de caractériser pour quelles familles de graphes $\mathcal{F}$ il existe un système de réétiquetage qui, pour tout graphe, se stabilise (terminaison implicite) dans une configuration telle que l'on puisse déterminer, en regardant uniquement quelles sont les étiquettes finales, si le graphe est dans $\mathcal{F}$ ou non. En d'autres termes, il s'agit de déterminer quelles fonctions du type "le graphe sous-jacent a-t-il ou n'a-t-il pas telle propriété ?" sont réalisables par système de réétiquetage de graphes. Nous appellerons ces types de problèmes "problème de reconnaissance".

Nous commencerons par aborder la reconnaissance sans information structurelle *a priori* sur le réseau sous-jacent afin de présenter la démarche qui sera la notre dans la suite de ce mémoire et de montrer précisément à quoi correspondent les conditions de "similarité" formalisées par les revêtements. Ensuite nous verrons comment généraliser les résultats obtenus lorsqu'est disponible une connaissance structurelle arbitraire. Chacun des deux chapitres se terminera par des exemples simples d'application des théorèmes. Un dernier chapitre d'applications présentera les résultats concernant la non-reconnaissabilité des classes fermées par mineurs même en connaissant la taille et la hiérarchie de pouvoir d'expression qu'induisent les connaissances structurelles.

Les résultats obtenus ici ont été présentés en tant que résumé dans [GMM00] (avec Y. Métivier et A. Muscholl) pour la partie condition nécessaire de la caractérisation et applications aux mineurs de graphes ; et dans [GM01] (avec Y.Métivier) pour la partie condition suffisante et hiérarchie de reconnaissabilité induite par les connaissances structurelles. Une version complète est présentée dans [GMM02].

Chapitre 4

Reconnaissance sans Connaissance Structurelle

Ce chapitre présente la version initiale du problème de la reconnaissance, c'est-à-dire lorsque l'on ne dispose d'absolument aucune information sur le graphe sous-jacent. On présentera tout d'abord les définitions puis la caractérisation obtenue.

4.1 Reconnaissance de Graphes Étiquetés

Soit $\mathcal{F}$ une famille de graphes étiquetés. On dira que $\mathcal{F}$ peut être localement reconnue si il existe un système de réétiquetage de graphes tel que, partant de n'importe quel graphe étiqueté $\mathbf{G}$ un étiquetage final peut être atteint, par réétiquetages successifs, permettant de décider si $\mathbf{G}$ appartient à $\mathcal{F}$ ou non. On commence par donner une définition formelle pour cette notion, puis on expliquera comment ceci peut être appliqué aux graphes non étiquetés.

4.1.1 Définitions

Définition 4.1. *Un* système de reconnaissance de graphes étiquetés *est un couple* $(\mathcal{R}, \mathcal{K})$ *où* $\mathcal{R}$ *est un système de réétiquetage de graphes,* $\mathcal{K}$ *est un ensemble d'étiquettes. Un graphe* $\mathbf{G}$ *est* reconnu *par* $(\mathcal{R}, \mathcal{K})$ *si il existe* $\mathbf{G}_f \in \mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$*, tel que l'ensemble des étiquettes de* $\mathbf{G}_f$ *est inclus dans* $\mathcal{K}$*.*

Un système de reconnaissance sera dit déterministe*, si pour tout graphe* $\mathbf{G}$*,*

- *soit pour tout* $\mathbf{G}_f \in \mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$*, l'ensemble des étiquettes de* $\mathbf{G}_f$ *est inclus dans* $\mathcal{K}$
- *soit pour tout* $\mathbf{G}_f \in \mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$*, l'ensemble des étiquettes de* $\mathbf{G}_f$ *est inclus dans le complémentaire de* $\mathcal{K}$*.*

On peut maintenant définir une famille de graphes reconnaissable.

Définition 4.2. *Une famille* $\mathcal{F}$ *de graphes étiquetés est dit* reconnaissable *si il existe un système de reconnaissance de graphes* $(\mathcal{R}, \mathcal{K})$ *dont l'ensemble des graphes reconnus est exactement* $\mathcal{F}$*.*

On s'intéresse uniquement aux familles reconnues par un système déterministe.

4.1.2 Le cas des graphes (non étiquetés)

On étendra ces notions à la reconnaissance de graphes non étiquetés de la manière suivante. Un système de reconnaissance de graphes (non étiquetés) est défini par un triplet $(\mathcal{R}, \mathcal{K}, l_0)$ où $(\mathcal{R}, \mathcal{K})$ est un système de reconnaissance de graphes étiquetés et l_0 une étiquette. Un graphe est reconnu par $(\mathcal{R}, \mathcal{K}, l_0)$ si le graphe étiqueté (G, Λ_{l_0}) est reconnu par $(\mathcal{R}, \mathcal{K})$, où Λ_{l_0} est l'étiquetage uniforme de chaque sommet par l_0. Ici l_0 peut donc être n'importe quelle étiquette car elle n'aura pas de signification particulière. Par la suite, on utilisera systématiquement le mot vide ε pour l_0.

Un exemple simple de reconnaissance est donné ci-dessous,

Exemple 4.3. **Reconnaissance des arbres** Le système suivant reconnaît les arbres. L'ensemble des étiquettes est $\{\varepsilon, \text{ÉLAGUÉ}, \text{ARBRE}\}$, et l'étiquetage d'un sommet v est notée $\lambda(v)$. L'étiquette initiale de tous les sommets est $l_0 = \varepsilon$. Les conventions usuelles s'appliquent.

REC_ARBRES1 : **Élagage**

Condition préalable :

- $\lambda(v_0) = \varepsilon$
- $\exists! v \in B(v_0), \lambda(v) = \varepsilon$

Réétiquetage :

- $\lambda'(v_0) :=$ ÉLAGUÉ

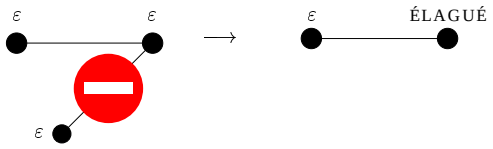


FIG. 4.1 – Règle d"élagage"

REC_ARBRES2 : **Détection de l'acyclicité**

Condition préalable :

- $\lambda(v_0) = \varepsilon$
- $\forall v \in B(v_0) \backslash \{v_0\}, \lambda(v) \neq \varepsilon$

Réétiquetage :

- $\lambda'(v_0) :=$ ARBRE.

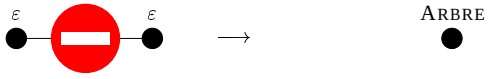


FIG. 4.2 – Apparition finale de l'étiquette "ARBRE"

REC_ARBRES3 : **Diffusion finale de l'étiquette ARBRE.**
Condition préalable :
– $\lambda(v_0) = \text{ARBRE}$
– $\exists v \in B(v_0), \lambda(v) \neq \text{ARBRE}$
Réétiquetage :
– $\forall v \in B(v_0), \lambda'(v) := \text{ARBRE}$.

On notera que dans les représentations (Fig. 7.1, 7.2) le "sens interdit" est un raccourci graphique exprimant la condition qu'il ne doit pas y avoir d'arêtes de ce type. Formellement, rappelons que ce sont en fait des règles génériques.

La condition finale est le singleton $\{\text{ARBRE}\}$. Le fonctionnement de ce système de reconnaissance repose sur le fait que la règle d'élagage "élague" récursivement les feuilles (sommets de degré 1 en ε), et cela ne laissera qu'un seul sommet étiqueté ε que si le graphe est sans cycle. La dernière règle permet, *si le graphe est un arbre,* d'informer tous les autres sommets "élagués" du fait que le graphe sous-jacent est un arbre. Si le graphe est un arbre alors l'étiquette finale de tout sommet sera ARBRE. Si le graphe n'est pas un arbre, sa forme irréductible contiendra des étiquettes ε. Il faut souligner que si le graphe sous-jacent n'est pas un arbre alors aucun sommet ne le saura explicitement. Voir section 8.3 sur l'équivalence des connaissances structurelles et en particulier la Prop. 8.25 pour voir que cela est en fait impossible.

4.2 Caractérisation des Familles de Graphes Reconnaissables

Cette section présente la caractérisation des familles reconnaissables de graphes étiquetés.

4.2.1 Une condition nécessaire classique

On rappelle que $\sim$ est la fermeture réflexive, symétrique et transitive de la relation "être revêtement de". Depuis les travaux d'Angluin [Ang80], il est bien connu que les familles de graphes reconnaissables doivent être fermées pour la relation revêtement. I.e. soit $\mathcal{F}$ une famille de graphes étiquetés, si $\mathcal{F}$ est reconnaissable alors $\mathcal{F}$ est clos pour $\sim$. Montrons ce résultat avec notre formalisme. C'est en fait un corollaire direct du Lemme 0.31 de Relèvement :

Lemme 4.4. *Soit $(\mathcal{R}, \mathcal{K})$ un système de reconnaissance de graphes. Soit $\mathbf{G}$, $\mathbf{H}$ deux graphes étiqueté, si $\mathbf{G}$ est un revêtement de $\mathbf{H}$ alors :*

1. *Si $\mathbf{H}$ est reconnu par $(\mathcal{R}, \mathcal{K})$ alors $\mathbf{G}$ est reconnu par $(\mathcal{R}, \mathcal{K})$.*
2. *Si $\mathbf{G}$ est reconnu de façon déterministe par $(\mathcal{R}, \mathcal{K})$ alors $\mathbf{H}$ est reconnu de façon déterministe par $(\mathcal{R}, \mathcal{K})$.*

Démonstration. 1. Par application du Lemme 0.31.

2. Considérons une chaîne de réétiquetages de $\mathbf{H}$. Par Lemme 0.31, celle-ci peut être relevée en une chaîne de réétiquetage de $\mathbf{G}$. Comme le système est nœthérien, cette chaîne relevée est finie et mène à un étiquetage final dont les étiquettes sont dans $\mathcal{K}$. Par conséquent, la chaîne de réétiquetage sur $\mathbf{H}$ est également finie mène à un étiquetage final qui utilise le même ensemble d'étiquettes, par conséquent l'ensemble des étiquettes finales est inclus dans $\mathcal{K}$.

□

Remarque 4.5. Si la preuve de 1 est une preuve basée sur la simulation, il faut bien noter que la preuve de la réciproque doit plutôt être vue comme une conséquence du déterminisme du système de reconnaissance : n'importe quelle exécution permet de conclure quant à la reconnaissance ou non-reconnaisance du graphe sous-jacent.

4.2.2 La condition d'Angluin est suffisante pour la reconnaissance

Le but de cette section est de prouver la réciproque du résultat d'Angluin et par conséquent d'obtenir :

Théorème 4.6. *Soit $\mathcal{F}$ une famille de graphes étiquetés. Les assertions suivantes sont équivalentes.*

1. *$\mathcal{F}$ est localement reconnaissable.*
2. *$\mathcal{F}$ est clos pour $\sim$ et $\mathcal{F}$ est un ensemble récursif.*

On suppose désormais qu'il existe un algorithme pour décider l'appartenance à $\mathcal{F}$.

Alors, nous allons décrire un algorithme qui reconnaît une famille de graphes donnée close pour $\sim$. Nos résultats sont obtenus à partir de l'algorithme de Mazurkiewicz décrit Section 3.1.4. Il vaut la peine de noter que, malgré sa simplicité, on va voir que cet algorithme fournit toute l'information sur le réseau sous-jacent que l'on peut calculer par système de réétiquetage de graphes.

Comme présenté précédemment l'algorithme de Mazurkiewicz appliqué à un graphe $\mathbf{G}$ permet de calculer distributivement un graphe $\mathbf{H}$ tel que $\mathbf{G}$ est un revêtement de $\mathbf{H}$. Par conséquent, $\mathcal{F}$ étant close pour $\sim$, $\mathbf{G} \in \mathcal{F}$ si et seulement si $\mathbf{H} \in \mathcal{F}$. Il "suffit" donc de tester si $\mathbf{H} \in \mathcal{F}$.

Dans cette partie, on décrit l'algorithme distribué permettant la reconnaissance d'une famille de graphes étiquetés donnée close pour $\sim$. On supposera, sans le rementionner expressément que $\mathcal{F}$ est récursif et clos pour $\sim$.

Dans le but de reconnaître le graphe sous-jacent, on utilisera le graphe $\mathbf{H}_\rho$ calculé par $\mathcal{M}$. On a

Proposition 4.7. *Soit $\mathcal{F}$ une famille de graphes étiquetés fermée pour $\sim$. Soit $\mathbf{G}$ un graphe. Alors pour toute exécution ρ de $\mathcal{M}$:*

$$\mathbf{G} \in \mathcal{F} \text{ si et seulement si } \mathbf{H}_\rho \in \mathcal{F}.$$

Démonstration. D'après la Prop. 3.9, on a $\mathbf{G} \sim \mathbf{H}_\rho$ pour toute exécution ρ. □

Pour compléter le système de reconnaissance, on ajoute une étiquette $r \in \{?, \underline{\text{VRAI}}, \underline{\text{FAUX}}\}$, qui contiendra le résultat du calcul effectué par la règle de test suivante

---

TEST$_{\mathcal{F}}$: **Test d'appartenance**
Condition préalable :
– r= ?
Réétiquetage :
– Calculer $\mathbf{H}_{M(v_0)}$
– Attribuer à r le résultat du test "$\mathbf{H}_{M(v_0)} \in \mathcal{F}$?"

Le système de reconnaissance $(\mathcal{R}, \mathcal{K})$ est le suivant :
– étiquettes : (λ, n, N, M, r),
– règles $\mathcal{R}$:
  – une règle de conversion initiale qui remplace, pour tout sommet v, l'étiquette $\lambda(v)$ par $(\lambda(v), 0, \emptyset, \emptyset, ?)$;
  – les règles de $\mathcal{M}$ travaillent sur (λ, n, N, M), mettant $r(v)$ à ? chaque fois qu'une règle est appliqué à v ;
  – la règle TEST$_{\mathcal{F}}$.
– condition finale $\mathcal{K}$: le registre r est $\underline{\text{VRAI}}$ sur tous les sommets.

Ce système de reconnaissance permet donc de prouver le théorème. Il est à noter que l'hypothèse sur la calculabilité de $\mathcal{F}$ est nécessaire et ne découle pas de la fermeture pour $\sim$, puisque l'on pourrait prendre un ensemble non récursif de classes d'équivalence de $\sim$, par exemple des arbres. Voir la section suivante pour plus de précision.

4.3 Applications

4.3.1 Caractérisation équivalente en termes de raffinement de degré étiqueté

On va pouvoir utiliser le critère de Leighton du Th. 0.53 pour étudier les familles closes pour $\sim$. Le raffinement de degré étant calculable pour tout graphe, on a le corollaire suivant :

Corollaire 4.8. *Soit G un graphe étiqueté. La classe d'équivalence de G est reconnaissable sans connaissance structurelle.*

En définitive, les familles de graphes reconnaissables peuvent être vues comme des ensembles récursifs de raffinements de degré étiqueté.

4.3.2 Cas particuliers

Graphes non-étiquetés

Pour les arbres, on a

Proposition 4.9. *Soit* $\mathbf{T}$ *un arbre. Alors la classe d'équivalence de* $\mathbf{T}$ *pour* $\sim$ *est le singleton* $\{\mathbf{T}\}$.

Le corollaire immédiat est

Corollaire 4.10. *Soit* $\mathcal{F}$ *une famille récursive d'arbres. Alors* $\mathcal{T}$ *est reconnaissable.*

Le système RECARBRES est un exemple de système reconnaissant $\mathcal{T}$, la famille des arbres. Soit d un entier. Les graphes $d-$réguliers sont une autre classe d'équivalence classique de $\sim$. On peut d'ailleurs considérer que le raffinement de degré généralise, d'un point de vue local, la notion de "régularité" des graphes $d-$réguliers.

Corollaire 4.11. *Soit* $d \in \mathbb{N}$. *La famille des graphes* d*-réguliers est reconnaissable.*

En particulier, la famille des anneaux est reconnaissable.

Graphes étiquetés

Un paradigme très intéressant de l'algorithme distribué peut s'exprimer en termes de reconnaissance de graphes étiquetés : le problème de la majorité (voir [LMS95b]). Le problème est le suivant, étant donné un graphe dont les sommets sont étiquetés par l'étiquette a ou b, décider si il y a autant, ou plus d'étiquettes a que de b. Une réponse positive est donné dans [LMS95b] où un système de reconnaissance avec 15 règles est donné. On peut maintenant voir ce résultat comme un corollaire du Théorème 4.6 puisque l'on constate par un calcul simple s'appuyant sur le nombre de feuillets (lemme 0.27) que la proportion $\frac{|\mathbf{G}|_\mathsf{a}}{|\mathbf{G}|_\mathsf{b}}$ est conservée par $\sim$, où $|\mathbf{G}|_\alpha$ dénote le nombre de fois où l'étiquette α apparaît sur $\mathbf{G}$. Une question ouverte de [LMS95b] était de savoir si l'on pouvait décider si $|\mathbf{G}|_\mathsf{a} \leq k|\mathbf{G}|_\mathsf{b}$ pour un certain k. La remarque précédente permet également de répondre positivement à cette question.

Nous reviendrons bien plus en détail sur ce sujet section 8.4, avec des contraintes de terminaisons explicites.

Chapitre 5

Reconnaissance avec Connaissance Structurelle

Concernant les problèmes de reconnaissance, on cherche maintenant à savoir ce que que peut apporter la connaissance *a priori* de certaines informations supplémentaires concernant des caractéristiques structurelles (la taille par exemple). Cette information sera encodée dans l'étiquetage initial.

5.1 Définitions

Soit $\mathbf{G} = (\mathbf{G}, \lambda)$ un graphe étiqueté et α une étiquette. Alors Λ_α est le sur-étiquetage uniforme de $\mathbf{G}$ avec l'étiquette α, *i.e.* chaque sommet v est étiqueté par le couple $(\alpha, \lambda(v))$.

La reconnaissance avec connaissance structurelle est alors définie par

Définition 5.1. *Un* système de reconnaissance de graphes étiquetés avec connaissance structurelle ι *est un triplet* $(\mathcal{R}, \mathcal{K}, \iota)$ *où* $(\mathcal{R}, \mathcal{K})$ *est un système de reconnaissance de graphes étiquetés (éventuellement non déterministe), et* ι *est une fonction qui associe à chaque graphe étiqueté* $\mathbf{G}$ *une étiquette* $\iota(\mathbf{G}) \in L$. *L'ensemble des graphes étiquetés* reconnus *par* $(\mathcal{R}, \mathcal{K}, \iota)$ *est l'ensemble* $\{\mathbf{G} \mid (\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$ *est reconnu par* $(\mathcal{R}, \mathcal{K})\}$.

Un système de reconnaissance de graphes $(\mathcal{R}, \mathcal{K}, \iota)$ *est dit* déterministe *si, pour toutes les entrées* $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$ *on a les deux propriétés suivantes*

- $\mathcal{R}$ *est nœthérien sur* $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$.
- *Soit pour tout graphe* $\mathbf{G}_f \in \mathrm{Irred}(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$ *l'ensemble des étiquettes de* $\mathbf{G}_f$ *est dans*$\mathcal{K}$, *soit pour tout graphe* $\mathbf{G}_f \in \mathrm{Irred}(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$ *l'ensemble des étiquettes de* $\mathbf{G}_f$ *est dans le complémentaire de* $\mathcal{K}$.

Le système ne doit être déterministe que sur les entrées correctement étiquetées de la forme $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$. On peut alors définir les classes de graphes reconnaissables avec information structurelle.

Définition 5.2. *Une famille* $\mathcal{F}$ *de graphes est* reconnaissable avec connaissance structurelle ι *si il existe un système de reconnaissance déterministe* $(\mathcal{R}, \mathcal{K}, \iota)$ *reconnaissant exactement* $\mathcal{F}$.

On pourra parler de familles $\iota-$reconnaissable. En particulier si ι dénote la taille, on parlera de familles TAILLE-reconnaissable.

5.2 Une Configuration Généralisée

On commencera par remarquer que la configuration par revêtement du lemme 0.31 ne peut plus être utile puisque l'on n'est pas assuré que les deux graphes en question partage la même valeur de connaissance structurelle ι, par exemple si ι encode la taille. Nous proposons donc une configuration généralisée(figure 5.1) qui nous permettra de conclure et d'aboutir à une caractérisation semblable à celle obtenue au chapitre précédent.

Nous commencerons par la proposition suivante qui établit comment la reconnaissabilité avec connaissance structurelle peut-être transmise par revêtements.

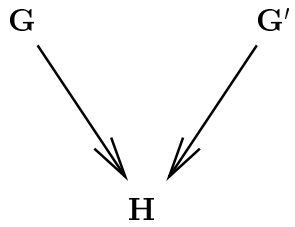


FIG. 5.1 – La configuration de revêtements généralisée.

Proposition 5.3. *Soit $(\mathcal{R}, \mathcal{K}, \iota)$ un système de reconnaissance de graphes. Soit $\mathbf{G}, \mathbf{G}'$ deux graphes tels que $\iota(\mathbf{G}) = \iota(\mathbf{G}')$ et qu'il existe $\mathbf{H}$ tel que $\mathbf{G}$ et $\mathbf{G}'$ soient tout deux revêtements de $\mathbf{H}$. Alors $\mathbf{G}$ est reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$ si et seulement si $\mathbf{G}'$ est reconnu.*

Démonstration. L'équivalence étant symétrique en $\mathbf{G}$ et $\mathbf{G}'$, on ne prouvera que la partie "seulement si".

Supposons que $\mathbf{G}$ soit reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$. Alors on peut appliquer la seconde partie du Lemme 4.4 à $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$ et $(\mathbf{H}, \Lambda_{\iota(\mathbf{G})})$. Par suite $(\mathbf{H}, \Lambda_{\iota(\mathbf{G})})$ est reconnu par $(\mathcal{R}, \mathcal{K})$. La première partie du lemme 4.4 et le fait que $\iota(\mathbf{G}) = \iota(\mathbf{G}')$ implique que $(\mathbf{G}', \Lambda_{\iota(\mathbf{G}')})$ est reconnu par $(\mathcal{R}, \mathcal{K})$. Donc, par définition, $\mathbf{G}'$ est reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$. □

Remarque 5.4. La valeur de $\iota(\mathbf{H})$ ci-dessus n'a aucune espèce d'importance, et on ne peut absolument pas dire que $\mathbf{H}$ est reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$, car $\iota(\mathbf{H})$ n'est pas nécessairement égal à $\iota(\mathbf{G})$ (par exemple, si ι encode la taille). $\mathbf{H}$ est juste un "intermédiaire de calcul".

Le résultat précédent est complètement généralisé comme suit. On définit sur les graphes étiquetés, une relation σ^ι en posant $\mathbf{G}\sigma^\iota\mathbf{G}'$ si

– $\iota(\mathbf{G}) = \iota(\mathbf{G}')$

– il existe un graphe $\mathbf{H}$ tel que $\mathbf{G}$ et $\mathbf{G}'$ soient revêtements de $\mathbf{H}$.

Soit $\sim^\iota$ la fermeture transitive de σ^ι. Une famille de graphes $\mathcal{F}$ sera fermée pour $\sim^\iota$ si pour tous graphes $\mathbf{G}$ et $\mathbf{G}'$ tel que $\mathbf{G} \sim^\iota \mathbf{G}'$, $\mathbf{G}$ appartient à $\mathcal{F}$ si et seulement si $\mathbf{G}'$ appartient à $\mathcal{F}$.

Par itération du lemme 5.3, il vient

Proposition 5.5. *Soit $(\mathcal{R}, \mathcal{K}, \iota)$ un système de reconnaissance. Soient deux graphes $\mathbf{G}$ et $\mathbf{G}'$ tels que $\mathbf{G} \sim^{\iota} \mathbf{G}'$. Alors $\mathbf{G}$ est reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$ si et seulement si $\mathbf{G}'$ est reconnu par $(\mathcal{R}, \mathcal{K}, \iota)$.*

Corollaire 5.6 (Condition nécessaire de reconnaissabilité). *Soit $\mathcal{F}$ une famille de graphes reconnaissable avec connaissance structurelle ι. Alors $\mathcal{F}$ est fermée pour $\sim^{\iota}$.*

En particulier, on utilisera $\sim^{\text{TAILLE}}$ lorsque ι encode la taille.

5.3 Réciproque

5.3.1 Savoir reconnaître des familles de graphes fermées pour $\sim^{\iota}$

Dans cette section, nous allons décrire un algorithme distribué pour reconnaître une famille $\mathcal{F}$ de graphes étiquetés, récursive et fermée pour $\sim^{\iota}$. Au cours de cette section, on supposera toujours que $\mathcal{F}$ est récursive et fermée pour $\sim^{\iota}$ sans le rappeler expressément.

On tente d'étendre la méthode utilisée au chapitre précédent, et on commencera par remarquer que, pour reconnaître le graphe sous-jacent, on dispose de deux "indices" :

– le graphe $\mathbf{H}_{\rho}$ calculé à l'aide de l'algorithme de Mazurkiewicz $\mathcal{M}$,
– la connaissance structurelle $\iota(G)$.

On définit l'ensemble $\mathcal{F}_{\iota}$ suivant. Soit

$$\mathcal{F}_{\iota} = \{(\mathbf{H}, \iota(\mathbf{G})) \mid \mathbf{G} \in \mathcal{F} \text{ et } \mathbf{G} \text{ est un revêtement de } \mathbf{H}\}.$$

D'après la définition de $\mathcal{F}_{\iota}$ et la Proposition 5.3, on obtient le lemme fondamental suivant qui lie l'appartenance à $\mathcal{F}$ et les deux "indices".

Lemme 5.7. *Soit $\mathcal{F}$ une famille de graphes fermée pour $\sim^{\iota}$. Soit $\mathbf{G}$ un graphe. Alors pour toute exécution ρ de $\mathcal{M}$:*

$$\mathbf{G} \in \mathcal{F} \text{ si et seulement si } (\mathbf{H}_{\rho}, \iota(\mathbf{G})) \in \mathcal{F}_{\iota}.$$

Maintenant, la propriété très intéressante c'est que l'on peut statuer sur la partie droite de l'équivalence sans connaître le graphe sous-jacent $\mathbf{G}$. En effet, $\mathcal{F}$ étant close pour $\sim^{\iota}$, il vient

Lemme 5.8. *Pour tout $(\mathbf{H}, \alpha) \in \mathcal{G}_{L\iota}$, on a*

1. *$(\mathbf{H}, \alpha) \in \mathcal{F}_{\iota}$ si et seulement si il existe $\mathbf{K} \in \mathcal{F}$ tel que $\mathbf{K}$ soit un revêtement de $\mathbf{H}$ et $\iota(\mathbf{K}) = \alpha$*
2. *$(\mathbf{H}, \alpha) \notin \mathcal{F}_{\iota}$ si et seulement si il existe $\mathbf{K} \notin \mathcal{F}$ tel que $\mathbf{K}$ soit un revêtement de $\mathbf{H}$ et $\iota(\mathbf{K}) = \alpha$*

Nous allons maintenant donner la description d'un semi-algorithme APPART qui sera exécuté *en interne* sur tous les nœuds. Son exécution sera remise à zéro à chaque fois que le nœud sera impliqué dans l'application d'une règle de $\mathcal{M}$. On se choisit un ordre d'énumération de l'ensemble des graphes étiquetés $\mathcal{G}_L$, on pose

---

Semi-Algorithme 1: APPART$(\mathbf{H}, \alpha)$

Données : un graphe $\mathbf{H} \in \mathcal{G}_L$,
une étiquette $\alpha \in L$

Sortie : étiquette r /* VRAI *ou* FAUX */

répéter
  /* *Énumérer tous les graphes étiquetés :* */
  $\mathbf{K} \in \mathcal{G}_L$
jusqu'à $\iota(\mathbf{K}) = \alpha$ *et* $\mathbf{K}$ *est un revêtement de* $\mathbf{H}$
retourner *le résultat de* "$\mathbf{K} \in \mathcal{F}$ *?*" /* $\mathcal{F}$ est récursive */

Nous ferons également des hypothèses d'équité classiques sur l'éxécution de cet algorithme. Nous supposerons que l'éxécution du semi-algorithme (interne) APPART est entrelacée avec celles des règles (externes) de Mazurkiewicz de telle manière que le **répéter jusqu'à** ne boucle pas indéfiniment. Ces hypothèses, bien que n'étant pas déraisonnables (voir [Lyn96] par exemple) ne donnent pas un système de réétiquetage "pur". Elles peuvent cependant être omises au moyen d'une amélioration que nous verrons dans la dernière partie, Remarque **??**.

On rappelle que l'éxécution de l'algorithme de Mazurkievicz sur un graphe étiqueté $\mathbf{G}$ utilise pour chaque sommet v le tuple $(\lambda(v), n(v), N(v), M(v))$. De plus, $\mathbf{H}_M$ représente le graphe que l'on peut reconstruire d'après les informations extraitres d'une boîte aux lettres M.

Le système de reconnaisance est alors le suivant :

- étiquettes : $(\iota(\mathbf{G}), \lambda, n, N, M, r)$,
- règles $\mathcal{R}_{\mathcal{F}}$:
  - une règle de réétiquetage initale qui convertit les étiquettes $(\iota(\mathbf{G}), \lambda)$ en $(\iota(\mathbf{G}), \lambda, 0, \emptyset, \emptyset, ?)$ sur chaque sommet ;
  - la règle (interne) APPART s'éxécute sur chaque nœud v comme un processus concurrent *équitable* avec en entrée $< \mathbf{H}_{M(v)}, \iota(\mathbf{G}) >$ et en sortie r ;
  - les règles de $\mathcal{M}$ s'éxécutent sur (n, N, M), remettant APPART à zéro et r à $?$ chaque fois qu'elles sont appliquées.
- condition finale $\mathcal{K}$: l'étiquette r est à VRAI sur tous les nœuds.

Finalement on obtient :

Proposition 5.9. *Soit* $\mathbf{G}$ *un graphe. Le système de réétiquetage* $\mathcal{R}_{\mathcal{F}}$ *est nœthérien sur* $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})$. *Sur tout étiquetage final,* $r(v) =$ VRAI *pour tout sommet* v *si et seulement si* $\mathbf{G} \in \mathcal{F}$.

Démonstration. Comme APPART est éxécuté équitablement, $\mathcal{M}$ se termine avec $\mathbf{H}_M = \mathbf{H}_\rho$ pour une certaine éxécution ρ. APPART sera alors éxécuté sur l'entrée $< \mathbf{H}_\rho, \iota(\mathbf{G}) >$,

la boucle **répéter jusqu'à** sera finie car au moins $\mathbf{G}$ vérifie les deux conditions d'arrêt. Et donc, d'après les lemmes 5.7 et 5.8, $r(v) = \underline{\text{VRAI}} \iff \mathbf{G} \in \mathcal{F}$, pour tout v. □

Par conséquent, $(\mathcal{R}_{\mathcal{F}}, \mathcal{K}, \iota)$ est un système de reconnaissance qui reconnaît $\mathcal{F}$.

5.3.2 Une condition nécessaire et suffisante de reconnaissabilité

Pour résumer la Prop. 5.9 et le Cor. 5.6,

Théorème 5.10. *Soit $\mathcal{F}$ une famille de graphes et ι une connaissance structurelle. Les assertions suivantes sont équivalentes.*

1. *$\mathcal{F}$ est localement reconnaissable avec connaissance structurelle ι.*
2. *$\mathcal{F}$ est fermée pour $\sim^{\iota}$ et $\mathcal{F}$ est un ensemble récursif.*

Un corollaire très intéressant est la stabilité pour les opérations booléennes.

Corollaire 5.11. *Les familles de graphes qui sont localement reconnaissable avec connaissance structurelle ι sont stables pour l'union, l'intersection et le passage au complément.*

Si les familles obtenues sont récursives, on a également stabilité pour l'union et l'intersection infinies.

Remarque 5.12. Excepté pour le passage au complément, qui découle directement de la définition (prenez le complément de la condition finale), la preuve de cette stabilité est plutôt délicate et technique à obtenir directement. Dans le cas fini, on éxécute "en parallèle" les différents systèmes de reconnaisance. Dans le cas infini, c'est assez peu réalisable directement.

Remarque 5.13. On notera bien que la terminaison est ici *implicite*.

5.4 Extension aux Réétiquetages de Rayon k

En utilisant $\mathcal{M}_k$ l'extension de l'algorithme de Mazurkiewicz aux réétiquetages sur des boules de rayon k, et en redéfinissant $\sim^{\iota}$ par $\sim^{\iota}_k$ pour utiliser des k-revêtements, on obtient exactement la même caractérisation.

Le corollaire intéressant de cette extension c'est qu'elle permet de répondre à la question de la comparaison de l'expressivité des règles de différents rayons. Cela montre qu'il existe une hiérarchie stricte pour ces règles, puisque l'on a, pour tout $k \neq k'$, des classes d'équivalence qui ne coïncident pas. Par exemple, soit $k \in \mathbb{N}$, la classe d'équivalence, pour $\sim^{\iota}_k$, de l'anneau A_{2k} contient tous les anneaux de taille supérieure à $2k$, alors que c'est un singleton dans le cas de $\sim^{\iota}_{k+1}$.

5.5 Quelques Preuves Simples d'Impossibilité

Nous allons appliquer la Proposition 5.3 dans le cas où ι représente la taille du graphe.

Proposition 5.14. *Les classes de graphes suivantes ne sont pas* TAILLE*-reconnaissables :*

1. *graphes bipartis,*
2. *graphes de connexité de sommet 1,*
3. *graphes ayant un isthme,*
4. *graphes Hamiltoniens.*

Démonstration. Nous appliquons la Proposition 5.3 aux graphes $G \sim^{\iota} G'$ (tout deux étant revêtements de H). Notons que, dans chaque cas, seul G est dans la famille à laquelle on s'intéresse. Et donc aucun système de reconnaissance ne peut reconnaître ces classes de graphes.
Figure 5.2. Un graphe est biparti si et seulement si il ne contient aucun cycle impair. Par conséquent G est biparti, mais G' ne l'est pas.
Figure 5.3. Le graphe G a deux sommets déconnectants et G' n'en a aucun.
Figure 5.4. Le graphe G a deux isthmes (arêtes déconnectantes) et G' n'en a aucun.
Figure 5.5. Le graphe G possède un chemin hamiltonien (chemin simple passant par tous les sommets), mais G' n'en a aucun.

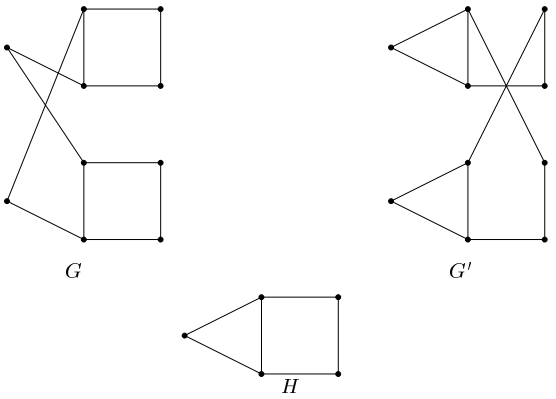


FIG. 5.2 – Les graphes bipartis ne sont pas localement reconnaissables en connaissant la taille.

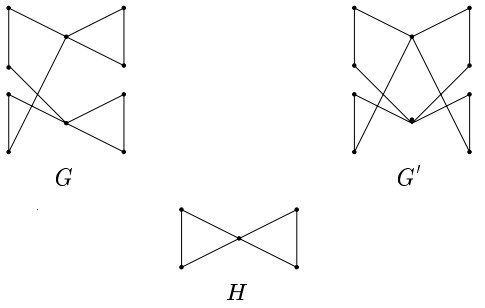


FIG. 5.3 – La présence de sommet déconnectant n'est pas localement reconnaissable en connaissant la taille.

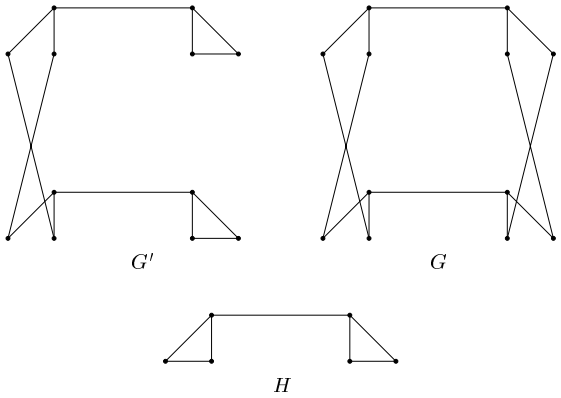


FIG. 5.4 – La présence d'isthme n'est pas localement reconnaissable en connaissant la taille.

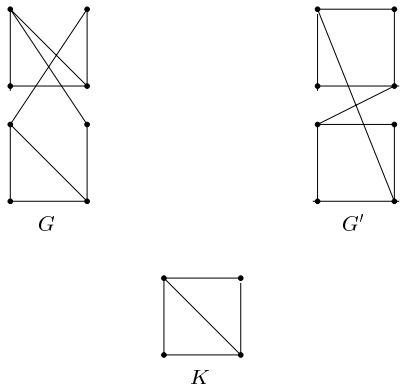


FIG. 5.5 – La présence de chemin hamiltonien n'est pas localement reconnaissable en connaissant la taille.

□

Chapitre 6

Applications

Ce chapitre étudie tout d'abord un exemple conséquent d'étude du cas où l'on connaît la taille du graphe. Afin de tenter de déterminer comment peuvent se placer les familles de graphes reconnaissables par rapport à d'autres familles de graphes classiques, nous étudierons les classes de graphes fermées par mineurs.

Ensuite pour apprécier le "grain" qu'apporte l'utilisation de différentes connaissance initiales, nous donnerons un aperçu de la hiérarchie de reconnaissabilité induite par la connaissance structurelle.

6.1 Classes Fermées par Mineurs

Rappelons certaines définitions concernant les opérations sur les arêtes. *Contracter une arête* reliant deux sommets u et v consiste à identifier u et v, en effaçant la boucle résultante et en remplaçant les arêtes multiples éventuelles par une arête simple. *Effacer une arête* signifie supprimer cette arête et le sommet isolé éventuellement créé. Un graphe G est un mineur d'un graphe H, noté $G \lhd H$, si il existe une suite d'opérations de contractions et effacements de H qui aboutit à G.

Une classe de graphes est dite *fermée par mineur* si elle contient tous les mineurs de ses membres. S'il s'agit d'une classe de graphes connexes, elle sera fermée par mineur si elle contient tous les mineurs connexes de ses membres.

Les classes de graphes connexes fermées par mineur peuvent être caractérisées par un ensemble finis de mineurs interdits, également appelés obstructions [RS88, Fel89]. Dans ce qui suit, nous prouverons que les classes de graphes fermées par mineurs ne sont, en général, pas reconnaissable par calculs locaux même en connaissant la taille. Les exceptions sont des familles simples comme les arbres, les anneaux et chaînes, ou encore, la famille des graphes toute entière.

Le résultat concernant la non-reconnaissabilité sans connaissance structurelle était déjà connu [CM94].

6.1.1 Configuration de revêtements

La première étape consiste à prouver ce qui suit :

Théorème 6.1. *Soit K et H deux graphes connexes et supposons que l'on ait l'une des conditions suivantes*

1. *K est planaire et H a au moins deux cycles,*
2. *H possède au moins trois cycles.*

Alors il existe un entier p tel que pour tout $q \geq p$, il existe un revêtement connexe G_1 à q feuillets de H qui contient K comme mineur.

La preuve nécessitera les propositions suivantes qui sont des améliorations de propositions analogues issues de [CM94]. Soient n et i deux entiers strictement positifs $mod_n(i)$ est l'unique entier appartenant à $[0, n-1]$ tel que $i = kn + mod_n(i)$, pour un certain entier positif k.

Proposition 6.2. *Pour tout graphe planaire K, pour tout graphe connexe H ayant au moins deux cycles, il existe un entier p tel que pour tout $q \geq p$, il existe un revêtement connexe G_1 à q feuillets de H qui contient K comme mineur.*

Démonstration. Un tore est un graphe de la forme $T(m,n)$ dont l'ensemble des sommets est

$$V(m,n) = [0, m-1] \times [0, n-1],$$

et l'ensemble d'arêtes

$$\{(x_1,y_1),(x_2,y_2)\} \quad \text{où} \quad \begin{array}{llll} \text{soit} & x_1 = x_2, & \text{et} & y_2 = \mathrm{mod}_n(y_1+1), \\ \text{soit} & y_1 = y_2, & \text{et} & x_2 = \mathrm{mod}_m(x_1+1). \end{array}$$

La figure 6.1 représente le tore $T(4,5)$.

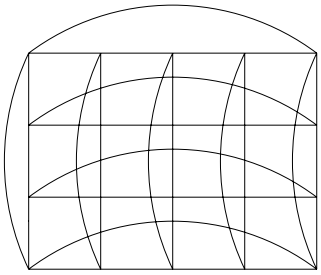


FIG. 6.1 – Le tore $T(4,5)$

K étant planaire, il existe m et n tel que $K \trianglelefteq T(m,n)$. Ceci provient du fait que tout graphe planaire est un mineur d'une grille carré suffisamment grande (ou de manière équivalente, tout graphe planaire peut être dessiné dans le plan avec des coordonnées entières), et la grille $m \times n$ est un sous-graphe de $T(m,n)$. Plus précisément, un graphe planaire avec p sommets peut être plongé dans une grille $2p-4 \times p-2$, voir [DFPP88, vL90].

Étant donné que H possède au moins deux cycles, il existe deux arêtes e et e' de H telles que $H \setminus \{e,e'\}$ est connexe. Soit $e = \{x,y\}$ et $e' = \{z,u\}$. On peut avoir $y = z$

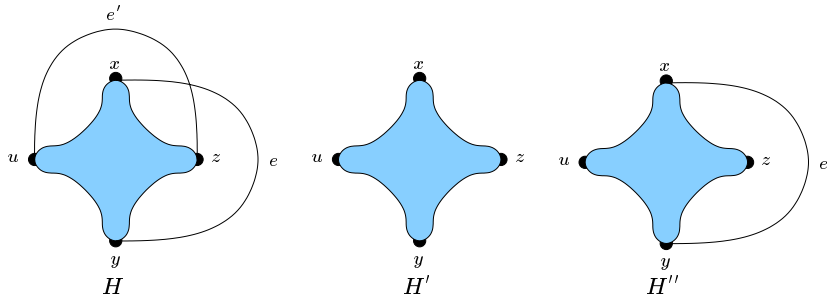


FIG. 6.2 – H, H' et H''

mais pas $e = e'$. Soit H' le graphe obtenu à partir de H en effaçant e et e'; soit H'' celui obtenu en effaçant e'. Se reporter à la figure 6.2.

Pour tout $q \geq mn$, nous allons construire un graphe connexe G_1 tel que $K \trianglelefteq G_1$ et que G_1 soit un revêtement à q feuillets de H. Il s'agit de la réunion de mn copies disjointes de H' reliées en forme de tore et d'une "queue" de $q - mn$ copies disjointes de H''. Se reporter à la Figure 6.3 pour un exemple de cette construction avec $m = 5$, $n = 4$, et $q = 23$.

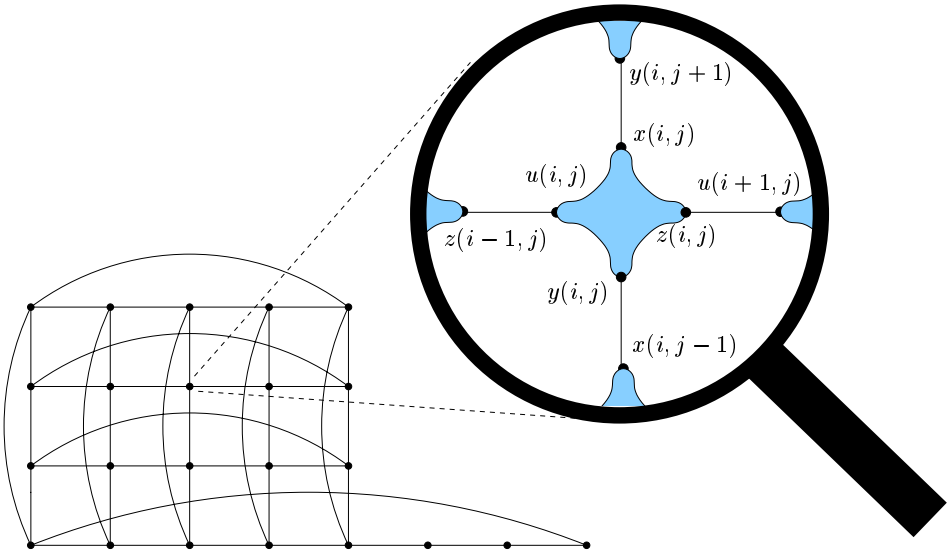


FIG. 6.3 – G_1, vu à la loupe

Formellement, pour tout $(i, j) \in V(m, n)$, on note $H'(i, j)$ une copie isomorphe de H' telle que l'ensemble des éléments soient deux à deux disjoints

$$V(H'(i, j)) \cap V(H'(i', j')) = \emptyset \quad \text{si} \quad (i, j) \neq (i', j').$$

Pour tout $0 \leq l \leq (q - p - 1)$, on note $H''(l)$ une copie isomorphe de H'' telle que

$$V(H'(i, j)) \cap V(H''(l)) = \emptyset,$$

$$V(H''(l)) \cap V(H''(l')) = \emptyset \quad \text{si} \quad l \neq l'.$$

On notera $w(i,j)$ la copie correspondante dans $H'(i,j)$ d'un sommet w de H'. On notera $w(l)$ la copie dans $H''(l)$ d'un sommet w de H''.

Maintenant G_1 consiste en :

– l'union des graphes $H'(i,j)$ et $H''(l)$,

– des arêtes

Partie tore :

– $\{x(i,j), y(i, \mathrm{mod}_n(j+1))\}\ 1 \leq i \leq m-1,\ 0 \leq j \leq n-1$,

– $\{z(i,j), u(\mathrm{mod}_m(i+1), j)\}\ 0 \leq i \leq m-1,\ 0 \leq j \leq n-1, (i,j) \neq (m-1, 0)$

Partie queue :

– $\{z(m-1,0), u(0)\}$,

– $\{z(i), u(i+1)\} \qquad 0 \leq i \leq q-mn-2$,

– $\{z(q-mn-1), u(0,0)\}$.

Il est assez clair que, en contractant simultanément toutes les arêtes des sous-graphes $H'(i,j)$ et $H''(l)$ de G_1, puis en contractant toutes les arêtes de la "queue", on obtient $T(m,n)$. Par conséquent $K \trianglelefteq T(m,n) \trianglelefteq G_1$. L'application qui affecte un sommet $w(i,j)$ de $H'(i,j)$ ou un sommet $z(l)$ de $H''(l)$ à w de H est un revêtement : G_1 est un revêtement à q feuillets de H. □

L'autre cas du Th. 6.1 est traité de manière similaire,

Proposition 6.3. *Pour tout graphe K, Pour tout graphe connexe H ayant au moins trois cycles, il existe un entier p tel que pour tout $q \geq p$ il existe un revêtement connexe G_1 à q feuillets de H qui contient K comme mineur.*

Démonstration. La preuve est une simple extension de la précédente. Au lieu de plonger K comme mineur dans un tore, on le plonge comme mineur dans un "parallélépipède torique", ou tore tridimensionnel $P(l,m,n)$ avec comme ensemble de sommets $V(l,m,n) = [0, l-1] \times [0, m-1] \times [0, n-1]$

On prend $p = l \times m \times n$, et soit $q \geq p$. On sélectionne trois arêtes $e = \{x,y\}$, $e' = \{z,u\}$, et $e'' = \{v,w\}$ telles que le graphe H' obtenu à partir de H en effaçant e, e' et e'' soit connexe. Soit H'' obtenu à partir de H en effaçant e. On construit K comme étant l'union de copies disjointes $H'(i,j,k)$ de H' pour $(i,j,k) \in V(l,m,n)$ et de copies disjointes $H''(l)$ de H'' pour $0 \leq l \leq q-p-1$ augmentée des arêtes ad hoc. On obtient $K \trianglelefteq P(l,m,n) \trianglelefteq G_1$ et G_1 est un revêtement de H à q feuillets. □

6.1.2 Application à la reconnaissance en connaissant la taille

Il est montré dans [CM94] que l'on ne peut déterminer par calculs locaux si un graphe contient ou ne contient pas un graphe donné comme mineur. L'idée de la preuve était de choisir un graphe H tel que K ne soit pas mineur de H, et ensuite de construire un revêtement G_1 de H qui contient K comme mineur.

Pour la reconnaissance avec connaissance de la taille, on ne peut reprendre cette construction car G_1 et H ont des tailles différentes. On introduit alors une autre construction (voir Fig. 6.4) et nous montrons que l'on ne peut, non plus, en général, décider par

calculs locaux si un graphe donné est un mineur d'un autre graphe, même en ayant connaissance de la taille de ce dernier. Par conséquent, à part des exceptions simples, les classes de graphes fermées par mineurs ne sont pas reconnaissables en connaissant la taille.

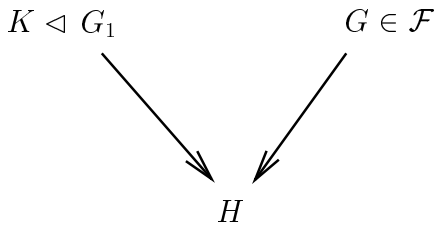


FIG. 6.4 – Configuration des revêtements pour la preuve du Th. 6.4

Notre résultat principal est le suivant :

Théorème 6.4. *Soit $\mathcal{F}$ une classe de graphes connexes fermée par mineur qui ne contient pas tous les graphes. Si l'une des conditions suivantes est satisfaite,*

1. *$\mathcal{F}$ contient tous les graphes planaires,*
2. *Il existe un graphe K planaire de taille p qui n'est pas dans $\mathcal{F}$ et il existe un graphe connexe $H = (V, E)$ tel que*
   - *H a un revêtement $G \in \mathcal{F}$ avec au moins $2p^2$ feuillets,*
   - *H possède au moins deux cycles.*

alors $\mathcal{F}$ est pas localement reconnaissable avec connaissance de la taille.

Démonstration. Supposons que $\mathcal{F}$ soit reconnaissable en connaissant la taille.

1. Soit $\mathbf{K}$ n'appartenant pas à $\mathcal{F}$. Si $\mathcal{F}$ contient tous les graphes planaires, alors cette classe contient le "trèfle à 3 feuilles" S_3 et une sous famille infinie de ses revêtements planaires $\{S^q \mid q \in \mathbb{N}\}$. Le graphe S^q est le revêtement à q feuillets de S^3 comme défini par la Figure 6.5. Si on applique la Proposition 6.3 avec $H = S_3$, on obtient que G_1 est un revêtement de S_3 tel que $K \trianglelefteq G_1$. Si q est le nombre de feuillets du revêtement G_1 alors on obtient que $G_1 \sim^{\text{TAILLE}} S^q$. Par conséquent $G_1 \in \mathcal{F}$ d'après le Cor. 5.6. $\mathcal{F}$ étant fermée par mineurs, on abouti à une contradiction avec le fait que $K \notin \mathcal{F}$.
2. Une configuration et un raisonnement identique aux précédents peut également être utilisés, en appliquant la Proposition 6.2 à la place.

□

Remarque 6.5. Les hypothèses du théorème précédent sont minimales dans le sens suivant : si on autorise H dans le cas 2 à avoir au plus un cycle alors $\mathcal{F}$ peut être reconnu en connaissant la taille. Considérons par exemple la famille des arbres, ou la famille des anneaux et chaînes. Voir la section 4.3.2 pour un preuve complète de la reconnaissabilité de ces familles (même en ne connaissant pas la taille d'ailleurs).

Notons également qu'il doit y avoir une borne inférieure sur le nombre de feuillets de G, car, si ce nombre n'est pas assez grand, alors la classe peut être reconnaissable

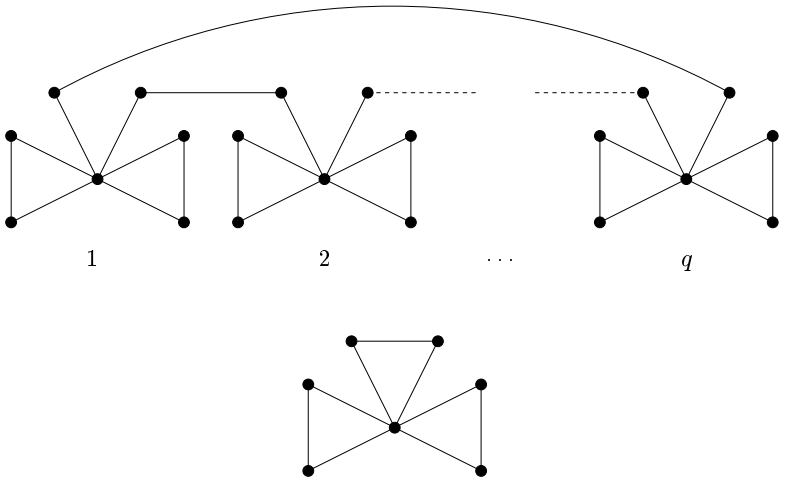


FIG. 6.5 – S_3 et un de ses revêtements planaires S^q

avec connaissance de la taille. La famille des graphes avec au plus n sommets (pour un entier n donné) est fermée par mineur et est, évidemment, reconnaissable si l'on connaît la taille.

6.2 Hiérarchie des Connaissances Structurelles

L'intérêt de la reconnaissance avec connaissance structurelle est en particulier de dégager quelles sont les informations minimales à avoir pour pouvoir effectuer certaines reconnaissances. Il est donc intéressant de comparer ce qu'apportent les différentes connaissances possibles. On notera tout d'abord que l'on peut, pour n'importe quelle classe $\mathcal{F}$, trouver une connaissance structurelle ι qui permet de rendre $\mathcal{F}$ reconnaissable, il suffit de prendre la fonction caractéristique de cette famille, pour ne pas parler de l'information topologique totale. Il importe donc de se limiter à une certaine catégorie de connaissance structurelle. Nous avons choisi d'étudier des connaissances "métriques" classiques : borne sur le diamètre, borne sur la taille, bonne borne sur la taille, le diamètre, la taille. Nous précisons quelles sont les familles reconnaissables pour chacune de ces informations structurelles, et comparons les différentes puissances d'expression de celles-ci.

Remarque 6.6. On commencera par remarquer que si $\mathcal{F}$ est reconnaissable en ayant la taille, alors $\mathcal{F}$ est reconnaissable avec une bonne borne, et par conséquent en ayant une borne sur le diamètre, et par conséquent en n'ayant aucune information structurelle.

L'étude de la reconnaissance sans information structurelle a été réalisée section 4.3.1.

6.2.1 Borne supérieure sur la taille et le diamètre

On définit une famille reconnaissable avec borne supérieure comme une famille reconnaissable par un même système de réétiquetage quelque soit la fonction bornant

choisie. Il est alors équivalent de dire qu'une famille reconnaissable avec borne supérieure est fermée pour toute relation $\sim^b$. Par conséquent si l'on définit $\sim^{\text{BORNE}} = \bigcup_b \sim^b$, où l'union est prise sur toutes les fonctions b bornant supérieurement la taille de $\mathbf{G}$, on a

Proposition 6.7. *Soit $\mathcal{F}$ une famille de graphes. Alors les propositions suivantes sont équivalentes :*

6.7.i $\mathcal{F}$ est reconnaissable avec borne supérieure,

6.7.ii $\mathcal{F}$ est fermée pour $\sim^{\text{BORNE}}$,

6.7.iii $\mathcal{F}$ est fermé pour $\sim^{\varepsilon}$,

6.7.iv $\mathcal{F}$ est $\varepsilon-$reconnaissable.

Démonstration. Si $\mathcal{F}$ est reconnaissable avec une borne supérieure alors, pour toute fonction bornant la taille b, $\mathcal{F}$ est $b-$reconnaissable et donc $\mathcal{F}$ est fermé pour $\sim^b$. Par conséquent, $\mathcal{F}$ est fermée pour $\sim^{\text{BORNE}}$.

Soient deux graphes tels que $\mathbf{G} \sim^{\varepsilon} \mathbf{G}'$, alors transitivité sur σ^{ι}, on passe par un nombre *fini* de revêtements de $\mathbf{G}$ à $\mathbf{G}'$ et il existe b une fonction bornant la taille (de tous ces revêtements en particulier) tel que $\mathbf{G} \sim^b \mathbf{G}'$. Par conséquent la fermeture pour $\sim^{\text{BORNE}}$ implique la fermeture pour $\sim^{\varepsilon}$.

Les implications restantes sont immédiates. □

Remarque 6.8. En d'autres termes, avoir une borne quelconque sur le nombre de sommets est la même chose que de n'avoir aucune information dans le contexte de la reconnaissance. Ce résultat assez surprenant est similaire à celui obtenu dans [YK96b] (Prop. 18). Ceci sera assez différent dans le contexte de l'élection et de la détection de la terminaison (parties suivantes)

Le précédent raisonnement s'applique de manière identique à la reconnaissance en ayant une borne sur le diamètre.

6.2.2 Bonne borne sur la taille

Le paragraphe précédent indique que l'on ne gagne rien à connaître une borne sur la taille qui peut être aussi élevée que possible. Que se passe-t-il si l'on contraint la borne à n'être que relativement mauvaise. On rappelle qu'une bonne borne est une borne supérieure de la taille b telle que, pour tout graphe G, $|V(G)| \leq b(G) < 2|V(G)|$. On définit, comme précédemment, une famille de graphes reconnaissable en ayant une bonne borne sur la taille comme étant une famille reconnaissable par un même système de réétiquetage quelque soit la bonne borne choisie. Il est alors équivalent de dire qu'une famille reconnaissable avec bonne borne est fermée pour toute relation $\sim^b$, pour toute bonne borne b.

Proposition 6.9. *Soit $\mathbf{G}$ un graphe minimal. Le singleton $\{\mathbf{G}\}$ est reconnaissable en connaissant une bonne borne.*

Démonstration. Étant donné qu'un revêtement propre a toujours au moins deux feuillets, la classe d'équivalence $[\mathbf{G}]_{\sim \texttt{BBorne}}$ est un singleton pour tout graphe minimal $\mathbf{G}$. □

En corollaire, la classe des graphes minimaux est reconnaissable en connaissant une bonne borne.

6.2.3 Connaissance de la taille ou du diamètre

On se reportera Sect. 5.5 et 6.1 pour des exemples approfondis de ce qui est ou n'est pas reconnaissable en connaissant la taille.

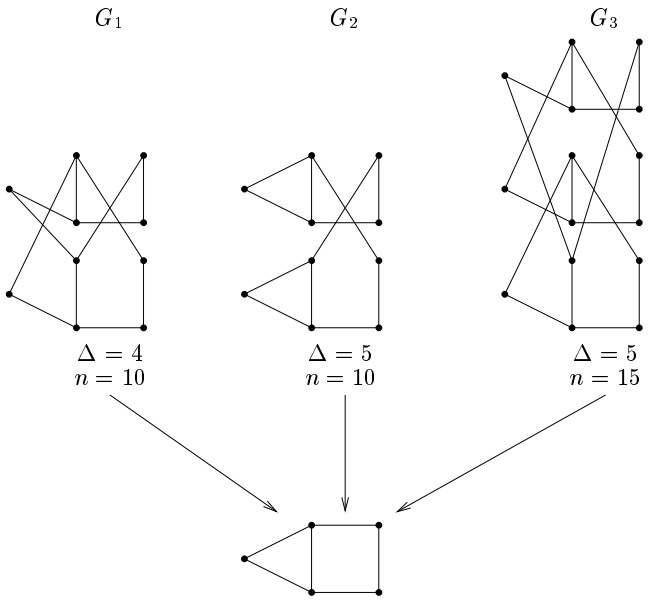


FIG. 6.6 – Connaissance de la taille et du diamètre sont incomparables

Intéressons nous maintenant à la question de savoir si il y a un lien entre les familles reconnaissables avec le diamètre et celles reconnaissables avec la taille, il suffit de regarder la Fig. 6.6. On y constate que

– $G_1 \sim^{\text{TAILLE}} G_2$ mais $G_1 \not\sim^{\Delta} G_2$.
– $G_2 \sim^{\Delta} G_3$ mais $G_2 \not\sim^{\text{TAILLE}} G_3$.

Remarque 6.10. On notera également en utilisant le Lemme 0.27 que si $\mathbf{G}$ et $\mathbf{G}'$ sont tout deux revêtement d'un graphe $\mathbf{H}$ et ont le même nombre de sommets, alors ils ont le même nombre d'arêtes. C'est à dire que connaître le nombre de sommets et connaître le nombre d'arêtes revient au même, puisque les relations d'équivalence associées sont identiques.

6.2.4 Tableau récapitulatif

Nous allons résumer les résultats obtenus précédemment dans le tableau suivant où OUI et NON correspondra à la reconnaissabilité de la famille considérée en ayant une connaissance structurelle donnée. Les numéros entre parenthèses correspondent

aux preuves ci-dessous. Les $--$ de part et d'autre de la diagonale sont complétée d'après la remarque 6.6

Proposition 6.11. *On a*

	Aucune connaissance =Borne sur la taille =Borne sur le diamètre	*Bonne borne*	*Nombre de sommets = Nombre d'arêtes*	*Topologie*
Arbres	OUI*(1)*	–	–	–
Graphes minimaux	NON*(2)*	OUI*(3)*	–	–
Graphes de taille paire	–	NON*(4)*	OUI*(5)*	–
Graphes planaires	–	–	NON*(6)*	OUI

Démonstration.

1. Prop. 4.9
2. La famille des graphes minimaux n'est pas close par revêtements.
3. Prop. 6.9
4. Soit n un entier impair. On a $A_{2n} \sim^{\text{BBORNE}} A_{3n}$ car ces deux anneaux sont tout deux revêtement de A_n et ont comme bonne borne commune $3n$ par exemple.
5. Trivial.
6. Th. 6.4

□

Ce tableau montre comment pour des connaissances structurelles relativement simples, on obtient des notions de reconnaissance strictement différentes. On notera que information zéro, bonne borne, taille et topologie forme une hiérarchie stricte mais que la connaissance du diamètre est incomparable avec celle de la taille ou d'une bonne borne. Pour ce dernier point on remarquera que, pour tout $n \in \mathbb{N}^*$, $A_{2n} \sim^{\text{BBORNE}} A_{3n}$ et que $\Delta(A_{2n}) = n \neq \lceil \frac{3n}{2} \rceil = \Delta(A_{3n})$.

6.3 Conclusion

Cette partie nous permet de préciser les différentes puissances d'expression en fonction de la connaissance structurelle : les familles reconnaissables sont des réunions d'ensembles "atomiques" de graphes.

L'aspect atomique est intrinsèquement lié à l'aspect distribué des calculs. En particulier, la tentative de définir, par des concepts issus de l'algorithmique distribuée, une "bonne" notion de familles de graphes reconnaissable (avec des propriétés similaires à celles des familles de mots ou d'arbres reconnaissables et des automates associés) n'a pas abouti puisqu'en fait ce qu'on manipule au moyen des systèmes de reconnaissance, ce sont les ensembles atomiques (les classes d'équivalences) et non pas les graphes directement, puisque certains graphes sont en fait "indistinguables". Le problème pour

tenter de définir des “automates de graphes” étant en particulier que, au contraire des mots et des arbres, on ne sait pas par où commencer et comment parcourir, l’idée d’utiliser une approche distribuée (on commence partout et on parcourt tout en parallèle en terminant de manière implicite) apparaissait assez riche de perspectives. Ce ne sera cependant pas sous la forme présentée ici.

Troisième partie

Tâches Réalisables par Calculs Locaux : Élection et Fonctions Scalaires

Chapitre 7

Élection et Cartographie Partielle

Dans ce chapitre nous allons étudier le problème de l'élection dans un réseau. Ce problème, que d'aucun considère comme un véritable paradigme de l'algorithmique distribuée, sera étudié et résolu en mettant en œuvre un algorithme de cartographie partielle du réseau sous-jacent. Nous verrons ensuite que cet algorithme peut également être utilisé pour effectuer, de manière générale, le calcul d'une fonction scalaire.

Ce chapitre a été effectué en collaboration avec Y. Métivier et a été publié sous forme de résumé dans [GM02a].

7.1 Élection

L'élection dans un réseau consiste à distinguer par calcul distribué, parmi l'ensemble des nœuds un unique nœud. Celui-ci pourra ensuite effectuer une opération n'autorisant qu'un seul accès exclusif, ou encore centraliser et coordonner un comportement général du réseau, par exemple pour un redémarrage après une panne. Un grand nombre de problèmes se réduise souvent à "élire" un coordinateur pour ensuite faire effectuer au réseau, sous sa coordination, les opérations idoines.

Ce problème a été étudié très largement, et un nombre très important d'algorithmes est disponible dans la littérature. Les premiers résultats théoriques où il est apparu que ce problème n'avait parfois pas de solution sont ceux d'Angluin. Nous allons rappeler ces résultats ainsi que ceux qui ont suivi.

7.1.1 Définition

On va utiliser deux étiquettes particulières $\boxed{\text{ÉLU}}$ et $\boxed{\text{PAS ÉLU}}$. Celles-ci seront des étiquettes terminales, *i.e.*, au cours de toute chaîne de réétiquetages, si l'une de ces étiquettes apparaît sur un sommet u, alors ce sommet conservera cette étiquette.

Définition 7.1. *Soit $\mathbf{G}$ un graphe. On dira qu'un système de réétiquetage de graphes $\mathcal{R}$ élit sur $\mathbf{G}$ si pour toute chaîne de réétiquetage, on a :*

- *la chaîne est finie*
- *l'étiquetage final est de la forme $(\mathbf{G}, \lambda)$ avec*

- *il existe un unique sommet u_0 tel que $\lambda(u_0) =$* ÉLU,
- *pour tout $u \neq u_0$, $\lambda(u) =$* PAS ÉLU.

– *les étiquettes* ÉLU *et* PAS ÉLU *sont terminales : pour tout sommet v, si à un moment donné, l'étiquette de v est* ÉLU *(resp.* PAS ÉLU*) alors, pour tous les réétiquetages suivants, l'étiquette de v demeure* ÉLU *(resp.* PAS ÉLU*).*

Définition 7.2. *Soit $\mathcal{F}$ une famille de graphe. On dira qu'un système de réétiquetage de graphes $\mathcal{R}$ est un algorithme d'élection* universel *pour $\mathcal{F}$ si, pour tout graphe $\mathbf{G}$ de $\mathcal{F}$, $\mathcal{R}$ élit sur $\mathbf{G}$.*

L'étude de l'élection effective, c'est-à-dire en fonction de la connaissance structurelle, sera réalisée dans la section 7.4.

Le système de réétiquetage suivant, ELECT_ARBRE, est un algorithme d'élection universel pour la famille des arbres. Il consiste en deux règles, l'une "élague" l'arbre, c'est-à-dire que chaque feuille peut devenir PAS ÉLU ; et l'autre qui élit le dernier sommet étiqueté ε. La correction de l'algorithme repose sur le fait, qu'à tout moment, le sous-graphe des sommets étiquetés ε est connexe.

ELECT_ARBRE1 : **Élagage**

Condition préalable :

– $\lambda(v_0) = \varepsilon$

– $\exists! v \in B(v_0), \lambda(v) = \varepsilon$

Réétiquetage :

– $\lambda'(v_0) :=$ PAS ÉLU

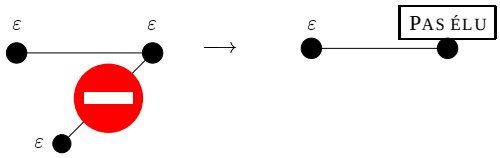


FIG. 7.1 – "Élagage"

ELECT_ARBRE2 : **Élection finale**

Condition préalable :

– $\lambda(v_0) = \varepsilon$

– $\forall v \in B(v_0), \lambda(v) \neq \varepsilon$

Réétiquetage :

– $\lambda'(v_0) :=$ ÉLU.

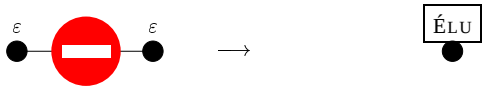


FIG. 7.2 – Élection du nœud restant

Ce système est très similaire à REC_ARBRES et fonctionne correctement pour des raisons similaires.

Le schéma suivant présente un exemple d'éxécution du système ELECT_ARBRE.

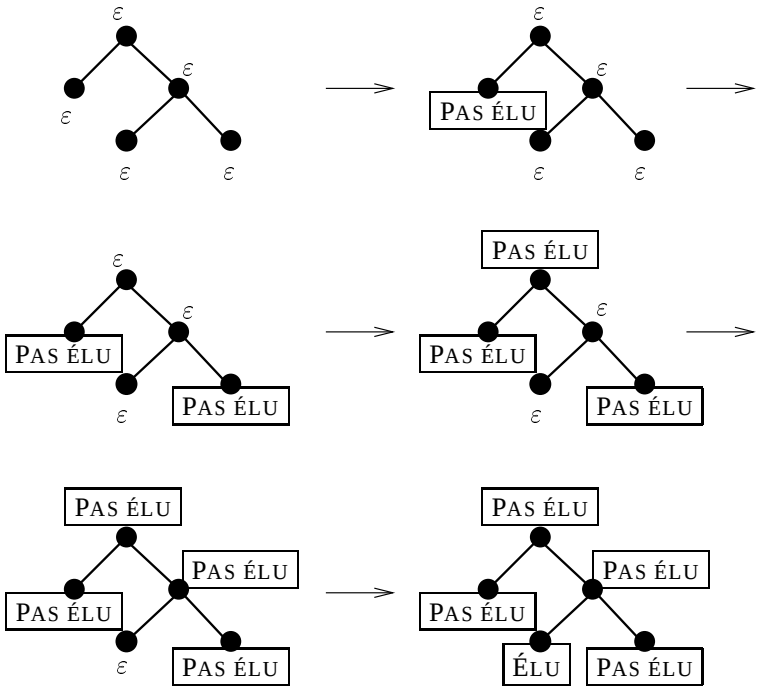


FIG. 7.3 – Une élection

Il faut noter que l'on connaît un certain nombre d'autres résultats concernant l'élection. Outre pour les arbres, il existe des algorithmes d'élection pour des réseaux anonymes tels que les anneaux orientés premiers [Maz88], les graphes T-premiers [GMMW01], les polyominos, les graphes complets...

7.1.2 La condition nécessaire d'Angluin

Dans [Ang80], Angluin a présenté une preuve d'impossibilité de l'existence d'algorithme d'élection pour certains types de réseaux. C'est à cette occasion que le concept de revêtement fut appliqué à l'algorithmique distribuée pour la première fois.

Théorème 7.3 (Angluin, 1980). *Soit* $\mathbf{G}$ *un graphe non minimal pour la relation revêtement. Alors il n'existe pas d'algorithme d'élection pour* $\mathbf{G}$.

Démonstration. Par lemme 0.31, dit de relèvement. En effet, supposons que $\mathbf{G}$ ne soit pas minimal. Alors il existe $\mathbf{H}$ tel que $\mathbf{G}$ soit un revêtement de $\mathbf{H}$ à q feuillets, avec $q \geq 2$. Soit $\mathcal{R}$ un système de réétiquetage de graphes pour $\mathbf{G}$. Soit $\mathbf{H}'$ une forme $\mathcal{R}$-irréductible de $\mathbf{H}$. D'après le lemme de relèvement, il existe une exécution de $\mathcal{R}$ sur $\mathbf{G}$ qui mène à $\mathbf{G}'$ où $\mathbf{G}'$ est un revêtement à q feuillets de $\mathbf{H}'$. Cela signifie que si l'étiquette ÉLU apparaît sur $\mathbf{G}'$ alors elle apparaît au minimum q fois. $\mathcal{R}$ ne peut donc pas être un système d'élection pour le graphe $\mathbf{G}$. □

C'est en fait une des conséquences les plus connues du Lemme de Relèvement, même si à strictement parler l'énoncé de [Ang80] (Th. 4.5) était un peu moins précis, la démonstration par contre était identique. L'énoncé était si deux graphes sont tels que $\mathbf{G}$ est un revêtement propre de $\mathbf{H}$, alors il n'existe pas d'algorithme d'élection (établissant un centre dans la terminologie d'Angluin) pour $\mathbf{G}$ et $\mathbf{H}$ à la fois.

Pendant longtemps la réciproque de ce théorème, c'est-à-dire la question de savoir si l'on pouvait élire dans tous les graphes minimaux est restée non résolue.

7.1.3 Élection avec l'algorithme de Mazurkiewicz

L'algorithme d'énumération de Mazurkiewicz se révèle être en fait un algorithme d'élection dès que l'on connaît la taille du réseau.

Théorème 7.4 (Mazurkiewicz, 1997). *Soit $\mathbf{G}$ un graphe minimal. Il existe un algorithme qui élit sur $\mathbf{G}$.*

Démonstration. Soit $t \in \mathbb{N}$. Notons $\mathcal{M}_t$ le système de réétiquetage de graphes obtenu en ajoutant à l'algorithme de Mazurkiewicz les deux règles suivantes :

$\mathcal{M}_t-$ELECT : **Règle d'élection**
Condition préalable :
– Il existe N tel que $(t, N) \in M(v_0)$.
– $n(v_0) = 1$,
Réétiquetage :
– v_0 prend l'étiquette ÉLU.
– Les voisins $v \in B(v_0) \backslash \{v_0\}$ prennent l'étiquette PAS ÉLU.

$\mathcal{M}_t-$TERM : **Règle de diffusion terminale**
Condition préalable :
– v_0 n'est ni étiqueté par ÉLU ni par PAS ÉLU
– Il existe $v \in B(v_0)$, étiqueté par PAS ÉLU.
Réétiquetage :
– Tous les sommets $v \in B(v_0)$ prennent l'étiquette PAS ÉLU.

Soit $t = |V(\mathbf{G})|$. Alors $\mathcal{M}_t$ élit sur $\mathbf{G}$. En effet, d'après le Lemme 3.6, $\mathbf{G}$ étant minimal, quand le numéro t apparaît, cela signifie que $\mathcal{M}$ a effectué une énumération des sommets du graphe, la règle de changement de numéro n'est plus applicable, et il y a un et un seul sommet numéroté 1 dans le graphe. La règle $\mathcal{M}_t-$TERM achève d'étiqueter tous les autres sommets par PAS ÉLU. □

Remarque 7.5. Dans $\mathcal{M}_t$, on peut également choisir d'élire un autre numéro. Classiquement on élit le sommet numéro t, mais on peut prendre n'importe quel entier entre 1 et t. Cependant, si l'on ne choisit pas d'élire le numéro t, la situation est assez intéressante pour qu'elle soit explicitement décrite.

D'après les règles de $\mathcal{M}_t$, dès qu'un sommet prend un numéro supérieur ou égal à deux, il "sait" qu'il ne pourra être élu et que son étiquette finale sera PAS ÉLU. Ce sommet ne passe pas immédiatement en mode terminal uniquement pour permettre aux sommets encore numéroté 1 de se départager.

Savoir si l'on pourrait utiliser cette remarque pour améliorer la complexité de l'algorithme d'élection n'est pas évident. Le fait que celui-ci soit essentiellement autostabilisant, *dans la même complexité*, laisse supposer qu'exécuter l'algorithme sur quelques pas puis le figer et le relancer sur le graphe étiqueté par les premiers numéros calculés semble, a priori, n'apporter rien de plus.

7.1.4 Algorithmes Universels d'Election

L'algorithme $\mathcal{M}_t$ est universel pour la familles des graphes de taille t et l'existence d'un algorithme universel, élisant sur tous les réseaux minimaux reste une question ouverte à ce stade.

Un théorème d'impossibilité

Un arrêt fut mis à la recherche de cet éventuel algorithme universel en 1997 suite à [MMW97] qui prouvait qu'en particulier il n'y avait pas d'algorithme universel d'élection pour les anneaux minimaux (ceux de taille première), et donc *a fortiori* pour l'ensemble des graphes minimaux. A proprement parler, encore, l'énoncé ci-dessous n'est pas exactement dans [MMW97], puisque l'article traitait de la détection de la terminaison des algorithmes distribués. Cependant les idées et la démarche de la preuve sont identiques.

Théorème 7.6 (Métivier Muscholl Wacrenier, 1997). *Soit $\mathcal{F}$ une famille de graphes. Si il existe un graphe $\mathbf{G}$ de $\mathcal{F}$ admettant des quasi-revêtements d'étendue arbitrairement grande dans $\mathcal{F}$, alors il n'existe pas d'algorithme universel d'élection pour $\mathcal{F}$.*

Démonstration. Cette fois-ci, il s'agit d'une conséquence du Lemme 0.35 de Quasi-Relèvement. Il est à noter que la preuve de [MMW97] était différente et n'utilisait pas ce lemme. L'un des avantages de la preuve qui suit est qu'elle est constructive.

On pourra supposer que $\mathcal{F}$ est une famille de graphes minimaux sinon le résultat est immédiat d'après le théorème d'Angluin. D'après le lemme 0.45, l'hypothèse

signifie en particulier que $\mathbf{G}$ admet dans $\mathcal{F}$ des quasi-revêtements de rayon arbitrairement grand. Supposons qu'il existe un algorithme d'élection $\mathcal{R}$ pour $\mathcal{F}$. Soit n la longueur d'une chaîne de réétiquetages aboutissant à une élection sur $\mathbf{G}$. Notons $\mathbf{G}'$ le graphe final. Par hypothèse, il existe un quasi-revêtement $\mathbf{K}$ de $\mathbf{G}$ de rayon supérieur à $2 * n + 2|V(\mathbf{G})|$. $\mathbf{K}$ étant minimal, il ne peut pas être revêtement de $\mathbf{G}$, et donc $\mathbf{K}$ est nécessairement un quasi-revêtement strict.

En effectuant n applications du Lemme de Quasi-Relèvement, il vient qu'il existe une chaîne de réétiquetage de $\mathbf{K}$ menant au graphe étiqueté $\mathbf{K}'$, $\mathbf{K}'$ étant un quasi-revêtement strict de $\mathbf{G}'$ de rayon $2|V(\mathbf{G}')|$. D'après le lemme 0.44, ce quasi-revêtement a au moins 2 feuillets. Par conséquent, l'étiquette $\boxed{\text{ÉLU}}$ apparaît au moins deux fois sur $\mathbf{K}'$. Ce qui contredit le fait que $\mathcal{R}$ élise sur $\mathbf{K}$. □

Corollaire 7.7. *Il n'existe pas d'algorithme universel d'élection pour la famille des anneaux.*

Une autre conséquence de ce théorème est la non-stabilité pour l'union.

Corollaire 7.8. *L'ensemble des familles de graphes admettant un algorithme universel d'élection n'est pas stable pour l'union.*

Démonstration. Par exemple, on pourra considérer $\mathcal{T}$ la famille des arbres et la famille constituée par l'anneau de taille 3, A_3.

Soit r un entier. Une $r-$chaîne est l'arbre défini par :

$$\begin{aligned} V_r &= [1, r] \\ E_r &= \{\{i, j\} \mid 1 \leq i, j \leq r,\ |i - j| = 1\} \end{aligned}$$

Pour tout r, une $r-$chaîne est quasi-revêtement de rayon $\lfloor \frac{r}{2} \rfloor$ de A_3. □

En fait, on remarquera, puisque les troncatures du revêtement universel de tout graphe sont des arbres (cf Remarque 0.41), que l'ajout de la famille des arbres à toute famille contenant un graphe qui ne soit pas un arbre provoque l'impossibilité d'avoir un algorithme universel d'élection pour l'union de ces deux familles.

Entre possibilités et impossibilités

On a remarqué précédemment que l'algorithme de Mazurkiewicz permet d'élire pour la classe des graphes minimaux d'une taille donnée. Cependant le Théorème 7.6 a pour corollaire qu'il n'existe pas d'algorithme universel d'élection pour la famille des graphes minimaux, la famille des graphes pour lesquels l'élection est possible.

La question de savoir comment améliorer cet algorithme, par exemple en arrivant à élire pour une famille de graphes minimaux de diamètre borné (on peut légitimement espérer détecter la terminaison de $\mathcal{M}$ si l'on a une telle borne) se pose alors. De même que se pose la question des familles de graphes ne correspondant pas au critère du théorème 7.6. Nous allons en fait montrer dans les sections suivantes la réciproque de ce théorème. Pour cela nous allons de nouveau exploiter $\mathcal{M}$ pour lui soutirer davantage d'informations.

7.2 Un Algorithme de Cartographie Partielle

La cartographie consiste à calculer en chaque nœud une image isomorphe du réseau sous-jacent. Cette section présente un algorithme de cartographie la plus complète possible d'un réseau. En effet, les différents théorèmes d'impossibilité (reconnaissance, élection, élection dans la famille des graphes minimaux) montrent qu'il n'existe pas d'algorithme universel de cartographie complète d'un réseau. Nous allons ici présenter un algorithme qui tente de fournir le maximum d'information que l'on peut théoriquement obtenir sur l'état du réseau sous-jacent, en utilisant si possible aucune connaissance structurelle. Comme noté au chapitre Reconnaissance, sans connaissance structurelle, et dans le cas de la terminaison implicite, l'algorithme de Mazurkiewicz permet d'obtenir le maximum d'information que l'on peut sur le réseau sous-jacent. Nous allons maintenant montrer qu'en extrayant encore davantage d'information de cet algorithme, tout ce qui peut être calculé de manière distribué peut l'être via l'algorithme de Mazurkiewicz.

Nous allons par conséquent nous intéresser au déroulement de l'algorithme et non plus seulement à son étiquetage final. Afin de détailler ce déroulement, nous allons chercher à savoir comment les réétiquetages s'effectuent et plus précisément quelle est, au cours du calcul, la taille d'un "îlot de stabilité", c'est-à-dire une région connexe où aucune règle n'est applicable. Ceci permettra à chaque nœud de déterminer jusqu'à quelle distance les autres nœuds effectue la même cartographie *partielle* du réseau. L'utilisation d'une extension de l'algorithme de Szymanski, Shi et Prywes à des fins similaires a été initialement introduite dans [MT00]. La présentation qui suit est cependant entièrement originale.

7.2.1 Un algorithme pour détecter les propriétés stables

Dans cette section, nous allons décrire un algorithme de Szymanski, Shi et Prywes (que nous noterons SSP) [SSP85]. On considère un algorithme distribué qui se termine quand tous les nœuds ont atteint leurs conditions locales de terminaison. Chaque nœud n'est capable de détecter que sa propre condition de terminaison. L'algorithme SSP est capable, sous certaines conditions, de détecter un instant où le calcul sous-jacent est entièrement terminé.

Soit $\mathbf{G}$ un graphe, à chaque sommet v est associé un prédicat $P(v)$ et un entier $a(v)$. Le prédicat correspond à la condition locale de terminaison et l'entier est un compteur associé, qui sera d'autant plus grand que la région constituée des sommets "proches" où tous les prédicats sont à VRAI sera importante. Initialement $P(v)$ est à FAUX et $a(v)$ est égal à -1. Chaque opération sous-jacente agit sur le prédicat et les transformations de $a(v_0)$ sont définies par les compteurs associés aux sommets de $B_G(v_0, 1)$. Plus précisément, soit v_0 un sommet et soient $\{v_1, ..., v_d\}$ l'ensemble des sommets adjacents de v_0. Si $P(v_0) =$ FAUX alors $a(v_0) = -1$; si $P(v_0) =$ VRAI alors $a(v_0) = 1 + Min\{a(v) \mid v \in B_{\mathbf{G}}(v_0)\}$.

Ceci peut être décrit dans le système de réétiquetage suivant, avec les conventions usuelles. Chaque règle du calcul sous-jacent doit être modifiée pour que chaque réétiquetage réinitialise éventuellement le prédicat et le compteur.

RÈGLE_ÀLASSP : **Règle modifiée pour SSP**

Condition préalable :

– ...
– idem
– ...

Réétiquetage :

– ...
– idem
– ...
– si le prédicat P est non vérifié après réétiquetage alors
  – $P'(v_0) :=$ FAUX,
  – $a'(v_0) := -1$,
  sinon
  – $P'(v_0) :=$ VRAI,
  – $a'(v_0) := 1 + \min\{a(v) \mid v \in B_{\mathbf{G}}(v_0)\}$.

SSP : **Règle SSP**

Condition préalable :

– $P(v_0) =$ VRAI,
– $a(v_0) - 1 \neq \min\{a(v) \mid v \in B_{\mathbf{G}}(v_0)\}$.

Réétiquetage :

– $a'(v_0) := 1 + \min\{a(v) \mid v \in B_{\mathbf{G}}(v_0)\}$.

Dans [SSP85] l'hypothèse suivante est faite : pour tout sommet v la valeur de $P(v)$ peut devenir VRAI et demeure alors VRAI pour toujours. Ici, nous présentons une généralisation de cette hypothèse sous laquelle nous allons utiliser les règles SSP. Pour tout sommet v, la valeur de $P(v)$ n'est plus nécessairement un booléen et peut prendre une valeur quelconque. De plus, la fonction P doit alors vérifier la propriété suivante, pour toute valeur α, si $P(v)$ prend la valeur α puisqu'il la perd, alors il ne pourra plus la reprendre. En d'autres termes sous cette hypothèse, la fonction est constante entre deux moments où elle prend une même valeur. On dira dans ce cas que la fonction P a la propriété de *connexité par valeurs*.

Nous allons étendre les règles de SSP et nous noterons cette généralisation GSSP. Dans GSSP, le compteur de v n'est incrémenté que si P est constante sur la boule $B(v)$. Comme précédemment, chaque règle du calcul sous-jacent, qui calcule en particulier $P(v)$, doit être modifiée pour que chaque réétiquetage réinitialise éventuellement le compteur.

RÈGLE_ÀLAGSSP : **Règle modifiée pour GSSP**

Condition préalable :

– ...
– idem

– ...
Réétiquetage :
– ...
– idem
– ...
– Pour tout sommet v de $B(v_0)$,
si $P'(v) \neq P(v)$, alors
– $a'(v) := -1$,
sinon
– $a'(v) := a(v)$.

GSSP : **Règle GSSP**
Condition préalable :
– Pour tout $v \in B(v_0),\ P(v) = P(v_0)$,
– $a(v_0) - 1 \neq \min\{a(v) \mid v \in B(v_0)\}$
Réétiquetage :
– $a'(v_0) := 1 + \min\{a(v) \mid v \in B(v_0)\}$.

On va maintenant utiliser la notation suivante. Soit $(\mathbf{G}_i)_{0 \leq i}$ une chaîne de réétiquetage associée à l'algorithme GSSP. On notera $a_i(v)$ (resp. $P_i(v)$) la valeur du compteur (resp. de la fonction) associé au sommet v de $\mathbf{G}_i$. D'après la définition de la règle GSSP, on remarque que pour tout sommet v, $a(v)$ peut être augmenté d'au plus 1 à chaque pas et que si $a(v)$ passe de h à $h+1$, cela signifie qu'à l'étape précédente, tous les voisins w de v étaient tels que $a(w) \geq h$ et $P(w) = P(v)$. Le lemme suivant est la version itérée de cette remarque.

Lemme 7.9. *Pour tout j, pour tout v, pour tout $w \in B(v, a_j(v))$, il existe un entier $i \leq j$ tel que*

$$\begin{aligned} a_i(w) &\geq a_j(v) - d(v,w) \\ P_i(w) &= P_j(v) \end{aligned}$$

Démonstration. La démonstration est faite par induction sur le rayon $k \in [0, a_j(v)]$ de la boule. Pour $k = 0$, le résultat est trivialement vrai.

Supposons que le résultat soit vrai pour tout sommet dans la boule $B(v,k)$, $k \leq a_j(v) - 1$. Maintenant, considérons un sommet w à distance $k+1$ de v. Le sommet w a pour voisin u tel que $d(v,u) = k$. Par hypothèse de récurrence, il existe $i_u \leq j$ tel que $a_{i_u}(u) \geq a_j(v) - k$ et $P_{i_u}(u) = P_j(v)$.

Soit i une étape, parmi les étapes précédant i_u, où u a atteint $a_{i_u}(u)$ avec $P_i(u) = P_{i_u}(u)$. Cet instant existe car le compteur progresse d'au plus 1 à chaque fois, et à chaque fois que $P(u)$ est modifié, le compteur $a(u)$ est réinitialisé à -1 (Règles modifiées pour GSSP).

Entre outre, d'après la règle GSSP, nous avons alors nécessairement, $a_i(w) \geq a_{i_u}(u) - 1$ et $P_i(w) = P_i(u)$. Par conséquent $a_i(w) \geq a_j(v) - k - 1$ et $P_i(w) = P_j(v)$. Le résultat est vrai pour w et donc, de la même manière pour tous les sommets à distance $k+1$. □

En particulier, cela prouve qu'à tout instant j, pour tout v, pour tout $w \in B(v, a_j(v))$, il existe un instant i_w dans le passé tel que $P_{i_w}(w) = P_j(v)$. Nous allons maintenant montrer que, pour les sommets w dans la boule de rayon $\lfloor \frac{a_j(v)}{3} \rfloor$, nous pouvons choisir le même i_w. Il s'agit d'une propriété fondamentale de l'algorithme GSSP.

Lemme 7.10 (GSSP). *Soit une exécution de l'algorithme GSSP sous les hypothèses de connexité par valeur pour la fonction P. Pour tout j, pour tout v, il existe $i \leq j$ tel que pour tout $w \in B(v, \lfloor \frac{a_j(v)}{3} \rfloor)$, $P_i(w) = P_j(v)$.*

Démonstration.

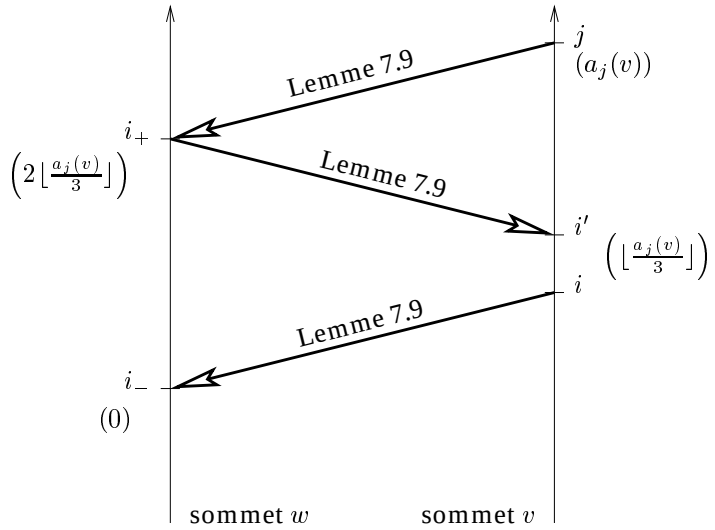


FIG. 7.4 – Schéma de la preuve : les axes verticaux correspondent au temps, les valeurs entre parenthèses sont des minorants des valeurs de a.

Soit i la première étape où $a_i(v) = \lfloor \frac{a_j(v)}{3} \rfloor$ et $P_i(v) = P_j(v)$. Soit $w \in B(v, \lfloor \frac{a_j(v)}{3} \rfloor)$, et notons i_+ une étape, dont l'existence nous est donnée par le lemme 7.9, tel que $a_{i_+}(w) \geq a_j(v) - d(v,w) \geq 2\lfloor \frac{a_j(v)}{3} \rfloor$ et $P_{i_+}(w) = P_j(v)$. Maintenant, appliquons le lemme 7.9 centré en w à l'étape i_+. Nous obtenons alors $i' \leq i_+$ tel que $a_{i'}(v) \geq a_{i_+}(w) - d(w,v) \geq \lfloor \frac{a_j(v)}{3} \rfloor = a_i(v)$ et $P_{i'}(v) = P_{i_+}(w) = P_j(v)$. Ainsi par minimalité de i, $i \leq i'$ et finalement $i \leq i_+$.

Maintenant, nous appliquons de nouveau le lemme 7.9, centré en v, à l'étape i. Nous obtenons alors $i_- \leq i$, tel que $a_{i_-}(w) \geq a_i(v) - d(v,w) \geq 0$ et $P_{i_-}(w) = P_i(v)$.

En conclusion, nous obtenons deux étapes i_- et i_+ telles que $P_{i_-}(w) = P_{i_+}(w) = P_j(v)$, et $i_- \leq i \leq i_+$. Par propriété de connexité par valeur de la fonction P, $P_i(w) = P_j(v)$. □

Remarque 7.11. Le tiers du compteur est un rayon optimal de stabilité. On peut construire des exemples où la fonction P n'est pas nécessairement constante sur la boule de centre v et de rayon $\lfloor \frac{a_j(v)}{3} \rfloor + 1$.

Remarque 7.12. Chaque sommet "connaît" la valeur exacte de i, cela apparaît dans la démonstration. Pourtant, cela ne sera en fait pas utilisé par la suite.

7.2.2 Algorithme de Mazurkiewicz + Algorithme GSSP = Algorithme de Cartographie Partielle

La principale idée que nous allons mettre en œuvre dans cette section est d'utiliser l'algorithme GSSP pour calculer le rayon de stabilité de $\mathcal{M}$ pour tout nœud. En d'autres mots, tout sommet saura jusqu'à quelle distance les autres nœuds sont en accord avec le graphe reconstruit $\mathbf{H}_{M(u)}$. Soit $\mathbf{G} = (G, \lambda)$ un graphe étiqueté, soit $(\mathbf{G}_i)_{0 \leq i}$ une chaîne de réétiquetage associée à une exécution de l'algorithme de Mazurkiewicz sur le graphe $\mathbf{G}$. Au sommet v de $\mathbf{G}_i$ est associé l'étiquette $(\lambda(v), (n_i(v), N_i(v), M_i(v)))$. En utilisant l'interprétation de la section 3.2.2, cet étiquetage permet de reconstruire un graphe. On rappelle la définition de ce graphe reconstruit à partir du contenu d'une boîte à lettre M. Pour une boîte M, on a défini le prédicat $\underline{\text{CHAMPION}}(l, n, N, M)$ qui est vrai si il n'y a aucun $(n, l', N') \in M$ vérifiant

$$l' > l \text{ ou } (l = l' \text{ et } N \prec N')\}.$$

Le graphe $\mathbf{H}_M$ est alors défini par

$$\begin{aligned} V(\mathbf{H}_M) &= \{n \mid \exists N, l, \underline{\text{CHAMPION}}(l, n, N, M)\} \\ E(\mathbf{H}_M) &= \{\{n, n'\} \mid \exists N, l, \underline{\text{CHAMPION}}(l, n, N, M), \text{ et } N = (..., (n', l', l''), ...)\} \end{aligned}$$

On a défini également un étiquetage sur $\mathbf{H}_M$ comme suit $\lambda_M(n) = (l, n, N, M)$, avec $\underline{\text{CHAMPION}}(n, l, N, M)$. On étend le prédicat $\underline{\text{CHAMPION}}$ aux sommets par $\underline{\text{CHAMPION}}(v) = \underline{\text{CHAMPION}}(l(v), n(v), N(v), M(v))$ et en chaque sommet v du graphe $\mathbf{G}_i$, la fonction $\mathbf{H}$ sera définie par

$$\mathbf{H}(v) = \begin{cases} \mathbf{H}_{M(v)} \text{ si } \underline{\text{CHAMPION}}(v) \\ \perp \text{ sinon.} \end{cases}$$

On utilisera comme fonction d'entrée de GSSP la fonction $\mathbf{H}$. Le compteur associé a est calculé comme précédemment par l'algorithme GSSP. La propriété de connexité par valeur de $\mathbf{H}$, excepté pour $\perp$, est une conséquence du fait que, par construction, la suite des boîtes aux lettres $M(v)$ est strictement croissante au sens de l'inclusion. Or, à tout instant, $M(v)$ est la quatrième composante de l'étiquette de tout sommet de $\mathbf{H}(v)$.

On appelle système *PCarto* (pour cartographie partielle) l'ajout du calcul de $\mathbf{H}$ à l'algorithme de Mazurkiewicz et les modifications correspondantes des règles pour GSSP. Notons que tant que $\mathbf{H}(v) = \perp$, le nœud sait qu'il n'a pas encore atteint son état final. La"sortie utile" de *PCarto* est, sur chaque sommet v, $(\mathbf{H}(v), r_{\text{conf}}(v))$, où r_{conf} dénote le "rayon de confiance" de la cartographie partielle, $r_{\text{conf}}(v) = \lfloor \frac{a(v)}{3} \rfloor$.

La propriété principale de *PCarto* est alors une conséquence directe du lemme 7.10 :

Théorème 7.13 (cartographie en quasi-revêtement). *A tout moment j, pour tout sommet v, si $\mathbf{H}_j(v) \neq \perp$, alors il existe une étape précédente $i \leq j$, telle que $\mathbf{G}_i$ soit un quasi-revêtement de $\mathbf{H}_j(v)$ de centre v et de rayon $r_{conf_j}(v)$.*

Démonstration. On abrège $r_{\text{conf}_j}(v)$ en r_{conf}.

D'après le lemme 7.10, il existe $i < j$ tel que, pour tout $w \in B_{\mathbf{G}_i}(v, r_{\text{conf}})$, $\mathbf{H}_i(w) = \mathbf{H}_j(v)$, *i.e.* $M_i(w) = M_j(v)$ et $\underline{\text{CHAMPION}}(w)$ en particulier. Notons δ le morphisme partiel de $\mathbf{G}_i$ sur $\mathbf{H}_j(v)$ projetant chaque sommet w de $B_{\mathbf{G}_i}(v, r_{\text{conf}})$ sur $n_j(w) = n_i(w)$. On va montrer que δ définit un quasi-revêtement de rayon r_{conf}.

Pour cela il nous faut définir un revêtement associé $\mathbf{F}$. Nous allons le construire à partir d'une copie isomorphe de $B_{\mathbf{G}_i}(v, r_{\text{conf}})$ à laquelle nous ajouterons des arbres infinis idoines pour en faire un revêtement de $\mathbf{H}_j(v)$.

On note $\mathbf{H} = \mathbf{H}_j(v)$. Formellement, pour tout sommet u de $\mathbf{H}$, pour tout $S \subset N_{\mathbf{H}}(u)$, on définit maintenant $\overline{\mathbf{H}}(u, S)$ comme étant le sous-arbre de $\widehat{\mathbf{H}}$, le revêtement universel de $\mathbf{H}$, induit par l'ensemble des chemins commençant en u suivi d'un sommet de S. Pour tout w tel que $d(v, w) = r_{\text{conf}}$, on définit $\mathbf{H}_w$ comme étant une copie isomorphe de $\overline{\mathbf{H}}(\delta(w), S_w)$, où $S_w = \{s \in N_{\mathbf{H}}(\delta(w)) \mid \forall x \in N_{\mathbf{G}_i}(w) \cap B_{\mathbf{G}_i}(v, r_{\text{conf}}), \delta(x) \neq s\}$. Les copies sont disjointes, si $w \neq w'$ alors $\mathbf{H}_w \cap \mathbf{H}_{w'} = \emptyset$

On note B une copie isomorphe de $B_{\mathbf{G}_i}(v, r_{\text{conf}})$, que l'on complète par les $\mathbf{H}_w$, pour tous les w tels que $d(v, w) = r_{\text{conf}}$, en identifiant à chaque fois w et la racine de $\mathbf{H}_w$. Le graphe $\mathbf{F}$ est le graphe (éventuellement infini) ainsi obtenu, φ est la bijection entre $B_{\mathbf{G}_i}(v, r_{\text{conf}})$ et sa copie disjointe B. Le morphisme γ est défini en tout sommet u par :

- $\delta(\varphi^{-1}(u))$ si $u \in B$
- t si $u \in \mathbf{H}_w$ et u est un chemin de $\delta(w)$ à t.

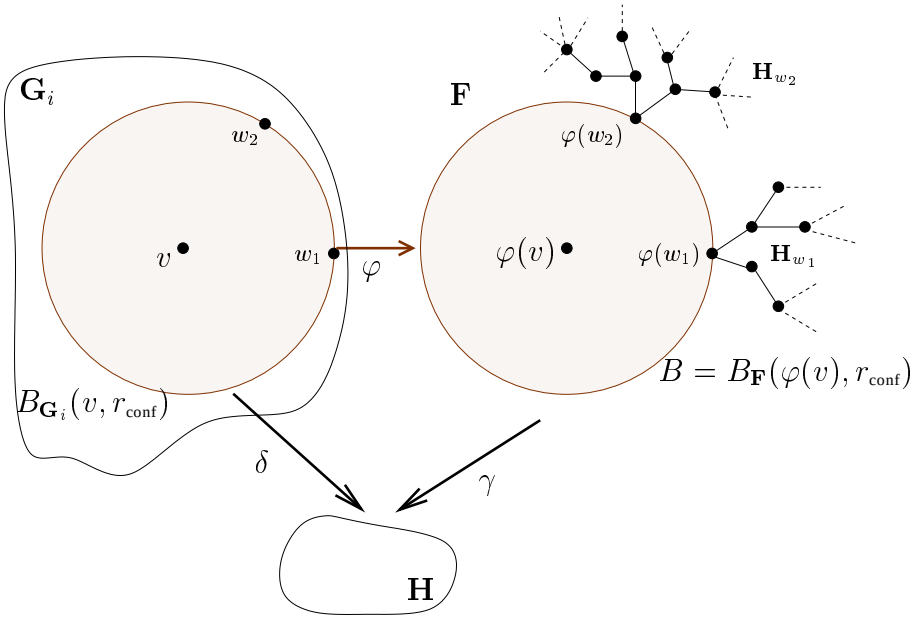


FIG. 7.5 – Construction du graphe $\mathbf{F}$

Il reste à vérifier que $\gamma : \mathbf{F} \longrightarrow \mathbf{H}$ définit bien un revêtement. Pour $u \in B_{\mathbf{F}}(\varphi(v), r_{\text{conf}} - 1)$, il vient que $B_{\mathbf{F}}(u)$ est en bijection avec $B_{\mathbf{H}}(\varphi^{-1}(u))$. En effet, soit $w, w' \in B_{\mathbf{F}}(u)$ tel que $\gamma(w) = \gamma(w')$. Comme $w, w' \in B$, cela implique que $\delta(\varphi^{-1}(w)) = \delta(\varphi^{-1}(w'))$, et donc que $n_i(\varphi^{-1}(w) = n_i(\varphi^{-1}(w')$. Comme $\underline{\text{CHAMPION}}(\varphi^{-1}(w))$ et $\underline{\text{CHAMPION}}(\varphi^{-1}(w'))$ sont

vérifiés, il vient que $n_i(\varphi^{-1}(w)$ et $n_i(\varphi^{-1}(w')$ sont non nuls et donc, par lemme 1.9, $\varphi^{-1}(w) = \varphi^{-1}(w')$, et donc $w = w'$. Pour la surjectivité, on se base sur le lemme suivant :

Lemme 7.14. *Soit $u, u' \in B_{\mathbf{G}_i}(v, r_{conf})$. Si $\delta(u) = \delta(u')$ alors $\lambda(u) = \lambda(u')$ et $N(u) = N(u')$.*

Démonstration. Tout d'abord, par hypothèse sur i, on a $M_i(u) = M_i(u')$. De plus les prédicats $\underline{\text{CHAMPION}}(u)$ et $\underline{\text{CHAMPION}}(u')$ sont vérifiés. Or $\delta(u) = \delta(u')$ implique que $n_i(u) = n_i(u')$, et donc $\lambda(u) = \lambda(u')$ et $N(u) = N(u')$ par définition de $\underline{\text{CHAMPION}}$. □

Maintenant, soit $s \in B_{\mathbf{H}}(\gamma(u))$. Par définition de $\mathbf{H}$, il existe t, w tels que $N_i(w) = (\cdots, (n_i(t), \lambda(t)), \cdots)$ et $\delta(t) = s$ et $\delta(w) = \gamma(u) = \delta(\varphi^{-1}(u))$. Donc, d'après le lemme précédent, $N_i(\varphi^{-1}(u)) = (\cdots, (n_i(t), \lambda(t)), \cdots)$. Les voisins de $\varphi^{-1}(u)$ étant dans $B_{\mathbf{G}_i}(v, r_{\text{conf}})$, il vient que u a un voisin t' tel que $\gamma(t') = s$.

Pour $u \in \mathbf{H}_w$, et différent de la racine w, cela découle de la définition (revoir la construction du revêtement universel section 0.4.5). Pour u tel que $d(\varphi(v), u) = r_{\text{conf}}$, on constate que, par construction de $S_{\varphi^{-1}(u)}$, les sommets de $N_{\mathbf{H}}(\gamma(u))$ qui n'ont pas d'antécédents dans B en ont exactement un dans $\mathbf{H}_{\varphi^{-1}(u)}$. □

Il faut noter que *PCarto* ne se termine jamais. Nous utiliserons $\mathbf{H}$ et r_{conf} pour détecter, *sous certaines conditions*, la terminaison de l'algorithme de Mazurkiewicz sous-jacent. En effet, de la terminaison de $\mathcal{M}$, on obtient

Corollaire 7.15. *Il existe i tel que pour tout $j \geq i$, pour tout sommet v*
- $\mathbf{H}_j(v) = \mathbf{H}_i(v)$
- $\mathbf{G}_j$ *est un revêtement de* $\mathbf{H}_j(v)$
- $(a_j(v))_{j \geq i}$ *est suite croissante divergente*

En particulier, si $\mathbf{G}$ est minimal alors pour tout $j > i$, $\mathbf{H}$ est une cartographie de $\mathbf{G}_f$, l'état final de l'exécution de $\mathcal{M}$. Par suppression des registres associés à l'algorithme de Mazurkiewicz, on obtient une image isomorphe de $\mathbf{G}$, une cartographie. Si l'on prend en compte les numéros calculés par $\mathcal{M}$, un sommet sait également où il se trouve exactement sur cette cartographie. La difficulté est de détecter cet instant i. Nous allons montrer à quelles conditions l'on peut y parvenir dans la partie suivante.

7.3 Algorithmes d'Election Universels

Nous allons pouvoir utiliser l'algorithme de cartographie partielle, qui, dans le cas des graphes minimaux, finit par fournir une cartographie *complète*, pour élire dans une famille de graphes ne satisfaisant pas aux conditions du Théorème 7.6.

7.3.1 Caractérisation

Théorème 7.16. *Soit $\mathcal{F}$ une famille de graphes. Alors il existe un système de réétiquetage de graphes universel d'élection pour $\mathcal{F}$ si et seulement si :*
- *$\mathcal{F}$ est une famille de graphes minimaux pour les revêtements,*

– *il existe une fonction récursive* $r : \mathcal{F} \longrightarrow \mathbb{N}$ *telle que tout graphe* $\mathbf{G}$ *n'admette aucun quasi-revêtement de rayon* $r(\mathbf{G})$ *dans* $\mathcal{F}$*, excepté lui-même.*

Démonstration. La condition nécessaire est le théorème 7.6. Le caractère récursif de r découle de la construction du rayon de valeur $2(n + |V(\mathbf{G})|)$ employé dans la preuve.

Pour montrer la condition suffisante, nous allons utiliser l'algorithme de cartographie partielle. On suppose donc que

– $\mathcal{F}$ est une famille de graphes minimaux pour les revêtements,
– il existe une fonction (calculable) $r : \mathcal{F} \longrightarrow \mathbb{N}$ telle que tout graphe $\mathbf{G}$ n'admette aucun quasi-revêtement de rayon $r(\mathbf{G})$ dans $\mathcal{F}$, excepté lui-même.

Soit UNIVELECT le système de réétiquetage de graphes constitué des règles de *PCarto* et des règles additionnelles suivantes, en reprenant les même notations et en notant $\overline{\mathbf{H}}$ (resp. $\overline{\mathbf{G}_i}$) le graphe obtenu en supprimant les registres n, N, M associés à $\mathcal{M}$ de $\mathbf{H}$, (resp. $\mathbf{G}_i$) :

ELECT : **Détection de la terminaison de $\mathcal{M}$ et élection**

Condition préalable :

– $\overline{\mathbf{H}(v_0)} \in \mathcal{F}$,
– $r_{\text{conf}}(v_0) \geq r\left(\overline{\mathbf{H}(v_0)}\right)$,
– $n(v_0) = 1$.

Réétiquetage :

– v_0 prend l'étiquette ÉLU.
– Les voisins $v \in B(v_0)$ prennent l'étiquette PAS ÉLU.

TERM : **Règle de diffusion terminale**

Condition préalable :

– v_0 n'est étiqueté ni par ÉLU ni par PAS ÉLU
– Il existe $v \in B(v_0)$ étiqueté par PAS ÉLU.

Réétiquetage :

– Tous les sommets $v \in B(v_0)$ prennent l'étiquette PAS ÉLU.

Ce système est correctement défini car l'entier $r\left(\overline{\mathbf{H}(v_0)}\right)$ est défini dès que $\overline{\mathbf{H}(v_0)} \in \mathcal{F}$. De plus, la terminaison de l'algorithme distribué est alors assuré par le Corollaire 7.15. Pour tout sommet v_0, il existe un moment i où $\mathbf{H}(v_0) \simeq \mathbf{G}_i$, car $\mathbf{G}$ est minimal, et où $r_{\text{conf}}(v_0) \geq r(\mathbf{H})$. Et d'après le lemme 3.6, il existe toujours au moins un sommet numéroté 1.

La correction provient du théorème 7.13. En effet, $\mathbf{G}$ est un quasi-revêtement de rayon $r_{\text{conf}}(v_0)$ de $\overline{\mathbf{H}(v_0)}$ et donc par hypothèse appliquée à $\overline{\mathbf{H}}(v_0)$, si $\overline{\mathbf{H}(v_0)}$ appartient à $\mathcal{F}$ et que $r_{\text{conf}}(v_0) \geq r\left(\overline{\mathbf{H}(v_0)}\right)$, alors $\mathbf{G} \simeq \overline{\mathbf{H}(v_0)}$. Par conséquent, le numéro 1 apparaît une et une seule fois, et l'on a bien procédé à une élection. □

Remarque 7.17. Il est important de noter que cette preuve est *constructive*, c'est-à-dire que si l'on connaît effectivement l'algorithme alors on peut calculer $r(\mathbf{G})$ pour tout $\mathbf{G}$. Inversement, si l'on connaît effectivement la fonction r, alors on peut donner effectivement l'algorithme.

Remarque 7.18. Pour étudier la complexité de cet algorithme, il faut remarquer qu'il se partage en deux phases. La première consiste en l'exécution de $\mathcal{M}$, la seconde est la phase GSSP de détection de la terminaison de $\mathcal{M}$. La complexité de $\mathcal{M}$ a été étudiée au chapitre 2 et on peut considérer qu'au cours de la première phase, on applique prioritairement les règles de $\mathcal{M}$, et que GSSP ne s'exécute donc que dans les régions où $\mathcal{M}$ ne s'exécute pas, et par conséquent n'ajoute aucune complexité d'un point de vue parallèle.

En ce qui concerne la seconde phase, c'est-à-dire jusqu'à ce qu'un nœud atteigne $r_{\text{conf}} = r(\mathbf{G})$, chaque nœud fait évoluer son rayon de confiance de 1 en 1. Il y a donc, d'un point de vue séquentiel, au plus $r(\mathbf{G}) \times n$ pas de calcul. D'un point de vue parallèle, le calcul exact reste à faire, mais on peut l'estimer de l'ordre de $r(\mathbf{G})$.

Ces deux remarques permettent d'estimer le coût d'utilisation d'un algorithme "générique" en place d'un algorithme spécifique à $\mathbf{G}$. En effet, en utilisant le même algorithme pour $\mathbf{G}$ et tous ses quasi-revêtements de rayon au plus la complexité de $\mathcal{M}$ sur $\mathbf{G}$, la complexité totale d'une exécution sur $\mathbf{G}$ reste alors du même ordre de grandeur. En particulier, si l'on ne connaît pour $\mathbf{G}$ qu'une borne b de son diamètre Δ, alors si $b \leq \Delta^2$, la complexité de UNIVELECT en ne connaissant que b n'est pas supérieure à celle en connaissant la taille. Car en ce cas, la complexité de l'élection se situe essentiellement dans $\mathcal{M}$ et non pas dans la stabilisation de GSSP.

On peut également noter que la complexité de l'algorithme de Mazurkiewicz, calculée au chapitre 2, est assez souvent pire que les bornes optimales de complexité des algorithmes d'élection connus. Ce manque d'optimalité est une contrepartie de l'universalité de l'algorithme.

7.3.2 Résultats connus

Un certain nombre de résultats connus apparaissent comme des corollaires directs du théorème précédent. En particulier, les familles suivantes admettent un algorithme universel d'élection.

- Les arbres, les grilles, les réseaux avec identités, les graphes complets : ces réseaux sont minimaux et n'admettent aucun quasi-revêtement à plus de 2 feuillets. La fonction r peut donc être 2 fois la taille du graphe (cf Lemme 0.44).
- Les graphes minimaux de taille donnée : en prenant $r(\mathbf{G}) = |V(\mathbf{G})|$.

7.3.3 Nouveaux Résultats

On obtient également de nouveaux résultats en terme d'existence ou d'absence d'existence d'algorithme d'élection universel.

Possibilité

Proposition 7.19. *Soit d un entier. La famille des graphes minimaux de diamètre au plus d admet un algorithme d'élection.*

Démonstration. $r(\mathbf{G}) = 2d + 1$ convient. □

Ceci est nouveau et ne peut être déduit de [Maz97]. Ce qui est également nouveau, c'est que l'on peut élire dans les réseaux avec au moins 1 et au plus k nœuds distingués, avec k donné.

Proposition 7.20. *Soit k un entier. La famille des graphes dont les sommets sont étiquetés par 0 ou 1, minimaux (en tant que graphes étiquetés) et possédant au moins un et au plus k sommets étiquetés par 1 admet un algorithme d'élection.*

Démonstration. Par définition, les quasi-revêtements dans cette famille ont au plus k feuillets, par conséquent, en utilisant le lemme 0.44, il vient que $r(\mathbf{G})$ défini par $k * |V(\mathbf{G})|$ convient pour appliquer le Th. 7.16. □

Impossibilité

En terme d'impossibilité, on retrouve la non-existence d'algorithme universel d'élection pour les anneaux de taille première et plus généralement pour les tores $d-$dimensionnels minimaux. Nous allons étudier ceci plus en détail. On note $\mathbb{Z}_n$ l'anneau des entiers modulo n.

Définition 7.21. *Soit $d \in \mathbb{N}$. Le tore d-dimensionnel de longueurs $n_1, \ldots, n_d \in \mathbb{N}^*$ est le graphe $T_{n_1,\ldots,n_d}$ défini par :*

$$\begin{aligned} V(T_{n_1,\ldots,n_d}) &= \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_d} \\ E(T_{n_1,\ldots,n_d}) &= \{\{(x_1,\ldots,x_d),(y_1,\ldots,y_d)\} \mid \exists i \forall j \neq i\ x_j = y_j;\ x_i - y_i = \pm 1 \mod n_i\} \end{aligned}$$

Proposition 7.22. *Soit $T_{n_1,\ldots,n_d}$ et $T_{n'_1,\ldots,n'_d}$ deux tores $d-$dimensionnels. Alors $T_{n'_1,\ldots,n'_d}$ est un revêtement de $T_{n_1,\ldots,n_d}$ si et seulement si il existe une permutation σ telle que pour tout $i \leq d$, $n_{\sigma(i)}$ divise n'_i.*

Démonstration. Condition suffisante. Immédiat.

Condition nécessaire.

Notons γ le revêtement et 0_d (resp. $0'_d$) le sommet de coordonnées nulles dans $T_{n_1,\ldots,n_d}$ (resp. $T_{n'_1,\ldots,n'_d}$). Par symétrie on peut supposer que l'image de $0'_d$ est 0_d. Soit $i \leq d$. Notons C l'image du cycle C' issu de $0'_d$ et de direction i. L'image du sommet $1'_i = (0,\ldots,1,\ldots,0)$ (1 en $i^{\text{ème}}$ coordonnée sur $T_{n'_1,\ldots,n'_d}$) de C' est $1_j = (0,\ldots,\pm 1,\ldots,0)$ (± 1 en $j^{\text{ème}}$ coordonnée sur $T_{n_1,\ldots,n_d}$). Posons $\sigma(i) = j$. Montrons que C est inclus dans le cycle C_j de direction j. Par l'absurde, supposons qu'il existe v' dans $T_{n'_1,\ldots,n'_d}$ tel que $v = \gamma(v') \notin C_j$. On peut supposer, par symétrie encore, que v' est voisin de $1'_i$. Cf figure 7.6.

Les sommets v et 0_d ont par conséquent un voisin commun distinct de 1_j. Notons celui-ci u, et notons u' son antécédent par γ. Par propriété de morphisme de graphe, u'

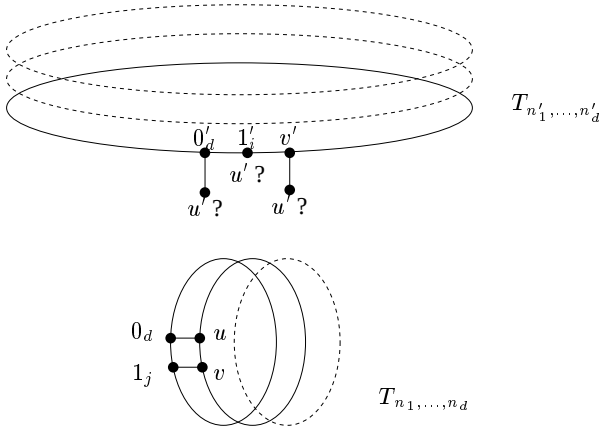


FIG. 7.6 – Divisibilité des longueurs des tores $d-$dimensionnels.

doit être adjacent à $0'_d$ et v', c'est-à-dire $u' = 1'_i$. Ce qui signifie que $u = \gamma(u') = \gamma(1'_i) = 1_j$, ce qui est contradictoire avec le fait que u est distinct de 1_j.

Finalement il vient que $C \subseteq C_j$ et que γ induit un revêtement de C' sur C_j. On conclue donc par Prop. 0.29. □

Cependant cela ne signifie pas que les tores dont les longueurs sont premières soient minimaux. En particulier, on peut montrer que si deux longueurs sont identiques alors le tore n'est pas minimal. Il suffit également de se rappeler que le cube est un revêtement du tétraèdre.

On a également le lemme suivant

Lemme 7.23. *Soit $T_{n_1,\ldots,n_d}$ un tore $d-$dimensionnel. Soit r un entier. Pour tout $i \leq d$, pour tout $n > n_i + 2r$, $T_{n_1,\ldots,n_{i-1},n,n_{i+1},\ldots,n_d}$ est un quasi-revêtement de rayon r de $T_{n_1,\ldots,n_d}$.*

Démonstration. On prend la projection canonique pour δ. Pour revêtement associé on prend $T_{n_1,\ldots,n_{i-1},p,n_{i+1},\ldots,n_d}$, où p est un multiple de n_i supérieur à n. La vérification est immédiate. □

Ce lemme a pour corollaire

Corollaire 7.24. *Soit $d \in \mathbb{N}$. Il n'existe pas d'algorithme d'élection universel dans la famille des tores d-dimensionnel minimaux.*

7.4 Élection avec Connaissance Structurelle

Comme précédemment remarqué, l'algorithme de Mazurkiewicz permet d'élire dans tout graphe minimal *si l'on connaît la taille*. En effet, on peut utiliser cet algorithme et l'information concernant la taille comme *paramètre* : $\mathcal{M}_t$. Ou encore, si le graphe a connaissance du fait qu'il est un arbre, il peut utiliser le système ELECT_ARBRE. Ceci nous amène à étudier quelles connaissances structurelles permettent d'élire dans la famille des graphes minimaux.

Définition 7.25. *Soit $\kappa : \mathcal{G}_{\min} \longrightarrow L$ une information structurelle. Un système de réétiquetage de graphes $\mathcal{R}$* élit avec connaissance structurelle κ *si pour tout graphe minimal $\mathbf{G}$, $\mathcal{R}$ élit sur $(\mathbf{G}, \Lambda_{\kappa(\mathbf{G})})$.*

Revenons sur [Maz97], l'énoncé complet est en fait :

Théorème 7.26 (Mazurkiewicz, [Maz97]). *$\mathcal{M}$ élit avec connaissance de la taille.*

A noter par conséquent que supposer que κ soit défini sur l'ensemble des graphes minimaux n'est pas restrictif puisque si l'on souhaite étudier l'élection avec une connaissance structurelle κ sur un sous-ensemble $\mathcal{F}$, il suffira d'étendre κ comme encodant la taille sur $\mathcal{G}_{\min} \setminus \mathcal{F}$.

Le théorème suivant donne la caractérisation des connaissances structurelles permettant d'élire avec un algorithme "paramétré" par cette connaissance.

Théorème 7.27. *Soit $\kappa : \mathcal{G}_{\min} \longrightarrow L$ une information structurelle. Il existe un système de réétiquetage de graphes $\mathcal{R}$ élisant avec connaissance structurelle κ si et seulement si il existe une application récursive $r : \mathcal{G}_{\min} \longrightarrow \mathbb{N}$ telle que pour tout graphe $\mathbf{G} \in \mathcal{G}_{\min}$, il n'existe dans $\kappa^{-1}(\kappa(\mathbf{G}))$ aucun quasi-revêtement de $\mathbf{G}$ de rayon $r(\mathbf{G})$, excepté $\mathbf{G}$ lui-même.*

Démonstration. Ceci est une conséquence du caractère constructif des preuves des Théorèmes 7.6 et 7.16, et en particulier du fait que la preuve du Th. 7.6 donne pour tout graphe $\mathbf{G}$ une borne $r(\mathbf{G})$ qui ne dépend que de $\mathbf{G}$ et $\mathcal{R}$. □

En particulier UNIVELECT permet d'élire en connaissant une borne sur le diamètre. Par contre, il n'est pas possible d'élire sur $\mathcal{G}_{\min}$ sans aucune connaissance. D'autre part, on peut constater qu'il n'existe pas de "plus petite" connaissance structurelle permettant d'élire. Plus petite au sens où toute autre connaissance permettant d'élire pourrait être déduite logiquement de celle-ci. Pour pouvoir élire il faut et il suffit de pouvoir distinguer un graphe de ses quasi-revêtements *à partir* d'un certain rayon, aussi grand soit-il. Cela constitue une suite infinie de connaissances structurelles de plus en plus fine. En fait, elles sont toutes équivalentes d'un point de vue distribué, voir la section 8.2.

Il faut bien remarquer que le théorème 7.27 est différent du fait que, pour chaque étiquette α, $\kappa^{-1}(\alpha)$ vérifie la condition du Théorème 7.16. En effet l'existence d'un système de réétiquetage de graphes pour chaque famille $\kappa^{-1}(\alpha)$ n'assure pas l'existence d'un système de réétiquetage de graphes paramétré par κ. Pour s'en convaincre, étudions l'exemple suivant, où l'on verra qu'une famille d'algorithmes d'élection universels ne donne pas nécessairement un algorithme d'élection avec connaissance structurelle.

Soit ψ une fonction récursive injective dont l'image n'est cependant pas récursive. De telles fonctions existent, on pourra se reporter à tout bon livre de cours sur les fonctions récursives, par exemple [CL93], Chap. 5, Ex. 13.

On va utiliser une familles de tores particulières. Pour tout $p \in \mathbb{N}$, on note T_p^d le tore $d+1$-dimensionnel $T_{p \times \underbrace{3 \times \cdots \times 3}_{d \text{ fois}}}$.

On définit κ par

$$\kappa(\mathbf{G}) = \begin{array}{ll} d & \text{si } \mathbf{G} = T_3^d \\ d & \text{si } \mathbf{G} = T_p^d \text{ et } \psi(p) = d \\ \mathbf{G} & \text{sinon.} \end{array}$$

Cette application est récursive et avec cette connaissance structurelle, chaque famille de préimages vérifie la condition du Th. 7.16. Cela est évident pour ceux dont on encode la topologie. Pour les autres, il suffit de remarquer que pour tout $d \in \mathbb{N}$, $\kappa^{-1}(d)$ est de cardinal 1 ou 2. Cependant, il n'existe pas de fonction *récursive* $r : \mathcal{G}_{\min} \longrightarrow \mathbb{N}$ satisfaisant au Th. 7.27, car cela contredirait la non-récursivité de $\psi(\mathbb{N})$. En effet si une telle fonction r existait alors pour tout $d, p \in \mathbb{N}$, si $\psi(p) = d$ alors $p < r(T_d^d) + 4$ par définition de r et d'après le lemme 7.23. Et donc pour tester l'appartenance de d donné à $\psi(\mathbb{N})$, il suffirait de rechercher son antécédent éventuel parmi les entiers de $[0, r(T_d^d)+4]$.

7.5 Conclusion : Résoudre un Problème en Fonction de la Connaissance Structurelle

L'étude complète du problème de l'élection est très instructive en ce sens qu'elle illustre ce que peut signifier résoudre un problème en algorithmique distribuée, c'est-à-dire résoudre un problème en fonction de la connaissance structurelle que l'on peut avoir sur le réseau.

On peut distinguer plusieurs étapes. La première consiste à décrire de façon exhaustive les réseaux pour lesquels il existe, au sens mathématique, une solution. Cela peut passer en fait par la description de l'ensemble complémentaire des réseaux pour lesquels le problème est impossible à résoudre. Cette description s'effectue en deux sous-étapes la première donnant une condition nécessaire d'existence (Th. 7.3 d'Angluin), la seconde prouvant la réciproque, la condition suffisante (Th. 7.4 de Mazurkiewicz). Dans le cas de l'élection, nous pouvons ainsi décrire les réseaux admettant une solution : les réseaux minimaux pour les revêtements. Cette condition de minimalité est intrinsèque au réseau et peut être vue d'une certaine manière comme spécifique au problème de l'élection.

Ensuite, on cherche à décrire les solutions sur les réseaux pour lesquels il en existe. Cette étude s'effectue de manière à donner les solutions les plus générales. Ce qui peut caractériser le degré de généralité d'une construction de solution est de déterminer quelle caractéristique précise du réseau est en fait utilisé dans la solution. La solution peut être tout à fait spécifique au réseau : pour un réseau différent, il faudra utiliser un autre algorithme ; ou encore universel pour une famille de réseau : le *même* algorithme fonctionnera pour tous les réseaux de la famille. Par exemple, on peut concevoir un algorithme qui fonctionnera pour tous les réseaux en forme d'anneau.

La recherche de solutions générales en ce sens s'effectue non seulement pour des raisons théoriques, mais aussi parce que cela a des implications du point de vue de la fiabilité des algorithmes. D'un point de vue pratique, on cherchera à diminuer les conditions que le réseau doit vérifier afin que l'algorithme fonctionne correctement

sur celui-ci. En d'autres termes, le réseau doit vérifier un certain nombre de spécifications et restreindre ces spécification revient à augmenter la fiabilité de l'algorithme puisque des modifications du réseau sous-jacent n'entraînent pas nécessairement d'incorrection pour l'algorithme. Pour ce qui nous concerne, un algorithme d'élection en ne connaissant qu'une borne sur le diamètre est plus fiable qu'un algorithme d'élection en connaissant la taille au sens où l'on peut dans le premier cas enlever ou ajouter un certain nombre de nœuds, et conserver un algorithme d'élection fonctionnant correctement.

Si l'on souhaite ainsi "diminuer" les spécifications requises pour résoudre le problème, c'est-à-dire utiliser le même algorithme pour résoudre le problème sur plusieurs réseaux, l'étude des solutions universelles est alors un préalable au même titre que les études d'impossibilités précédemment décrites, puisque cela permet de préciser les formes de connaissances structurelles admissibles. Il faut en ce sens voir le Th. 7.16 comme un théorème complet d'impossibilité au même titre que la réunion des théorèmes 7.3 et 7.4. La condition énoncée au Th. 7.16 peut être vue comme une condition de changement d'échelle (scalability) : le problème de l'élection supporte mal la "montée en charge".

Le dernier point consiste finalement à décrire effectivement les solutions en fonction de la connaissance structurelle, c'est-à-dire de donner un algorithme distribué "paramétré" par la connaissance structurelle, ou encore un algorithme qui fournit le système de réétiquetage en fonction de la valeur de la connaissance structurelle. Le théorème 7.27 est lui un théorème complet de possibilité au sens où il donne toutes les connaissances structurelles permettant d'élire effectivement. La connaissance d'une borne sur le diamètre en est sans doute un des cas particuliers les plus intéressant en pratique, mais la condition nécessaire et suffisante énoncée est en fait bien plus générale.

Il est sans doute délicat et peut-être artificiel de diviser notre travail en ces trois étapes puisque, en général, les preuves d'existence de solution fournissent des algorithmes (plus ou moins effectifs). Cependant notre démarche a été de rechercher à aboutir au Th. 7.27 en en délimitant progressivement les conditions par les théorèmes complets d'impossibilités 7.3 et 7.4, et 7.16.

Chapitre 8

Fonctions Scalaires Localement Calculables

Le but de ce chapitre est d'étudier et de caractériser les fonctions que l'on peut calculer en utilisant des règles de réétiquetages. Avant de définir formellement ces notions, nous allons évoquer intuitivement ce que l'on entend par de telles fonctions. Dans la deuxième partie sur la reconnaissance de familles de graphes, nous avons étudié des fonctions d'un type très particulier : les fonctions booléennes avec terminaison implicite. L'utilisation de la relation $\sim$, basée sur l'utilisation des revêtements, a permis de mettre en valeur les contraintes *intrinsèquement* dues à l'aspect distribué des calculs. On peut également voir ce chapitre comme la mise en pratique de techniques particulières, les revêtements, à l'étude des fonctions les plus simples : le résultat ne peut prendre que deux valeurs, OUI(le réseau sous-jacent possède la propriété recherchée) et NON(le réseau n'a pas la propriété recherchée). Pour l'étude plus générale des fonctions, nous allons maintenant autoriser les résultats à prendre des valeurs dans un ensemble quelconque. Ceci, nous le verrons, ne change pas fondamentalement les résultats, tout est encore dépendant des similarités entre réseau, ces similarités étant exprimées par les revêtements. Ce qui apporte une modification plus substantielle des techniques à mettre en œuvre est l'introduction d'une terminaison explicite des calculs. Une étiquette particulière, dite terminale, indique au nœud que le résultat du calcul est obtenu. Une telle contrainte de terminaison explicite est également présente dans le problème de l'élection, puisque les étiquettes $\boxed{\text{ÉLU}}$ et $\boxed{\text{PAS ÉLU}}$ n'apparaissent qu'une seule fois, en fin de processus. La technique utilisée alors, l'algorithme de quasi-cartographie, va permettre de prouver les caractérisations, qui seront basées, de manière non surprenante, sur des quasi-revêtement.

Nous commencerons par revenir sur les tâches à terminaison implicite et montrerons comment les techniques utilisées dans la partie Reconnaissance se généralisent facilement, en particulier pour les tâches autostabilisante. Nous verrons également comment utiliser l'algorithme *PCarto* pour améliorer ses techniques. Nous formaliserons ensuite les fonctions scalaires, les tâches dont le calcul sur un réseau s'achève par un étiquetage uniforme par une étiquette fonction de ce réseau. Nous reviendrons ensuite sur les différences qu'il peut exister entre différentes saveurs de "terminaisons expli-

cites". Nous présentons ensuite une caractérisation des fonctions scalaires, puis nous verrons que ce type de calcul, si l'on y intègre l'utilisation de connaissances structurelles, revient à étudier l'équivalence entre connaissance structurelle. Nous verrons ensuite un type de calcul lié aux graphes non uniformément étiquetés, les problèmes de type statistique comme par exemple la détection de l'étiquette majoritaire.

Le théorème 8.16 de ce chapitre est paru dans [GM02b] sous la forme de la caractérisation de l'équivalence des connaissances structurelles. Les parties concernant les tâches à terminaison implicite et les problèmes statistiques n'ont pas encore été soumises à la critique.

8.1 Tâches à Terminaison Implicite et Autostabilisation

Une fonction calculée sans détection de la terminaison a souvent une portée limitée, sauf dans le cas où la terminaison explicite n'est pas attendue mais plutôt une stabilisation du réseau. C'est le cas de l'autostabilisation. On va par conséquent étudier simultanément ces deux types de problèmes. Nous nous restreignons ici au cas où l'on cherche à stabiliser les nœuds dans un certain état, et non pas dans un certain comportement.

8.1.1 Définitions

On commence par définir quelques notations afin de distinguer les différentes partie de l'étiquette d'un sommet. On définit deux registres particuliers ENTREE et SORTIE pour, respectivement, y lire les étiquettes initiales (numéro d'identification éventuels, connaissance structurelle,..), et écrire les résultats du calcul distribué. Un troisième registre, CALCUL, servira pour le stockage des valeurs intermédiaires de calcul. L'étiquette d'un sommet a alors exactement trois composantes : ENTREE, SORTIE et CALCUL. On supposera que, pour cette section, $\mathcal{G}_L = \{(G, \text{ENTREE} \times \text{SORTIE} \times \text{CALCUL}) \mid G \in \mathcal{G}\}$. On note $\mathcal{G}_{\text{ENTREE}}$ (resp. $\mathcal{G}_{\text{SORTIE}}$) la sous-partie de $\mathcal{G}_L$ des graphes tels que les composantes SORTIE et CALCUL (resp. ENTREE et CALCUL) soient à $\perp$. Pour tout graphe $\mathbf{G} \in \mathcal{G}_L$, on notera $\text{ENTREE}(\mathbf{G})$, $\text{SORTIE}(\mathbf{G})$ et $\text{CALCUL}(\mathbf{G})$ le graphe de $\mathcal{G}_{\text{ENTREE}}$ obtenu à partir de $\mathbf{G}$ en mettant les registres SORTIE et CALCUL,ENTREE et CALCUL, ENTREE et SORTIE, respectivement, à $\perp$. Par extension, si v est un sommet de $\mathbf{G}$, $\text{SORTIE}(\mathbf{G})(v)$ dénote la valeur du registre SORTIE de l'étiquette de v.

Étant fixé une connaissance structurelle ι, on notera $\text{ENTREE}_\iota(\mathbf{G})$ le graphe obtenu à partir de $\mathbf{G}$ en remplaçant, pour tout sommet v, le registre $\text{ENTREE}(v)$ par $(\text{ENTREE}(v), \iota(\mathbf{G}))$, et les registres SORTIE et CALCUL par $\perp$.

On définit une tâche comme étant une fonction qui associe à tout graphe d'entrée un ensemble de configurations admissibles.

Définition 8.1. *Une* tâche *est une fonction* $T : \mathcal{G}_{\text{ENTREE}} \longrightarrow \mathcal{P}(\mathcal{G}_{\text{SORTIE}})$ *telle que pour tout graphe* $\mathbf{G}$*, pour tout graphe* $\mathbf{G}' \in T(\mathbf{G})$, $\text{ENTREE}(\mathbf{G}') = \text{ENTREE}(\mathbf{G})$.

Étant donnés deux graphes $\mathbf{G}$ et $\mathbf{H}$ tels que le graphe $\text{ENTREE}(\mathbf{G})$ soit revêtement de $\text{ENTREE}(\mathbf{H})$, on note $\uparrow(\mathbf{G}, \mathbf{H})$ le graphe $\mathbf{K}$ tel que $\text{ENTREE}(\mathbf{G}) = \text{ENTREE}(\mathbf{K})$,

$\text{CALCUL}(\mathbf{G}) = \text{CALCUL}(\mathbf{K})$ et $\text{SORTIE}(\mathbf{K})$ soit un revêtement de $\text{SORTIE}(\mathbf{H})$. En quelque sorte, $\mathbf{K}$ est obtenu en "relevant" sur $\mathbf{G}$ la composante SORTIE de $\mathbf{H}$.

Définition 8.2. *Une tâche $T : \mathcal{G}_{\text{ENTREE}} \longrightarrow \mathcal{P}(\mathcal{G}_{\text{SORTIE}})$ est* réalisable avec terminaison implicite *avec connaissance structurelle ι si il existe un système de réétiquetage de graphes $\mathcal{R}$ tel que pour tout graphe $\mathbf{G}$ de $\mathcal{G}_{\text{ENTREE}}$, pour tout graphe $\mathbf{K}$ de $\text{Irred}_{\mathcal{R}}(\text{ENTREE}_{\iota}(\mathbf{G}))$, $\text{SORTIE}(\mathbf{K}) \in T(\mathbf{G})$.*

Définition 8.3. *Une tâche $T : \mathcal{G}_{\text{ENTREE}} \longrightarrow \mathcal{P}(\mathcal{G}_{\text{SORTIE}})$ est* autostabilisante *si il existe un système de réétiquetage de graphes $\mathcal{R}$ tel que pour tout graphe $\mathbf{G}$ de $\mathcal{G}_L$, pour tout graphe $\mathbf{K}$ de $\text{Irred}_{\mathcal{R}}(\mathbf{G})$, $\text{SORTIE}(\mathbf{K}) \in T(\text{ENTREE}(\mathbf{G}))$.*

8.1.2 Sans connaissance structurelle

La méthode utilisée dans la partie Reconnaissance de Graphes s'étend très simplement, et on obtient :

Théorème 8.4. *Soit une tâche $T : \mathcal{G}_L \longrightarrow \mathcal{P}(\mathcal{G}_L)$. Alors les trois assertions suivantes sont équivalentes :*

8.4.i T est autostabilisante,

8.4.ii T est réalisable avec terminaison implicite sans connaissance structurelle,

8.4.iii Il existe une fonction (calculable) $t : \mathcal{G}_{\text{ENTREE}} \longrightarrow \mathcal{G}_L$ telle que pour tout graphe $\mathbf{G}$ et $\mathbf{H}$ de $\mathcal{G}_{\text{ENTREE}}$ tels que $\mathbf{G}$ soit revêtement de $\mathbf{H}$, $\text{SORTIE}(\uparrow(\mathbf{G}, t(\mathbf{H}))) \in T(\mathbf{G})$

Démonstration.

($i \Rightarrow ii$) Immédiat d'après les définitions. $\mathcal{R}$ terminant toujours dans $T(\mathbf{G})$ pour n'importe quel étiquetage initial, il y termine a fortiori si l'étiquetage initial est dans $\mathcal{G}_{\text{ENTREE}}$.

($ii \Rightarrow iii$) Notons $\mathcal{R}$ le système de réétiquetage de graphes réalisant T. On va conclure en utilisant de nouveau le Lemme 0.31 de relèvement. Soit $\mathbf{H}$ un graphe de $\mathcal{G}_{\text{ENTREE}}$. On effectue $\mathcal{R}$ sur $\mathbf{H}$ et on aboutit à un graphe irréductible $\mathbf{H}_f$. On pose $t(\mathbf{H}) = \mathbf{H}_f$.

Soit $\mathbf{G}$ est un revêtement de $\mathbf{H}$. Par lemme de relèvement $\mathbf{G}\mathcal{R}^* \uparrow(\mathbf{G}, \mathbf{H}_f)$. Comme $\mathbf{H}_f$ est irréductible, $\uparrow(\mathbf{G}, \mathbf{H}_f)$ l'est également. Comme $\mathcal{R}$ réalise T, $\text{SORTIE}(\uparrow(\mathbf{G}, t(\mathbf{H}))) \in T(\mathbf{G})$.

($iii \Rightarrow i$) On va procéder comme pour l'algorithme de reconnaissance $\mathcal{R}_{\mathcal{F}}$, page 61 mais en se basant cette fois sur $\mathcal{M}_{\text{a.s.}}$. On va adjoindre à celui-ci quelque règles de manière à utiliser la renormalisation des numéros calculés et de reconstruire $\mathbf{H}_M$. Se reporter à 3.1.4 pour les détails du calcul de cette renormalisation et reconstruction. L'étiquette SORTIE sera à tout moment réécrite en utilisant le relèvement de $t(\mathbf{H}_M)$.

Le système de réétiquetage de graphes sera défini comme suit :

- étiquettes : $(\text{ENTREE}, \text{SORTIE}, \text{CALCUL})$ où $\text{CALCUL} = (n, N, M)$,
- règles $\mathcal{R}_T^{\text{a.s.}}$:
  - les règles de $\mathcal{M}_{\text{a.s.}}$,

– on ajoute à chaque règle précédente le réétiquetage

$$\text{SORTIE}'(v) := \text{SORTIE}\left(t(\mathbf{H}_{M(v_0)})\right)(n(v_0)).$$

Vérifions que $\mathcal{R}_T^{\text{a.s.}}$ réalise bien T pour tout étiquetage initial. $\mathcal{M}_{\text{a.s.}}$ étant autostabilisant, $\mathcal{R}_T^{\text{a.s.}}$ se stabilise dans une configuration $\mathbf{G}_f$ telle que $\mathbf{G}_f$ est un revêtement de $\mathbf{H}_{M(v)}$ pour tout sommet v. Par réétiquetage du registre SORTIE, on a $\text{SORTIE}(\mathbf{G}_f) = \text{SORTIE}\,(\uparrow(\mathbf{G},\ t(\mathbf{H}_M)))$ et l'hypothèse (iii) assure alors que $\text{SORTIE}(\mathbf{G}_f) \in T(\mathbf{G})$. □

Remarque 8.5. Boldi et Vigna ont obtenu un résultat similaire sur l'équivalence des tâches réalisables et réalisables de manière autostabilisante [BV02b].

8.1.3 Avec connaissance structurelle

Si l'on ajoute une connaissance structurelle, on pose

$$\mathcal{G}_\iota = \{(\mathbf{H}, \iota(\mathbf{G})) \mid \mathbf{G} \in \mathcal{G}_{\text{ENTREE}} \text{ et } \mathbf{G} \text{ est un revêtement de } \mathbf{H} \in \mathcal{G}_{\text{ENTREE}}\}.$$

alors il vient la caractérisation suivante :

Théorème 8.6. *Une tâche $T : \mathcal{G}_L \longrightarrow \mathcal{P}(\mathcal{G}_L)$ est réalisable avec connaissance structurelle ι si et seulement si il existe une fonction (calculable) $t : \mathcal{G}_\iota \longrightarrow \mathcal{G}_L$, telle que, pour tout graphe $\mathbf{G}$ et $\mathbf{H}$ de $\mathcal{G}_{\text{ENTREE}}$ tels que $\mathbf{G}$ soit revêtement de $\mathbf{H}$,* $\text{SORTIE}(\uparrow(\mathbf{G}, t(\mathbf{H}, \iota(\mathbf{G})))) \in T(\mathbf{G})$.

Démonstration. La preuve de la condition nécessaire est identique à celle de $(ii) \Rightarrow (iii)$, ci-dessus.

Pour la condition suffisante on va utiliser une version légèrement modifiée de l'algorithme *PCarto*. Pour cela, nous avons besoin d'une fonction intermédiaire qui sera définie par un semi-algorithme. Celui-ci se terminera et sera correctement défini sur les entrées issues des valeurs calculées par *PCarto*. On se donne une énumération arbitraire de l'ensemble des graphes étiquetés $\mathcal{G}_L$ et on définit le semi-algorithme RECH.

Semi-Algorithme 2: $\underline{\text{RECH}}(\alpha, \mathbf{H}, s)$.

Entrée : $\alpha \in L$
$\mathbf{H} \in \mathcal{G}_L$
$s \in \mathbb{N}$
répéter
  / Énumérer tous les graphes étiquetés : */*
  obtenir $\mathbf{K}$ de l'énumération de $\mathcal{G}_L$
jusqu'à $\begin{cases} \iota(\mathbf{K}) = \alpha \\ \mathbf{K} \textit{ est un quasi-revêtement de } \mathbf{H} \textit{ de rayon } s. \end{cases}$ *(*)*
retourner $\mathbf{K}$

Informellement, le principe est de "geler" localement *PCarto* lorsqu'on constate que $\mathcal{M}$ *pourrait* être terminé, *i.e.* lorsque le rayon de confiance r_{conf} est supérieur à la taille de

$\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$. Lorsque tous les sommets sont "gelés", le graphe est sous forme irréductible, et $\mathcal{M}$ est effectivement terminé. Pour geler intentionnellement et provisoirement l'algorithme GSSP, on va l'effectuer en prenant comme entrée $(\mathbf{H}, d)$ où d sera une sorte de drapeau qui prendra, lorsque r_{conf} sera suffisamment grand, une valeur négative différente de celle de tous ses voisins, ceci afin d'"inhiber" l'application des règles de GSSP. Initialement $d(v) = 0$, pour tout v.

GEL : **Détection de la terminaison possible de $\mathcal{M}$ et "gel" de GSSP**

Condition préalable :

$-\ r_{\text{conf}}(v_0) \geq |\ \underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))\ |$,

Réétiquetage :

$-\ d'(v_0) := \min_{v \in B_{\mathbf{G}}(v_0)} (d(v)) - 1$

A chaque règle de $\mathcal{M}$ on ajoute le réétiquetage suivant :

Réétiquetage

- ...
- Pour tout $v \in B_{\mathbf{G}}(v_0)$, $d'(v) := 0$,
- $\text{SORTIE}'(v_0) := \begin{cases} \perp, & \text{si } r_{\text{conf}}(v_0) <| \underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))\ |, \\ \text{SORTIE}\left(t(\mathbf{H}_{M(v_0)}, \iota(\mathbf{G}))(n(v_0))\right), & \text{sinon.} \end{cases}$

Il faut remarquer tout d'abord, d'après le Lem. 7.13, que $\mathbf{G}_i$, avec les notations du lemme, satisfait à la contrainte de terminaison de la boucle de $\underline{\text{RECH}}$ (condition *(*)*). Par conséquent, $\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$ est toujours défini au cours de l'exécution. De plus, lorsque $r_{\text{conf}}(v_0) \geq |\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))|$, $\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$ est un revêtement de $\mathbf{H}(v_0)$ et $t(\mathbf{H}_{M(v_0)}, \iota(\mathbf{G}))$ est bien défini.

Ensuite, ce système de réécriture s'arrête. Pour cela, on remarque que par définition des règles gérant d, on a le lemme suivant.

Lemme 8.7. *Pour tout i, si $d_i(v_0) \neq 0$ alors $d_i(v_0) \neq d_i(v)$, pour tout $v \in B_{\mathbf{G}}(v_0)$.*

Comme le comportement de $\mathcal{M}$ est non modifié (on n'a ajouté aucune règle modifiant les valeurs de (n, N, M)), alors $\mathcal{M}$ termine, et par conséquent, à partir de cette étape là, $\mathbf{H}_M$ est constant et égal à un certain $\mathbf{H}_0$ sur tout le graphe et seul GSSP sur la fonction $\mathbf{H}, d)$ s'applique. Supposons que GSSP ne se termine pas. Comme, sur l'entrée $(\mathbf{H}, d)$, GSSP ne peut s'appliquer que si $d(v) = 0$, d'après le lemme précédent, cela signifie que $(\mathbf{H}, d)$ demeure constant à partir d'un certain pas. Par conséquent, r_{conf} croit vers l'infini sur tout sommet v. Comme tout revêtement est un quasi-revêtement de rayon arbitraire et que tout revêtement strict n'est pas, à partir d'un certain rayon, un revêtement, il vient qu'à partir d'un certain r_0, $\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_0)$ est constant (et égal au premier revêtement de $\mathbf{H}_0$ dans l'ordre d'énumération de $\mathcal{G}_L$). Par conséquent, la règle de gel finit par être applicable en un certain v_0, et par conséquent $d(v_0) \neq 0$. Contradiction.

Notons $\mathbf{G}_f$ la forme irréductible de $\mathbf{G}$ obtenue. Après terminaison de *PCarto*, $\mathbf{H}_M$ est uniforme, ainsi que r_{conf} (en tout sommet v, $r_{\text{conf}}(v)$ est le plus petit entier r solution

de $r \geq \underline{\text{RECH}}(l(v_0), \mathbf{H}_0, r)$). Par conséquent en posant $\mathbf{K} = \underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$, le graphe $\mathbf{K}$ est un revêtement de $\mathbf{H}$ et $\mathbf{K} \sim^{\iota} \mathbf{G}_f$. L'affectation de SORTIE est correcte par définition de t. □

Remarque 8.8. Il faut souligner que le point clef de la démonstration qui précède est l'assurance, en se fondant sur la valeur du rayon de confiance, d'effectuer le calcul de t sur une entrée valide. C'est en fait la résolution du problème équivalent que l'on avait rencontré dans la partie Reconnaissance au sujet de la non terminaison éventuelle du semi-algorithme APPART. Cette preuve fournit par conséquent une alternative "100%" réétiquetage de graphes à la caractérisation des familles de graphes reconnaissables, effectuée en faisant une hypothèse d'équité sur les règles "internes" et "externes", section 5.3.

Remarque 8.9. La caractérisation éventuelle des tâches autostabilisantes avec connaissance structurelle, si cela a un sens – il faut supposer que certains registres sont plus "incorruptibles" que d'autres – dépend de la réponse à la question ouverte de l'autostabilisation de GSSP. Car dans ce cas, ce qui pose alors problème c'est que l'on ne semble pas, *a priori*, pouvoir faire confiance aux rayons de confiance du point de vue de l'existence et de l'appartenance des paramètres au domaine de définition de t, c'est-à-dire à $\mathcal{G}_{\iota}$.

8.2 Fonctions Scalaires

8.2.1 Étiquettes terminales et fonctions scalaires

Pour rendre explicite la terminaison d'un calcul, nous allons utiliser des étiquettes spéciales qui n'apparaîtront sur un sommet que lors de son dernier réétiquetage. On suppose dans toute la suite que l'on se fixe un sous-ensemble L_F de l'ensemble des étiquettes L.

Définition 8.10. *On appelle* étiquette terminale *tout élément de* L_F.

Définition 8.11. *Une* fonction scalaire *de domaine de définition* $\mathcal{F}$ *est une fonction* $f : \mathcal{F} \longrightarrow L_F$.

Le domaine de définition $\mathcal{F}$ sera toujours supposé récursif. Une fonction scalaire sera calculable localement si il existe un système de réétiquetage de graphes, calculant cette fonction et où les étiquettes terminales n'apparaissent qu'en fin de calcul. Formellement,

Définition 8.12. *Une fonction scalaire* f *est* localement calculable *sur* $\mathcal{F}$ *si il existe un système de réétiquetage de graphes* $\mathcal{R}$ *tel que*

- $\mathcal{R}$ *est nœthérien sur* $\mathcal{F}$
- *Pour tout graphe* $\mathbf{G}$ *de* $\mathcal{F}$, $\mathit{Irred}_{\mathcal{R}}(\mathbf{G}) = (\mathbf{G}, \Lambda_{f(\mathbf{G})})$.
- *Pour tous graphes* $\mathbf{G} \in \mathcal{F}$ *et réétiquetages* $\mathbf{G}' = (\mathbf{G}, \lambda'), \mathbf{G}'' = (\mathbf{G}, \lambda'')$ *tels que* $\mathbf{G}\mathcal{R}^*\mathbf{G}'\mathcal{R}^*\mathbf{G}''$, *pour tout sommet* u, $\lambda'(u) \in L_F \Rightarrow \lambda''(u) = \lambda'(u)$.

La fonction f est localement calculable avec connaissance structurelle ι *si f est localement calculable sur* $\{(\mathbf{G}, \Lambda_{\iota(\mathbf{G})}) \mid \mathbf{G} \in \mathcal{F}\}$.

Il faut bien souligner que cette définition implique que le nœud "sait" que son étiquette ne sera plus modifiée, mais qu'en particulier, il "ne sait rien" sur la terminaison globale du calcul. Nous allons y revenir en détail.

Remarque 8.13. On commencera par remarquer que la définition de terminaison ainsi donnée, si elle interdit la modification de toute étiquette terminale, n'interdit pas sa lecture par les voisins du sommet ayant "terminé", ce qui peut paraître paradoxal. On pourrait en fait rajouter dans la définition la condition qu'aucun sommet étiqueté terminalement ne soit plus impliqué dans aucun réétiquetage. Cela ne reviendrait en fait qu'à compliquer la description des algorithmes, puisqu'on peut parfaitement supposer que chaque sommet se réétiquetant terminalement, transmet, avant arrêt complet de ses activités, cette étiquette à tous ses voisins qui l'ajoutent dans un registre dédié. Avant d'appliquer une règle, un sommet collecte donc les étiquettes de ses voisins n'ayant pas encore terminé et y ajoute les valeurs provenant du registre dédié au stockage des valeurs terminales des sommets voisins. Cela ne change donc pas le pouvoir d'expression et nous faisons par conséquent le choix de la définition la plus simple.

8.2.2 Terminaison explicite et détection de la terminaison globale

L'autre point important concernant cette définition est son aspect local : l'information de terminaison ne concerne que le sommet lui-même. Que se passe-t-il si l'on s'intéresse à la terminaison globale du calcul, c'est-à-dire à la détection du moment où le réseau a atteint une forme irréductible. De nombreuses recherches concernant la terminaison et la détection locale de celle-ci ont été réalisées depuis le début de l'étude de la puissance d'expression des systèmes distribués, voir par exemple [Tel00] pour une revue. En particulier, dans le cadre des systèmes de réétiquetage de graphes, cela fut étudié dans [MMW97], où la notion de quasi-revêtement fut pour la première fois introduite. Une condition nécessaire pour la détection de la terminaison y était donnée. Récemment la réciproque fut publiée dans [MT00]. Cependant le critère de détection locale de la terminaison globale donné dans ces articles est plus exigeant que le nôtre. Ce critère requiert qu'un nœud du réseau puisse détecter, avec des informations *locales* la terminaison *globale* du calcul. Par conséquent cette détection ne peut être réalisée sur un ensemble restreint de classes de graphes, les classes de graphes de rayon de quasi-revêtement borné. Pour ce qui nous concerne, nous utilisons un critère de terminaison plus faible, nous demandons que chaque nœud détecte le moment où son propre état de calcul demeurera inchangé. Quand cela a été détecté, le sommet peut par exemple démarrer un autre algorithme, en utilisant la valeur qui vient d'être calculée.

Pour souligner la différence entre ces deux variantes, on doit noter qu'avec le critère le plus fort, un corollaire de la caractérisation de [MT00] est qu'il ne peut y avoir d'algorithme distribué universel avec terminaison explicite qui, pour tout réseau, calcule le degré en chaque nœud. Avec notre critère de terminaison locale, le résultat est trivial : chaque nœud calcule la valeur désirée en comptant ses voisins. Le point fondamental est que, si l'on ne sait rien du réseau sous-jacent, le moment où tous les nœuds

ont effectué leur comptage ne peut être *localement* détecté. On retrouve cette différence également sur notre premier exemple $\textsc{Colo}_d$, page 7, de $(d+1)$coloration des graphes $d-$réguliers. Chaque sommet n'applique qu'une seule règle, et pourtant la terminaison globale ne peut être détectée. En fait, en règle générale, on ne peut même pas détecter non seulement la terminaison mais également le commencement global des réétiquetages sur tous les sommets.

8.2.3 Relations de quasi-simulation

Définition 8.14. *Pour toute fonction $f : \mathcal{G}_L \longrightarrow L$, on notera $\approx^f$ la relation d'équivalence définie par $\mathbf{G} \approx^f \mathbf{G}'$ si $f(\mathbf{G}) = f(\mathbf{G}')$.*

Sur le modèle de ce qui a été fait dans le chapitre sur la reconnaissance, on définit maintenant quelques relations entre graphes qui permettront de traduire le fait que certains calculs sur un réseau peuvent être simulés partiellement sur d'autres.

Définition 8.15. *Soit κ une connaissance structurelle, soit $r : \mathcal{G}_L \longrightarrow \mathbb{N}$ une fonction. Étant donnés deux graphes $\mathbf{G}$ et $\mathbf{G}'$, on dit que $\mathbf{G}\ \sigma_r^\kappa\ \mathbf{G}'$ si les deux conditions suivantes sont satisfaites :*

1. $\kappa(\mathbf{G}) = \kappa(\mathbf{G}')$
2. *il existe $\mathbf{H}$ tel que $\mathbf{G}$ et $\mathbf{G}'$ soit tout deux quasi-revêtements de $\mathbf{H}$ de rayon $r(G)$.*

On note $\sim_r^\kappa$ la relation d'équivalence définie par la fermeture symétrique et transitive de σ_r^κ.

8.2.4 Caractérisation

Théorème 8.16 (Fonctions scalaires localement calculables). *Soit ι une connaissance structurelle et f une fonction de domaine de définition $\mathcal{G}_L$. Alors il existe un système de réétiquetage de graphes calculant f en connaissant ι si et seulement si il existe une fonction récursive $r : \mathcal{G}_L \longrightarrow \mathbb{N}$, telle que $\sim_r^\iota \subset \approx^f$.*

Démonstration.
Condition Nécessaire : Notons $\mathcal{R}$ le système de réétiquetage calculant f connaissant ι.

Soit $\mathbf{G}$ un graphe. Soit $\mathcal{C}(\mathbf{G}) = \{\mathbf{H} | \mathbf{G}$ est un revêtement de $\mathbf{H}\}$. Comme $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})\mathcal{R}^*(\mathbf{G}, \Lambda_{f(\mathbf{G})})$, on obtient d'après le lemme de relèvement, que pour tout $\mathbf{H} \in \mathcal{C}(\mathbf{G})$, $(\mathbf{H}, \Lambda_{\iota(\mathbf{G})})\mathcal{R}^*(\mathbf{H}, \Lambda_{f(\mathbf{G})})$. Pour tout $\mathbf{H}$, on note $n_{\mathbf{H},\mathbf{G}}$ la longueur d'une telle chaîne de réétiquetage. On définit ensuite $r(\mathbf{G}) = 1 + \max_{\mathbf{H} \in \mathcal{C}(\mathbf{G})} (2 * n_{\mathbf{H},\mathbf{G}}, |V(\mathbf{G})|)$. Comme $\mathcal{C}(\mathbf{G})$ est fini, ce maximum est toujours défini.

Maintenant, considérons deux graphes $\mathbf{G}$ et $\mathbf{G}'$ tels que $\mathbf{G}\ \sigma_r^\iota\ \mathbf{G}'$. Il existe $\mathbf{H}$ tel que $\mathbf{G}$ et $\mathbf{G}'$ sont tout deux quasi-revêtements de $\mathbf{H}$ de rayon $r(\mathbf{G})$. Comme par construction $r(\mathbf{G}) > |V(\mathbf{G})|$, $\mathbf{G}$ est en fait un revêtement de $\mathbf{H}$. Ou, en d'autres mots, $\mathbf{H} \in \mathcal{C}(\mathbf{G})$. En effectuant $n_{\mathbf{H},\mathbf{G}}$ itérations du lemme de Quasi-Relèvement pour $\mathbf{G}'$ et $\mathbf{H}$, et pour la chaîne de réétiquetage associée à $n_{\mathbf{H},\mathbf{G}}$, on obtient qu'il existe Λ tel que $(\mathbf{G}', \Lambda_{\iota(\mathbf{G}')})\mathcal{R}^*(\mathbf{G}', \Lambda)$, le graphe $(\mathbf{G}', \Lambda_{\iota(\mathbf{G}')})$ étant un quasi-revêtement de $(\mathbf{H}, \Lambda_{f(\mathbf{G})})$ de rayon au moins 1. Par conséquent, l'étiquette $f(\mathbf{G})$ apparaît au moins sur un sommet de $(\mathbf{G}', \Lambda)$. Comme $\mathcal{R}$ calcule f, on en déduit que $f(\mathbf{G}') = f(\mathbf{G})$. Par conséquent

$\mathbf{G}\,\sigma_r^\iota\,\mathbf{G}' \Rightarrow \mathbf{G} \approx^f \mathbf{G}'$ pour tout $\mathbf{G}$ et $\mathbf{G}'$. Par fermeture symétrique et transitive, on obtient que $\sim_r^\iota \subset \approx^f$.

Condition Suffisante : On va utiliser *PCarto* pour calculer $f(\mathbf{G})$ en utilisant de nouveau RECH.

Le système de réétiquetage de graphes que nous allons utiliser est constitué des règles de *PCarto* auxquelles nous ajoutons les règles suivantes.

CALCUL : **Détection de la terminaison de *PCarto* et calcul de $f(\mathbf{G})$**

Condition préalable :

– $r_{\text{conf}}(v_0) \geq r(\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0)))$,

Réétiquetage :

– Tous les sommets de $B(v_0)$ prennent l'étiquette $f(\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$.

TERM : **Règle de diffusion terminale**

Condition préalable :

– Si il existe $v \in B(v_0)$ dont l'étiquette λ est telle que $\lambda \in L_F$.

Réétiquetage :

– Tous les sommets de $B(v_0)$ prennent l'étiquette λ.

Comme précédemment, $\underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$ est toujours défini au cours de l'exécution.

D'après le corollaire 7.15, comme $\mathcal{M}$ se termine, $\mathbf{H}$ devient constant et r_{conf} croit alors vers l'infini. Par conséquent si l'on note $\mathbf{G}_f$ le graphe final (pour $\mathcal{M}$), on a, d'après la Prop. 1.16 que $\mathbf{G}_f$ est une solution de *(*)* pour tout rayon $s \geq r(\mathbf{H})$. De plus, si l'on note r_0 le maximum de r sur l'ensemble des premiers graphes énumérés jusqu'à atteindre $\mathbf{G}_f$, on obtient que la condition préalable de 1.16 sera satisfaite si $r_{\text{conf}} = r_0$. Par conséquent, la règle CALCUL sera applicable à un certain moment et l'exécution du système se terminera.

Vérifier la correction est alors immédiat. On a, par construction, lorsque CALCUL est appliquée, en posant $\mathbf{K} = \underline{\text{RECH}}(l(v_0), \mathbf{H}(v_0), r_{\text{conf}}(v_0))$, que $\mathbf{K}$ est un quasi-revêtement de $\mathbf{H}$ de rayon $r_{\text{conf}}(v_0) \geq r(\mathbf{K})$, donc de rayon $r(\mathbf{K})$. Et d'après le Th. 7.13, $\mathbf{G}$ est également un quasi-revêtement de $\mathbf{H}$ de rayon $r(\mathbf{K})$. Par conséquent, $\mathbf{K}\ \sigma_r^\iota\ \mathbf{G}$, d'où par hypothèse, appliquée à $\mathbf{K}$, $f(\mathbf{G}) = f(\mathbf{K})$. □

Remarque 8.17. Il est important de bien noter que la condition énoncée ci-dessus *ne permet pas* de détecter la terminaison (globale) de l'exécution de l'algorithme de Mazurkiewicz sous-jacent. Cette condition nous permet juste de conclure quand l'exécution sous-jacente est "suffisamment avancée".

8.2.5 Une Application Simple : Reconnaissance avec Terminaison Explicite

Au cours de la partie Reconnaissance, nous avons fait remarquer que la définition donnée correspondait en fait à un calcul distribué avec terminaison *implicite*. Nous allons voir qu'il est simple d'étendre cette définition en exigeant cette fois une terminaison *explicite* et de donner une caractérisation des familles reconnaissables avec terminaison explicite en utilisant le théorème précédent.

Nous allons utiliser comme ensemble d'étiquettes terminales l'ensemble $L_F = \{\underline{\text{VRAI}}, \underline{\text{FAUX}}\}$, et une famille de graphes sera reconnue avec terminaison explicite s'il existe un système de reconnaissance, avec terminaison explicite relativement à L_F, de condition finale égale au singleton $\{\underline{\text{VRAI}}\}$.

Définition 8.18. *Une famille de graphes $\mathcal{F}$ est explicitement reconnaissable avec connaissance structurelle ι si il existe un système de réétiquetage de graphes $\mathcal{R}$ tel que :*

- *$\mathcal{R}$ est nœthérien sur $\{(\mathbf{G}, \Lambda_{\iota(\mathbf{G})}) \mid \mathbf{G} \in \mathcal{F}\}$,*
- *Pour tout graphe $\mathbf{G}$ de $\mathcal{F}$,*
  $Irred_{\mathcal{R}}(\mathbf{G}, \Lambda_{\iota(\mathbf{G})}) = \{(\mathbf{G}, \Lambda_{\underline{\text{VRAI}}})\}$ si et seulement si $\mathbf{G} \in \mathcal{F}$, et $Irred_{\mathcal{R}}(\mathbf{G}, \Lambda_{\iota(\mathbf{G})}) = \{(\mathbf{G}, \Lambda_{\underline{\text{FAUX}}})\}$ si et seulement si $\mathbf{G} \notin \mathcal{F}$;
- *Pour tous graphes $\mathbf{G} \in \mathcal{F}$ et réétiquetages $\mathbf{G}' = (\mathbf{G}, \lambda'), \mathbf{G}'' = (\mathbf{G}, \lambda'')$ tels que $(\mathbf{G}, \Lambda_{\iota(\mathbf{G})})\mathcal{R}^*\mathbf{G}'\mathcal{R}^*\mathbf{G}''$, pour tout sommet u, $\lambda'(u) \in L_F \Rightarrow \lambda''(u) = \lambda'(u)$.*

En d'autres termes,

Proposition 8.19. *Une famille de graphes $\mathcal{F}$ est explicitement reconnaissable avec connaissance structurelle ι si et seulement si la fonction caractéristique de $\mathcal{F}$, $\chi_{\mathcal{F}}$, est localement calculable avec connaissance structurelle ι.*

Par conséquent, d'après le théorème 8.16, on a la caractérisation suivante :

Théorème 8.20. *Une famille de graphes $\mathcal{F}$ est explicitement reconnaissable avec connaissance structurelle ι si et seulement si il existe une fonction récursive $r : \mathcal{F}_L \longrightarrow \mathbb{N}$ telle que $\mathcal{F}$ est fermée pour $\sim_r^\iota$.*

8.3 Équivalences des Connaissances Structurelles

Les algorithmes distribués font en général certaines hypothèses sur la structure du graphe sous-jacent (une topologie spécifique : le réseau est un arbre, est triangulé, est un anneau ; une certaine métrique : la taille, le diamètre, des bornes sur ces valeurs). Nous avons appelé connaissances structurelles de telles informations. Les algorithmes distribués peuvent, et en général le font, utiliser une connaissance structurelle spécifique [Tel00, Pel00]. Une question naturelle se pose alors : étant donné une certaine connaissance structurelle, est-il possible de calculer une autre connaissance structurelle. Cela peut-être utile, par exemple, si l'on souhaite exécuter un algorithme $\mathcal{A}$ dont le bon fonctionnement dépend de la connaissance explicite d'une certaine propriété

structurelle, alors que l'on en possède une autre. On définit des connaissances structurelles équivalentes comme étant des connaissances structurelles que l'on peut calculer de manière distribuée l'une connaissant l'autre. On présente dans cette partie, une caractérisation de cette équivalence.

Une étude sur le sujet de la comparaison des "affectations initiales", dans le modèle de Yamashita et Kameda, a été réalisée par Sakamoto [Sak99]. Certaines connaissances structurelles particulières y sont comparées, et il est montré principalement que celles-ci forme un treillis infini.

On s'aperçoit rapidement que déduire une connaissance structurelle d'une autre connaissance structurelle est directement relié à la détection de la terminaison d'algorithmes de reconnaissance. Par conséquent, de façon guère surprenante, l'équivalence sera exprimée en terme de fermeture pour certaines relations basée sur des quasi-revêtements.

8.3.1 Définition

Pour cette section, une *connaissance structurelle* est désormais une application récursive $\kappa : \mathcal{G}_L \longrightarrow L_F$.

Une connaissance structurelle sera déductible d'une autre si elle est localement calculable connaissant cette dernière. Pour distinguer les étiquettes initiales des étiquettes terminales, on suppose qu'il existe une copie, noté L_F^0 de L_F dans $L \backslash L_F$. Pour chaque étiquette terminale $\alpha \in L_F$, on note α^0 sa copie "initiale" dans $L \backslash L_F$. Cette manipulation est nécessaire pour ne pas se retrouver à faire débuter nos systèmes de réétiquetage de graphes sur des étiquettes terminales.

Définition 8.21. *Étant données deux connaissances structurelles ι et κ, on dit que κ est* déductible *de ι, ce qui est noté $\iota \vdash \kappa$, si il existe une fonction localement calculable qui calcule κ connaissant ι^0.*

On dit que ι et κ sont équivalents *si $\iota \vdash \kappa$ et $\kappa \vdash \iota$. On note $\kappa \approx \iota$ dans ce cas.*

8.3.2 Caractérisation

Nous établissons maintenant les théorèmes de déductibilité et d'équivalence. Le théorème suivant est la conséquence immédiate du théorème 8.16.

Théorème 8.22. *Soit κ et ι deux connaissances structurelles. Alors $\iota \vdash \kappa$ si et seulement si il existe une fonction calculable $r : \mathcal{G}_L \longrightarrow \mathbb{N}$, telle que $\sim_r^\iota \subset \sim_r^\kappa$.*

Démonstration. On remarque que $\sim_r^{\iota^0} = \sim_r^\iota$, et par théorème 8.16, il vient que $\sim_r^{\iota^0} \subset \approx^\kappa$. Par conséquent, pour tous graphes $\mathbf{G}$ et $\mathbf{G}'$, si $\mathbf{G}\ \sigma_r^\iota\ \mathbf{G}'$ alors $\kappa(\mathbf{G}) = \kappa(\mathbf{G}')$ et donc $\mathbf{G}\ \sigma_r^\kappa\ \mathbf{G}'$.

La réciproque provient de 8.16 et du fait que $\sim_r^\kappa \subset \approx^\kappa$. □

Le théorème suivant se déduit de la double application du précédent.

Théorème 8.23 (équivalence des connaissances structurelles). *Soit κ et ι deux connaissances structurelles. Alors $\kappa \approx \iota$ si et seulement si il existe une fonction calculable $r : \mathcal{G}_L \longrightarrow \mathbb{N}$, telle que $\sim_r^\iota = \sim_r^\kappa$.*

Étant donnée une connaissance structurelle κ et une fonction $r : \mathcal{G}_L \longrightarrow \mathbb{N}$, la relation $\sim_r^\kappa$ peut également être vue comme la fermeture de $\sigma_r^\varepsilon \cap \approx^\kappa$. On remarque alors que $\approx^\kappa$ ne dépend pas des valeurs *réelles* de $\kappa(\mathbf{G})$, ce qui n'est pas surprenant puisque changer les étiquettes de manière isomorphes amènent évidemment à une connaissance équivalente, il s'agit en fait de changer l'encodage de cette connaissance. Cela suggère d'étudier plus particulièrement les propriétés de σ_r^ε.

8.3.3 Applications

Le cas zéro connaissance

On commence par étudier les propriétés de $\sim_r^\varepsilon$. Le cas où l'on a aucune connaissance structurelle est assez surprenant puisque l'on a le résultat suivant

Proposition 8.24. *Soit $r : \mathcal{G}_L \to \mathbb{N}$. Pour tout graphe $\mathbf{G}$, il existe un arbre $\mathbf{T}$ tel que $\mathbf{G}\ \sigma_r^\varepsilon\ \mathbf{T}$.*

Démonstration. $\mathbf{T}$ est défini en tronquant le revêtement universel de $\mathbf{G}$ à une boule de rayon $r(\mathbf{G})$. □

Ceci amène certains résultats d'impossibilités

Proposition 8.25. $\varepsilon \nvdash \chi_{\mathcal{T}}$.

Cela signifie qu'il n'y a aucun moyen, sans aucune connaissance de savoir si le graphe sous-jacent possède un cycle ou non. De plus si $\mathbf{G}$ et $\mathbf{G}'$ ne sont pas des arbres, on peut choisir le même $\mathbf{T}$. En d'autres mots, pour tous graphes, qui ne soient pas des arbres, $\mathbf{G}$ et $\mathbf{G}'$, $\mathbf{G} \sim_r^\varepsilon \mathbf{G}'$. Pour une meilleure description, on va utiliser la définition suivante provenant de la théorie des automates d'arbres. Elle décrit les familles d'arbres présentant une certaine régularité.

Définition 8.26 ([CDG+99]). *Une famille* régulière[4]. *d'arbres est une famille telle qu'il existe un automate d'arbre qui reconnaît cette famille*

Une famille d'arbres $\mathcal{S}$ est *fortement irrégulière* s'il n'existe aucun ensemble régulier constitué de sous-boules d'arbres de $\mathcal{S}$. La famille suivante d'"*étoile*"$\{S_n, n \in \mathbb{N}\}$ est un exemple d'une telle famille.

$$\begin{aligned} V(S_n) &= \{0, 1, \ldots, n\} \\ E(S_n) &= \{\{0, i\} \mid i \in [1, n]\} \end{aligned}$$

[4]Le terme consacré en théorie des automates d'arbres est plutôt *reconnaissable*, on lui substitue le terme *régulier* pour qu'il n'y ait aucune confusion avec notre propre notion de reconnaissabilité de la Partie II

La famille suivante, où les sommets non feuilles ont tous des degrés différents en est également un exemple, de diamètre non borné cette fois-ci.

$$\begin{aligned} V(T_n) &= [1,n] \cup \{(i,j) \mid 1 \leq i \leq n, 1 \leq j \leq i\} \\ E(T_n) &= \{\{i,(i,j)\} \mid i \in [1,n], 1 \leq j \leq i\} \end{aligned}$$

Une conséquence du lemme de pompage, qui exprime la régularité sous-jacente, est la suivante :

Proposition 8.27. *Soit κ une connaissance structurelle. Alors $\kappa \equiv \varepsilon$ si et seulement si il existe une famille fortement irrégulière d'arbres $\mathcal{S}$ tel que, pour tout graphe $\mathbf{G}$ qui ne soit pas un arbre $[\mathbf{G}]_{\approx^\kappa} = \mathcal{G}_L \backslash \mathcal{S}$.*

Diamètre borné

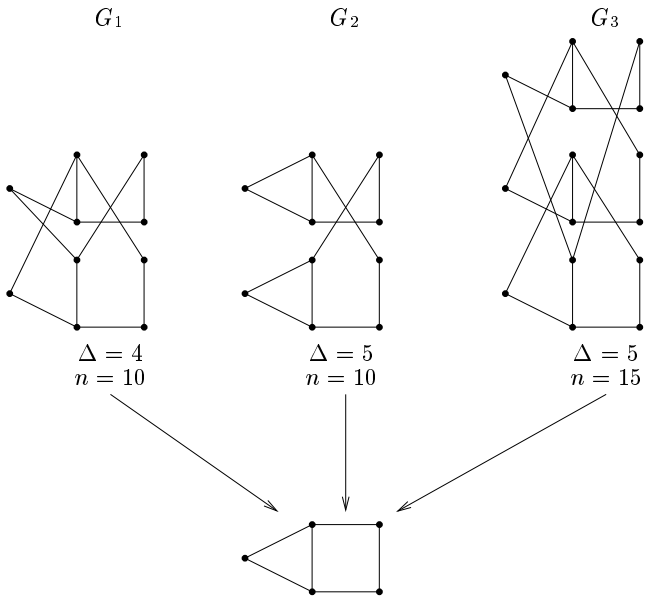


FIG. 8.1 – Diamètre et taille ne sont pas déductibles l'un de l'autre.

D'après la propriété fondamentale de *PCarto*, Si l'on connaît une borne sur le diamètre, on peut détecter la terminaison de $\mathcal{M}$. La difficulté provient alors du fait que le graphe reconstruit $\mathbf{G}_\rho$ n'est pas nécessairement isomorphe au graphe sous-jacent. Cette restriction provient exclusivement de l'aspect distribué des calculs. Mais l 'on peut noter que si l'on se restreint aux graphes minimaux (qui forme une famille assez large de graphes) cette connaissance permet de calculer précisément la topologie du graphe. Ce qui signifie en particulier que sur les graphes minimaux la hiérarchie induites par les connaissances structurelles s'achève très rapidement, et est *très* différente de celle des systèmes avec terminaison implicite.

Plus généralement, on remarque, par théorème de Reidemeister, que pour un graphe donné $\mathbf{H}$, il existe un nombre fini de revêtements avec un diamètre donné. En prenant

par conséquent pour $r(\mathbf{H})$, la taille maximale d'un revêtement de $\mathbf{H}$ et de diamètre inférieur à une certaine borne„ on voit que la connaissance d'une borne sur la taille est déductible, et par conséquent équivalente, à la connaissance d'une borne sur le diamètre. Cependant comme remarqué section 6.2, diamètre et taille ne sont pas déductibles l'une de l'autre. Reprenons la figure 6.6 en 8.1, où les trois graphes du haut sont revêtement de celui du bas. On a, pour toute fonction r, $G_1 \sim_r^{\text{TAILLE}} G_2$ et pas $G_1 \sim_r^{\Delta} G_2$. On a également $G_2 \sim_r^{\Delta} G_3$ et pas $G_2 \sim_r^{\text{TAILLE}} G_3$.

8.4 Statistique des Étiquettes

8.4.1 Consensus, Majorité et autres

Le problème auquel nous allons nous intéresser est de savoir s'il est possible de calculer certains aspects concernant la distribution des étiquettes. En d'autres termes, est-il possible de connaître, par exemple, le nombre d'étiquettes différentes, la valeur de l'étiquette la plus fréquente (problème de la Majorité), de choisir pour tous les sommets une même étiquette apparaissant parmi les étiquettes initiales (problème du Consensus), ou encore et plus généralement, est-il possible de calculer la densité des étiquettes c'est à dire leur fréquence d'apparition sur le graphe entier (problème du calcul des statistiques). Le problème de la majorité a été étudié, sous la forme d'un problème de décision, dans [LMS95b], où une solution a été proposée. Cette partie fournit une autre démonstration du résultat et répond aux questions ouvertes de cet article. La méthode est donnée dans toute sa généralité. Ces problèmes ont des applications en particulier dans le cadre des bases de données distribuées, lorsqu'il s'agit, par exemple, de restaurer la base après une défaillance du système.

Pour la suite on se fixe une famille $\mathcal{F}$ de graphes étiquetés, et une famille particulière d'étiquettes X. On note $\mathcal{F}_X$ l'ensemble des graphes de $\mathcal{F}$ "surétiquetés" par des étiquettes de X,

$$\mathcal{F}_X = \{(\mathbf{G}, \lambda) \mid \mathbf{G} \in \mathcal{F},\ \lambda : \mathbf{G} \longrightarrow X\}$$

Pour un graphe $\mathbf{G} \in \mathcal{F}$, on conviendra de noter $\overline{\mathbf{G}}$ un graphe de $\mathcal{F}_X$ de graphe sous-jacent $\mathbf{G}$.

Définition 8.28. *Soit $\overline{\mathbf{G}} = (\mathbf{G}, \lambda)$ un graphe de $\mathcal{F}_X$, soit $\alpha \in X$. On appelle* densité *de α dans $\mathbf{G}$, notée $f_\alpha(\overline{\mathbf{G}})$ le rapport*

$$f_\alpha(\overline{\mathbf{G}}) = \frac{\text{card}(\lambda^{-1}(\alpha))}{\text{card}(V(\mathbf{G}))}$$

Définition 8.29. *On appelle* fonction de majorité *toute fonction $\underline{Maj} : \mathcal{F}_X \longrightarrow X$ telle que, pour tout $(\mathbf{G}, \lambda) \in \mathcal{F}_X$, pour toute étiquette $\alpha \in X$, $f_\alpha(\overline{\mathbf{G}}) \leq f_{\underline{Maj}(\overline{\mathbf{G}})}(\overline{\mathbf{G}})$.*

Définition 8.30. *On appelle* fonction de consensus *toute fonction $\underline{Consensus} : \mathcal{F}_X \longrightarrow X$ telle que pour tout graphe $(\mathbf{G}, \lambda)$, $\underline{Consensus}(\mathbf{G}, \lambda) \in \lambda(\mathbf{G})$.*

Définition 8.31. *Le problème* de la Majorité *(resp.* du Consensus*) admet une solution sur $\mathcal{F}$ si une fonction* $\underline{Maj}$ *(resp.* $\underline{Consensus}$*) est localement calculable.*

Remarque 8.32. Il est à noter que les définitions des problèmes sont effectuées sous forme fonctionnelle plutôt que relationnelle ce qui n'est pas restrictif, tant que l'on peut supposer que les étiquettes sont ordonnées.

Ces problèmes dépendent de la "statistique des étiquettes" du graphe :

Définition 8.33. *On appelle* statistique *des graphes la fonction* $\underline{Stat}$ *définie par*

$$\begin{array}{rccl} \underline{Stat}: & \mathcal{F}_X & \longrightarrow & [0,1]^X \\ & \overline{\mathbf{G}} & \longmapsto & \alpha \mapsto f_\alpha(\overline{\mathbf{G}}) \end{array}$$

Une fonction $f : \mathcal{F}_X \longrightarrow Y$ est dite statistique *si il existe une fonction $g : [0,1]^X \longrightarrow Y$ telle que $f = g \circ \underline{Stat}$. Un problème est dit* statistique *sur $\mathcal{F}$ s'il correspond à l'existence d'un système de réétiquetage de graphes $\mathcal{R}$ calculant une fonction statistique sur $\mathcal{F}_X$.*

Les problèmes du Consensus ou de la Majorité sont statistiques.

Une propriété très intéressante des revêtements est qu'ils conservent la statistique.

Lemme 8.34. *Soit $\gamma : \mathbf{G} \longrightarrow \mathbf{H}$ un revêtement. Alors pour tout surétiquetage $\overline{\mathbf{H}}$ de $\mathbf{H}$, si $\overline{\mathbf{G}}$ est le graphe obtenu par relèvement depuis $\overline{\mathbf{H}}$, alors*

$$\underline{Stat}(\overline{\mathbf{G}}) = \underline{Stat}(\overline{\mathbf{H}}).$$

Démonstration. La propriété remarquable des revêtements étant l'égalité des cardinaux des ensembles de préimages, corollaire 0.27, il vient qu'il existe q, le nombre de feuillets, tels que, pour tout sommet u de $\overline{\mathbf{H}}$, $\mathrm{card}(\gamma^{-1}(u)) = q$. Par conséquent, pour toute étiquette α, par simplification du numérateur et dénominateur, $f_\alpha(\overline{\mathbf{G}}) = f_\alpha(\overline{\mathbf{H}})$. □

8.4.2 Caractérisation des fonctions statistiques

Nous allons montrer, qu'en général, la solvabilité d'un problème statistique est équivalente à celle du calcul de $\underline{Stat}$. Nous allons devoir distinguer les fonctions dépendant "réellement" de la statistique de celle qui dépendent en fait de la structure du graphe sous-jacent.

Définition 8.35. *Une fonction statistique f est dite* purement statistique *si pour tout graphe $\mathbf{G}$, il existe deux surétiquetage $\overline{\mathbf{G}}$ et $\overline{\mathbf{G}'}$ tels que $f(\overline{\mathbf{G}}) \neq f(\overline{\mathbf{G}'})$.*

Outre les fonctions constantes, on peut également trouver des fonctions non purement statistiques telle que celle qui, sur la famille des anneaux de taille non divisible par 5 réunie avec l'étoile à 4 sommets S_4, renvoie 1 si toutes les étiquettes sont de même densité 1/5, et 0 sinon.

Théorème 8.36. *Soit $\mathcal{F}$ une famille de graphes, et f une fonction purement statistique. Les trois assertions suivantes sont équivalentes :*

8.36.i $\underline{Stat}$ *est calculable sur $\mathcal{F}$.*

8.36.ii f *est calculable sur* $\mathcal{F}$.

8.36.iii Il existe une fonction calculable $r : \mathcal{F} \longrightarrow \mathbb{N}$ *telle que tout graphe* $\mathbf{G} \in \mathcal{F}$ *n'admette aucun quasi-revêtement strict de rayon* $r(\mathbf{G})$ *sur* $\mathcal{F}$.

Démonstration.

$(i \Rightarrow ii)$. f est statistique.

Pour la suite, $\mathcal{F}$ et X étant fixé, on simplifiera, pour toute fonction $r : \mathcal{F} \longrightarrow \mathbb{N}$ les notations $\sigma_r^{\chi_{\mathcal{F}_X}}$ et $\sim_r^{\chi_{\mathcal{F}_X}}$ en σ_r et $\sim_r$.

$(ii \Rightarrow iii)$. D'après le théorème 8.16, f est calculable sur $\mathcal{F}_X$ implique qu'il existe $r : \mathcal{F}_X \longrightarrow \mathbb{N}$ tel que $\sim_r \subset \approx^f$. Montrons par l'absurde que cela implique bien ce qui est recherché. Supposons donc qu'il existe un graphe $\mathbf{G}$ tel que pour tout s, il existe $\mathbf{K}$ tel que $\mathbf{K}$ soit quasi-revêtement strict de $\mathbf{G}$ de rayon s. Comme f est purement statistique, il existe $\overline{\mathbf{G}}_1$ et $\overline{\mathbf{G}}_2$, de graphe sous-jacent $\mathbf{G}$, tel que $f(\overline{\mathbf{G}}_1) \neq f(\overline{\mathbf{G}}_2)$. Prenons $s = 1 + 2(r(\overline{\mathbf{G}}_1) + r(\overline{\mathbf{G}}_2))$, il existe $\mathbf{K}$ tel que $\mathbf{K}$ soit quasi-revêtement strict de $\mathbf{G}$ de rayon s_0 via γ. On va définir $\overline{\mathbf{K}}$ tel que $\overline{\mathbf{G}}_1 \sigma_r \overline{\mathbf{K}}$ et $\overline{\mathbf{G}}_2 \sigma_r \overline{\mathbf{K}}$.

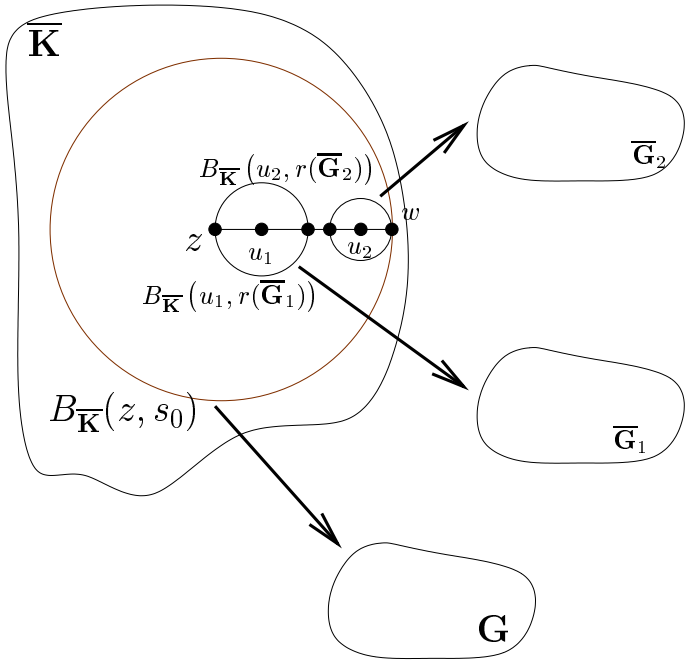


FIG. 8.2 – Le graphe $\overline{\mathbf{K}}$ est quasi-revêtement de $\overline{\mathbf{G}}_1$ (resp. $\overline{\mathbf{G}}_2$) de rayon $r(\overline{\mathbf{G}}_1)$ (resp. $r(\overline{\mathbf{G}}_2)$).

Comme $\mathbf{K}$ est strict, il existe un chemin de longueur s_0 du centre z du quasi-revêtement à un sommet w à distance exactement s_0 de z. Soit u_1 (resp. u_2) le sommet à distance $r(\overline{\mathbf{G}}_1)$ (resp. $1 + 2r(\overline{\mathbf{G}}_1) + r(\overline{\mathbf{G}}_2)$) de z sur ce chemin. Les boules $B_{\mathbf{K}}(u_1, r(\overline{\mathbf{G}}_1))$ et $B_{\mathbf{K}}(u_2, r(\overline{\mathbf{G}}_2))$ sont disjointes et incluses dans $B_{\mathbf{K}}(z, s_0)$. On définit $\overline{\mathbf{K}}$ comme étant le graphe de $\mathcal{F}_X$ obtenu en relevant par $_g amma$ l'étiquetage de $\overline{\mathbf{G}}_1$ (resp. de $\overline{\mathbf{G}}_2$) dans $B_{\mathbf{K}}(u_1, r(\overline{\mathbf{G}}_1))$ (resp. $B_{\mathbf{K}}(u_2, r(\overline{\mathbf{G}}_2))$), et en complétant pour les autres sommets par un étiquetage quelconque. Alors $\overline{\mathbf{K}}$ est un quasi-revêtement de $\overline{\mathbf{G}}_1$ et $\overline{\mathbf{G}}_2$ de rayon $r(\overline{\mathbf{G}}_1)$ et

$r(\overline{\mathbf{G}}_2)$ respectivement. Par conséquent, $\overline{\mathbf{G}}_1 \sigma_r \overline{\mathbf{K}}$ et $\overline{\mathbf{G}}_2 \sigma_r \overline{\mathbf{K}}$. Donc $\overline{\mathbf{G}}_1 \sim_r \overline{\mathbf{G}}_2$. Or $f(\overline{\mathbf{G}}_1) \neq f(\overline{\mathbf{G}}_2)$, d'où $\sim_r \not\subseteq \approx^f$, contradiction.

$(iii \Rightarrow i)$. On définit $\overline{r}$ sur $\mathcal{F}_X$, pour tout graphe $\overline{\mathbf{G}}$ de $\mathcal{F}_X$ de sous-jacent $\mathbf{G}$, par $\overline{r}(\overline{\mathbf{G}}) = \max\{r(\mathbf{H}) \mid \mathbf{G} \text{ est un revêtement de } \mathbf{H}\}$, l'ensemble étant fini le maximum est toujours défini. Il vient alors $\sim_{\overline{r}} \subset \approx^{\underline{\text{Stat}}}$. En effet, soit $\overline{\mathbf{G}}$ et $\overline{\mathbf{G}'}$ deux graphes tels que $\overline{\mathbf{G}} \sigma_{\overline{r}} \overline{\mathbf{G}'}$. Par hypothèse, $\overline{\mathbf{G}}$ et $\overline{\mathbf{G}'}$ sont deux quasi-revêtements de rayon $\overline{r}(\overline{\mathbf{G}})$ d'un même graphe $\overline{\mathbf{H}}$. Par définition de $\overline{r}$, $\overline{r}(\overline{\mathbf{G}}) \geq r(\mathbf{H})$, et $\overline{\mathbf{G}}$ et $\overline{\mathbf{G}'}$ sont donc, par hypothèse, des quasi-revêtements non stricts. En d'autres termes, ce sont des revêtements de $\overline{\mathbf{H}}$. D'après le lemme 8.34, il vient $\underline{\text{Stat}}(\overline{\mathbf{G}}) = \underline{\text{Stat}}(\overline{\mathbf{H}})$ et $\underline{\text{Stat}}(\overline{\mathbf{G}'}) = \underline{\text{Stat}}(\overline{\mathbf{H}})$. Par conséquent $\overline{\mathbf{G}} \approx^{\underline{\text{Stat}}} \overline{\mathbf{G}'}$, et $\underline{\text{Stat}}$ est calculable par Th. 8.16. □

Conclusions et Perspectives ...

La motivation première de cette thèse était l'étude du modèle des réétiquetages de graphes en lui-même, en essayant de déterminer exactement la puissance d'expression de ce modèle. Cette étude avait donc pour but d'établir si ce modèle pouvait être utile pour l'étude des algorithmes distribués. Les résultats très détaillés que nous avons obtenus ont permis, outre d'accroître la connaissance du modèle des réétiquetages de graphes, de montrer toute la pertinence de ce modèle. Par conséquent, les résultats obtenus concernant des problèmes très spécifiques à l'algorithmique distribuée et n'ayant pas, à notre connaissance, d'équivalents dans d'autres modèles, sont particulièrement précieux.

La recherche des limites à la puissance d'expression passe en particulier par l'étude de ce qui est impossible à réaliser dans le modèle. Nous avons donc entamé notre étude par les systèmes de reconnaissances de graphes puisque ceux-ci paraissaient isoler assez simplement certaines caractéristiques des calculs distribués et qu'un certain nombre de théorèmes d'impossibilités étaient déjà connus à leur sujet. En interprétant dans notre modèle un algorithme, extrêmement élégant, d'A. Mazurkiewicz, nous avons pu démontrer la réciproque de ces résultats d'impossibilités connus auparavant. Les théorèmes ont permis de montrer l'importance des similarités structurelles entre réseau. Un point fondamental de cette étude est qu'elle a pu être menée en supposant disponible pour l'algorithme une connaissance structurelle *arbitraire*, ce qui n'est pas le cas, par exemple, dans le modèle de Yamashita et Kameda.

C'est en tentant ensuite d'étudier un problème fondamental de l'algorithmique distribuée, l'élection d'un nœud dans un réseau, et en tentant, là encore, de compléter des résultats d'impossibilités déjà connus, que nous avons été amené à raffiner l'algorithme de Mazurkiewicz. Plus précisément, en nous appuyant sur la notion de quasi-revêtement développée pour la caractérisation de certains cas où l'élection n'était pas possible, nous avons pu approfondir la connaissance de l'algorithme de Mazurkiewicz. Nous avons donc établi que son comportement dynamique pouvait être décrit à l'aide de quasi-revêtements et d'un compteur très simple provenant d'un algorithme de Shi, Szymanski et Prywes. Cet algorithme, que nous avons dénommé algorithme de cartographie sous sa forme complète, nous a ensuite permis de réaliser tout ce qui n'était pas prouvé irréalisable. En effet, on doit souligner que cet algorithme permet de réaliser *autant que cela est possible* les opérations qui semblent préalables – du fait de l'absence *a priori* de centralisation – à tout algorithme distribué : rupture de la symétrie initiale (dont l'élection est un exemple clé) et collecte d'information concernant le graphe sous-jacent (comme le font les systèmes de reconnaissance). En conclusion, la

compréhension du domaine est grandement amélioré puisque les résultats de possibilités et d'impossibilités s'éclairent ainsi mutuellement.

Les théorèmes d'impossibilités, basés sur le lemme de Relèvement, ont été, grâce à notre approfondissement de la connaissance des propriétés de l'algorithme d'énumération de A. Mazurkiewicz, transformés en condition nécessaire et suffisante d'impossibilités, puis en théorèmes de possibilités. On peut détailler précisément ce qui est réalisable par calculs locaux et de nombreux exemples ont pu être donnés, montrant que les caractérisations sont bien pratiquables. En outre, certaines caractérisations sont similaires à celles données par exemple dans le modèle Yamashita et Kameda et Boldi et Vigna (équivalence des tâches à terminaison implicite et autostabilisantes, équivalence de la connaissance de borne sur la taille ou le diamètre) ce qui tend à renforcer la crédibilité du modèle des réétiquetages de graphes : on n'y prouve / trouve pas d'aberrations. Celui-ci fournit donc une alternative très crédible au modèle de Yamashita et Kameda.

Nous avons également pu, dans ce cadre, apporter de nouvelles idées concernant la résolution de certains problèmes fondamentaux de l'algorithmique distribuée. Nous avons ainsi caractérisé complètement (c'est-à-dire description complète des cas n'admettant pas de solution, et solution effective lorsqu'elle existe) le problème de l'élection, En particulier, le théorème 7.27 permet de dire quelles sont exactement les connaissances sur le réseau sous-jacent qui permettent d'élire. Ceci n'avait pas encore été fait dans d'autres modèles. En effet, dans le modèle de Yamashita et Kameda [YK96b, YK96c], le problème de l'élection a été étudié et une caractérisation presque complète a été réalisée dans [BCG$^+$96]. Le cas où l'on n'a aucune connaissance y est partiellement traité, sans doute parce qu'il manque un résultat d'impossibilité du type Th. 7.6. On peut raisonnablement espérer que, de même que les idées d'Angluin s'adaptent aisément dans la plupart des modèles, l'idée de "simulation bornée", formalisée ici avec les quasi-revêtements, puisse être utilisée dans de nombreux modèles et en particulier dans le modèle de Boldi et Vigna, sous un formalisme à déterminer.

Les perspectives de recherche sont encore nombreuses et on peut commencer par noter qu'un aspect faisant peut-être défaut est l'intégration des algorithmes probabilistes dans notre cadre. Cependant, avant toute tentative de modélisation, il faut sans doute garder à l'esprit qu'une des particularités des algorithmes distribués asynchrones est d'avoir une exécution non déterministe même si la description de l'algorithme ne fait intervenir aucun calcul aléatoire. Un algorithme distribué peut donc être déterministe en un certain sens mais également aléatoire en un autre. Un préalable est donc très certainement une évaluation très précise du taux d'échec dans l'assignation de numéros distincts par l'algorithme de Mazurkiewicz sur des réseaux non-minimaux.

Concernant l'étude et la caractérisation des tâches avec terminaison explicite (locale ou globale), il reste encore un peu de chemin à faire pour aboutir. Les outils possibles sont sans doute encore *PCarto* et les revêtements et quasi-revêtements, mais cela ne permet pas de conclure immédiatement. Il semble qu'il faille cette fois faire intervenir une relation en V faisant intervenir revêtement et quasi-revêtement. Une étude combinatoire de cette configuration est sans doute à mener afin d'unifier les caractérisations

et d'éviter des présentations différentes mais qui se révéleraient équivalentes.

Une autre direction de recherche future possible est l'étude des réécritures pour d'autres formes de règles : par exemple lecture de l'état de tous les voisins et réécriture du seul centre, ou encore lecture et écriture d'un sommet et d'un seul de ses voisins. Ce dernier exemple se rapproche davantage d'un modèle de communications par passage de messages et il semble intéressant d'étudier si, et sous quelle forme, peut-on adapter l'algorithme de Mazurkiewicz. En effet, une différence notable entre ces deux modèles principaux est que dans le cadre Yamashita et Kameda, l'algorithme fondamental ne se termine que si l'on connaît une borne sur la taille ou le diamètre du graphe, car il est essentiellement basé sur la construction du revêtement universel, arbre infini. En revanche, l'algorithme de Mazurkiewicz termine toujours, même si l'on ne connaît rien du réseau sous-jacent, et par conséquent il nous a été possible de donner des caractérisations combinatoires pour des connaissances structurelles arbitraires. Il parait donc très intéressant de pouvoir adapter $\mathcal{M}$ dans le cadre, ou dans un cadre proche, du passage de messages. Il est à noter que [BV02a] contient une étude catégorique des morphismes de graphes qui semblent être la base des preuves de réalisabilité en algorithmique distribuée, que ce soit dans le modèle de Boldi et Vigna ou dans le cadre des réétiquetages de graphes. Ainsi, des propriétés fondamentales des revêtements, et plus généralement des fibrations (extensions des revêtements aux graphes orientés), y sont produites.

Un aspect très important de l'algorithmique distribuée est cependant absent de ce panorama. Il s'agit de l'étude de la robustesse des algorithmes en cas de défaillances dynamiques d'un nœud (crash). Si l'aspect auto-stabilisation a été étudié relativement en détail, sa résolution tient au fait que l'algorithme de Mazurkiewicz est *très* robuste, il rompt autant que possible la symétrie de la situation initiale, quelle qu'elle soit. Ce qui est moins évident, c'est son comportement en cas de défaillance dynamique au cours du déroulement de l'algorithme. Y a-t-il reconstruction locale de la nouvelle topologie et propagation cette nouvelle configuration ? Ces aspects n'ont pas encore été abordés dans le cadre des réécritures de graphes et c'est une lacune importante qui peut apparaître difficile à intégrer tant les techniques utilisées (lemme de Relèvement et Quasi-Relèvement) paraissent fondées en grande partie sur le fait que le graphe sous-jacent n'est jamais modifié. Cependant, des techniques récentes de [HS99] permettent d'envisager une voie d'exploration. La méthode utilisée consiste à "coder" les systèmes de transition, là dans le cadre des systèmes concurrents, sous la forme extrêmement "compacte" de structures géométriques de grande dimension : les complexes simpliciaux. Les réécritures de graphes peuvent être vues entre autres comme un système de transition pour l'état global constitué de l'ensemble des états des processeurs : dans ce cadre, le système passe d'un état au suivant en appliquant les règles de réécritures qui définissent les transitions légales du système. L'évolution du système est donc décrite par une suite de configuration globales. L'idée de [HS99] est de remarquer que, puisque les transitions du système consistent en des modifications *locales* des états des processeurs, pour étudier l'évolution de la configuration globale, on ne va modifier que le strict minimum, c'est-à-dire uniquement le processeur changeant d'état, et l'on ne va pas "dupliquer" ce qui n'a pas été modifié. Une configuration globale est

un simplexe de dimension n (intuitivement un graphe complet à n sommets) et une transition entre deux configurations est décrite par le recollement des deux simplexes correspondants sur la face définie par les $n-1$ sommets dont l'état est inchangé. L'ensemble des transitions légales est décrit par un ensemble de simplexes ainsi recollés : un complexe simplicial. Une défaillance est modélisée par le passage d'un certain simplexe de dimension n à une de ces faces (de dimension $n-1$), celle qui contient les processeurs n'ayant pas crashé. Une exécution avec défaillances éventuelles est alors un parcours, continu en un certain sens, de simplexe en simplexe dans le complexe simplicial. L'étude de la possibilité et de la décidabilité de certains problèmes (en particulier celui du Consensus) se ramène à étudier les groupes d'homologie d'un "équivalent géométrique", sans préciser davantage, du complexe simplicial dans l'espace euclidien $\mathbb{R}^n$.

En ce qui concerne les systèmes de réécritures, il paraît judicieux de se demander ce qui se passe en termes géométriques si la cellule de base n'est plus, comme précédemment un simplexe - un graphe complet - mais un graphe quelconque (on parle dans ce cas de complexe cellulaire [Hat01]).

Table des Systèmes de Réétiquetage

Bibliographie

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.

[BCG+96] Paolo Boldi, Bruno Codenotti, Peter Gemmell, Shella Shammah, Janos Simon, and Sebastiano Vigna. Symmetry breaking in anonymous networks : Characterizations. In *Proc. 4th Israeli Symposium on Theory of Computing and Systems*, pages 16–26. IEEE Press, 1996.

[Ber83] C. Berge. *Graphes*. Gauthier Villars, 1983.

[BGM+01] M. Bauderon, S. Gruner, Y. Métivier, M. Mosbah, and A. Sellami. Visualization of distributed algorithms based on labeled rewriting systems. In *Second International Workshop on Graph Transformation and Visual Modeling Techniques, Crete, Greece*, volume 50 of *ENTCS*, pages 229–239, 2001.

[Bot97] A. Bottreau. *Réécritures de graphe et calculs distribués*. PhD thesis, Université Bordeaux I, 1997.

[BV01] Paolo Boldi and Sebastiano Vigna. An effective characterization of computability in anonymous networks. In Jennifer L. Welch, editor, *Distributed Computing. 15th International Conference, DISC 2001*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 2001.

[BV02a] Paolo Boldi and Sebastiano Vigna. Fibrations of graphs. *Discrete Math.*, 243, 2002.

[BV02b] Paolo Boldi and Sebastiano Vigna. Universal dynamic synchronous self-stabilization. *Distr. Computing*, (15), 2002.

[CDG+99] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D.lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. 1999.

[CGS95] B. Codenotti, P. Gemmel, and J. Simon. Symmetry breaking in anonymous networks. 1995.

[CL93] R. Cori and D. Lascar. *Logique mathématique II. Fonctions récursives, théorème de Gödel, théorie des ensembles, théorie des modèles*. Masson, 1993.

[CM94] B. Courcelle and Y. Métivier. Coverings and minors : Applications to local computations in graphs. *Europ. Journal of Combinatorics*, 15 :127–138, 1994.

[Dem98] P. Dembinski. Enumeration protocol in estelle : an exercise in stepwise development. In E. Najm S. Budkowski, A. Cavalli, editor, *Formal Description Techniques and Protocol Specification, Testing and Verification*, pages 147–162. Kluwer Academic Publishers, 1998.

[DFPP88] H. De Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fáry embeddings of planar graphs. In *Proc. of the 20th Ann. ACM Symp. Theory of Computing, Chicago, 1988*, pages 426–433. ACM Press, 1988.

[Dij74] E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *CACM*, 17(11) :643–644, 1974.

[Dol00] Schlomi Dolev. *Self-Stabilization*. MIT Press, 2000.

[DZ96] F. Demichelis and W. Zielonka. Decidability questions for graph k-coverings. *Technical Report, LaBRI*, 1996.

[Fel89] M. R. Fellows. The Robertson-Seymour theorems : a survey of applications. *Contemp. Math.*, 89 :1–18, 1989.

[Gib85] A. Gibbons. *Algorithmic graph theory*. Cambridge University Press, 1985.

[GM01] E. Godard and Y. Métivier. Characterization of classes of graphs recognizable by local computations with initial knowledge (*extended abstract*). In *SIROCCO 01 - 8th International Colloquium on Structural Information & Communication Complexity*, number 11 in Proceedings in Informatics, pages 179–194. Carleton Scientific, 2001.

[GM02a] E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible (*ext. abstract*). In M. Nielsen and U. Engberg, editors, *Proc. of Foundations of Software Science and Computation Structures, FOSSACS'02*, number 2303 in LNCS, pages 159–171. Springer-Verlag, 2002.

[GM02b] E. Godard and Y. Métivier. Equivalence of structural knowledges in distributed algorithms. In *SIROCCO 02 - 9th International Colloquium on Structural Information & Communication Complexity*, Proceedings in Informatics. Carleton Scientific, 2002. to appear.

[GMM00] E. Godard, Y. Métivier, and A. Muscholl. The power of local computations in graphs with initial knowledge. In *Proc. of Th. and App. of Graph Transformations'98*, number 1764 in LNCS, pages 71–84. Springer, 2000.

[GMM02] E. Godard, Y. Métivier, and A. Muscholl. Characterization of classes of graphs recognizable by local computations. *Theory of Computer Systems*, 2002. to appear.

[GMMW01] S. Gruner, Y. Métivier, M. Mosbah, and P.-A. Wacrenier. Distributed Algorithm for Computing a Spanning Tree in Anonymous T-prime Graphs. In *OPODIS'2001, International Conference On Principles of DIstributed Systems, Manzanillo, Mexico*, 2001.

[God02] E. Godard. A self-stabilizing enumeration algorithm. *Information Processing Letters*, 82(6) :299–305, 2002.

[Hat01] A. Hatcher. Algebraic topology, 2001.

[HS99] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. 1999.

[JS85] R.E. Johnson and F.B. Schneider. Symmetry and similarities in distributed systems. In *Proc. 4th conf. on Principles of Distributed Computing*, pages 13–22, 1985.

[Lav95] C. Lavault. *Evaluation des algorithmes distribués.* Hermès, Paris, 1995.

[Lei82] F. T. Leighton. Finite common coverings of graphs. *J. Combin. Theory, Ser. B*, 33 :231–238, 1982.

[LMS95a] I. Litovsky, Y. Métivier, and E. Sopena. Different local controls for graph relabelling systems. *Math. Syst. Theory*, 28 :41–65, 1995.

[LMS95b] I. Litovsky, Y. Métivier, and E. Sopena. Checking global graph properties by means of local computations : the majority problem. *Electronic Notes in Theoretical Computer Science*, 2, 1995.

[LMS99] I. Litovsky, Y. Métivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In H. Ehrig, H.J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of graph grammars and computing by graph transformation*, volume 3, pages 1–56. World Scientific, 1999.

[LMZ95] I. Litovsky, Y. Métivier, and W. Zielonka. On the Recognition of Families of Graphs with Local Computations. *Information and Computation*, 118(1) :110–119, 1995.

[Lyn89] N. Lynch. A hundred impossibility proofs for distributed computing. In *8th International conference on distributed computing systems*, pages 1–28, 1989.

[Lyn96] N. A. Lynch. *Distributed algorithms.* Morgan Kaufman, 1996.

[Mas91] W. S. Massey. *A basic course in algebraic topology*. Springer-Verlag, 1991. Graduate texts in mathematics.

[Maz87] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editor, *Petri nets, applications and relationship to other models of concurrency*, volume 255 of *Lecture notes in computer science*, pages 279–324. Spinger-Verlag, 1987.

[Maz88] A. Mazurkiewicz. Solvability of the asynchronous ranking problem. *Inf. Processing Letters*, 28 :221–224, 1988.

[Maz97] A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61 :233–239, 1997.

[MMW97] Yves Métivier, Anca Muscholl, and Pierre-André Wacrenier. About the local detection of termination of local computations in graphs. In D. Krizanc and P. Widmayer, editors, *SIROCCO 97 - 4th International Colloquium on Structural Information & Communication Complexity*, Proceedings in Informatics, pages 188–200. Carleton Scientific, 1997.

[Mos] M. Mosbah. communication personnelle.

[MS01] M. Mosbah and A. Sellami. Implementation of an enumeration protocol algorithm using local graph computations. In *Second International Workshop on Graph Transformation and Visual Modeling Techniques , Crete, Greece*, 2001.

[MSZ00] Y. Métivier, N. Saheb, and A. Zemmari. Randomized rendez-vous. In D. Gardy and A. Mokkadem, editors, *proc. of the Int. colloquium on Mathematics and computer Science : Algorithms, Trees, Combinatorics and Probabilities*, Birkhauser Trends in Mathematics Series, pages 183–194, 2000.

[MT00] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *SIROCCO 00 - 7th International Colloquium on Structural Information & Communication Complexity*, pages 237–251, 2000.

[Pel00] D. Peleg. *Distributed Computing - A locality-sensitive approach*. SIAM, 2000.

[Rei32] K. Reidemeister. *Einführung in die Kombinatorische Topologie*. Vieweg, Brunswick, 1932.

[Ros00] K. H. Rosen, editor. *Handbook of discrete and combinatorial mathematics*. CRC Press, 2000.

[RS88] N. Robertson and P. Seymour. Graph minors VII : Disjoint paths on a surface. *J. Combin. Theory, Ser. B*, 45 :212–254, 1988.

[Sak99] N. Sakamoto. Comparison of initial conditions for distributed algorithms on anonymous networks. In *Proc. of ISPD'99*, pages 173–179. ACM, 1999.

[SSP85] B. Szymanski, Y. Shy, and N. Prywes. Terminating iterative solutions of simultaneous equations in distributed message passing systems. In *Proc. of the 4th Symposium of Distributed Computing*, pages 287–292, 1985.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.

[vL90] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume A, B. Elsevier Science Publishers B. V., 1990.

[Wei62] P.M. Weichsel. The kronecker product of graphs. *Proc. Amer. Math. Soc.*, 8 :47–52, 1962.

[YK96a] M. Yamashita and T. Kameda. Computing functions on asynchronous anonymous networks. *Math. Systems Theory*, 29 :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 (formerly Math. systems Theory*, 31(1), 1998.

[Zem00] A. Zemmari. *Contribution à l'Analyse d'Algorithmes Distribués*. PhD thesis, Université Bordeaux I, 2000.

Index

Résumé

Un système distribué peut être représenté par un graphe étiqueté : les sommets correspondent aux processeurs, les arêtes aux liens de communication et les étiquettes associées aux sommets codent les états des processeurs. Un algorithme distribué est alors décrit par un système de règles de transition locale où l'étiquette suivante d'un sommet est fonction de son étiquette actuelle et de celles de ses voisins (réétiquetage local). Les réétiquetages opérant sur des voisinages disjoints se déroulent en parallèle, de manière asynchrone. Dans ce cadre, on étudie la réalisabilité et non-réalisabilité des tâches distribuées. Nous illustrerons notre méthode en nous intéressant en particulier à certains problèmes spécifiques aux systèmes distribués (élection d'un noeud, reconnaissance de certaines propriétés topologiques du graphe sous-jacent au réseau, calcul de métriques du réseau comme par exemple la taille ou le diamètre).

Dans tous ces cas, on présente une caractérisation complète de ce qui est réalisable par calcul distribué en fonction de la topologie du graphe sous-jacent mais également du degré de connaissance qu'a le réseau sur lui-même ("connaissance structurelle"). Ces conditions nécessaires et suffisantes sont principalement exprimées en termes de fermetures par "similarités" des familles de réseaux considérées. Ces "similarités" sont décrites de manière combinatoire à l'aide de morphismes de graphes particuliers : les revêtements et les quasi-revêtements. Les preuves des conditions nécessaires emploient des techniques de simulation à base de revêtements et quasi-revêtements. Les algorithmes distribués présentés pour les preuves des conditions suffisantes se fondent essentiellement sur un algorithme de cartographie du réseau sous-jacent. Celui-ci est construit à partir des extensions d'un algorithme d'énumération de A. Mazurkiewicz et d'un algorithme de détection des propriétés stables de Shy, Szymanski et Prywes.

Mots-clés: algorithme distribué, modélisation, réétiquetage de graphes, calculabilité, connaissance structurelle, graphes, revêtement, quasi-revêtement, réseaux anonymes, autostabilisation, terminaison implicite, terminaison explicite

Abstract

A distributed system can be modelized by a labelled graph : processes as vertices, communication links as edges and labels as processes' states. A distributed algorithm is then described as a local transition system where rules define the label of a vertex according to its current label and to its neighbours' labels (local relabelling). Relabellings operating on disjoint neighbourhood are executed in parallel, asynchronously. In this framework, we study the feasibility and non-feasibility of distributed tasks. Our method is illustrated by investigating in particular some problems that are specific to distributed systems (election of a node, recognition of some topological properties of the underlying graph, computations of mettrics of the network such as the size or the diameter).

In all above cases, we give a full characterization of what is feasible with distributed computations according to the topology of the underlying graph but also according to the amount of knowledge that the network has about its own topology ("structural knowledge"). These characterizations are mainly expressed in terms of "similarities" of the networks families that we consider. These "similarities" are combinatorially described with the help of special kind of graphs morphisms : coverings and quasi-coverings. The proof of necessary conditions are done by quasi-covering and covering-based simulation techniques. The distributed algorithms we describe for proving sufficient conditions are essentially based upon a cartography algorithm. This algorithm is constructed from the extensions of an enumeration algorithm by A. Mazurkiewicz and an algorithm for detecting stable properties by Shy, Szymanski et Prywes.

Keywords: distributed algorithms, modelization, graph relabelling, computability, structural knowledges, graphs, covering, quasi-covering, anonymous networks, selfstabilization, implicit termination, explicit termination