Contribution à l'analyse d'algorithmes distribués

Akka Zemmari

$N°$d'ordre : 2334

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Akka ZEMMARI

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

****

CONTRIBUTION À L'ANALYSE D'ALGORITHMES DISTRIBUÉS

****

Soutenue le : vendredi 15 décembre 2000

Après avis des rapporteurs :

MM.	Christian Lavault	Professeur
	Loÿs Thimonier	Professeur

Devant la commission d'examen formée de :

MM.	Christian Lavault	Professeur	.....	Président
	Yves Métivier	Professeur	.....	Examinateur
	Nasser Saheb	Maître de conférences	.....	Directeur
	Loÿs Timonier	Professeur	.....	Rapporteur

- 2000 -

La disponibilité de Nasser Saheb, l'enthousiasme dont il a fait preuve pour encadrer mon travail a été pour beaucoup dans l'élaboration de cette thèse. Ses conseils et les nombreuses discussions que nous avons eu m'ont été très précieux tout au long de mes recherches. Je suis très heureux de lui exprimer ici ma gratitude et mes remerciements.

Je tiens également à remercier Yves Métivier. Ses conseils et son assistance ne m'ont jamais été épargnés, tout aussi bien pour mes recherches que pour les problèmes que j'ai eu durant mes quatre années à Bordeaux. Qu'il trouve ici l'expression de mes remerciements et de ma reconnaissance.

Mon premier travail de recherche a été sous la direction de Cyril Gavoille pendant le DEA. Les discussions que j'ai eu avec lui m'ont fait aimer la recherche, qu'il veuille accepter mes remerciements.

Je remercie aussi Christian Lavault pour avoir accepté de rapporter cette thèse et pour l'honneur qu'il m'a fait en acceptant d'être président du jury.

Merci aussi à Loÿs Thimonier pour l'interêt qu'il a manifesté pour mon travail en rapportant cette thèse et en étant membre du jury.

La liste serait très longue si je voulais énumérer tous les gens que j'ai eu le plaisir de côtoyer au LaBRI durant ces trois années de thèse. Que ceux que j'aurais oubliés veuillent bien m'en excuser, et accepter mes remerciements.

D'abord, il y a Davy, le grand ami qui a toujours été là pendant les moments difficiles, puis Valère dont l'humour est à tout épreuve, Cédric à qui je promets de ne pas m'attirer d'ennuis lors de nos prochaines missions, Guillaume avec qui les matches de tennis et de ping-pong ont été de grands moments de détente, Olivier le compagnon de voyage, et le bon ami. Enfin, mes anciens collègues de bureau Pascal, Mourad, Anthony, Sylvain, Gabriel et Arnaud.

En dehors du LaBRI, il y a tous mes amis qui ont toujours été là quand j'avais besoin d'eux. Je remercie Aicha, Samira, Lamia, Ouafae, Meriem, Hanane, Mustapha et Mustapha2, ainsi que Jawad, Alaa, et Nawfel. Enfin Driss et Sami, les grands amis de toujours sans oublier bien sûr Loubna et Christelle.

Enfin, ma famille, mes parents, mes frères et mes sœures sans qui ma venue en France n'aurait été possible. Je pense notamment à Belhadj et Moulay. Je sais l'ampleur des sacrifices que cela leur a demandé. Je leur suis gré de leur dévotion pour mes souhaits, et les remercie pour les grands sacrifices qu'ils ont dû faire afin de m'aider à faire mes études ici à Bordeaux.

Table des matières

Introduction **1**

Notions de la théorie des graphes **7**

I Degré de parallélisme dans les monoïdes de commutation **11**

1 Degré de parallélisme **13**

1.1 Introduction 14
1.2 Monoïde de commutation 15
1.2.1 Définitions et notations 15
1.2.2 Graphe de conflit 15
1.3 Degré de parallélisme 16
1.3.1 Introduction 16
1.4 Comportement asymptotique 21
1.4.1 Modélisation stochastique 22
1.4.2 Facteurs saturés et théorie de renouvellement 23

2 Méthodes de calcul **25**

2.1 Profondeur moyenne et degré de parallélisme 25
2.2 Exemples 26
2.3 Méthode de diagonalisation 29
2.4 Chaînes de Markov et degré de parallélisme 31
2.5 Approche utilisant les marches aléatoires 35
2.6 Une méthode d'approximation 37
2.7 Relation avec la physique statistique 38
2.8 Conclusion et perspectives 40

II Synchronisations locales probabilistes 41

3 Synchronisations locales 43
3.1 Modèles et Définitions . . . 43
3.2 Généralités sur les algorithmes probabilistes . . . 44
3.3 Hypothèses et objectifs de l'étude . . . 45
3.4 Exemples de calculs locaux . . . 46
3.4.1 Calcul distribué d'un arbre couvrant . . . 46
3.4.2 Un algorithme pour détecter des propriétés stables . . . 47
3.4.3 Un algorithme d'énumération de Mazurkiewicz . . . 48
3.5 Implémentation avec les algorithmes probabilistes . . . 49

4 Analyse des algorithmes 53
4.1 Le rendez-vous . . . 53
4.1.1 Introduction . . . 53
4.1.2 Premiers résultats . . . 54
4.1.3 Probabilité de succès pour des cas particuliers . . . 63
4.1.4 Une borne uniforme pour la probabilité de succès . . . 66
4.1.5 Distribution du nombre de rendez-vous . . . 67
4.1.6 Graphes avec sommets passifs . . . 72
4.1.7 Résultats expérimentaux . . . 73
4.2 LS_1-Synchronisation . . . 75
4.2.1 Probabilité de synchronisation sur un sommet . . . 75
4.2.2 Nombre moyen de LS_1-synchronisations . . . 77
4.3 LS_2-Synchronisation . . . 79
4.3.1 Probabilité d'une LS_2-synchronisation sur un sommet . . . 79
4.3.2 Nombre moyen de LS_2-synchronisations . . . 80
4.4 Généralisation et remarques . . . 82

5 Efficacité des algorithmes 85
5.1 Efficacité de l'algorithme du rendez-vous . . . 85
5.2 Efficacité de l'algorithme des LS_1-synchronisations . . . 88
5.3 Efficacité de l'algorithme des LS_2-synchronisations . . . 90
5.4 Conclusion et perspectives . . . 91

III Analyse d'un algorithme d'élection dans les arbres 93

6 Problème d'élection 95
6.1 Définitions . . . 95
6.2 Problème d'élection . . . 97
6.2.1 Quelques algorithmes d'élection . . . 98

6.2.2 L'algorithme d'élection étudié 99

7 Analyse de l'algorithme 103
7.1 Analyse suivant la considération **(i)** 103
7.1.1 Ordre partiel des facteurs 104
7.1.2 Forme matricielle 105
7.1.3 Relation avec les monoïdes de commutation 106
7.1.4 Une propriété combinatoire 107
7.1.5 Calcul effectif de p 108
7.1.6 Algorithme d'élection et chaînes de Markov 111
7.2 Analyse suivant la considération **(ii)** 113
7.2.1 Forme matricielle 115
7.2.2 Une propriété combinatoire 115
7.2.3 Une implémentation de l'algorithme 116
7.3 Conclusion et perspectives 120

IV Routage compact et adaptatif 123

8 Routage par intervalles 125
8.1 Routage et tables de routage 125
8.2 Routage par intervalles 127
8.3 Tables de routage adaptatives 128
8.4 Définitions formelles et notations 129

9 Routage par intervalles adaptatif 131
9.1 Un schéma d'étiquetage général 131
9.2 Étiquetage de plus courts chemins 134
9.2.1 Comparaison entre IRS et IRS_α 134
9.2.2 Une borne inférieure pour la α-compacité 137
9.3 Conclusion et perspectives 139

Bibliographie 146

Introduction

Un système distribué est un ensemble d'entités pouvant communiquer entre elles. Ces entités peuvent être des puces, des machines multi-processeurs, des stations de travail, des réseaux locaux, etc. Elles sont indépendantes ou semi-indépendantes, certaines entités pouvant nécessiter une coordination.

La nature d'un système distribué, la coordination entre ses différentes entités ou bien encore la résolution d'un problème général génèrent différents types de problèmes : partage des ressources, tolérance aux pannes, disponibilité, centralisation de certaines informations, détection de la terminaison, routage, etc.

Des difficultés apparaissent liées à l'asynchronisme, à la connaissance locale des machines et aux pannes.

Des solutions peuvent être apportées par des modèles permettant de comprendre ces problèmes, de concevoir des solutions et de les analyser. Deux exemples de ces modèles sont les monoïdes de commutation et les calculs locaux.

Les algorithmes probabilistes permettent de fournir des solutions parfois plus "efficaces" que les solutions déterministes, ou encore des solutions pour des problèmes qui n'admettent pas de solution déterministe.

Plusieurs livres traitent de ces problèmes et fournissent de bons "états de l'art" sur ce que l'on sait faire tels que les livres [73] de G. Tel et [40] de C. Lavault pour l'algorithmique distribuée et du livre [55] de R. Motwani et P. Raghavan pour les algorithmes probabilistes.

Préférer un système distribué à un système séquentiel peut être motivé, entre autres, par les raisons suivantes :

- échanger les informations : plusieurs machines peuvent échanger des informations,
- partager les ressources : plusieurs machines ont accès aux ressources critiques,
- augmenter les performances : une tâche peut être partagée en plusieurs sous-tâches indépendantes, celles-ci peuvent être exécutées chacune sur une machine réduisant ainsi le temps nécessaire pour réaliser cette tâche.

Soit $\mathcal{T}$ une tâche à réaliser et soit $\mathcal{S}$ un système distribué, un *algorithme distribué* $\mathcal{A}$ pour la tâche $\mathcal{T}$ est un algorithme capable de faire collaborer tous, ou partie, des processeurs du système $\mathcal{S}$ afin de réaliser $\mathcal{T}$.

Nous abordons dans cette contribution certains aspects des thèmes suivants en liaison avec des algorithmes distribués et des problèmes de synchronisation :

Degré de parallélisme dans les monoïdes de commutation : Les monoïdes de commutation permettent de modéliser les exécutions d'algorithmes distribués. La forme normale de Foata permet d'obtenir le temps "optimal" d'exécution d'un algorithme si celui-ci est exécuté dans un système distribué. Le degré de parallélisme d'un mot est défini comme le rapport entre ce temps optimal et le temps d'une exécution séquentielle. Ces investigations interviennent dans une comparaison quantitative et aussi en physique statistique. Nous fournissons quelques méthodes pour calculer ce degré basées sur une approche probabiliste.

Synchronisations locales probabilistes : Pour implémenter un algorithme distribué, les processeurs d'un système distribué doivent réaliser des calculs locaux. Ces calculs nécessitent des échanges d'informations entre les processeurs voisins, lesquels nécessitent des synchronisations locales. Or, sous certaines hypothèses sur le réseau, ces synchronisations ne peuvent se faire que par des algorithmes probabilistes.

Problèmes d'élection dans les arbres : Le problème d'élection est un paradigme de l'algorithmique distribuée. Élire dans un système distribué revient à choisir "équitablement" une entité du système pour réaliser une tâche. Cette tâche pouvant être la centralisation des informations, la relance d'une procédure de réinitialisation du système, ou encore l'accès à une ressource critique telle qu'une imprimante. Nous analysons un algorithme d'élection pour les réseaux en arbre, ou plus généralement, pour les réseaux où un arbre couvrant est disponible.

Routage compact et adaptatif : On s'intéresse à l'étude de la taille mémoire requise pour router efficacement des messages dans un réseau de processeurs. Si de nombreuses solutions à ce problème ont été proposées dans le cadre du routage déterministe, très peu l'ont été dans un cadre adaptatif, c'est-à-dire lorsque plusieurs chemins peuvent être utilisés en fonction de la charge du réseau ou de tout autre paramètre. Parmi les méthodes de routage compact déterministe, le routage par intervalles consiste à numéroter chacun des n noeuds du réseau par un unique entier compris entre 1 et n, et à affecter à chacun des liens un intervalle de destinations. L'algorithme de routage étant : si le noeud x reçoit ou veut transmettre un message à y alors il choisit le lien associé à l'intervalle contenant le numéro de y. L'intérêt d'une telle méthode est que peu d'information doit être stockée au niveau de chaque noeud (les intervalles). Nous proposons une généralisation de la méthode décrite ci-dessus à un routage adaptatif.

Cette thèse est organisée comme suit :

Partie I : Degré de parallélisme dans les monoïdes de commutation

Cette partie est consacrée aux monoïdes de commutation et aux méthodes de calcul du degré de parallélisme dans ces monoïdes. Les résultats de cette partie ont été obtenus en collaboration avec Nasser Saheb [67]. Elle est organisée en deux chapitres :

Chapitre 1 : Ce chapitre est une introduction à la problématique qui nous intéresse. La section 1.1 est une introduction générale au problème. La section 1.2 présente de manière formelle les monoïdes de commutation et la notion de graphe de conflit. Dans la section 1.3, nous donnons le degré de parallélisme tel qu'il a été défini par N. Saheb dans [65], ainsi que la forme normale de Foata d'un mot quelconque. Nous présentons aussi quelques propriétés de ce paramètre déjà établies dans des travaux ultérieurs.

Chapitre 2 : Dans ce chapitre, nous présentons quelques méthodes pour calculer le degré de parallélisme. Nous commençons par quelques exemples simples, puis dans la section 2.3 nous présentons une méthode dite de diagonalisation pouvant être utilisée dans des cas particuliers tels que le cas où il y a une lettre qui ne commute avec aucune autre lettre de l'alphabet.

Dans la section 2.4, nous exploitons la modélisation du processus de factorisation par une chaîne de Markov, on verra comment l'espace des états de la chaîne peut être réduit, permettant ainsi de calculer la distribution stationnaire de probabilités et d'en déduire le degré de parallélisme.

Les sections 2.5 et 2.6 fournissent deux approches pour obtenir une approximation du degré. La première est fondée sur les marches aléatoires et la seconde sur des constructions successives.

Enfin, la section 2.7 donne une relation entre le degré étudié et une conjecture de la physique statistique.

Partie II : Synchronisations locales probabilistes

Dans cette partie, on considère un réseau asynchrone de processeurs anonymes à communication par échange de messages en mode asynchrone. On s'intéresse à l'implémentation d'algorithmes distribués dans de tels réseaux, en fournissant des procédures probabilistes pour synchroniser des processeurs entre eux. Les résultats de cette partie ont été obtenus en collaboration avec Yves Métivier et Nasser Saheb [51]. Cette partie est organisée en trois chapitres :

Chapitre 3 : Nous commençons ce chapitre par la section 3.1 dans laquelle nous présentons les modèles utilisés et nous rappelons quelques notions de réseau

synchrone, asynchrone, et anonyme, d'échange de message en mode asynchrone et synchrone.

Dans la section 3.3, on présente les hypothèses et les objectifs de notre étude, et dans la section 3.4, on donne quelques exemples d'algorithmes distribués classiques nécessitant des synchronisations locales pour être implémentés.

Dans la section 3.2, on présente des algorithmes probabilistes pour implémenter ces algorithmes. Le premier algorithme permet de synchroniser un processeur avec un de ses voisins, le deuxième permet de synchroniser un processeur avec tous ses voisins et le dernier de synchroniser un processeur avec tous les sommets à distance inférieure ou égale à 2 de ce processeur.

Chapitre 4 : Dans ce chapitre, nous analysons les algorithmes du chapitre précèdent. La section 4.1 est dédiée à l'étude de l'algorithme permettant de synchroniser un processeur avec un de ses voisins. Nous y étudions un certain nombre de paramètres liés à cet algorithme concernant les comportements probabilistes du nombre de synchronisations.

Dans la section 4.2 (resp. section 4.3), nous analysons l'algorithme permettant de synchroniser un processeur avec tous ses voisins, (resp. avec tous les processeurs à une distance inférieure ou égale à 2).

Chapitre 5 : Dans ce chapitre nous introduisons et étudions une mesure d'efficacité des algorithmes probabilistes présentés dans le chapitre précédent. L'objectif étant de fournir une mesure commune à tout algorithme probabiliste de synchronisation.

Partie III : Élection dans les arbres

Dans cette partie, nous analysons le problème d'élection dans un système distribué ayant la topologie d'arbre. Les résultats qui y sont présentés ont été obtenus en collaboration avec Yves Métivier et Nasser Saheb. Elle est organisée en deux chapitres :

Chapitre 6 : Dans ce chapitre, nous commençons par la section 6.1 où nous rappelons des notions fondamentales de l'algorithmique distribuée. Nous y rappelons les définitions et modèles utilisés dans cette partie, notamment les algorithmes en vagues et les systèmes de réécriture de graphes. Dans la section 6.2, nous présentons le problème d'élection et nous rappelons quelques algorithmes connus pour le résoudre. La dernière section est dédiée à la présentation de l'algorithme, nous montrons, en particulier, que c'est un algorithme en vagues qui peut être codé par un système de réécriture de graphes.

Chapitre 7 : Dans ce chapitre, nous effectuons une analyse probabiliste de l'algorithme proposé. Différentes considérations sont étudiées, le but étant de répondre à la question : quel est le sommet (de l'arbre représentant le réseau) qui a la plus grande probabilité d'être élu. Différentes techniques sont alors mises en œuvre, notamment les monoïdes de commutation et les chaînes de Markov. Nous terminons le chapitre par la section 7.2.3 où nous donnons quelques implémentations possibles de l'algorithme.

Partie IV : Routage compact et adaptatif

Dans cette partie, on s'intéresse à l'étude de la taille mémoire nécessaire pour coder les tables de routage adaptatives dans un réseau de processeurs. Les résultats qui y sont présentés ont été obtenus en collaboration avec Cyril Gavoille [32]. Elle est organisée en deux chapitre :

Chapitre 8 : Dans ce chapitre, nous présentons le problème de routage et nous rappelons le principe de la technique de routage par intervalles, ainsi que les principales notations et définitions formelles utilisées dans cette partie.

Chapitre 9 : Dans ce chapitre, nous nous intéressons à l'adaptativité du routages par intervalles et à sa compacité. Le théorème 15 de la section 9.1 permet de montrer que tout graphe supporte un routage par intervalles α-adaptatif dont la compacité est optimale, c'est-à-dire égale à 1. Dans la section 9.2, nous nous intéressons au routage par intervalles de plus court chemin, nous imposons alors qu'au moins un des chemins fournis par les tables permet de router en utilisant le plus court chemin. Le résultat principal de cette section est le théorème 17 selon lequel la différence entre la compacité dans le cas déterministe et la compacité dans le cas adaptatif peut être très grande.

Notions de la théorie des graphes

Dans notre étude, nous nous servons des graphes pour modéliser les réseaux de communications. Dans ce chapitre, nous rappelons quelques notions fondamentales de la théorie des graphes. Les définitions sont empruntées au livre de C. Berge [6] et à [63].

Définitions élémentaires

Un *graphe non orienté simple* est un couple (V, E), où E est un ensemble de paires d'éléments distincts de V, appelés *arêtes*.

Si $e = \{v, v'\} \in E$, on dit que e est *incident* à v et v' et que v et v' sont deux sommets *voisins*. Soit $N(v)$ l'ensemble des voisins de v. Le *degré* du sommet v, noté $d(v)$, est le cardinal de $N(v)$.

Le nombre de sommets d'un graphe G est appelé *taille* de G.

Soit $G = (V, E)$ un graphe quelconque. Un *sous-graphe* du graphe G est un graphe $H = (V', E')$ dont l'ensemble des sommets V' (respectivement l'ensemble des arêtes E') est un sous-ensemble de V (respectivement de E) tel que pour toute arête $e = \{u, v\}$ de E', les deux sommets u et v appartiennent à V'.

Un *sous-graphe couvrant* du graphe G est un sous-graphe de G qui contient tous les sommets de G.

Le graphe résultant de la suppression d'une arête e d'un graphe $G = (V, E)$ est le graphe G privé de l'arête e.

La suppression d'un sommet v d'un graphe $G = (V, E)$ est l'opération consistant à supprimer le sommet v et toutes les arêtes incidentes à v.

Deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ sont dits *sommets disjoints* si $V_1 \cap V_2 = \emptyset$. Ils sont dits *arêtes disjoints* si $E_1 \cap E_2 = \emptyset$.

Un *chemin* P de v_1 à v_i dans G est une suite $P = v_1, e_1, v_2, e_2, ..., e_{i-1}, v_i$ de sommets et d'arêtes, telle que, pour $1 \leqslant j < i$, e_j est incidente à v_j et v_{j+1} ; $i - 1$ est la *longueur* de P. Si $v_1 = v_i$, alors P est un *cycle*. Deux sommets v et v' sont dits *connectés* ou *liés*, s'il existe un chemin de v à v'. Un graphe est *connexe* si deux sommets quelconques u et v sont connectés.

Remarque. Les définitions que nous donnons ici sont relatives aux graphes non orientés. La plupart d'entre elles restent les mêmes pour les graphes orientés.

Arbres et forêts

Un *arbre* T est un graphe connexe sans cycle. Un arbre T' est un *facteur* d'un arbre T, si soit $T' = T$, soit T' est un facteur d'un arbre obtenu par la suppression d'une feuille de T.

Une *composante connexe* du graphe G est un sous-graphe connexe maximal du graphe G.

Un *arbre couvrant* d'un graphe G est un sous-graphe couvrant de G qui est un arbre.

Une *forêt* est un graphe dont les composantes connexes sont des arbres. Une *forêt couvrante* d'un graphe G est une forêt qui contient l'ensemble des sommets de G.

Classes particulières de graphes

Un graphe est *complet* si pour toute paire de sommets $\{u, v\}$, l'arête $\{u, v\}$ existe. On note K_n le graphe complet de taille n (ce graphe possède donc $(n \times (n-1))/2$ arêtes).

Un graphe G est *biparti* si V peut être partitionné en deux classes disjointes V_1 et V_2 telles que aucune arête de E ne relie deux sommets de V_1 ou deux sommets de V_2. Le graphe G est *biparti complet* si tout sommet de V_1 est adjacent à tout sommet de V_2. On note $K_{n,m}$ le graphe non orienté biparti complet tel que $|V_1| = n$ et $|V_2| = m$.

Une *étoile* de taille n est le graphe $K_{1,n-1}$.

Pour un entier positif r et un sommet v de G, la boule $B(v, r)$, de centre v et de rayon r est définie par induction sur r comme suit :

$$B(v, 0) = \{v\}, \text{ et } B(v, r+1) = B(v, r) \cup \bigcup_{w \in B(v,r)} N(w).$$

Distances sur un graphe

Soit $G = (V, E)$ un graphe connexe quelconque. La *distance* entre deux sommets quelconques u et v, notée $d(u, v)$, est la longueur du plus court chemin de u à v. Le *diamètre* du graphe G est la distance maximale entre deux de ses sommets :

$$D(G) = \max_{u,v \in V} \{d(u, v)\}.$$

L'*excentricité* d'un sommet v, notée $e(v)$, d'un graphe connexe G est la distance maximale entre v et les autres sommets de G. Le *rayon* du graphe G, noté $r(G)$, est

alors l'excentricité minimum sur tous ses sommets :

$$e(u) = \max_{v \in V}\{d(u,v)\} \text{ et } r(G) = \min_{u \in V} e(u).$$

Un sommet v est un *centre* d'un graphe connexe G si c'est un élément de l'ensemble des sommets d'excentricité minimum, c'est-à-dire de l'ensemble :

$$c(G) = \{v \mid e(v) = \min_{w \in V} e(w)\}.$$

Soit G un graphe quelconque, et soit k un entier positif. On définit la *k-densité* de G par le rapport :

$$\mathcal{D}_k(G) = \frac{\sum_{v \in V} d(v)^k}{|V|}.$$

Soit $T = (V, E)$ un arbre, pour un sommet v, on définit $Dist(v, T)$ par :

$$Dist(v, T) = \sum_{w \in V} D(v, w).$$

Un sommet de V est un *sommet médian*, si c'est un élément du sous-ensemble $m(T)$, défini par :

$$m(T) = \{v \mid Dist(v, T) = \min_{w \in V} Dist(w, T)\}.$$

Première partie

Degré de parallélisme dans les monoïdes de commutation

Chapitre 1

Degré de parallélisme dans les monoïdes de commutation

On représente une suite d'opérations par un mot sur l'alphabet des actions. Si les actions a et b agissent sur des ensembles disjoints, les exécutions séquentielles ab et ba produisent le même effet. Cette propriété est modélisée mathématiquement par la commutation de a et b : $ab \simeq ba$. Dans un contexte distribué cela implique que a et b peuvent être exécutées *en parallèle*.

Une exécution distribuée n'est donc plus un mot mais une classe d'équivalence pour la congruence engendrée par les relations de commutation. Cette classe d'équivalence est appelée *trace*, elle peut être identifiée au graphe d'une relation d'ordre partiel. L'ensemble des traces forme le monoïde de commutation. Mazurkiewicz propose les monoïdes des traces pour modéliser les processus parallèles [46].

Nous rappelons brièvement dans ce chapitre quelques concepts et résultats de [65]. Le chapitre suivant porte sur de nouvelles techniques de calcul de degré de parallélisme pour lequel aucune formule exacte n'est connue sauf dans des cas très particuliers, voir [27].

Un système de commutation est un alphabet d'actions A muni d'une relation binaire θ. Lorsque $(a, b) \in \theta$, les actions a et b ne sont pas causallement liées et, donc, peuvent être exécutées en parallèle. Ainsi, les éléments de θ sont les paires de lettres (ou d'actions) qui peuvent être exécutées parallèlement.

La forme normale de Foata permet de maximiser le taux de simultanéité pour un mot de A^* (ou une suite d'actions). Dans [65], N. Saheb définit le degré de parallélisme pour un mot non vide comme le rapport entre la longueur du mot et le nombre de facteurs une fois le mot écrit sous sa forme normale de Foata.

Il est montré que chaque monoïde de commutation a un degré de parallélisme commun à presque tous ses mots infinis. Or, même s'il existe des caractérisations de

ce degré, il n'y a pas de méthode effective connue pour le calculer. Ici, nous introduisons quelques nouvelles méthodes pour une évaluation exacte de ce paramètre dans des cas particuliers, et une méthode pour son approximation en général.

1.1 Introduction

Les monoïdes de trace ont été introduits par Mazurkiewicz pour permettre une étude algébrique du parallélisme [46], (voir [2, 13, 24, 25, 49, 59] pour les résultats obtenus dans ce cadre).

Arquès et Françon ont proposé une mesure de parallélisme, pour les processus concurrents fondée sur la possibilité de mélanger des mots représentants le comportement des processus. L'approche permet quelques comparaisons pour différents algorithmes de synchronisation, sans pour autant fournir une étude générale de mesure de parallélisme.

N. Saheb a introduit le degré de parallélisme pour les monoïdes de commutation, voir [65]. Ce degré se définit pour un mot comme le rapport entre la longueur du mot et son nombre de facteurs. Intuitivement ce degré est le rapport entre le temps d'exécution séquentielle et celui d'exécution parallèle de la suite d'instructions associée au mot.

Il est prouvé que si tous les mots de longueur n ont la même probabilité d'être tirés, quand $n \to \infty$, avec probabilité 1, le degré du mot choisi tend vers une limite fixée qui est complètement déterminée par le système.

Un système de commutation peut être caractérisé par son graphe de conflit, c'est-à-dire, le graphe dont les sommets sont les actions et les arêtes les paires de lettres qui ne commutent pas ; c'est le graphe complémentaire du graphe de commutation. Si le graphe de conflit est connexe, il a été montré que le degré est l'inverse de la distribution stationnaire de la probabilité d'incrémentation de la profondeur (c'est-à-dire, le nombre de facteurs de Foata), voir [65].

Dans [65], le processus de factorisation est considéré comme un processus aléatoire, où à chaque étape, une lettre est insérée. Toutes les lettres de l'alphabet d'actions ont la même chance d'être tirées. Si le graphe de conflit est connexe, le processus aléatoire admet une distribution limite et le degré de parallélisme s'écrit facilement en fonction de certains paramètres de cette distribution.

Si le graphe de conflit est non connexe, alors chaque composante forme un système de commutation et le degré de parallélisme de tout le système peut alors être calculé en fonction des degrés de parallélisme des systèmes représentés par les composantes connexes. Il suffit alors de chercher des méthodes pour calculer le degré de parallélisme dans le cas où le graphe de conflit est connexe.

1.2 Monoïde de commutation

1.2.1 Définitions et notations

A est un alphabet (d'actions) de cardinalité N. Un *monoïde ou système de commutation* est l'alphabet A muni d'une relation binaire $\theta \subset A \times A$ symétrique et irréflexible. Si $(a, b) \in \theta$, on dit que *a et b commutent* ou *a et b sont concurrentes*, et l'on note $ab \simeq ba$.

Intuitivement, quand a et b commutent, le résultat d'exécution ne change pas si on exécute ab ou bien ba. On est alors autorisé à considérer que a et b sont concurrentes, en effet, si l'exécution de a n'affecte en rien celle de b et vice versa, a et b peuvent être exécutées simultanément. L'irréflexivité de θ est imposée par le fait que deux actions identiques ne peuvent être exécutées de manière concurrente.

Le mot vide est noté $\mathbf{1}$. A^*, A^+, et A^ω désignent respectivement l'ensemble de mots, l'ensemble de mots non vides, et l'ensemble des mots infinis sur A. Pour tout $w \in A^*$, $|w|$ est la longueur de w. Enfin, si $B \subset A$, alors $(w)_B$ est le mot obtenu de w en supprimant les lettres n'appartenant pas à B, et $|w|_B$ est alors la longueur du mot $(w)_B$, (si B est réduit à une seule lettre a, $|w|_a$ est le nombre d'occurrences de la lettre a dans w).

La relation θ engendre une congruence (voir [13, 14, 46]). Si deux mots w_1 et w_2 sont équivalents par cette congruence, on écrit $w_1 \simeq w_2$; ce qui veut dire que w_1 et w_2 sont *opérationnellement équivalentes*.

1.2.2 Graphe de conflit

Considérons un système de commutation $S = (A, \theta)$. Le *graphe de conflit* de S est le graphe G dont les sommets sont les éléments de A et pour tout couple de sommets différents (a, b) de A, on met une arête entre a et b si $(a, b) \in \theta$.

Exemple 1.1 Soit $A = \{a, b, c, d\}$, où a, b, c, d représentent les instructions suivantes :

$$\begin{cases} (a) : x \leftarrow x * cos(y) \\ (b) : y \leftarrow y + t/2 \\ (c) : t \leftarrow (u + v)/2 \\ (d) : u \leftarrow x * ln(v) \end{cases} \tag{1.1}$$

Il est clair que dans une suite d'actions a et b doivent être exécutées dans l'ordre imposé, de même pour les instructions $\{a, d\}$, et $\{c, d\}$; par contre, toute occurrence de ac peut être remplacée par une occurrence de ca, et toute occurrence de bd peut être remplacée par une occurrence de db.

Le graphe de conflit correspondant à ce système est alors représenté par le graphe de la figure 1.1.

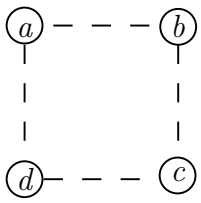


FIGURE 1.1 – Graphe de conflit de l'exemple 1.1.

1.3 Degré de parallélisme dans les monoïdes de commutation

1.3.1 Introduction

Soit w un mot de A^n. À w correspond un graphe d'ordonnancement G_w défini comme suit.
Soit $w = x_1x_2 \dots x_n$ avec $x_i \in A,\ \forall i = 1, \dots, n$. Le graphe G_w a n sommets étiquetés $1, 2, \dots, n$, et pour toute paire de sommets (i, j), il y a un arc de i vers j si et seulement si $i < j$ et x_i ne commute pas avec x_j.

Exemple 1.2 Soit $A = \{a, b, c, d\}$ et θ le relation définie par $ab \simeq ba$, $ac \simeq ca$, $bc \simeq cb$, $ad \simeq da$. Considérons le mot $w = dcabadbac$, le graphe G_w correspondant est donné par le figure 1.2, (pour la simplicité, nous supprimons des arcs qui peuvent être obtenus par transitivité).

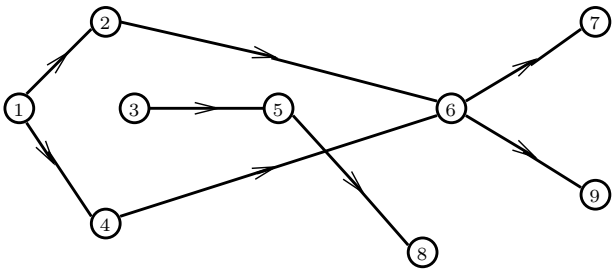


FIGURE 1.2 – Graphe d'ordonnancement de l'exemple 1.2.

Définition 1 Soit w un mot de A^n. La *profondeur* de w, notée $dep(w)$, est définie comme étant le nombre de sommets sur le plus long chemin du graphe G_w.

Sur l'exemple précédent, le plus long chemin est de longueur 4, la profondeur du mot w est donc $dep(w) = 4$.

Définition 2 Soit w un mot de A^n. Le *degré de parallélisme* de w noté $d(w)$ est la quantité

$$d(w) = \frac{|w|}{dep(w)}. \tag{1.2}$$

Dans l'exemple précédent, $d(w) = \frac{9}{4}$. Intuitivement, cette quantité illustre le fait que dans un temps nécessaire pour exécuter 4 actions, on peut en exécuter 9. Donc, en moyenne, $\frac{9}{4}$ actions peuvent être exécutées simultanément.

Le graphe G_w est une manière simple de montrer dans quel ordre les actions formant le mot w doivent être exécutées (tri topologique). Mais une autre caractérisation de $dep(w)$ semble plus appropriée à notre étude. Cette caractérisation utilise la *forme normale de Foata* [10].
Un mot de A^* est sous *forme normale de Foata* s'il est factorisé en $u_1u_2\cdots u_k$, $k \geqslant 0$, tel que :

(i) Tout u_i est un mot non vide et est produit de lettres qui commutent deux à deux.

(ii) Si a est une lettre dans u_{j+1} n'apparaissant pas dans u_j, alors il existe une lettre de u_j qui ne commute pas avec a.

Chaque u_i est alors appelé *facteur*. Un facteur est un mot non vide u dont les lettres commutent deux à deux. Une lettre sera dite *commuter* avec un facteur u si elle commute avec toutes les lettres de u. La profondeur d'un mot w, $dep(w)$, n'est autre que le nombre des facteurs de w écrit sous forme normale de Foata.

La forme normale de Foata d'un mot w est obtenue récursivement comme suit :

— si w est une lettre, alors sa forme normale de Foata est w lui même.

— si la forme normale de Foata de w est $u_1u_2\cdots u_k$, alors la forme normale de Foata de wa, $a \in A$, est $u_1u_2\cdots u_ka$ si a ne commute pas avec u_k ou bien $u_1u_2\cdots u'_i\cdots u_k$ avec $u'_i = u_ia$, où i est tel que a commute avec $u_k, u_{k-1}, \cdots, u_i$ mais pas avec u_{i-1}.

Dans l'exemple 1.2, on a $w \simeq (da)(cba)(da)(bc)$. Les actions au sein d'un même facteur peuvent être exécutées dans un ordre quelconque, mais aucun changement d'ordre entre les facteurs n'est possible. Il est à remarquer que $dep(w) = 4$, ce qu'on a obtenu avec le graphe G_w.

1.3.1.1 Propriétés

La proposition suivante fournit quelques propriétés du degré de parallélisme, voir [65].

Proposition 1 *Pour tous $w_1, w_2, \cdots, w_k \in A^+$, les propriétés suivantes sont vérifiées :*

(i) $1 \leqslant d(w) \leqslant N$, *($N = card(A)$),*
(ii) $d(w_1 w_2 \cdots w_k) = \sum_{i=1}^{k} \frac{dep(w_i) d(w_i)}{dep(w_1 w_2 \cdots w_k)}$,
(iii) $d(w_1 w_2 \cdots w_k)$ *est minoré par la moyenne de* $d(w_1), d(w_2) \cdots, d(w_k)$ *pondérés par les coefficients* $dep(w_1), dep(w_2), \cdots, dep(w_k)$ *respectivement.*
(iv) $d(w_1 \cdots w_k) \geqslant \min\{d(w_1), \cdots, d(w_k)\}$.

Preuve.

(i) Il est facile de voir que si le mot w est formé par des lettres ne commuttant pas entre elles, sa forme normale est donc lui même. On en déduit alors que $dep(w) \leqslant |w|$. Si toutes les lettres du mot commutent, sa forme normale est de longueur au moins $\frac{|w|}{N}$. On en déduit donc que

$$\frac{|w|}{N} \leqslant dep(w) \leqslant |w|.$$

D'où le résultat.

(ii) On a $|w_1 w_2 \cdots w_k| = |w_1| + |w_2| + \cdots + |w_k|$.
Donc, $d(w_1 w_2 \cdots w_k) = \sum_{i=1} k|w_i| / dep(w_1 w_2 \cdots w_k)$.
Le résultat en découle.

(iii) On a

$$d(w_1 w_2 \cdots w_k) = \frac{1}{dep(w_1 w_2 \cdots w_k)} \sum_{i=1}^{k} |w_i|.$$

Or

$$dep(w_1 w_2 \cdots w_k) \leqslant dep(w_1) + dep(w_2) + \cdots + dep(w_k).$$

Donc

$$d(w_1 w_2 \cdots w_k) \geqslant \frac{1}{dep(w_1) + dep(w_2) + \cdots + dep(w_k)} \sum_{i=1}^{k} dep(w_i) d(w_i).$$

(iv) Application directe de (iii).

□

1.3.1.2 Degré moyen et degré moyen relatif de parallélisme

Définition 3 Pour tout entier positif n, le *degré moyen de parallélisme* des mots de longueur n, est défini par :

$$d(n) = \frac{n}{N^n} \sum_{w \in A^n} \frac{1}{dep(w)}. \tag{1.3}$$

C'est l'espérance mathématique du degré de parallélisme si tous les mots de longueur n ont la même probabilité d'être choisis.

Définition 4 Pour tout entier positif n, la *profondeur moyenne relative* des mots de longueur n est définie par :

$$l(n) = \frac{1}{nN^n} \sum_{w \in A^n} dep(w). \tag{1.4}$$

C'est l'espérance mathématique de $dep(w)/n$ si tous les mots de longueur n ont la même probabilité d'être choisis.

On obtient

$$1 \leqslant d(n) \leqslant N \tag{1.5}$$

et

$$\frac{1}{n} \leqslant l(n) \leqslant 1. \tag{1.6}$$

Nous avons aussi :

Proposition 2 *Soient $n_1, n_2, \cdots, n_k$ des entiers positifs quelconques ; nous avons*
(i) *$l(\sum_{1 \leqslant n \leqslant k} n_i)$ est majorée par la moyenne de $l(n_1), \cdots, l(n_k)$ pondérés par $n_1, n_2, \cdots, n_k$ respectivement ; autrement dit, l est une fonction convexe,*
(ii) *$l(kn) \leqslant l(n)$,*
(iii) *$d(n) \geqslant l(n)$.*

Preuve.

(i) Soient $w_1, w_2, \cdots, w_k$ des mots de longueurs $n_1, n_2, \cdots, n_k$ respectivement. Posons $n = \sum_{i=1}^{k} n_i$, on a alors

$$\frac{1}{n} dep(w_1 w_2 \cdots w_k) \leqslant \frac{1}{n} \sum_{i=1}^{k} dep(w_i).$$

En passant à l'espérance mathématique, et en utilisant sa linéarité, nous obtiendrons :

$$l(n) \leqslant \frac{1}{n} \sum_{i=1}^{k} n_i l(n_i).$$

Le résultat en découle.

(ii) Il suffit d'appliquer (i) à $n_1 = n_2 = \cdots = n_k = n$.

(iii) La fonction f définie par $f(x) = \frac{1}{x}$ est convexe. En vertu de l'inégalité de Jensen ([17],pp. 153-154), nous avons

$$d(n) = E(\frac{1}{dep(w)}) \geqslant \frac{1}{E(dep(w))} = l(n).$$

□

Exemple 1.3 Soit $A = \{a, b\}$ avec $ab \simeq ba$. Le monoïde est alors commutatif et $dep(w) = \max\{|w|_a, |w|_b\}$. Par conséquent :

$$d(n) = \frac{n}{2^n} \sum_{i=0}^{n} \binom{n}{i} / max\{i, n-i\}, \tag{1.7}$$

et

$$l(n) = \frac{1}{n2^n} \sum_{0 \leqslant i \leqslant n} \frac{\binom{n}{i}}{max\{i, n-i\}}. \tag{1.8}$$

Le calcul de $l(n)$ donne :

$$l(n) = \frac{1}{2} + \binom{n-1}{\lfloor \frac{1}{2} n \rfloor} / 2^n. \tag{1.9}$$

Quand n tend vers l'infini, le second terme tend vers 0. D'où,

$$\lim_{n \to \infty} l(n) = \frac{1}{2}. \tag{1.10}$$

Par ailleurs, $d(n)$ étant majoré par 2, et minoré par $\frac{1}{l(n)}$, tend vers 2, lorsque $n \to \infty$.

Cet exemple montre la difficulté de trouver l'expression exacte du degré moyen. Il serait donc souhaitable de développer une méthode pour trouver sa valeur asymptotique.

1.4 Comportement asymptotique

Avant de développer cette section, reconsidérons le système de l'exemple précédent. Peut-on généraliser la définition de la notion de degré de parallélisme pour les mots infinis ? Considérons la suite $\{w_n\}$ de mots définis par la table suivante :

n	1	2	3	4	5	$\cdots$
w_n	ab	aba^2	aba^2b^2	$aba^2b^2a^6$	$aba^2b^2a^6b^6$	$\cdots$
$d(w_n)$	2	4/3	2	4/3	2	$\cdots$

Si n est impair, alors $d(w_n) = 2$ et l'on pose $w_{n+1} = w_n a^k$, où k est tel que $|w_n a^k|_a = 3|w_n|_b$. Alors $d(w_{n+1}) = \frac{4}{3}$. Si n est pair, alors $d(w_n) = \frac{4}{3}$ et l'on pose $w_{n+1} = w_n b^{k'}$, où k' est tel que $|w_n b^{k'}|_b = 3|w_n|_n$. Ainsi $d(w_{n+1}) = 2$.

Cet exemple simple montre la difficulté de définir une notion de degré de parallélisme pour les mots infinis en général, vue que si $\{w_n\}$ est la suite définie ci-dessus, $d(w_n)$ prend les valeurs 2 et $\frac{4}{3}$, donc ne peut converger quand n tend vers l'infini.

Cependant, si l'on suppose que a et b ont la même probabilité d'apparaître dans un mot infini w , utilisant la loi faible des grands nombres ([17],chap. X), on a, avec probabilité 1, :

$$|w|_a/|w| \to \frac{1}{2} \text{ et } |w|_b/|w| \to \frac{1}{2} \text{ quand } |w| \to \infty, \tag{1.11}$$

et donc avec probabilité 1 :

$$\begin{aligned} d(w) &= |w|/dep(w) \\ &= 1/\max\{|w|_a/|w|, |w|_b/|w|\} \\ &\to 2. \end{aligned} \tag{1.12}$$

Remarque. Notons que nous avons supposé que a et b ont la même probabilité d'apparaître dans le mot. Cependant, si on suppose que a et b apparaissent respectivement avec probabilités α et β, alors avec probabilité 1, :

$$\max\{|w|_a/|w|, |w|_b/|w|\} \to \max\{\alpha, \beta\}, \tag{1.13}$$

ainsi

$$d(w) \to 1/\max\{\alpha, \beta\}. \tag{1.14}$$

Le parallélisme est alors maximal si $\alpha = \beta = 1/2$.

Cette convergence est intéressante. Dans la suite, on verra que, avec probabilité 1, le degré de parallélisme existe et tend vers une limite qui ne dépend que du monoïde de commutation avec probabilité 1, [65].

1.4.1 Modélisation stochastique

Le processus de factorisation de la section 1.3 peut être vu comme un processus aléatoire : à la n^{ieme} étape, l'état de la forme normale du mot est $u_1u_2\cdots u_k$. Une lettre a est insérée avec la probabilité $1/N$, l'état deviendra :

(i) $u_1u_2\cdots u_ka$ si a ne commute pas avec u_k, ou bien

(ii) $u_1u_2\cdots u'_i\cdots u_k$ si a ne commute pas avec u'_i.

C'est une chaîne de Markov dont l'ensemble des états est l'ensemble infini, mais dénombrable, de toutes les factorisations possibles. Nous nous intéressons alors au nombre de facteurs, sa valeur est k à la $n^{ième}$ étape et il passe à $k+1$ (cas (i)) ou bien reste k, (cas (ii)).

Désignons par ν_n la probabilité que la $n^{ième}$ insertion incrémente la profondeur du mot. Alors, on peut démontrer que si le graphe de conflit est connexe, la chaîne de Markov est érgodique et ν_n converge vers une limite ν, (voir [65]).

Proposition 3 *Si le monoïde de commutation a un graphe de conflit connexe, alors le processus de factorisation admet une distribution de probabilité stationnaire ν pour l'incrémentation de la profondeur. Le degré de parallélisme est alors $1/\nu$.*

Exemple 1.4 Soit $S = (\{a,b,c,d\}, \{ab \simeq ba\})$. Sachant que le graphe de conflit est connexe, on peut utiliser la proposition 3 pour obtenir le degré de parallélisme. Il suffit de calculer la limite de ν_n quand n tend vers l'infini. Il faut donc calculer la probabilité d'incrémentation de la profondeur à la $n^{ième}$ insertion. On obtient facilement

$$\nu_n = 1/3 + 1/3 + (1/3) \sum_{1\leqslant i\leqslant (n-1)/2} (2/3)^{2i} \binom{2i}{i} (1/2)^{2i}. \tag{1.15}$$

Ainsi,

$$\begin{aligned} \nu &= \lim_{n\to\infty} \nu_n \\ &= 2/3 + (1/9) \textstyle\sum_{i\geqslant 0} \binom{2i}{i} (1/9)^i \\ &= 2/3 + (1/9)(1-4/9)^{-1/2} \\ &= 2/3 + 1/(3\sqrt{5}), \end{aligned} \tag{1.16}$$

Il en résulte que le degré de parallélisme du système est égal à $3/(2+(1/\sqrt{5}))$.

Remarque. La méthode ci-dessus peut être appliquée pour tout monoïde de commutation ; l'expression $(1/n)\sum_{1\leqslant i\leqslant n} \nu_i$ (ou bien ν_n si $\lim_{n\to\infty} \nu_n$ existe) donne une bonne approximation pour l'inverse du degré de parallélisme. Cependant, il n'est pas

toujours facile de l'appliquer pour des cas plus compliqué que l'exemple ci-dessus.

On reviendra sur ce calcul dans la section 2.4 du chapitre suivant en donnant plus d'outils en termes de chaînes de Markov.

1.4.2 Facteurs saturés et théorie de renouvellement

Revenons sur la modélisation par un processus stochastique présentée dans la section 1.3. On peut voir que quand la factorisation se termine avec un facteur u_k, son évolution future ne dépend pas seulement de u_k mais des facteurs précédents en général. Cependant, il y a des cas qui font exception. Il y a des facteurs qui *renouvellent* le processus de factorisation.

Définition 5 Un facteur u est dit *saturé* s'il est maximal pour l'inclusion. Ainsi, u est saturé si et seulement si
(i) $x, y \in u, x \neq y \Rightarrow xy \simeq yx$,
(ii) $x \in A, y \notin u \Rightarrow xy \not\simeq yx$.
On dit que le processus de factorisation *arrive à une saturation* si son dernier facteur est saturé.

Remarque.
(i) Si A est commutatif, alors le seul facteur saturé est A lui même. Si $\theta = \emptyset$, alors les facteurs saturés sont les singletons.
(ii) Un facteur saturé est une clique maximale dans le graphe de commutation.
(iii) Quand le processus rencontre une saturation, il est regénéré, c'est-à-dire que son futur ne dépend alors pas de ce qui s'est passé dans le passé.

Rappelons quelques résultats de [65] qui nous seront utiles dans le chapitre suivant.

Lemme 1 *Si le graphe de conflit G du système de commutation S est connexe, alors l'événement récurrent est apériodique et persistant et admet un temps d'attente à espérance mathématique finie.*

La proposition suivante montre comment ce lemme peut être utilisé pour étudier le degré de parallélisme d'un système de commutation quelconque.

Proposition 4 *Soit $\mathcal{R}$ un événement récurrent et persistant avec une espérance mathématique du temps d'attente finie μ. Soit h l'espérance mathématique de la*

profondeur dans une intervalle. Si le mot w_n *est choisi au hasard de* A^n*, alors quand* $n \to \infty$*,*

$$d(w_n) \to \mu/h \tag{1.17}$$

avec probabilité 1*.*

On finit cette partie par un théorème central.

Théorème 1 *On suppose que le graphe de conflit* G *du système de commutation* $S = (A, \theta)$ *a* k *composantes connexes* $G_1 = (A_1, E_1), \cdots, G_k = (A_k, E_k)$*. Soit* $\alpha_i = card(A_i)/N$ *pour* $i = 1, \cdots, k$*. Soit* w_n *un mot de longueur* n *choisi au hasard. On a avec probabilité* 1

$$d(w_n) \to DEG(G), \text{ quand } n \to \infty, \tag{1.18}$$

où $DEG(G) = DEG(S)$ *est défini par*

$$DEG(G) = \min\{DEG(G_1)/\alpha_1, \cdots, DEG(G_k)/\alpha_k\}. \tag{1.19}$$

Ainsi, l'étude du degré de parallélisme de tout monoïde de commutation se ramène à l'étude de celle des monoïdes de commutation dont le graphe de conflit est connexe.

Chapitre 2

Méthodes pour calculer le degré de parallélisme dans les monoïdes de commutation

Dans ce chapitre, nous nous intéressons aux comportements asymptotiques du degré moyen $d(n)$. Cette étude permet le calcul exact de degré dans des cas particuliers. Rappelons que, d'après le chapitre précédent, si les lettres sont tirées au hasard avec la même probabilité $\frac{1}{N}$, où N est la cardinalité de l'alphabet A, c'est-à-dire toutes les lettres apparaissent avec la même probabilité $1/N$, alors

$$\lim_{n\to\infty} d(n) = DEG. \tag{2.1}$$

DEG est complètement déterminé par le système de commutation.

2.1 Profondeur moyenne et degré de parallélisme

Revenons à la profondeur moyenne relative des mots de longueur n $l(n)$, introduite dans le chapitre précédent (définition 4). Avec $l(n)$ ainsi définie, l'espérance mathématique du nombre de facteurs (ou de la profondeur) des mots de longueur n est $nl(n)$.

Soit ν_i la probabilité que la $i^{ième}$ lettre insérée ne commute pas avec le dernier facteur. ν_i est donc la probabilité pour la profondeur d'être incrémentée à la $i^{ième}$ insertion. On a

$$nl(n) = \sum_{1\leqslant i\leqslant n} \nu_i,\ n = 1, 2, 3, \cdots \tag{2.2}$$

Nous avons le théorème suivant :

Théorème 2 *Pour tout système de commutation $S = (A, \theta)$, on a*

$$\lim_{n\to\infty} l(n) = \frac{1}{DEG}. \tag{2.3}$$

Preuve. Par la propriété de convergence, on a

$$d(w_n) \to DEG \text{ avec probabilité 1 , quand } n \to \infty.$$

Donc, quand $n \to \infty$, avec probabilité 1

$$dep(w_n)/n \to 1/DEG.$$

Par ailleurs, $l(n)$ étant l'espérance mathématique de la variable aléatoire $dep(w_n)/n$, cette v.a. converge avec probabilité 1 vers $l(n)$. D'où le théorème, voir Feller [17]. □

2.2 Exemples

Les exemples présentés dans cette section sont relativement simples. L'objectif étant d'illustrer l'application des propositions et théorèmes ci-dessus pour calculer le degré de parallélisme.

Exemple 2.1 Soit $A = \{a, b, c\}$ et soit θ la relation de commutation définie par $ac \simeq ca$, $bc \simeq cb$. Le graphe de conflit de ce système est donné par la figure 2.1. Il est formé de deux composantes connexes , et le degré de parallélisme de chacune est 1. Donc par le théorème 1, le degré de parallélisme de tout le système est $\frac{3}{2}$.

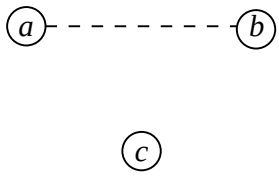


FIGURE 2.1 –

Exemple 2.2 Soit $A = \{a, b, c\}$ et θ définie par $ab \simeq ba$. Le graphe de conflit est donné par le figure 2.2. Le facteur c est saturé ; l'événement de la rencontre d'une occurrence de c est un événement récurrent qui est persistant positif. Donc en

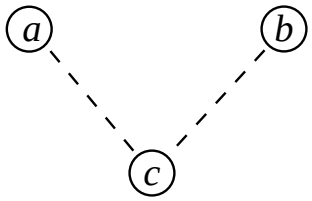


FIGURE 2.2 –

vertue de la proposition 3, il suffit d'étudier les paramètres qui nous intéressent dans un intervalle entre deux occurrences de c (la première occurrence est exclue et la dernière est incluse). Soit M la longueur d'un tel intervalle. Il admet une distribution géométrique

$$P(M=m)=\frac{1}{3}\left(\frac{2}{3}\right)^{m-1},\ \ m=1,2,3,\cdots$$

avec

$$\mu=E(M)=3.$$

Il reste à calculer l'espérance mathématique de la profondeur dans un intervalle. Pour un intervalle de longueur fixée, nous somme dans la situation de l'exemple 1.3, et la formule 1.9 permet d'écrire l'espérance conditionnelle (conditionnée par la longueur de l'intervalle égale à m) de *dep* :

$$D_m=\frac{1}{2}(m+1)+(m-1)\frac{\left(\begin{array}{c} m-2 \\ \lfloor\frac{1}{2}(m-1)\rfloor \end{array}\right)}{2^{m-1}},\ m=1,2,... \tag{2.4}$$

ou en termes de séries génératrices :

$$D(z)=\sum_{m\geqslant 1}D_m z^m=\frac{z}{1-z}+\frac{z^2}{2(1-z)^2}+z\sum_{m\geqslant 1}m\left(\begin{array}{c} m-1 \\ \lfloor\frac{1}{2}m\rfloor \end{array}\right)(\frac{z}{2})^m. \tag{2.5}$$

Et nous aurons

$$h=\sum_{m\geqslant 1}\frac{1}{3}(\frac{2}{3})^{m-1}D_m=\frac{1}{2}D(\frac{2}{3}). \tag{2.6}$$

La somme du second membre de 2.5 est de la forme $A(z)+B(z)$, avec

$$A(z)=\sum_{n\geqslant 1}\frac{(2n)!}{n!(n-1)!}(\frac{z^2}{4})^n \tag{2.7}$$

et

$$B(z)=z\sum_{n\geqslant 0}\frac{(2n+1)!}{n!n!}(\frac{z^2}{4})^n. \tag{2.8}$$

Ces fonctions se calculent facilement à l'aide des opérations élémentaires sur la série génératrice des nombres de Catalan $C(z) = \frac{1-\sqrt{1-4z^2}}{2z^2}$. Un calcul simple permet d'obtenir

$$A(z) = \frac{1}{2}z^2(1-z^2)^{-3/2} \tag{2.9}$$

et

$$B(z) = \frac{1}{2}z(1-z^2)^{-1/2} + \frac{1}{2}z^3(1-z^2)^{-3/2}. \tag{2.10}$$

Nous obtiendrons finalement de 2.4, 2.7 et 2.8 :

$$h = 2 + \frac{1}{\sqrt{5}}, \tag{2.11}$$

et donc

$$DEG = \frac{3}{2+1/\sqrt{5}} = 1,225884003552664.... \tag{2.12}$$

Exemple 2.3 Comparons, comme un dernier exemple de cette section, les deux systèmes de commutation S_1 et S_2 donnés par leurs graphes de conflit G_1 et G_2 des figures 2.3 et 2.4.

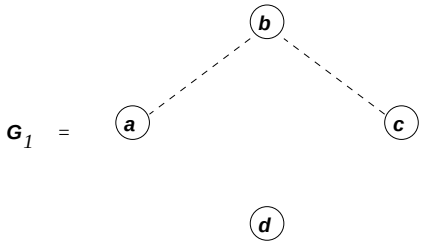


FIGURE 2.3 – Graphes de conflit de S_1 de l'exemple 2.3

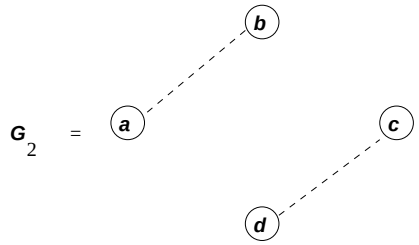


FIGURE 2.4 – Graphes de conflit de S_2 de l'exemple 2.3

Pour S_1, nous aurons, par le théorème 1 du chapitre précédent et le calcul de l'exemple précédent,

$$DEG_1 = min\{4, \frac{4}{3}\frac{3}{2+1/\sqrt{5}}\} = \frac{4}{2+1/\sqrt{5}} = 1.63451200473... \tag{2.13}$$

Chacune des composantes de G_2 correspond à un monoïde non commutatif et est donc de degré 1. On en déduit alors par le théorème 1 du chacune précédent,

$$DEG_2 = 2. \tag{2.14}$$

Par conséquent S_2 est nettement "plus parallèle" que S_1.

2.3 Méthode de diagonalisation

Cette section exploite la proposition 4. Supposons qu'il existe une lettre a qui ne commute avec aucune autre lettre. L'événement "rencontrer a dans le processus de factorisation" constitue un événement renouvelant d'espérance mathématique de durée d'attente $\mu = N$. Pour appliquer la proposition 4, on doit calculer h qui est l'espérance mathématique de la profondeur dans un intervalle. Il y a des cas où le nombre de facteurs peut être exprimé comme la répétition maximale de m événements. Si la distribution des m événements est connue par une fonction génératrice à m variables, la méthode de diagonalistation des séries permet de calculer la distribution du nombre maximal d'occurrences d'une lettre dans un intervalle conditionnée par la longueur d'un intervalle.

Rappelons d'abord la diagonalistation d'une série, pour plus de détails, voir [25]. Étudions l'exemple suivant, ce qui nous permettra de clarifier les circonstances dans lesquelles la méthode peut être appliquée.

Considérons une série formelle à m variables à des coefficients sur l'ensemble des nombres complexes :

$$f(x_1, x_2, \cdots, x_n) = \sum_{n_i \geqslant 0} a_{n_1 n_2 \cdots n_m} x_1^{n_1} x_2^{n_2} \cdots x_m^{n_m} \tag{2.15}$$

La diagonale de f, notée $\mathcal{D}_f$, est une série formelle à une variable définie par :

$$\mathcal{D}_f(t) = \sum a_{nn\cdots n} t^n. \tag{2.16}$$

Pour $m = 2$, on a

$$\mathcal{D}_f(t) = (1/2\pi i) \int_{|\zeta|=\varepsilon} f(\zeta, t/\zeta) d\zeta/\zeta, \tag{2.17}$$

où ε et $|t|$ sont suffisamment petits pour que $f(z, x)$ soit régulière lorsque $|z| < \varepsilon$ et $|x| < t/\varepsilon$. Comme dans [25], notons que si f est rationnelle, alors $\mathcal{D}_f$ est une fonction algébrique. Pour f rationnelle et $m \geqslant 3$, l'intégration sur plusieurs contours de la forme ci-dessus aboutit en général à une fonction transcendante.

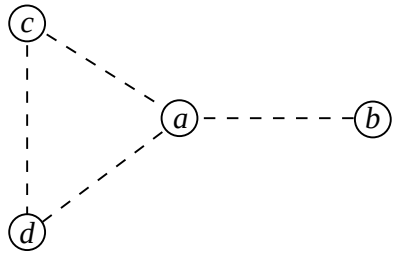


FIGURE 2.5 – Graphe de conflit de l'exemple 2.4.

Exemple 2.4 Soit $S = (A, \theta)$ le monoïde de commutation représenté par le graphe de conflit de la figure 2.5.

On a $\mu = 4$. Pour calculer h, soit w un mot de longueur n sur l'alphabet $\{b, c, d\}$. La profondeur de w est $\max\{|w|_b, |w|_{\{c,d\}}\}$. On a

$$\begin{aligned} g(x,y) &= \sum Pr(|w|_b = \alpha, |w|_{\{c,d\}} = \beta) x^\alpha y^\beta \\ &= \tfrac{1/4}{1-(3/4)(x/3+2y/3)} \end{aligned} \tag{2.18}$$

La fonction génératrice correspondant aux probabilités cumulées est

$$\begin{aligned} f(x,y) &= \sum Pr(|w|_b \leqslant \alpha, |w|_{\{c,d\}} \leqslant \beta) x^\alpha y^\beta \\ &= \tfrac{1}{(1-x)(1-y)(4-(x+2y))}. \end{aligned} \tag{2.19}$$

Ainsi,

$$\begin{aligned} q(t) &= \sum Pr(\max\{|w|_b, |w|_{\{c,d\}}\} \leqslant n) t^n \\ &= \tfrac{1}{2\pi i} \textstyle\int_{|\zeta|=\varepsilon} f(\zeta, t/\zeta) \tfrac{d\zeta}{\zeta}, \end{aligned} \tag{2.20}$$

où ε et t satisfont les conditions énoncées ci-dessus. Un calcul standard permet de trouver

$$q(t) = \frac{1}{(1-t)(2-t)} + \frac{(2-r)}{2(r-1)(2-t-t)r}, \tag{2.21}$$

avec $r = \sqrt{4-2t}$.

On en déduit le nombre moyen de facteurs entre deux occurrences de a,

$$\begin{aligned} h &= 1 + \tfrac{d}{dt}[(1-t)q(t)]_{t=1} \\ &= \tfrac{5+\sqrt{2}}{2}. \end{aligned} \tag{2.22}$$

Donc

$$DEG = \mu/h = \frac{8}{5+\sqrt{2}}. \tag{2.23}$$

2.4 Chaînes de Markov et degré de parallélisme

Si le graphe de conflit est connexe, le processus de factorisation peut être modélisé par une chaîne de Markov érgodique. La probabilité ν_n d'incrémentation de la profondeur admet une probabilité stationnaire ν. Le degré de parallélisme étant l'inverse de la probabilité stationnaire d'incrémentation de la profondeur, il suffit alors de calculer cette probabilité.

Cependant, ν n'est pas toujours facile à calculer. Mais, il est possible dans certains cas de réduire l'ensemble des états possibles du processus, afin d'obtenir une marche aléatoire simple.

Considérons l'exemple suivant :

Exemple 2.5 Soit $S = (\{a, b, c, d, \}, \theta)$, où θ est représentée par le graphe de la figure 2.6.

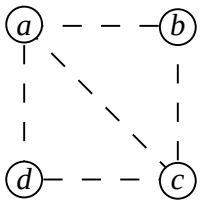


FIGURE 2.6 – Graphe de conflit de l'exemple 2.5

On identifie l'ensemble des états de la chaîne avec l'ensemble des entiers $\mathbb{Z}$. Par définition, l'état actuel de la chaîne est k, si on peut insérer exactement k occurrences consécutives de b sans créer de nouveau facteur ; il sera $-k$, si on peut insérer exactement k occurrences consécutives de d sans créer de nouveau facteur. Le premier état correspond au fait que le processus de factorisation se termine par exactement k occurrences du facteur (d) et le second au fait que le processus se termine par exactement k occurrences de (b).

Les facteurs (a) et (c) sont deux facteurs saturés et donc, une insertion de l'un de ces facteurs remet le processus à l'état 0. On obtient la marche aléatoire dont le graphe de transitions est donné par la figure 2.7.

Ainsi, la matrice de transition est :

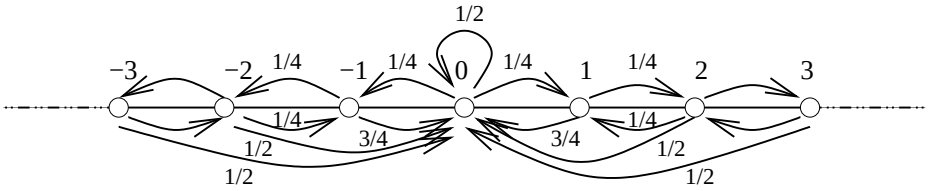


FIGURE 2.7 – La marche aléatoire sur $\mathbb{Z}$ correspondant à l'exemple 2.5

	...	-2	-1	0	1	2	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	...	0	0	1/2	1/4	1/4	...
1	...	0	0	3/4	0	1/4	...
0	...	0	1/4	1/2	1/4	0	...
-1	...	1/4	0	3/4	0	0	...
-2	...	0	1/4	1/2	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Il est clair qu'il n'y a pas besoin de créer un nouveau facteur à l'état k si et seulement si ou bien la lettre insérée est b et $k < 0$ ou bien d et $k > 0$. On a donc

$$\nu = \frac{1}{4} + \frac{3}{4}\pi_0, \tag{2.24}$$

où π_0 est la probabilité stationnaire de l'état 0. On obtient facilement $\pi_0 = 1/\sqrt{3}$ et par conséquent

$$\nu = \frac{1+\sqrt{3}}{4}. \tag{2.25}$$

Donc le degré de parallélisme recherché est

$$DEG = 2\sqrt{3} - 2. \tag{2.26}$$

Exemple 2.6 La même méthode peut être appliquée pour l'exemple 1.1. On identifie les états de la chaîne de Markov avec le nombre des derniers facteurs qui peuvent absorber une lettre donnée dans le processus de factorisation. Ainsi, l'état S_n est l'état du processus se terminant avec $(a)^n, (b)^n, (c)^n$ et $(a)^n$, appelés *facteurs dominants*, tel que le facteur situé à gauche de ces n facteurs ne commute avec aucune lettre qui commute avec les facteurs.

Par symétrie de la structure de commutation, on peut identifier les états de ces facteurs. Le prochain facteur dominant sera :

(i) $(a)^{n+1}$ si la lettre insérée est a,

(ii) (b) si la lettre insérée est b, sachant que tout ce qui commute avec b ne commute pas avec a,
(iii) $(a)^{n-1}$ si la lettre insérée est c, ou encore
(iv) (d) si la lettre insérée est d, sachant que tout ce qui commute avec d ne commute pas avec a,

Ainsi, pour tout $n \geqslant 1$, les probabilités de transition sont

$$p_{n,n+1} = 1/4,\ p_{n,n-1},\ p_{n,1} = 1/2. \tag{2.27}$$

Pour $n = 0$, la troisième close échoue et $(a)^{n-1}$ doit être remplacé par (c).

Donc :

$$p_{0,1} = 1. \tag{2.28}$$

La matrice de transition est alors

$$\begin{array}{c} \\ S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} S_0 & S_1 & S_2 & S_3 & S_4 & \cdots \end{array} \\ \left(\begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & \cdots \\ 1/4 & 1/2 & 1/4 & 0 & 0 & \cdots \\ 0 & 3/4 & 0 & 1/4 & 0 & \cdots \\ 0 & 1/2 & 1/4 & 0 & 1/4 & \cdots \\ 0 & 1/2 & 0 & 1/4 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \right) \end{array}$$

D'autre part, il est facile de voir que la lettre insérée est absorbée par un facteur déjà existant dans le cas (iii), mais la création d'un nouveau facteur est nécessaire dans tous les autres cas. Ainsi, la probabilité stationnaire d'incrémentation de la profondeur ν est donnée par

$$\nu = \frac{3}{4} + \frac{1}{4}\pi_0, \tag{2.29}$$

où π_0 est la probabilité stationnaire de S_0. On obtient facilement

$$\pi_0 = (1/\sqrt{3})(2 - \sqrt{3}), \tag{2.30}$$

et par conséquent

$$\nu = 1/2 + 1/(2\sqrt{3}). \tag{2.31}$$

Ainsi, le degré de parallélisme recherché est

$$DEG = 2/(1 + 1/\sqrt{3}). \tag{2.32}$$

La méthode utilisée dans ces deux exemples peut être généralisée pour l'étude du cas suivant. Soit $S = (A, \theta)$ le système de commutation dont le graphe de conflit G est défini comme suit : soit K_1 et K_2 deux graphes complets et soit G l'union disjointe de K_1 et K_2, comme représenté dans la figure 2.8. Dans ce graphe, les lettres représentées par les sommets blancs ne commutent avec aucune autre lettre, mais celles en gris commutent avec les lettres de même couleur mais pas avec les autres.

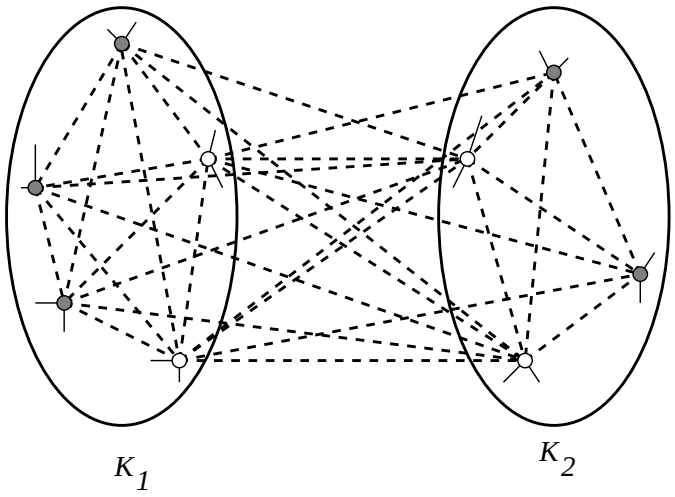


FIGURE 2.8 –

Formellement, on a deux graphes complets $K_1 = (V_1, E_1)$ et $K_2 = (V_2, E_2)$, tels que $V_1 = V_1^1 \cup V_1^2$, $V_2 = V_2^1 \cup V_2^2$ et $\forall (a, b) \in V_1 \times V_2$, $(a, b) \in \theta$ si et seulement si $a \in V_1^1$ et $b \in V_2^1$.

$$\text{Soient } p = \frac{|V_1^1|}{|V_1| + |V_2|}, \quad q = \frac{|V_2^1|}{|V_1| + |V_2|}, \quad \text{et } r = \frac{|V_1^2| + |V_2^2}{|V_1| + |V_2|}, \quad (p + q + r = 1).$$

Considérons la chaîne de Markov dont les états sont les éléments de $\mathbb{Z}$. Le processus est à l'état $k > 0$ si k lettres de V_1^2 peuvent être insérées sans avoir à créer un nouveau facteur. C'est le cas si la factorisation se termine avec exactement k facteurs, chacun étant réduit à une lettre de V_2^2. Le processus est à l'état $k < 0$ si k lettres de V_2^2 peuvent être insérées sans avoir à créer un nouveau facteur. C'est le cas si la factorisation se termine avec exactement k facteurs, chacun étant réduit à une lettre de V_2^1. On obtient une marche aléatoire similaire à celle de l'exemple 2.5, avec la matrice de transition suivante.

	...	−2	−1	0	1	2	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	...	0	0	r	q	0	...
1	...	0	0	$r+q$	0	p	...
0	...	0	q	r	p	0	...
−1	...	q	0	$r+p$	0	0	...
−2	...	0	p	r	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Un calcul simple similaire à celui des exemples précédents donne $\pi_0 = rx_2/(x_2 - 2p)$, où x_1 et x_2 sont les racines réelles de l'équation $qx^2 - x + p = 0$, (rappelons que $r > 0$ et donc $1 - 4pq > 0$), c'est-à-dire

$$x_1 = \frac{1 - \sqrt{1-4pq}}{2q} \text{ et } x_2 = \frac{1 + \sqrt{1-4pq}}{2q}. \tag{2.33}$$

D'autre part, il est facile de voir qu'une nouvelle lettre insérée n'est pas absorbée par un facteur déjà existant si la marche aléatoire va de n à $n+1$ ou 0 si $n > 0$, ou bien de n à $n-1$ ou 0 si $n < 0$. Ainsi

$$\nu = \pi_0 + \sum_{n>0}(1-q)\pi_n + \sum_{n<0}(1-p)\pi_n, \tag{2.34}$$

où π_n est la probabilité stationnaire de l'état n. Ainsi, le degré de parallélisme du système est

$$DEG = \frac{r}{r + ((1-r)qx_1 - 2pq)\pi_0}. \tag{2.35}$$

Les exemples étudiés dans cette section montrent comment l'ensemble des états de la chaîne peut se réduire à des marches aléatoires unidimensionnelles sur $\mathbb{Z}$. Comme on verra dans la prochaine section, il est toujours possible de construire une marche aléatoire sur $\mathbb{N}^N$, simulant le comportement du processus de factorisation pour tout système de commutation.

2.5 Approche utilisant les marches aléatoires

Dans cette section, nous donnons une méthode générale pour une approximation du degré de parallélisme. Soit $M = (A, \theta)$ un monoïde de commutation. Les lettres de A sont notées $a_1, a_2, ..., a_N$. Au processus de factorisation correspond une marche aléatoire sur $\mathbb{N}^N$ comme suit : A chaque étape de la factorisation, l'état est déterminé

par un $N-$vecteur $x = (x_1, ..., x_N)$, de $\mathbb{N}^N$, où au moins une composante est nulle. L'état $(x_1, x_2, \cdots, x_N)$ correspond au fait que le processus peut absorber exactement x_i occurrences de la lettre a_i, sans avoir à créer un nouveau facteur. L'état initial est $(0, ..., 0)$. Si une nouvelle lettre a_i est à insérer, le nouvel état sera calculé par l'algorithme 1.

Algorithme 1: L'algorithme pour obtenir le nouvel état de la marche aléatoire.

```
début
    si x_i > 0 alors
        x_i ← x_i − 1;
        pour j de 1 à N faire
            si a_j ne commute pas avec a_i alors x_j ← min{x_j, x_i};
    sinon
        pour j de 1 à N faire
            si a_j commute avec a_i alors x_j ← x_j + 1;
            sinon x_j ← 0;
fin
```

On a alors la proposition suivante :

Proposition 5 *La marche aléatoire ci-dessus simule une étape d'insertion dans le processus de factorisation du système $S = (A, \theta)$. Si le graphe de conflit de S est connexe, alors la chaîne de Markov obtenue en affectant une probabilité $1/N$ à chaque occurrence de a_i, $i = 1, 2, .., N$, à chaque étape de la factorisation, admet une distribution limite (qui est la distribution stationnaire). Le degré de parallélisme est alors $1/\nu$, où ν est l'espérance mathématique du nombre de composantes égales à 0 par rapport à la distribution stationnaire.*

Preuve. Premièrement, on remarque qu'à chaque étape, $x = (x_1, ..., x_N)$ contient toutes les informations dans la mesure où chacun des x_i compte le nombre de a_i successifs que le processus peut absorber sans avoir à créer un nouveau facteur. Soit a_i la lettre insérée à une étape donnée.

(i) Si $x_i > 0$, alors x_i doit être réduit à $x_i - 1$. Et pour toute autre composante x_j, si a_j commute avec a_i, le même x_j reste valide et, si a_j ne commute pas avec a_i, x_j ne peut dépasser $x_i - 1$ car la lettre insérée a_i, introduit une barrière gauche pour une a_j absorption et, ainsi, la nouvelle valeur de la $j^{ième}$ composante sera $\min\{x_i, x_j\}$.

(ii) Si $x_i = 0$, alors un nouveau facteur (a_i) doit être crée. L'introduction de ce facteur augmente toute j^{ieme} composante pour qui a_j commute avec a_i, les autres composantes sont réduites à 0.

Donc, l'évolution du processus de factorisation correspond à la description algorithmique ci-dessus. Un argument similaire à celui du lemme 1 dans [65] montre que si le graphe de conflit est connexe, alors la chaîne admet une distribution de probabilité stationnaire. On a à créer un nouveau facteur dans (ii). Donc, la probabilité stationnaire d'incrémentation de la longueur est la même que l'espérance mathématique du nombre de composantes nulles dans le vecteur x. □

Remarque. La méthode permet d'écrire le degré de parallélisme d'un système de commutation comme une distribution de probabilité stationnaire qui elle même est solution d'un système fini d'équations. Des méthodes très connues de l'analyse numérique donnent une bonne approximation.

2.6 Une méthode d'approximation

Dans cette section, nous proposons une méthode fondée sur la notion de gain et de réduction pour approcher le degré de parallélisme dans les monoïdes de commutation.

Soit $S = (A, \theta)$ un monoïde de commutation, et $G = (V, E)$ son graphe de conflit. Si w et w' sont deux mots donnés de A^n, $(n > 0)$ le *gain* sur w et w' est défini comme étant

$$g(w, w') = dep(w) + dep(w') - dep(ww'). \tag{2.36}$$

À présent, étant donné un entier positif n, on définit la *réduction* par rapport à n comme étant :

$$red(n) = \frac{1}{2nN^{2n}} \sum_{w,w' \in A^n} g(w, w'). \tag{2.37}$$

Il est alors clair que la réduction ainsi définie correspond à l'espérance mathématique de $\frac{g(w,w')}{2n}$.

Proposition 6 *Pour tout entier positif n :*

(i) $red(n) = l(n) - l(2n)$.

(ii) $\sum_{i=0}^{n} red(2^i) = 1 + l(2^{n+1})$.

Preuve.

(i) Se déduit directement de la définition.

(ii) Résulte de (i).

□

On a alors le théorème suivant :

Théorème 3 *On a*

$$\frac{1}{DEG} = 1 - \sum_{n \geqslant 0} red(2^n). \tag{2.38}$$

Preuve. La relation se déduit de (ii) de la proposition 6, et de la relation suivante.

$$\lim_{n \to \infty} l(2^{n+1}) = \frac{1}{DEG}. \tag{2.39}$$

□

Le théorème 3 donne un algorithme approximatif pour le calcul du degré de parallélisme d'un système de commutation. L'algorithme a à l'entrée le graphe de conflit, il simule alors un certain nombre de pas. Les étapes suivantes de l'algorithme sont fondées sur la construction d'un graphe $G_n = (V_n, E_n)$ comme suit :

— $V_n = \{w \in A^n\}$, c'est-à-dire, chaque sommet correspond à un mot de longueur n. Il est clair que $|V_n| = N^n$.

— E_n est défini comme suit. Étant donné deux mots w et w' de A^n, Il y a un arc orienté de w vers w' si $g(w, w') \neq 0$, on étiquette cet arc avec $g(w, w')$.

La construction correspondant au système $S = (A, \theta)$, avec $A = \{a, b, c\}$ et $\theta = \{(b, c)\}$ est illustrée dans la figure 2.9. Si on applique la méthode à ce cas particulier, on obtient après juste deux itérations

$$DEG \geqslant 1.1654676. \tag{2.40}$$

Par ailleurs, un calcul exact du degré de parallélisme de ce système [65] donne

$$DEG = 1.225884003552664... \tag{2.41}$$

Cela montre que dans cet exemple, l'algorithme fournit une suite qui converge rapidement vers la valeur exacte du degré recherché. Cette confirmation reste à prouver dans le cas général et pourra être l'objet de nouvelles investigations.

2.7 Relation avec la physique statistique

Les investigations menées dans les chapitres 1 et 2 peuvent être utiles dans la conception d'algorithmes de synchronisation ; elles permettent en effet de faire une

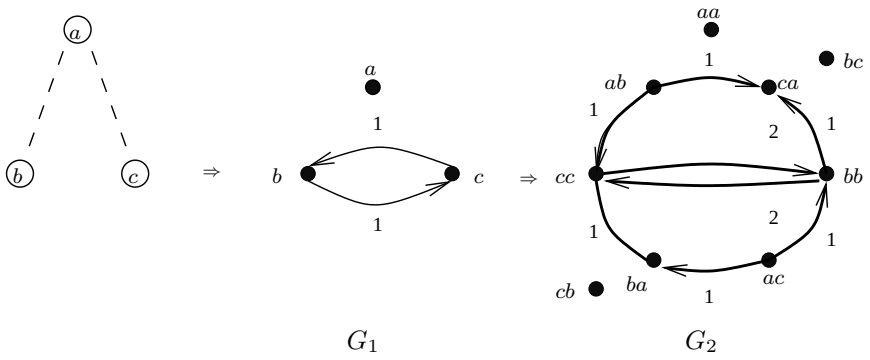


FIGURE 2.9 – La construction pour $n = 1$ et $n = 2$.

étude comparative de ces algorithmes. Il existe d'autres applications dans le domaine de la physique statistique. Ce lien est fort bien expliqué dans les travaux de Viennot sur les tas de pièces, [76].

À titre d'exemple, considérons le cas particulier du tas de pièces. On va montrer comment notre méthode peut être utilisée pour étudier ce problème. Dans [12, 15], A. Comtet et S. Nechaev s'intéressent au problème suivant. Considérons n colonnes, une particule tombe verticalement dans une colonne choisie au hasard, avec la contrainte que si deux particules tombent dans deux colonnes adjacentes, alors elles ne peuvent pas se trouver au même niveau. Le paramètre d'intérêt est la valeur moyenne de la hauteur maximale du tas résultant.

Ce problème peut être modélisé par le monoïde de commutation suivant. À chaque colonne i, on associe une lettre a_i, ce qui définit l'alphabet A. Le fait que deux particules tombant dans deux colonnes voisines ne peuvent pas être au même niveau est illustré par $a_i a_{i+1} \not\equiv a_{i+1} a_i,\ \forall i = 1, \cdots, n-1$. On a donc un monoïde de commutation (A, θ), où θ contient tout couple $(a_i, a_j),\ i \neq j$, sauf ceux de la forme $(a_i, a_{i+1});\ i = 1, 2, \cdots, n-1$ (le graphe de conflit est donc une chaîne à n sommets).

Le paramètre auquel on s'intéresse est la valeur de la hauteur maximale, H_{max}, si N particules sont tombées (à noter que les auteurs utilisent N pour la longueur des mots et n pour la taille de l'alphabet). La hauteur du tas est le nombre de facteurs de Foata si le mot est écrit sous sa forme normale de Foata. Dans une communication privée avec X. Viennot, les auteurs conjecturent le comportement asymptotique suivant de la valeur moyenne de la hauteur maximale, $< H_{max} >$: $< H_{max} > \sim \frac{N}{n} 4$, quand n et N tendent vers l'infini. La hauteur maximale n'étant autre que le nombre de facteurs si le mot est en sa $F.N.F$ et nous avons montré que le rapport entre ce nombre et N tend vers une valeur fixée déterminée par le système de commutation. Cependant, plus d'investigations sont nécessaires pour calculer ν introduite dans la marche aléatoire présentée dans la section 2.6. Et ainsi, pouvoir vérifier la conjecture.

2.8 Conclusion et perspectives

Soit $S = (A, \theta)$ un système de commutation. Si le graphe de conflit de S est connexe, le processus de factorisation (sous la forme normale de Foata) peut être modélisé par une chaîne de Markov (section 2.4). Le degré de parallélisme étant l'inverse de la probabilité stationnaire d'incrémenter la profondeur (proposition 3), nous avons pu, pour des cas particuliers où l'ensemble des états de la chaîne de Markov peut être réduit, calculer cette probabilité, et par conséquent calculer le degré de parallélisme.

Néanmoins, il est toujours possible d'écrire le degré de parallélisme en fonction de la distribution stationnaire de probabilités d'une marche aléatoire généralisée sur le treillis $\mathbb{N}^N$, (proposition 6 de la section 2.5). Elle permet ainsi d'écrire le degré de parallélisme en fonction d'une distribution stationnaire de probabilités, elle même solution d'un système fini d'équations. En utilisant des méthodes de l'analyse numérique, on peut alors obtenir une bonne approximation.

Le théorème 3 de la section 2.6 donne un algorithme, fondé sur la notion de gain et de réduction, pour faire une approximation du degré de parallélisme, cet algorithme est fondé sur des constructions successives. Sur des exemples simples, l'algorithme fournit une suite qui converge rapidement vers la valeur exacte du degré recherché. Cependant, cette affirmation reste à prouver dans le cas général.

Enfin, la section 2.7 établit un lien étroit entre le degré étudié et un problème classique de la physique statistique. La vérification de la conjecture sur la valeur moyenne de la hauteur maximale du tas de pièces nécessite plus d'investigations pour calculer la distribution introduite dans la marche aléatoire de la section 2.5.

Deuxième partie

Synchronisations locales probabilistes

Chapitre 3

Synchronisations locales dans les réseaux anonymes

Dans cette partie, on considère un réseau asynchrone de processeurs anonymes communiquant par échange de messages en mode asynchrone. Nous proposons et analysons des algorithmes probabilistes pour synchroniser des sommets voisins et permettant d'implémenter des calculs locaux dans un graphe.

3.1 Modèles et Définitions

Nous considérons le modèle standard des réseaux de communications point-à-point. Un réseau est décrit par un graphe orienté, symétrique, connexe et fini $G = (V, E)$. Les sommets représentent les nœuds ou les processeurs du réseau, les arcs représentent les liens de communications entre les processeurs. Le graphe étant supposé orienté symétrique, on le représentera sous la forme d'un graphe non orienté. Un sommet ne peut communiquer directement qu'avec ses voisins.

Rappelons quelques définitions ; plus de détails peuvent être trouvés dans [73].

Réseau synchrone, réseau asynchrone

Un réseau *synchrone* est un réseau ayant une horloge globale. Le temps est divisé en un nombre infini $0, 1, 2, \cdots$ d'*unités de temps*. Les processeurs commencent une exécution à l'instant $t = 0$. À chaque unité de temps t, un processeur p lit un message qui lui a été envoyé par un de ses voisins à l'instant $t-1$, si un tel message existe, il change alors d'état (c'est-à-dire, effectue un calcul quelconque) et envoie éventuellement au plus un message à l'un de ses voisins. Le nouvel état du processeur p et le message envoyé dépendent de son état à l'instant $t-1$ et du message qu'il a

reçu.

Un réseau asynchrone est un réseau sans horloge globale, chaque processeur a sa propre horloge, et peut évoluer dans le temps indépendamment de l'évolution des autres processeurs.

Réseau anonyme

Un réseau de processeurs est dit anonyme si les processeurs ne connaissent pas leurs identités.

Échange de messages en mode synchrone

Un réseau est à communication par échange de messages en mode *synchrone* si l'échange des messages entre deux processeurs voisins se fait après l'obtention d'un rendez-vous. L'émetteur est prêt à émettre et le récepteur est prêt à recevoir.

Échange de messages en mode asynchrone

Un réseau est dit à communication par échange de messages en mode *asynchrone* si :

— un processeur envoie un message à un autre processeur en le déposant dans le canal correspondant,

— il n'y a pas de borne supérieure sur le temps que met un message pour être délivré.

L'existence d'algorithmes permettant de résoudre certains problèmes dépend des hypothèses faites sur le réseau ou la connaissance qu'a chaque processeur sur le réseau, par exemple : réseau synchrone ou asynchrone, réseau anonyme ou non, connaissance de la topologie, de la taille.

3.2 Généralités sur les algorithmes probabilistes

Les algorithmes probabilistes sont des algorithmes qui utilisent des tirages de type pile ou face ou des générateurs de nombres aléatoires.

À la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'évolution de tels algorithmes.

Les algorithmes probabilistes sont utilisés pour "accroître" la rapidité de la résolution du problème posé ou encore, tout simplement, pour résoudre des problèmes n'admettant pas de solution déterministe. Par exemple, on a le théorème suivant dû à Angluin [1] et Itai et al [38] :

Théorème 4 *Sur un anneau anonyme de taille n, aucun protocole déterministe ne peut résoudre le problème de l'élection sans connaissance de la valeur de n par les processeurs.*

Cependant, étant donné un paramètre $\varepsilon > 0$ quelconque et quelque soit la taille n de l'anneau, il est possible de déterminer exactement n avec une probabilité d'erreur inférieure à ε grâce à un algorithme probabiliste se terminant par messages (sans détection de la terminaison).

Bien sûr, à cause du "hasard", de tels algorithmes peuvent fournir de fausses solutions ou encore ne pas se terminer du tout, cependant, si la probabilité que de tels événements se produisent est faible, il suffit d'exécuter l'algorithme plusieurs fois et on est "presque" sûr d'obtenir le "bon" résultat en un temps "réduit". Ce qui amène à distinguer deux catégories d'algorithmes probabilistes :

- Algorithmes de type *Las Vegas* : ce sont des algorithmes probabilistes qui
  - se terminent en un temps dont l'espérance mathématique est bornée et
  - retournent une solution correcte.
- Algorithmes de type *Monte Carlo* : ce sont des algorithmes probabilistes qui
  - se terminent en un temps fini et
  - retournent une solution correcte avec probabilité $\geqslant 1/3$.

3.3 Hypothèses et objectifs de l'étude

On considère un réseau asynchrone de processeurs anonymes, communiquant par échange de messages en mode asynchrone. Le réseau est modélisé par un graphe $G = (V, E)$.

L'objectif de cette étude est de fournir des outils pour implémenter des algorithmes classiques de l'algorithmique distribuée.

Un algorithme distribué est un ensemble de *transitions locales* correspondant à des calculs locaux effectués par les processeurs du réseau. Ces calculs sont exécutés de manière séquentielle par chaque processeur. Cet algorithme peut être vu comme une collection d'exécutions sur des processeurs qui, par échange d'informations, contribuent à la réalisation d'une tâche commune.

Nous nous intéressons à l'implémentation de trois types de calculs locaux :

- Calcul local de type LC_0 : dans ce type de calcul, un processeur échange des informations avec un de ses voisins et effectue une transition.
- Calcul local de type LC_1 : dans ce type de calcul, l'échange d'informations se fait entre un processeur et tous ses voisins, sa transition est fonction des informations qu'il a collecté. Deux calculs de ce type ne peuvent avoir lieu en parallèle que s'ils se produisent sur deux étoiles arêtes disjointes.
- Calcul local de type LC_2 : dans ce type de calcul, tous les processeurs d'une même étoile échangent des informations et effectuent une transition. Deux calculs de ce type ne peuvent avoir lieu en même temps que s'ils se produisent sur deux étoiles sommets disjointes.

Deux processeurs ne peuvent communiquer que s'ils sont voisins et s'ils sont tous les deux prêts à communiquer, on parle alors de *rendez-vous*. Or, dans [1], Angluin a prouvé qu'il n'existe pas d'algorithme déterministe pour implémenter un passage de messages en mode synchrone dans un réseau anonyme qui passe les messages en mode asynchrone.

On est donc amené à introduire des solutions *probabilistes* pour implémenter le rendez-vous, c'est-à-dire la synchronisation d'un processeur avec un de ses voisins, d'un processeur avec tous ses voisins, ou plus généralement, d'un processeur avec tous les processeurs à une distance $k \geqslant 1$.

Pour pouvoir implémenter les trois types de calculs locaux décrits ci-dessus, on a besoin d'algorithmes permettant de réaliser des synchronisations d'un processeur avec un de ses voisins, avec tous ses voisins, ou avec tous les processeurs à une distance inférieure ou égale à 2 de ce processeur. On distingue naturellement trois types de synchronisations : les synchronisations LS_0 ou $rendez-vous$ qui permettent de synchroniser un processeur avec un voisin afin de réaliser un calcul local de type LC_0, les synchronisations LS_1 pour synchroniser un processeur avec tous ses voisins et réaliser un calcul local de type LC_1 et enfin les synchronisations LS_2 implémentant les calculs locaux de type LC_2.

3.4 Exemples de calculs locaux

Dans cette section, on présente des exemples d'algorithmes distribués codés par des calculs locaux. Les exemples traités sont issus de [72, 46, 47]. Ils illustrent les trois cas vus précédemment.

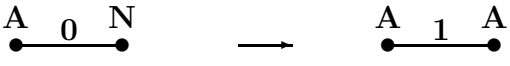
3.4.1 Calcul distribué d'un arbre couvrant

Dans cette section, on considère un algorithme distribué calculant un arbre couvrant d'un graphe connexe.

On suppose que l'on dispose d'un processeur dans un état "actif" (codé par l'étiquette **A**), les autres processeurs sont supposés être dans un état "neutre" (étiquette **N**) et les liens sont dans un état "passif" (étiquette **0**).

Initialement, l'arbre contient le seul sommet actif. À chaque étape de calcul, un sommet actif active un de ses voisins neutres et change l'étiquette du lien qui le relie à ce voisin en **1**. Le calcul s'arrête dès que tous les processeurs ont été activés. La figure 3.1 représente l'exécution de l'algorithme sur un graphe simple.

Une étape de l'algorithme peut être décrite par une réécriture utilisant la règle de réécriture R suivante :

R : 

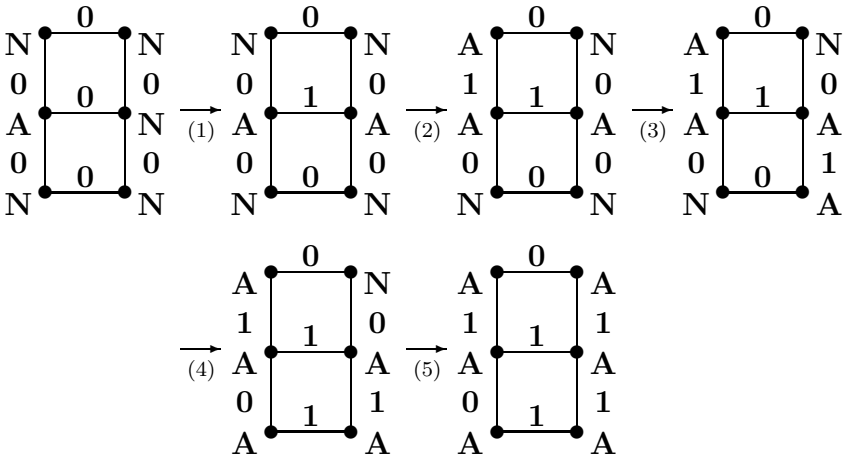


FIGURE 3.1 – Calcul d'un arbre couvrant.

Une application de cette règle de réécriture dans un graphe (ou un réseau) donné consiste en :

1. trouver dans le graphe un sous-graphe isomorphe à la partie gauche de la règle (appelé *occurrence* de la règle),
2. modifier les étiquettes comme décrit par la partie droite de la règle.

Pour implémenter cet algorithme dans un système distribué, avec les hypothèses ci-dessus, chaque sommet doit exécuter les actions suivantes : il obtient un rendez-vous avec l'un de ses voisins, ils échangent alors leurs états, s'ils réalisent une occurrence de la règle R, ils changent leurs états en appliquant cette règle, sinon ils recommencent.

3.4.2 Un algorithme pour détecter des propriétés stables

Dans cette section, nous décrivons l'algorithme de Szymanski, Shi et Prywes (algorithme SSP) [72].

Considérons un système (S) d'équations à résoudre dans un système distribué représenté par un graphe $G = (V, E)$. Chaque processeur dispose d'une équation du système (S) et s'emploie à la résoudre. Il s'agit de détecter la fin des calculs dans le réseau.

À chaque nœud v est associé un prédicat $P(v)$ (qui indique si l'équation a été résolue ou pas) et un entier $a(v)$ (qui indique la distance au delà de laquelle v ne sait pas si les processeurs ont fini leurs calculs ou non).
Initialement, $P(v)$ est faux et $a(v)$ est égal à -1. Les transformations des valeurs de $a(v)$ sont définies par les règles suivantes.

Soit v_0 un sommet, chaque calcul local agit sur l'entier $a(v_0)$ associé au sommet v_0 ; la nouvelle valeur de $a(v_0)$ dépend des valeurs associées aux sommets de $B(v_0, 1)$, boule de centre v_0 et de rayon 1. Plus précisément, soit v_0 un sommet et soit $\{v_1, ..., v_d\}$ l'ensemble de ses voisins.

— Si $P(v_0) = faux$ alors $a(v_0) = -1$;

— si $P(v_0) = vrai$ alors $a(v_0) = 1 + Min\{a(v_k) \mid 0 \leqslant k \leqslant d\}$.

Pour chaque sommet v, la valeur de $P(v)$ peut devenir vrai (l'équation est résolue), et elle le reste pour toujours.

L'algorithme SSP permet à un sommet v de savoir qu'un prédicat P est vrai pour chaque sommet de la boule de rayon $a(v)$. Dans [72], il est montré que le diamètre du graphe est une valeur seuil atteinte par un sommet dès que tous les prédicats de tous les sommets sont vrais.

Pour implémenter cet algorithme dans un réseau asynchrone, anonyme, à communication par passage asynchrone de messages, un sommet peut se synchroniser avec tous ses sommets voisins, collecter leurs états, et se réécrire en fonction de son état et de l'état de ses voisins.

3.4.3 Un algorithme d'énumération de Mazurkiewicz

C'est un algorithme pour énumérer les nœuds dans un graphe minimal pour les revêtements anonyme si on connaît sa taille [48].

Un *homomorphisme* entre deux graphes $G = (V(G), E(G))$ et $H = (V(H), E(H))$ est une application $\gamma : V(G) \to V(H)$ telle que si (u, v) est une arête de G, alors $(\gamma(u), \gamma(v))$ est une arête de H.
γ est un homomorphisme *surjectif* si γ est surjectif pour les sommets et les arêtes. C'est un *isomorphisme* si il est bijectif pour les sommets et les arêtes. Enfin, γ est localement bijectif si pour tout sommet v de $V(G)$, la restriction de γ à l'ensemble $N_G(v)$ des voisins de v est une bijection vers $N_H(\gamma(v))$.

Un graphe G est un *revêtement* d'un graphe H si il existe un homomorphisme surjectif de G sur H qui est localement bijectif. Un graphe G est minimal pour les revêtements s'il n'existe pas de graphe H tel que G est un revêtement de H et H est non isomorphe à G.

Description de l'algorithme

Une application d'une règle de transformation est faite sur un sous-graphe défini par un sommet v et les sommets voisins de v ; l'application d'une règle :

— dépend de l'étiquette de v et des étiquettes des sommets voisins de v,

— change l'étiquette de v et les étiquettes des nœuds voisins de v.

Soit G un graphe, à chaque nœud est associé un triplet (p, N, M) où :

— p est un entier positif,

— N est un ensemble fini d'entiers positifs,

— M est un ensemble de couples ordonnées de la forme (p, N).

Soit (p, N, M) le triplet associé au sommet v. L'ensemble $K(v)$ dénote l'ensemble des entiers utilisés dans le graphe et connus par v ; plus précisément :

$$K(v) = \{p\} \cup N \cup \{p_i | (p_i, N_i) \in M\} \cup \bigcup_{(p_i,N_i)\in M} N_i. \tag{3.1}$$

On a besoin de la relation d'ordre suivante. Soient N_1 et N_2 deux ensembles finis d'entiers positifs :

$$N_1 \prec N_2 \qquad si \qquad Max(N_1 \setminus N_2) < Max(N_2 \setminus N_1).$$

Initialement, chaque triplet associé à un sommet de G est égal à $(0, \emptyset, \emptyset)$. Chaque calcul local agit sur un sommet et les sommets voisins de ce sommet. Soit v_0 un sommet et soit $\{v_1, ..., v_d\}$ l'ensemble des sommets voisins de v. Soit (p_i, N_i, M_i) le triplet associé au nœud v_i.

1. S'il existe i, j tels que $0 \leqslant i < j \leqslant d$ et $M_i \neq M_j$ alors soit :
$$M = \bigcup_{k=0}^{k=d} M_k$$
pour tout $0 \leqslant i \leqslant d \quad M_i := M$.
2. Supposons que pour tous i, j tels que $0 \leqslant i < j \leqslant d$: $\quad M_i = M_j = M$,
si $(v_0 = 0)$ ou s'il existe $(p, N) \in M$ tel que $p_0 = p$ et $N_0 \prec N$
alors soit :
$$k = Max(\bigcup_{i=0}^{i=d} K(v_i))$$
pour tout $0 \leqslant i \leqslant d$, ajouter $(k + 1, N_0)$ à M_i, et pour tout $0 < i \leqslant d$, remplacer p_0 par $k + 1$ dans N_i et enfin, remplacer p_0 par $k + 1$ dans la première coordonnée du triplet associé à v_0.

Mazurkiewicz a prouvé qu'après une exécution de cet algorithme dans un graphe minimal pour les revêtements G, il y a une bijection entre les nœuds et les entiers de $[1, n]$ (où n est la taille de G).

Pour implémenter cet algorithme dans un réseau asynchrone, anonyme, communiquant par échange asynchrone de messages, un sommet v doit se synchroniser avec tous les sommets à une distance $\leqslant 2$ de v.

3.5 Algorithmes probabilistes pour l'implémentation des calculs locaux

Les algorithmes de la section précédente permettent de calculer des paramètres du réseau. Chaque processeur effectue un calcul local et le résultat global dépend de

la collection des calculs locaux.

Ces calculs ne peuvent se faire de manière déterministe dans un réseau asynchrone de processeurs anonymes communiquant par échange de messages en mode asynchrone [1], on est donc amené à proposer des solutions probabilistes.

Dans cette section, nous proposons trois algorithmes permettant de réaliser des synchronisations locales afin d'implémenter les calculs locaux de types LC_0, LC_1 et LC_2. Le premier algorithme permet de réaliser des synchronisations de type rendez-vous, c'est-à-dire d'un processeur avec un de ses voisins, ce qui permet d'implémenter l'algorithme de calcul d'arbre couvrant. Le deuxième algorithme permet de synchroniser un processeur avec tous ses voisins, ce qui permet d'implémenter l'algorithme SSP, et le dernier permet de synchroniser un processeur avec tous les processeurs à une distance $\leqslant 2$, ce qui permet d'implémenter l'algorithme de Mazurkiewicz. Si on utilise les systèmes de réécriture de graphes, on peut voir qu'un algorithme est codé par des réétiquetages locaux : les étiquettes des sommets sont changées localement ; dans des sous-graphes isomorphes à une étoile ou bien au graphe complet K_2. Les trois types de calculs locaux décrits ci-dessus peuvent être interprétés dans la réécriture de graphes par :

- RV : dans une étape du calcul, les étiquettes attachées aux sommets de K_2 sont modifiées par application d'une règle en fonction des étiquettes dans K_2.
- LC_1 : dans une étape du calcul, l'étiquette attachée au centre de l'étoile est modifiée en fonction des étiquettes des sommets de l'étoile, les étiquettes des feuilles de l'étoile ne sont pas modifiées.
- LC_2 : dans une étape du calcul, les étiquettes attachées au centre et aux feuilles de l'étoile peuvent être modifiées en fonction des étiquettes des sommets des sommets de l'étoile.

Dans la figure 3.2, (a) représente une synchronisation entre les deux sommets 1 et 2 permettant de réaliser une étape de calcul de type RV, dans (b), les sommets 3 et 5 sont centres de synchronisations permettant de réaliser une étape de calcul de type LC_1, et finalement dans (c), le sommet 3 est centre d'une synchronisation permettant d'implémenter une étape de calcul de type LC_2. On peut remarquer que dans ce cas, et pour ce graphe particulier, à un instant donné il ne peut y avoir qu'une seule synchronisation de type LC_2.

Implémentation du rendez-vous RV

Pour implémenter des calculs de type RV, nous proposons la procédure probabiliste suivante de l'algorithme 2, exécutée de manière distribuée.

Intuitivement, chaque sommet choisit un de ses voisins au hasard, les choix étant équiprobables, (tous de probabilité $\frac{1}{d(v)}$ si $d(v)$ est le degré du sommet v). Un rendez-vous a lieu entre deux sommets voisins u et v si et seulement si u et v se choisissent mutuellement.

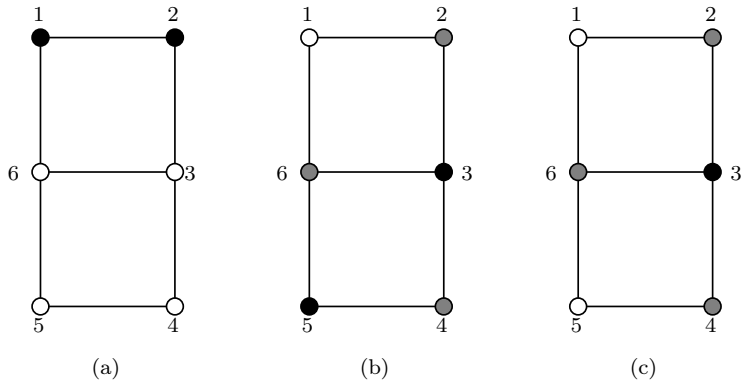


FIGURE 3.2 – Un graphe, et des synchronisations (a) RV, (b) LS_1 et (c) LS_2.

Algorithme 2: Procédure Rdv pour obtenir un rendez-vous

Chaque sommet v exécute indéfiniment les actions suivantes :
début
le sommet v choisit un de ses voisins $c(v)$ au hasard et uniformément parmi ses voisins;
le sommet v envoie 1 à $c(v)$;
le sommet v envoie 0 à ses voisins différents de $c(v)$;
le sommet v reçoit les messages de tous ses voisins;
(* il y a rendez-vous entre v et $c(v)$ si v reçoit 1 de $c(v)$ *)
fin

Implémentation de LC_1

Si un sommet est synchronisé avec tous ses voisins, cette synchronisation sera dite faible ou de type LS_1 s'il n'y a que le sommet centre qui change d'état en fonction des états de ses voisins, ce qui correspond à une implémentation d'un calcul de type LC_1.

La procédure de l'algorithme 3 permet de réaliser une telle synchronisation. Chaque processeur tire un entier entre 1 et une constante C et l'envoie à tous ses voisins. Les sommets de l'étoile centrée en v, notée S_v, sont synchronisés si pour toute feuille w de S_v : $rand(v) > rand(w)$; Cette synchronisation sera dite une LS_1-*synchronisation*. Dans ce cas, une étape de calcul dans S_v est possible : le centre de l'étoile collecte les étiquettes des feuilles et calcule sa nouvelle étiquette en fonction.

Implémentation de LC_2

Si un sommet est synchronisé avec tous ses voisins, cette synchronisation sera dite forte ou de type LS_2 si tous les sommets de l'étoile changent d'état en fonction de leurs états actuels, ce qui correspond à une implémentation d'un calcul de type

Algorithme 3: Procédure pour obtenir une LS_1-synchronisation

Chaque sommet v exécute indéfiniment les actions suivantes :
début
le sommet v choisit un entier $rand(v)$ au hasard;
le sommet v envoie $rand(v)$ à ses voisins;
le sommet v reçoit les messages de tous ses voisins;
(* il y a une synchronisation LS_1 centrée en v si $rand(v)$ est strictement plus grand que tous les entiers reçus par v ; dans ce cas, une étape de calcul peut se faire en S_v.*)
fin

LC_2.

La procédure de l'algorithme 4 permet de réaliser une telle synchronisation. Comme la précédente procédure, elle est basée sur le tirage de nombres entiers entre 1 et une constante C, suffisamment grande, chaque processeur v tire un entier $rand(v)$ au hasard dans l'ensemble $\{1, \cdots, C\}$. Il l'envoie à tous ses voisins. Les sommets de l'étoile S_v centrée en v, sont synchronisés si pour toute feuille w de S_v et pour tout sommet w_i voisin de w et différent de v : $rand(v) > rand(w)$, et $rand(v) > rand(w_i)$; Cette synchronisation sera dite une *LS_2-synchronisation*. Dans ce cas, une étape de calcul dans S_v est possible : tous les sommets de l'étoile changent d'étiquettes en fonction de leurs étiquettes et de celles de leurs voisins.

Algorithme 4: Procédure pour obtenir une LS_2-synchronisation

Chaque sommet v exécute indéfiniment les actions suivantes :
début
le sommet v choisit un entier $rand(v)$ au hasard;
le sommet v envoie $rand(v)$ à ses voisins;
le sommet v reçoit les messages de tous ses voisins;
Pour tout voisin w, soit $m_{v,w}$ le max des entiers que v a reçu des sommets différents de w ; le sommet v envoie $m_{v,w}$ à w;
le sommet v reçoit les messages de tous ses voisins;
(* Il y a une synchronisation LS_2 centrée en v si $rand(v)$ est strictement plus grand que tous les entiers reçus par v ; dans ce cas, une étape de calcul peut se faire en S_v.*)
fin

Chapitre 4

Analyse des algorithmes

Dans ce chapitre, nous analysons les algorithmes présentés dans le chapitre précèdent. On commence par une étude de la procédure permettant d'implémenter le rendez-vous, puis celles permettant d'implémenter les synchronisations LS_1 et LS_2.

Notre analyse est basée sur la considération de rounds : pour mesurer la performance des algorithmes en terme de nombre de rendez-vous, de synchronisations LS_1 ou de synchronisations LS_2 qui ont lieu, on suppose que, à un instant donné, chaque nœud envoie puis reçoit les messages. Ainsi, le paramètre qui nous intéresse, qui est le nombre de synchronisations réalisées, est le nombre maximum (sous l'hypothèse que tous les sommets sont actifs) autorisé par l'algorithme.

4.1 Le rendez-vous

4.1.1 Introduction

On s'intéresse tout d'abord au nombre moyen de rendez-vous, et à la probabilité d'obtenir au moins un rendez-vous dans le graphe, on montre en particulier que cette probabilité est minorée par $1 - e^{-1/2}$, ce qui montre que l'algorithme est un algorithme de type Las Vegas.

Des bornes inférieures plus fines sont données pour des classes particulières de graphes, notamment pour la classe des graphes à degré borné.

Comme conséquence directe des définitions, nous calculerons facilement la probabilité de rendez-vous pour les sommets. On en déduira alors le temps moyen d'attente pour un sommet avant d'obtenir un rendez-vous. Des calculs simples donnent aussi le temps d'attente entre deux rendez-vous sur une arête.

On étudie aussi l'impact du rajout de nouvelles arêtes sur le comportement du nombre moyen de rendez-vous. On verra que ce comportement n'est pas monotone,

la même remarque est vraie pour le comportement de la probabilité d'obtenir au moins un rendez-vous.

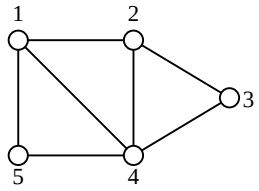
On étudiera aussi le comportement asymptotique de la distribution du nombre de rendez-vous dans des cas particuliers, à savoir, le cas du graphe complet et du cycle.

4.1.2 Premiers résultats

4.1.2.1 Définition et caractérisation d'un appel

Définition 6 Soit $G = (V, E)$ un graphe. Un *appel* sur G est une fonction c de V dans V qui associe à chaque sommet un de ses voisins.

Exemple 4.1 La figure 4.1 représente un graphe et un appel sur ce graphe.



$$c : \begin{array}{l} 1 \to 2 \\ 2 \to 3 \\ 3 \to 2 \\ 4 \to 5 \\ 5 \to 1 \end{array}$$

FIGURE 4.1 – Un graphe et un appel sur ce graphe.

Soit c un appel, par définition, un rendez-vous dans le graphe G est un couple de sommets (v, w) tels que $c(v) = w$ et $c(w) = v$.

Un appel c sur G est un *succès*, s'il contient au moins un rendez-vous, sinon, ce sera un *échec*.

On peut représenter un appel sur un graphe G par le graphe orienté $G_c = (V, A)$, où A est tel que pour tout $\{v, w\} \in E$, l'arc $\{v, w\}$ appartient à A si et seulement si $w = c(v)$, voir la figure 4.2.

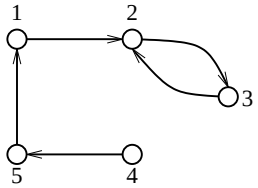


FIGURE 4.2 – Le graphe G_c correspondant au graphe et à l'appel de la figure 4.1.

Clairement, G_c est un graphe orienté, sans cycle de longueur 1 dont tous les sommets sont de degré sortant 1. Il possède donc $n = |V|$ arcs. De plus, il est facile

de voir que :

Lemme 2 *Soit c un appel sur le graphe G, c est un échec si et seulement si G_c est sans cycle de longueur 2.*

Lemme 3 *Si G est un arbre, alors tout appel sur G est un succès.*

Preuve. Pour tout appel c sur G, le graphe orienté G_c a $n = |V|$ arcs. Chaque arc correspond à une arête de E, pour laquelle une orientation a été choisie. Comme G a $n-1$ arêtes, nécessairement au moins une arête a été orientée dans les deux sens. □

Un *quasi-arbre* τ est un graphe connexe contenant exactement un seul cycle. Une *quasi-forêt* est un graphe dont les composantes connexes sont des quasi-arbres.

Soit G un graphe quelconque, c un appel sur G, et G_c le graphe orienté correspondant. Soit G'_c le graphe non orienté associé à G_c. Alors, G'_c est une quasi-forêt couvrante de G. Inversement, soit ϕ une quasi-forêt couvrante de G, soit τ un quasi-arbre de ϕ, on choisit une orientation de l'unique cycle de ϕ, ce qui induit une orientation de chaque arête du cycle ; les autres arêtes de τ sont orientées de façon à ce que pour tout couple de sommets (u, v) de τ, il y a un chemin traversant le cycle et reliant u à v. On obtient un graphe orienté correspondant à un échec.

Si on considère le cas du graphe complet K_n de taille n, un appel correspond à une endofonction [7].

Soit U un ensemble, une *endofonction* ψ de U est définie par un sous-ensemble γ de $U \times U$ vérifiant :

$$\forall x \in U \ \exists! y \in U \text{ tel que } (x, y) \in \gamma. \tag{4.1}$$

Il y a une bijection entre les appels sur K_n et l'ensemble des endofonctions sur les entiers de $[1 \cdots n]$ sans point fixe. Un appel est un succès (resp. un échec) si et seulement si il correspond à une endofonction sans point fixe contenant au moins un cycle de longueur 2 (resp. sans cycle de longueur 2).

Exemple 4.2 Soit $V = \{a, b, \cdots, q\}$, et soit G le graphe complet sur V. La figure 4.3 représente le graphe G_c pour l'appel c sur G avec $c(a) = b$, $c(b) = f$, $c(f) = a$, $c(g) = f$, $c(h) = i$, $c(i) = h$, etc.

Pour cet appel, il y a deux rendez-vous : le premier entre les sommets h et i et le deuxième entre les sommets l et m.

Si on considère le graphe non orienté correspondant à G_c, il contient une quasi-forêt définie par les sommets a, b, c, ... g.

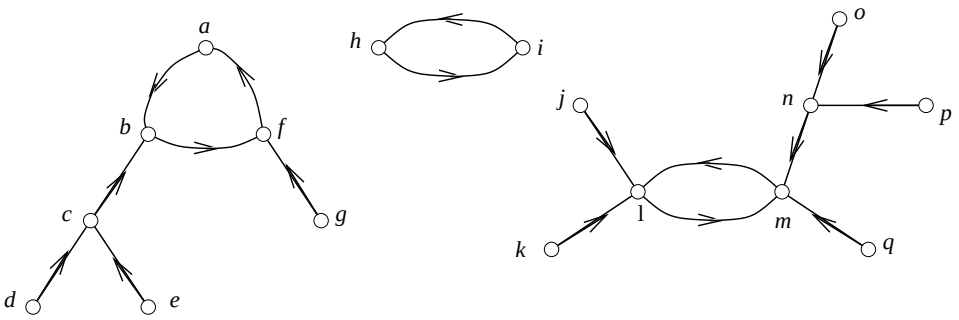


FIGURE 4.3 –

4.1.2.2 Probabilité d'obtenir au moins un rendez-vous

On suppose que tous les sommets adjacents à v ont la même probabilité $\frac{1}{d(v)}$ d'être choisis, où $d(v)$ est le degré du sommet v. Ainsi, chaque arête $e = \{v, w\} \in E$ a une probabilité $\frac{1}{d(v)d(w)}$ d'être un lien sur lequel a lieu un rendez-vous entre v et w. Dans la suite de ce chapitre, on suppose que chaque sommet se comporte indépendamment des autres et sans se rappeler de ses choix précédents.

Chaque sommet v a $d(v)$ choix possibles, considérons maintenant la mesure de probabilité, qui associe à chaque appel sur G la probabilité $\alpha(G)$ donnée par :

$$\alpha(G) = \prod_{v \in V} \frac{1}{d(v)}. \tag{4.2}$$

Soit $s(G)$ la probabilité de succès et $f(G)$ celle d'échec. Du lemme 2, on déduit que :

Lemme 4 *On a*

$$f(G) = \alpha(G)N(G) \tag{4.3}$$

et

$$s(G) = 1 - \alpha(G)N(G), \tag{4.4}$$

où $N(G)$ est le nombre d'appels c sur G pour lesquels G_c n'a pas de cycle de longueur 2.

La probabilité $f(G)$ peut être obtenue en utilisant les quasi-forêts. Soit $F(G)$ l'ensemble des quasi-forêts couvrantes de G, si ϕ est une quasi-forêt couvrante de G, alors $|\phi|$ dénote le nombre de quasi-arbres de ϕ. Avec ces notations, et en utilisant la caractérisation des échecs par des quasi-forêts, on obtient :

$$f(G) = \prod_{v \in V} \frac{1}{d(v)} \left(\sum_{\phi \in F(G)} 2^{|\phi|} \right). \tag{4.5}$$

Pour obtenir une expression exacte de la distribution de probabilité du nombre de rendez-vous pour un appel aléatoire, on considère les couplages.

Un couplage sur $G = (V, E)$ est un sous-ensemble M de E tel que pour toute paire d'arêtes e et e' de M, $e \cap e' = \emptyset$. On associe à un couplage M le rendez-vous correspondant aux rendez-vous entre les extrémités des arêtes du couplage.

Soit $e = \{v, w\}$ une arête, soit $e^{(1)}$ l'événement correspondant à un rendez-vous sur e, et $e^{(0)}$ son complémentaire. La probabilité d'un rendez-vous sur e est

$$Pr(e^{(1)}) = Pr(\{v, w\}^{(1)}) = \frac{1}{d(v)d(w)}. \tag{4.6}$$

Soit $M = \{e_1, ..., e_k\}$ un couplage, de la même façon, la probabilité $Pr(M)$ d'avoir des rendez-vous sur M est

$$Pr(M) = Pr(e_1^{(1)} \wedge e_2^{(1)} \wedge ... \wedge e_k^{(1)}) = \prod_{\{v,w\} \in M} \frac{1}{d(v)d(w)} = \prod_{e \in M} Pr(e^{(1)}). \tag{4.7}$$

Pour un entier k, un *k-couplage* sur G est un couplage de taille k. Soit $\mathcal{M}_k$ l'ensemble des k-couplages.

Soit

$$q_k = \sum_{M \in \mathcal{M}_k} Pr(M), \ k = 0, 1, ..., \lfloor n/2 \rfloor. \tag{4.8}$$

Avec cette définition, on peut noter que $q_0 = 1$. Par une application directe du principe d'inclusion-exclusion, on obtient :

Proposition 7 *Soit la suite $q_k, k = 0, 1, ..., \lfloor n/2 \rfloor$ comme définie ci-dessus pour un graphe connexe G de taille n. Alors, pour tout entier l entre 1 et $\lfloor n/2 \rfloor$, la probabilité d'obtenir au moins l rendez-vous sur G est :*

$$P_l = \sum_{0 \leqslant i \leqslant \lfloor n/2 \rfloor - l} (-1)^i q_{l+i}. \tag{4.9}$$

En particulier, la probabilité de succès est :

$$s(G) = P_1 = \sum_{0 \leqslant i \leqslant \lfloor n/2 \rfloor - 1} (-1)^i q_{i+1}. \tag{4.10}$$

Il est aussi possible de dériver des expressions simples pour la probabilité de succès $s(G)$ et l'espérance mathématique du nombre d'appels nécessaires pour obtenir un rendez-vous dans des classes spéciales de graphes. En effet :

Exemple 4.3 Soit G un graphe anneau (cycle) de taille $n \geqslant 2$. Le nombre $N(G)$, introduit dans le lemme 4, est égal à 2. Ainsi :

$$f(G) = \frac{1}{2^{n-1}} \tag{4.11}$$

et

$$s(G) = 1 - \frac{1}{2^{n-1}}. \tag{4.12}$$

Le nombre moyen d'appels nécessaires pour obtenir un succès est donc :

$$(1 - \frac{1}{2^{n-1}}) + 2\frac{1}{2^{n-1}}(1 - \frac{1}{2^{n-1}}) + 3(\frac{1}{2^{n-1}})^2(1 - \frac{1}{2^{n-1}}) + ... \tag{4.13}$$

c'est-à-dire :

$$\frac{2^{n-1}}{2^{n-1} - 1}. \tag{4.14}$$

Il est intéressant de voir l'impact de l'addition d'une nouvelle arête au graphe sur le comportement de la probabilité de succès. En effet, ce comportement n'est pas monotone :

- Si on ajoute une arête à un arbre, la probabilité d'avoir au moins un rendez-vous ne peut que diminuer.
- Le graphe G de la figure 4.4 dû à H. Austinat et V. Diekert [4], montre que l'ajout d'une arête peut augmenter cette probabilité. En effet, pour ce graphe, on a $s(G) = 1156/1600 = 0.7225$. Soit G' le graphe obtenu à partir de G en rajoutant l'arête $\{1, 2\}$. On a $s(G') = 4742/6400 = 0.7409....$

4.1.2.3 Espérance mathématique du temps d'attente entre deux rendez-vous successifs pour un sommet et pour une arête

Pour un sommet v, la probabilité $p(v)$ que v obtienne un rendez-vous peut être calculée facilement en utilisant l'indépendance des choix des sommets et le fait que les événements associés aux rendez-vous sur les arêtes incidentes sont disjoints :

$$p(v) = \sum_{e \text{ incident avec } v} Pr(e^{(1)}) \tag{4.15}$$

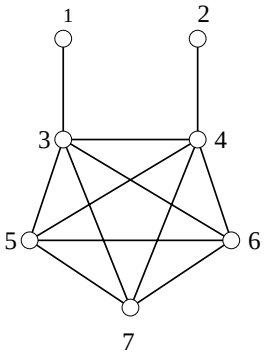


FIGURE 4.4 – Un exemple où l'ajout d'une arête augmente la probabilité d'obtenir au moins un rendez-vous.

c'est-à-dire

$$p(v) = \frac{1}{d(v)} \sum_{w \text{ adjacent à } v} \frac{1}{d(w)}. \tag{4.16}$$

De cette formule, il est clair que $p(v) = 1$ si et seulement si tous les sommets adjacents à v sont des feuilles.

Si on considère les appels successifs, on peut définir l'espérance du temps d'attente entre deux rendez-vous successifs pour un sommet ou pour une arête.

Partant des relations ci-dessus, on peut voir que le temps moyen d'attente entre deux rendez-vous successifs pour un sommet v est :

$$d(v) \frac{1}{\displaystyle\sum_{w \text{ adjacent à } v} \frac{1}{d(w)}}. \tag{4.17}$$

Ainsi, pour un graphe de degré maximum d, le temps moyen d'attente entre deux rendez-vous successifs pour un sommet est majoré par d.

Le temps moyen d'attente entre deux rendez-vous sur une arête $e = \{v, w\}$ est $d(v)d(w)$. Encore une fois, si on considère un graphe à degré borné par d, cette valeur est majorée par d^2.

4.1.2.4 Nombre moyen de rendez-vous

Soit X le nombre de rendez-vous pour un appel sur G, le nombre moyen de rendez-vous sur G, noté $\overline{M}(G)$, est $E(X)$: c'est l'espérance mathématique de X. Ce

paramètre peut être considéré comme une mesure du degré de parallélisme réalisé par l'algorithme de rendez-vous.

Pour tout arête $e \in E$, on définit χ_e par :

$$\chi_e = \begin{cases} 1 & \text{s'il y a rendez-vous sur } e, \\ 0 & \text{sinon.} \end{cases} \tag{4.18}$$

On a

$$X = \sum_{e \in E} \chi_e. \tag{4.19}$$

En appliquant le principe de la linéarité de l'espérance mathématique, on a :

$$E(X) = \sum_{e \in E} E(\chi_e). \tag{4.20}$$

Or

$$E(\chi_{\{v,w\}}) = \frac{1}{d(v)d(w)}, \tag{4.21}$$

donc :

$$E(X) = \sum_{\{v,w\} \in E} \frac{1}{d(v)d(w)}, \tag{4.22}$$

et finalement :

Proposition 8 *Le nombre moyen de rendez-vous sur G est :*

$$\overline{M}(G) = \sum_{\{v,w\} \in E} \frac{1}{d(v)d(w)}. \tag{4.23}$$

Considérons les cas particuliers suivants :

Exemple 4.4 Si G est le graphe complet de taille $n \geqslant 2$, alors

$$\overline{M}(G) = \binom{n}{2} \frac{1}{(n-1)^2} = \frac{n}{2(n-1)}. \tag{4.24}$$

Exemple 4.5 Si G est le cycle de taille $n \geqslant 3$, alors

$$\overline{M}(G) = \frac{n}{4}. \tag{4.25}$$

Exemple 4.6 Si $G = (V, E)$ est de degré borné par d, alors

$$\overline{M}(G) \geqslant \frac{|E|}{d^2}. \tag{4.26}$$

Si on considère le cas d'un arbre T de taille n et de degré borné par d, on obtient :

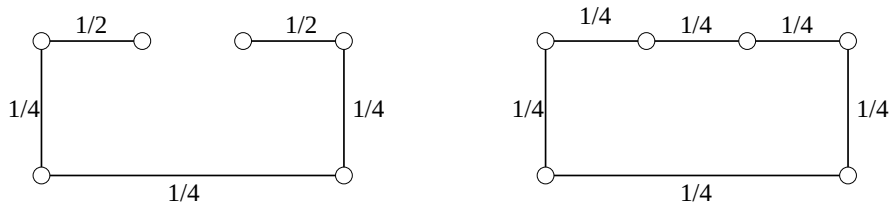
$$\overline{M}(T) \geqslant \frac{n-1}{d^2}. \tag{4.27}$$

Dans le cas des graphes réguliers de degré d, on obtient :

$$\overline{M}(G) = \frac{n}{2d}, \tag{4.28}$$

où n est la taille du graphe.

On va s'intéresser maintenant à l'impact de l'addition d'une nouvelle arête sur le comportement de $\overline{M}(G)$. Les exemples suivants illustrent le fait que le nombre d'arêtes ne favorise pas nécessairement les rendez-vous. En effet, les figures 4.5 et 4.6 montrent que le nombre moyen de rendez-vous n'est pas monotone en fonction du nombre d'arêtes.



$\overline{M}(G) = 7/4$ $\overline{M}(G) = 3/2$

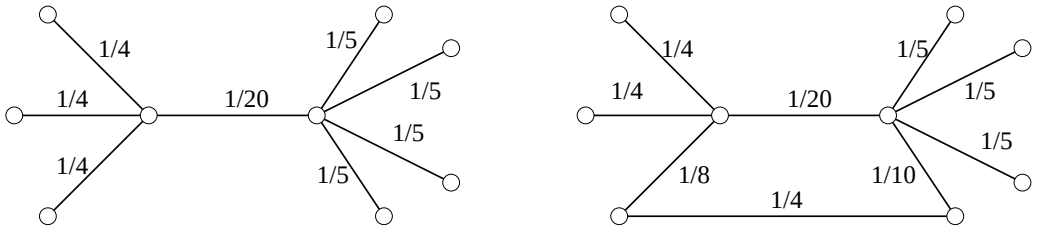
FIGURE 4.5 – Un exemple où le nombre moyen de rendez-vous diminue si on rajoute une arête.

La proposition 9 donne un minorant du nombre moyen de rendez-vous.

Proposition 9 *Pour un entier positif fixé n, le graphe complet K_n minimise le nombre moyen de rendez-vous sur les graphes de taille n. La valeur moyenne minimale réalisée par K_n est $\frac{n}{2(n-1)}$.*

Preuve. Soit $G = (V, E)$ un graphe, et $\overline{M}(G)$ le nombre moyen de rendez-vous sur G, on a

$$\overline{M}(G) = \frac{1}{2}\sum_{v \in V} p(v), \tag{4.29}$$



$\overline{M}(G) = 8/5$ $\overline{M}(G) = 13/8$

FIGURE 4.6 – Un exemple où le nombre moyen de rendez-vous augmente si on rajoute une arête.

où $p(v)$ est comme définie dans la section 4.1.2.3. Sachant que :

$$p(v) = \frac{1}{d(v)} \sum_{\{w,v\} \in E} \frac{1}{d(w)}, \tag{4.30}$$

et que $d(w) \leqslant n-1$, on a $p(v) \geqslant \frac{1}{n-1}$. En sommant sur tous les sommets, on obtient

$$\overline{M}(G) \geqslant \frac{n}{2(n-1)}. \tag{4.31}$$

Or, on sait que si G est le graphe complet de taille n, alors

$$\overline{M}(G) = \frac{n}{2(n-1)}. \tag{4.32}$$

La proposition en résulte. □

4.1.2.5 Nombre moyen de rendez-vous dans les arbres

Dans les sections précédentes, on a vu que les arbres maximisent la probabilité de succès, puisqu'elle est égale à 1. Dans cette section, nous étudions le nombre moyen de rendez-vous dans un arbre.

Proposition 10 *Soit $T = (V, E)$ un arbre et $T' = (V', E')$ l'arbre obtenu à partir de T par ajout d'une nouvelle feuille. Si $\overline{M}(T')$ est le nombre moyen de rendez-vous sur T' et $\overline{M}(T)$ celui de T, alors $\overline{M}(T') \geqslant \overline{M}(T)$.*

Preuve. Soit a la feuille ajoutée et v le sommet attaché à a. Soit d le degré de v dans T. On a :

$$\overline{M}(T') = \overline{M}(T) + \frac{1}{d+1} \left(1 - \frac{1}{d} \sum_{w \text{ adjacent à } v} \frac{1}{d(w)} \right), \tag{4.33}$$

puisque $\sum_{w \text{ adjacent à } v} \frac{1}{d(w)} \leqslant d$, il en résulte que $\overline{M}(T') \geqslant \overline{M}(T)$. □

Corollaire 1 *Soit T un arbre dont le degré maximum est k et dont le diamètre est D. Alors*

$$\overline{M}(T) \leqslant \overline{M}(T(k, \frac{D}{2})), \tag{4.34}$$

où $T(k,h)$ représente l'arbre k-aire dont la hauteur est h.

Preuve. Par des applications successives de la proposition 10. □

4.1.3 Probabilité de succès pour des cas particuliers

Dans cette section, on s'intéresse en détail à la probabilité d'obtenir au moins un rendez-vous dans le graphe. Soit e une arête, rappelons que $e^{(1)}$ est l'événement *un rendez-vous sur* e et $e^{(0)}$ son événement complémentaire.

4.1.3.1 Graphes à degrés bornés

On commence par le cas où $G = (V, E)$ est un graphe de degré au plus d. La proposition suivante donne une borne inférieure pour la probabilité d'échec.

Proposition 11 *Soit $G = (V, E)$ un graphe à degré d-borné, et $s(G)$ sa probabilité de succès. Alors*

$$s(G) \geqslant 1 - \left(1 - \frac{1}{d^2}\right)^{|E|}. \tag{4.35}$$

Preuve. Si $E = \{e_1, e_2, \ldots, e_m\}$, on a

$$\begin{aligned} f(G) &= Pr(e_1^{(0)} \wedge e_2^{(0)} \wedge \cdots \wedge e_m^{(0)}) \\ &= Pr(e_1^{(0)}) Pr(e_2^{(0)} \mid e_1^{(0)}) Pr(e_3^{(0)} \mid e_1^{(0)} \wedge e_2^{(0)}) \ldots Pr(e_m^{(0)} \mid e_1^{(0)} \wedge e_2^{(0)} \wedge \cdots \wedge e_{m-1}^{(0)}). \end{aligned}$$

Un simple calcul montre que la probabilité conditionnelle $Pr(e_i^{(0)} \mid e_{i_1}^{(0)} \wedge e_{i_2}^{(0)} \wedge \cdots \wedge e_{i_k}^{(0)})$, $i \neq i_1, i_2, \cdots, i_k$, est égale à la probabilité $Pr(e_i^{(0)})$ si e_i n'est incidente à aucune des arêtes $e_{i_1}, \cdots, e_{i_k}$ et elle est supérieure à $Pr(e_i^{(0)})$, si elle est incidente à l'une des arêtes au moins. Pour s'en apercevoir, considérons une arête $e = \{v, w\}$. Soient

$e_1, ..., e_k$ les arêtes incidentes à v et $f_1, ..., f_l$ les arêtes incidentes à w différentes de e. On a clairement :

$$Pr(e^{(1)}) = \frac{1}{d(v)d(w)} \leqslant \frac{1}{(d(v)-k)(d(w)-l)} = Pr(e^{(1)} \mid e_1^{(0)} \wedge \cdots \wedge e_k^{(0)} \wedge f_1^{(0)} \wedge \cdots \wedge f_l^{(0)}).$$

Ainsi, pour tout $e \in E$

$$\forall j = 2, \ldots, m; \;\; Pr(e_j^{(0)} \mid e_1^{(0)} \wedge e_2^{(0)} \wedge \cdots \wedge e_{j-1}^{(0)}) \leqslant Pr(e_j^{(0)}) \tag{4.36}$$

et alors

$$f(G) \leqslant Pr(e^{(0)})^{|E|}. \tag{4.37}$$

Sachant que $Pr(e^{(0)}) \leqslant (1 - \frac{1}{d^2})$, on a

$$f(G) \leqslant \left(1 - \frac{1}{d^2}\right)^{|E|}. \tag{4.38}$$

La proposition en résulte. □

Le majorant ci-dessus devient très intéressant si le rapport $|E|$ sur d est important. En effet, la formule ci-dessus montre que

$$f(G) \leqslant e^{\frac{-|E|}{d^2}}. \tag{4.39}$$

En particulier, dans le cas des graphes d-réguliers, on a $|E| = \frac{nd}{2}$ et finalement :

Corollaire 2 *Soit G un graphe $d-$régulier, la probabilité d'échec $f(G)$ vérifie :*

$$f(G) \leqslant e^{\frac{-n}{2d}}. \tag{4.40}$$

4.1.3.2 Graphes complets

Pour cette classe de graphes, on a :

Proposition 12 *Soit K_n le graphe complet de taille n, alors :*
- $s(K_n) = \sum_{k \geqslant 1} (-1)^{k+1} \frac{n!}{k! 2^k (n-2k)!} \frac{1}{(n-1)^{2k}}$,
- $s(K_n)$ *est asymptotiquement* $1 - e^{-1/2}$,
- *et le nombre moyen d'appels nécessaires pour obtenir un succès vaut asymptotiquement* $\frac{\sqrt{e}}{\sqrt{e}-1}$.

Preuve. Pour k fixé, les k-couplages de K_n sont tous de même probabilité

$$q_k = \frac{1}{(n-1)^{2k}}. \tag{4.41}$$

D'autre part, un calcul simple donne leur nombre $\frac{n!}{k!2^k(n-2k)!}$.

On a alors :

$$s(K_n) = \sum_{k\geqslant 1}(-1)^{k+1}\frac{n!}{k!2^k(n-2k)!}\frac{1}{(n-1)^{2k}}. \tag{4.42}$$

Le nombre moyen d'appels nécessaires pour obtenir un succès est $\frac{1}{s(K_n)}$, où $s(K_n)$ est donné par l'expression ci-dessus. Cette expression n'est pas facile à calculer, néanmoins, si on utilise un raisonnement combinatoire, on peut en faire une estimation asymptotique.

D'après le lemme 2 de la section 4.1.2.1, un appel c sur G est un échec, si G_c est sans cycle de longueur 1 ou 2. Les objets combinatoires représentant de tels objets peuvent être exprimés par la grammaire suivante :

$$\begin{aligned} \textit{échec} &= \texttt{ensemble}(\textit{quasi}-\textit{arbre});\\ \textit{quasi}-\textit{arbre} &= \textit{cycle}_{\geqslant 3}(\textit{arbre});\\ \textit{arbre} &= \textit{nœud} * \texttt{ensemble}(\textit{arbre});\\ \textit{nœud} &= \texttt{z}; \end{aligned} \tag{4.43}$$

Notons alors $F(z)$ la série génératrice exponentielle (SGE) des échecs, $c(z)$ celle des cycles de longueur $\geqslant 3$ et enfin $t(z)$ celle des arbres. Une traduction des spécifications des types ci-dessus en SGE donne :

$$\begin{aligned} F(z) &= c(t(z)),\\ c(z) &= \tfrac{1}{1-z}e^{-z-z^2/2},\\ t(z) &= ze^{t(z)}. \end{aligned} \tag{4.44}$$

L'unique singularité de $F(z) = c(t(z))$ est $z_0 = 1/e$, en effet $t(z_0) = 1$ si et seulement si $z_0 = 1/e$.

Dans [19, 69], les auteurs montrent que

$$t(z) \sim 1 - 2^{1/2}\sqrt{1-ez}. \tag{4.45}$$

Ainsi

$$F(z) \sim \frac{1}{\sqrt{2e^3}}(1-ez)^{-1/2}. \tag{4.46}$$

On en déduit alors que

$$[z^n]F(z) \sim \frac{1}{\sqrt{2e^3}} e^n \frac{1}{\sqrt{\pi n}}. \tag{4.47}$$

Ce qui donne le nombre d'échecs $N(K_n)$ dans le graphe complet K_n. Pour obtenir la probabilité d'échec sur K_n, il suffit de diviser $N(K_n)$ par $(n-1)^n$ qui est le nombre total d'appels sur K_n. En utilisant la formule de Stirling, on obtient

$$f(K_n) \sim e^{-1/2}, \tag{4.48}$$

ou

$$s(K_n) \sim 1 - e^{-1/2}. \tag{4.49}$$

Et le nombre moyen d'appels nécessaires pour obtenir un succès est asymptotiquement

$$\frac{\sqrt{e}}{\sqrt{e}-1}. \tag{4.50}$$

□

4.1.4 Une borne uniforme pour la probabilité de succès

La proposition 11 donne une borne inférieure pour la probabilité de succès si le graphe est à degré borné par d. Le corollaire 2 montre combien cette borne est importante si d est suffisamment petit par rapport à $n = |V|$. Cependant, cette borne devient trop large si d est trop grand et si $|E|$ n'est pas suffisamment grand. Il est donc intéressant de trouver une borne uniforme ne dépendant ni de d, ni de $|E|$. Cette section a pour objectif de donner un tel minorant. En effet, on a le théorème suivant :

Théorème 5 *La probabilité $s(G)$ de succès dans un appel sur tout graphe $G = (V, E)$ est minorée par $1 - e^{-\overline{M}(G)}$, où $\overline{M}(G)$ est le nombre moyen de rendez-vous dans G.*

Preuve. Par un même raisonnement que pour la proposition 11, on a

$$\begin{aligned} f(G) &= Pr(e_1^{(0)} \wedge e_2^{(0)} \wedge \cdots \wedge e_m^{(0)}) \\ &= Pr(e_1^{(0)}) Pr(e_2^{(0)} \mid e_1^{(0)}) Pr(e_3^{(0)} \mid e_1^{(0)} \wedge e_2^{(0)}) \ldots Pr(e_m^{(0)} \mid e_1^{(0)} \wedge e_2^{(0)} \wedge \cdots \wedge e_{m-1}^{(0)}), \end{aligned}$$

avec la même notation que pour la preuve de la proposition, on a

$$f(G) \leqslant \prod_{i=1}^{m} (1 - Pr(e_i^{(1)})). \tag{4.51}$$

Or, d'après la proposition 8, on a

$$\sum_{i=1}^{m} Pr(e_i^{(1)}) = \overline{M}(G). \tag{4.52}$$

Ainsi, $f(G)$ est maximisée lorsque

$$Pr(e_i^{(1)}) = \frac{\overline{M}(G)}{m},\ \forall i = 1, \ldots, m. \tag{4.53}$$

Donc,

$$f(G) \ \leqslant \ \prod_{i=1}^{m}(1 - \frac{\overline{M}(G)}{m}) = (1 - \frac{\overline{M}(G)}{m})^m \leqslant e^{-\overline{M}(G)}. \tag{4.54}$$

Le théorème en résulte. □

En utilisant le théorème 5 et la proposition 9, on obtient :

Corollaire 3 [1] *La probabilité $s(G)$ d'un succès sur tout graphe $G = (V, E)$ est minorée par $1 - e^{-1/2}$.*

Et par conséquent :

Corollaire 4 *Le nombre moyen d'appels nécessaires pour obtenir un succès est majoré par $\frac{\sqrt{e}}{\sqrt{e}-1}$.*

Remarque. En dépit de ce corollaire et de la proposition 12, qui affirme que $f(K_n) \leqslant e^{-1/2}$, on ne sait pas si le graphe complet K_n minimise la probabilité de succès pour les graphes de taille n.

Remarque. Le corollaire ci-dessus montre que l'algorithme est de type Las Vegas.

4.1.5 Distribution du nombre de rendez-vous

Dans cette section, on s'intéresse à la distribution asymptotique du nombre de rendez-vous pour quelques classes de graphes. Essentiellement, on étudie le comportement asymptotique de ce nombre pour des classes particulières de graphes de taille très grande. Si le graphe est une étoile, ce nombre est toujours égal à 1. Si c'est un

1. *Une preuve directe de ce résultat a été donnée par M. Robson [62]*

cycle de taille n, cette variable prend ses valeurs dans l'intervalle $[0, \frac{n}{2}]$ et son espérance mathématique est $\frac{n+1}{4}$. Dans le cas des graphes complets, ses valeurs possibles sont dans l'intervalle $[0, \frac{n}{2}]$, et son espérance mathématique est asymptotiquement $\frac{1}{2}$. Cependant, même si, en principe, le calcul de la distribution est possible, il n'y a pas de technique facile et simple, et une application naïve de la technique classique basée sur l'énumération est assez compliquée.

Nous effectuons le calcul de la distribution asymptotique du nombre de rendez-vous dans les deux cas extrêmes, le cas du cycle et le cas du graphe complet. On verra que dans le cas du graphe complet, quand n tend vers l'infini, la valeur de ce nombre reste un entier fini dont la distribution sera calculée, alors que pour le cas du cycle, cette variable a un comportement différent, le nombre moyen tend vers l'infini, mais la distribution normalisée, tend vers la loi gaussienne.

4.1.5.1 Cas des anneaux et des chaînes

Nous commençons par le cas de la chaîne pour laquelle le calcul de la distribution est assez facile. On montrera ensuite comment la distribution du nombre de rendez-vous dans le cas du cycle peut s'obtenir à partir du cas de la chaîne. Soit $G = (V, E)$ une chaîne de taille n et X_n la variable aléatoire qui compte le nombre de rendez-vous dans G. Dans cette section, nous nous intéressons à la distribution asymptotique de cette variable aléatoire. Rappelons que l'on suppose que tous les sommets sont actifs. On commence par prouver le lemme suivant qui donne une expression exacte de la probabilité d'obtenir exactement k rendez-vous dans la chaîne de taille n.

Lemme 5 *Pour tout entier k, la probabilité d'obtenir exactement k rendez-vous est*

$$Pr(X_n = k) = \frac{1}{2^{n-2}} \binom{n-1}{2k-1}. \tag{4.55}$$

Preuve. Soit $\phi_n(x)$ la série génératrice ordinaire de la variable aléatoire X_n, c'est-à-dire $\phi_n(x) = \sum_{k=0}^{\infty} Pr(X_n = k)x^k$. Considérons aussi la série génératrice $\psi_{n-1}(x)$ d'une autre variable aléatoire qui compte le nombre de rendez-vous dans la chaîne si les deux extrémités de la chaîne sont passives. Un raisonnement combinatoire conduit à l'élaboration du système suivant :

$$\begin{cases} \phi_n(x) &= \frac{1}{2}\phi_{n-1}(x) + \frac{1}{2}x\psi_{n-1}(x), & \forall n \geqslant 2 \\ \psi_n(x) &= \frac{1}{2}\psi_{n-1}(x) + \frac{1}{2}\phi_{n-1}(x), & \forall n \geqslant 2 \end{cases} \tag{4.56}$$

avec $\phi_1(x) = \psi_1(x) = 1$.
Après application des techniques usuelles de résolution de tels systèmes, on obtient :

$$\phi_n(x) = \frac{2\sqrt{x}}{1+\sqrt{x}}\left(\frac{1+\sqrt{x}}{2}\right)^n - \frac{2\sqrt{x}}{1-\sqrt{x}}\left(\frac{1-\sqrt{x}}{2}\right)^n, \tag{4.57}$$

on en déduit alors que

$$[x^k]\phi_n(x) = \frac{1}{2^{n-2}}\binom{n-1}{2k-1}. \tag{4.58}$$

Le lemme en découle. □

La série génératrice ainsi établie permet aussi d'obtenir des paramètres intéressants pour X_n. En particulier, on a :

Corollaire 5 *La variance de la variable aléatoire X_n est égale à $\frac{n-1}{16}$.*

D'autre part, la fonction génératrice $\phi_n(x)$ peut être utilisée pour montrer que le comportement asymptotique de la variable aléatoire X_n est normal. En effet, on a le théorème suivant :

Théorème 6 *La variable normalisée définie par*

$$Y_n = \frac{4X_n - n}{\sqrt{n}}, \tag{4.59}$$

admet une distribution qui tend vers une distribution normale $\mathcal{N}(0,1)$, c'est-à-dire que pour tout intervalle de réels $[a,b]$,

$$Pr(a < Y_n \leqslant b) \to \frac{1}{\sqrt{2\pi}}\int_a^b e^{-x^2/2}dx. \tag{4.60}$$

Preuve. Pour tout entier k, soit $j = 2k-1$, et $l = n-2k+1$. Le lemme 5 montre que

$$Pr(X_n = k) = \frac{1}{2}\frac{n!}{j!l!}\left(\frac{1}{2}\right)^j\left(\frac{1}{2}\right)^l. \tag{4.61}$$

En appliquant la formule de Stirling, on obtient

$$Pr(X_n = k) \sim \frac{1}{2}\sqrt{\frac{n}{2\pi jl}}\left(\frac{n}{2j}\right)^j\left(\frac{n}{2l}\right)^l, \tag{4.62}$$

et

$$\frac{jl}{n} \sim \frac{(\sqrt{n}x+n)(n-\sqrt{n}x)}{4n} \sim \frac{4}{n}, \text{ où } x = \frac{4k-n}{\sqrt{n}}. \tag{4.63}$$

Ainsi

$$Pr(X_n = k) = \frac{1}{\sqrt{2\pi n}}\left(\frac{n}{2j}\right)^j \left(\frac{n}{2l}\right)^l \tag{4.64}$$

et

$$\ln\left(\frac{n}{2j}\right)^j \left(\frac{n}{2l}\right)^l \sim -\frac{x^2}{2}. \tag{4.65}$$

Le théorème est alors prouvé par un raisonnement similaire a celui de ([44], pp. 22).

Le théorème peut être prouvé aussi par une transformation de la fonction génératrice $\phi_n(x)$ en fonction caractéristique en effectuant le changement de variable $x = e^{it}$. En effet, on peut facilement montrer que la fonction caractéristique de la variable aléatoire Y_n définie dans le théorème tend vers $e^{-\frac{1}{2}t^2}$, quand n tend vers l'infini, or cette dernière fonction n'est autre que la fonction caractéristique de la distribution normale. Il en découle alors que la variable aléatoire Y_n converge en loi vers la variable aléatoire normale. □

Considérons maintenant la variable aléatoire Z_n comptant le nombre de rendez-vous dans un cycle de taille n. On a vu que son espérance mathématique est $E(Z_n) = n/2$. De plus, on a :

Théorème 7 *Définissons la variable aléatoire normalisée par* $V_n = \frac{4Z_n - n}{\sqrt{n}}$. *Alors, si* n *tend vers l'infini :*

$$Pr(a < V_n \leqslant b) \to \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx \tag{4.66}$$

pour tout intervalle de réels $[a, b]$.

Preuve. Considérons une chaîne dont les sommets sont étiquetés $1, 2, ..., n$ et les arêtes $\{i, i+1\}, 1 \leqslant i \leqslant n-1$. Le nombre de rendez-vous sur la chaîne peut alors être écrit sous forme de la somme $X_n = \sum_{i=1}^{n-1} R_i$, où la variable aléatoire R_i est définie par

$$R_i = \begin{cases} 1 & \text{s'il y a un rendez-vous sur } \{i, i+1\}, \\ 0 & \text{sinon.} \end{cases} \tag{4.67}$$

On a alors

$$\sum_{i=2}^{n-2} R_i \leqslant X_n \leqslant \sum_{i=2}^{n-2} R_i + 2. \tag{4.68}$$

Considérons maintenant un cycle dont les sommets sont étiquettes $1, 2, ..., n$ et les arêtes $\{i, i+1\}, 1 \leqslant i \leqslant n-1, \{n, 1\}$. De la même façon, on peut écrire le nombre de rendez-vous sur ce cycle sous forme de la somme $Y_n = \sum_{i=1}^{n} S_i$, où la variable aléatoire S_i est définie par :

$$S_i = \begin{cases} 1 & \text{s'il y a un rendez-vous sur } \{i, i+1\}, \\ 0 & \text{sinon.} \end{cases} \tag{4.69}$$

On a alors

$$\sum_{i=2}^{n-2} S_i \leqslant Z_n \leqslant \sum_{i=2}^{n-2} S_i + 3. \tag{4.70}$$

Or

$$\sum_{i=2}^{n-2} R_i = \sum_{i=2}^{n-2} S_i, \tag{4.71}$$

donc

$$|Z_n - X_n| \leqslant 3. \tag{4.72}$$

D'où

$$|V_n - Y_n| \leqslant \frac{12}{\sqrt{n}}. \tag{4.73}$$

Ainsi, si n tend vers l'infini, la différence entre les deux variables normalisées tend vers 0. Le théorème en découle. □

4.1.5.2 Nombre de rendez-vous dans le graphe complet

Dans cette section, on s'intéresse au comportement asymptotique du nombre de rendez-vous dans les graphes complets. Dans la suite, on considère que $K_n = (V, E)$ est un graphe complet, et X_n la variable aléatoire qui compte le nombre de rendez-vous dans K_n.

On commence par le lemme suivant :

Lemme 6 *Soit m un entier positif. Si $n \to \infty$, alors la probabilité d'avoir au moins m rendez-vous sur K_n, $Pr(X_n \geqslant m)$ tend vers $\sum_{k \geqslant m} (-1)^{k+m} \frac{1}{k! 2^k}$.*

Preuve. D'après la proposition 7, et en utilisant un raisonnement similaire à celui de la preuve du premier point de la proposition 12, la probabilité d'avoir au moins m rendez-vous sur K_n est

$$Pr(X_n \geqslant m) = \sum_{k \geqslant m} (-1)^{k+m} \frac{n!}{k! 2^k (n-2k)!} \frac{1}{(n-1)^{2k}}. \tag{4.74}$$

En utilisant la formule de Stirling, on obtient facilement le lemme. □

On en déduit alors une caractérisation simple de la distribution asymptotique :

Théorème 8 *Pour tout entier positif m, la probabilité pour X_n d'être égal à m est donnée par*

$$Pr(X_n = m) \longrightarrow 2\frac{(-1)^m}{\sqrt{e}} - \frac{1}{m! 2^m} - 2(-1)^m \sum_{k<m} (-1)^k \frac{1}{k! 2^k} \tag{4.75}$$

et

$$Pr(X_n = 0) \longrightarrow \frac{1}{\sqrt{e}}, \tag{4.76}$$

quand $n \to \infty$.

Preuve. Posons $X = \lim_{n \to \infty} X_n$. Alors

$$Pr(X = m) = Pr(X \geqslant m) - Pr(X \geqslant m+1). \tag{4.77}$$

Le théorème est alors une conséquence du lemme 6. □

4.1.6 Graphes avec sommets passifs

Dans les sections précédentes, nous avons étudié la probabilité de succès sous l'hypothèse que tous les sommets sont actifs. Nous avons alors obtenu une borne inférieure uniforme pour la probabilité de succès. Dans cette section, on va omettre cette hypothèse, on supposera que des sommets $A \subset V$ sont actifs, les autres sommets seront autorisés à être passifs, on verra que la borne dans le cas où tous les sommets sont actifs reste valide.

On considère donc qu'il y a rendez-vous entre deux sommets voisins si et seulement si un sommet actif envoie 1 à un de ses voisins $c(x)$ et que $c(x)$ est soit passif, soit envoie 1 à x.

Le résultat suivant généralise celui donné dans le corollaire 3 où $A = V$.

Proposition 13 *Soit $G = (V, E)$ un graphe avec un ensemble $A \subseteq V$ de sommets actifs ; on suppose que $A \neq \emptyset$. Alors la probabilité $s(G, A)$ de succès dans un appel sur G où les sommets de A sont actifs (les autres sont supposés être passifs) est minorée par $1 - e^{-1/2}$.*

Preuve. Soient $G_1 = (V_1, E_1), ..., G_k = (V_k, E_k), A = \cup_{1 \leqslant i \leqslant k} V_i$ les composantes connexes du sous-graphe induit par A. Il est facile de voir qu'un appel sur G est un échec si et seulement si sa restriction à chacune des composantes G_i est un échec (c'est-à-dire qu'il n'y a pas de rendez-vous sur ces composantes). Alors, la probabilité d'échec sur G est bornée par la probabilité d'échec sur $G_i, 1 \leqslant i \leqslant k$ comme composante de G. D'autre part, la probabilité qu'un échec sur G soit dû à un échec sur G_i est bornée par la probabilité d'échec sur G_i considéré comme un graphe indépendant où tous les sommets sont actifs. En effet, les événements aboutissant à des échecs sont les mêmes dans les deux cas, cependant, le degré d'un sommet considéré comme actif dans G_i est supérieure ou égal au degré du même sommet dans G_i considéré comme passif. Par conséquent, puisque la probabilité associée est l'inverse du produit des degré, la probabilité qu'un appel sur G soit un échec est bornée par $f(G_i), 1 \leqslant i \leqslant k$, où $f(G)$ est la probabilité d'échec dans un graphe G, où tous les sommets sont actifs. À présent, en utilisant le corollaire 3, on a $f(G_i) \leqslant e^{-1/2}$ et $f(G, A) \leqslant e^{-1/2}$, et $s(G, A) \geqslant 1 - e^{-1/2}$. □

Remarque. La proposition ci-dessus donne une borne uniforme pour la probabilité d'échec. Dans plusieurs cas, il est possible de donner une bien meilleure borne. Considérons les composantes connexes introduites ci-dessus, il est clair, d'après le lemme 3, que la probabilité de succès est 1 si et seulement si au moins une des composantes est un arbre (c'est-à-dire sans cycle).

Soit k le nombre des composantes connexes. La preuve ci-dessus donne une meilleure borne pour la probabilité d'échec *en général*. En effet, on a $f(G, A) \leqslant e^{-k/2}$.

4.1.7 Résultats expérimentaux

Cette section présente quelques résultats expérimentaux obtenus dans [70]. En plus des simulations de l'algorithme présenté ci-dessus, l'intérêt principal était de répondre à une question importante :

Quel est le graphe qui maximise le nombre moyen de rendez-vous ?

Dans un premier temps, il peut apparaître normal de conjecturer que les chaînes permettent de maximiser ce nombre. En effet, jusqu'à $n = 6$, cette conjecture est vérifiée, à partir de $n = 7$, on voit apparaître les arbres de la figure 4.7 comme étant les arbres maximisant ce nombre.

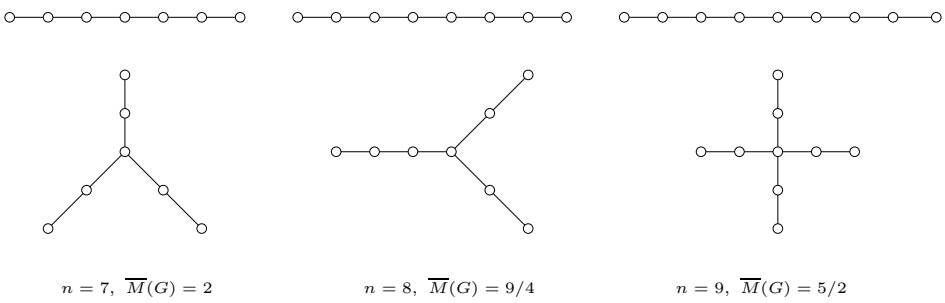


FIGURE 4.7 – Les graphes maximisant $\overline{M}(G)$, pour $n = 7, 8$, et 9.

En poussant les calculs un peu plus loin, une autre classe d'arbres paraît être candidate pour maximiser le nombre moyen de rendez-vous (figure 4.8).

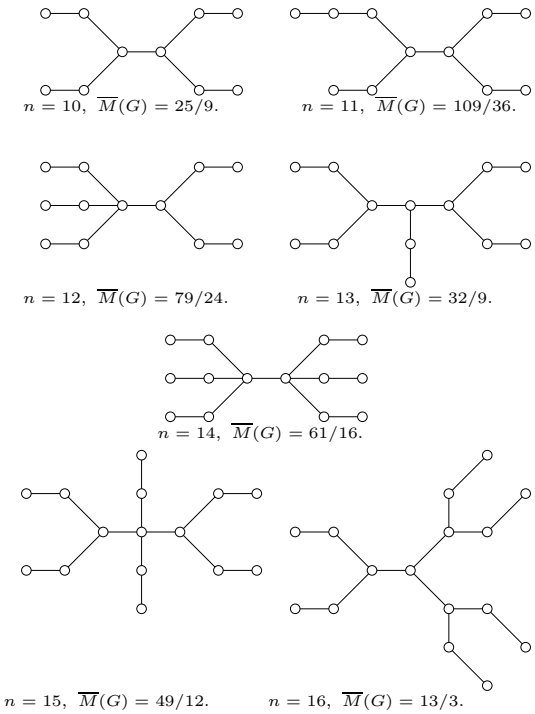


FIGURE 4.8 – Les graphes maximisant $\overline{M}(G)$, pour $n \in [10, 16]$.

4.2 LS_1-Synchronisation

Soit $G = (V, E)$ un graphe, et $B(v, 1)$ une boule de centre v et de rayon 1. Une LS_1-synchronisation de centre v est une synchronisation permettant d'implémenter un calcul local de type LC_1, c'est-à-dire un calcul permettant au sommet v d'effectuer une transition en fonction des informations collectées sur les sommets de la boule $B(v, 1)$. L'algorithme 3 du chapitre précédent permet d'obtenir de telles synchronisations. Dans cette section, on s'intéresse aux mêmes questions que pour le rendez-vous, c'est-à-dire, la probabilité d'obtenir au moins une LS_1-synchronisation dans le graphe, la probabilité pour un sommet d'être centre d'une LS_1-synchronisation ainsi qu'au nombre moyen des LS_1-synchronisations réalisées par l'algorithme.

On suppose que le tirage se fait dans un ensemble d'entiers très grand, il est assimilé à un tirage dans un ensemble de réels, donc la probabilité pour deux sommets de tirer le même nombre tend vers 0.
On a donc le fait suivant :

Fait 1 *La probabilité d'obtenir au moins une LS_1-synchronisation est égale à 1.*

Remarque. Un sommet peut savoir si un sommet voisin a tiré le même entier.

On commence par l'étude de la probabilité pour un sommet d'être centre d'une synchronisation LS_1, on en déduira la probabilité pour un sommet de participer à une telle synchronisation. Puis comme dans la section précédente, on s'intéressera au nombre moyen de synchronisations réalisées par l'algorithme.

Dans cette section, on supposera que tous les graphes sont connexes.

4.2.1 Probabilité de synchronisation sur un sommet

Soit $G = (V, E)$ un graphe quelconque. Chaque sommet $v \in V$ tire aléatoirement et uniformément un nombre $rand(v)$ de l'ensemble $\{1, \cdots, N\}$. Il y a une LS_1-synchronisation centrée en v si pour tout sommet $w \in B(v, 1) \setminus \{v\}$, on a $rand(w) < rand(v)$.

Soit $X \subseteq V$ un ensemble de sommets contenant un sommet donné v, et soit $|X| = K$. Alors :

$$Pr\left(rand(v) > rand(w), \forall w \in X \setminus \{v\}\right) = \frac{1}{N} \sum_{i=2}^{N} \left(\frac{i-1}{N}\right)^{K-1}. \tag{4.78}$$

N étant considéré très grand, on a la proposition suivante :

Proposition 14 *La probabilité pour un sommet v d'être centre d'une LS_1-synchronisation est donnée par :*

$$p_1(v) = \frac{1}{d(v)+1}. \tag{4.79}$$

Rappelons que $d(v)+1$ est le cardinal de $B(v,1)$.

Preuve. En prenant $X = B(v,1)$, on a

$$p_1(v) = Pr\left(rand(v) > rand(w), \forall w \in B(1,v) \setminus \{v\}\right) = \frac{1}{N}\sum_{i=2}^{N}\left(\frac{i-1}{N}\right)^{K-1}.$$

Ainsi,

$$\begin{array}{rcccl} p_1(v) & = & \sum_{i=1}^{N} \frac{1}{N}\left(\frac{i}{N}\right)^{d(v)} & = & \frac{1}{N}\left(\left(\frac{1}{N}\right)^{d(v)} + \left(\frac{2}{N}\right)^{d(v)} + \cdots + \left(\frac{N-1}{N}\right)^{d(v)}\right) \\ & = & \sum_{k=1}^{N-1} \frac{1}{D} f(a + k\frac{(b-a)}{N}) & & \end{array}$$

où $a = 0$, $b = 1$ et $f(x) = x^{N_2(v)-1}$.
En utilisant la formule de Darboux, on obtient :

$$p_1(v) = \int_0^1 x^{d(v)}dx = \frac{1}{d(v)+1}. \tag{4.80}$$

La proposition en résulte. □

De cette proposition, on déduit le corollaire suivant :

Corollaire 6 *Soit v un sommet de G, le temps moyen d'attente entre deux LS_1-synchronisations sur le sommet v est égale à $d(v)+1$.*

Considérons maintenant des classes simples de graphes :

Exemple 4.7 Si G est le graphe cycle de taille $n > 2$, alors, pour tout sommet v, on a :

$$p_1(v) = \frac{1}{3}. \tag{4.81}$$

Exemple 4.8 Si G est le graphe complet de taille $n > 1$, alors, pour tout sommet v, on a :

$$p_1(v) = \frac{1}{n}. \tag{4.82}$$

4.2.2 Nombre moyen de LS_1-synchronisations

Dans cette section, on s'intéresse au nombre moyen de LS_1-synchronisations réalisées par la procédure. Notons alors $\overline{M_1}(G)$ le nombre moyen de ces synchronisations. Ce paramètre peut être interprété comme le degré de parallélisme autorisé par l'algorithme. Il est facile de voir que si on pose, pour tout sommet v,

$$X_v = \begin{cases} 1 & \text{si } v \text{ est centre d'une } LS_1\text{-synchronisation,} \\ 0 & \text{sinon,} \end{cases} \tag{4.83}$$

alors

$$\overline{M_1}(G) = \sum_{v \in V} Pr(X_v = 1) = \sum_{v \in V} p_1(v), \tag{4.84}$$

on en déduit :

Proposition 15 *Le nombre moyen de LS_1-synchronisations réalisées par la procédure est*

$$\overline{M_1}(G) = \sum_{v \in V} \frac{1}{d(v)+1}. \tag{4.85}$$

Reconsidérons les exemples ci-dessus :

Exemple 4.9 Si G est le cycle de taille $n > 2$, alors

$$\overline{M_1}(G) = \frac{n}{3}. \tag{4.86}$$

Exemple 4.10 Si G est le graphe complet de taille $n > 1$, alors

$$\overline{M_1}(G) = 1. \tag{4.87}$$

Revenons à la k-densité définie dans le chapitre des notions sur la théorie des graphes, pour $k = 1$, on a

$$\mathcal{D}_1(G) = \frac{\sum_{v \in V} d(v)}{|V|} = 2\frac{|E|}{|V|}. \tag{4.88}$$

On voit bien que la 1-densité d'un graphe G est deux fois le rapport entre le nombre d'arêtes du graphe et celui de ses sommets.

On termine cette section en donnant quelques résultats afin de minorer le nombre moyen de LS_1-synchronisations dans un graphe quelconque.

Théorème 9 *Soit $G = (V, E)$ un graphe connexe de taille $n \geqslant 2$ avec $m = |E|$ arêtes. On a :*

$$\overline{M_1}(G) \geqslant \frac{n}{\mathcal{D}_1(G) + 1} = \frac{n^2}{2m + n}. \tag{4.89}$$

Preuve. Pour un graphe donné $G = (V, E)$, un calcul simple donne :

$$\sum_{v \in V} [d(v) + 1] = n\mathcal{D}_1(G) + n. \tag{4.90}$$

Ainsi,

$$\frac{1}{n} \sum_{v \in V} [d(v) + 1] = \mathcal{D}_1(G) + 1. \tag{4.91}$$

Cependant, on a

$$\frac{1}{n} \sum_{v \in V} [d(v) + 1] \geqslant \left(\prod_{v \in V} [d(v) + 1] \right)^{\frac{1}{n}}, \tag{4.92}$$

et alors,

$$\left(\prod_{v \in V} \frac{1}{d(v) + 1} \right)^{\frac{1}{n}} \geqslant \frac{1}{\mathcal{D}_1(G) + 1}. \tag{4.93}$$

Or,

$$\left(\prod_{v \in V} \frac{1}{(d(v) + 1)} \right)^{\frac{1}{n}} \leqslant \frac{1}{n} \sum_{v \in V} \frac{1}{(d(v) + 1)}, \tag{4.94}$$

et ainsi,

$$\overline{M_1}(G) = \frac{1}{n} \sum_{v \in V} \frac{1}{(d(v) + 1)} \geqslant \frac{1}{\mathcal{D}_1(G) + 1}. \tag{4.95}$$

Ce qui prouve le théorème. □

Si G est un graphe à degré borné par Θ, du théorème on déduit immédiatement :

Corollaire 7 *Soit G un graphe dont les sommets sont tous de degrés inférieurs ou égal à Θ. Alors,*

$$\overline{M_1}(G) \geqslant \frac{n}{\Theta + 1}. \tag{4.96}$$

D'autre part, dans le cas des arbres, l'expression du théorème devient :

Corollaire 8 *Soit T un arbre de taille $n \geqslant 2$. On a*

$$\overline{M_1}(T) \geqslant \frac{n^2}{3n-2} > \frac{n}{3}. \tag{4.97}$$

4.3 LS_2-Synchronisation

Soit $G = (V, E)$ un graphe, et $B(v, 2)$ une boule de centre v et de rayon 2. Une LS_2-synchronisation de centre v est une synchronisation permettant d'implémenter un calcul local de type LC_2, c'est-à-dire un calcul permettant à tous les sommets de la boule $B(v, 1)$ d'effectuer une transition en fonction des informations échangés par les sommets de la boule $B(v, 1)$. L'algorithme 4 du chapitre précédent permet d'obtenir de telles synchronisations. Remarquons que l'on peut détecter les coïncidences dans une boule de rayon 2, c'est-à-dire que l'on peut savoir si deux sommets ont tiré le même nombre.

De la même façon que pour la section précédente, on peut voir que la probabilité d'obtenir au moins une LS_2-synchronisation est égale à 1.

4.3.1 Probabilité d'une LS_2-synchronisation sur un sommet

Soit $G = (V, E)$ un graphe connexe. Chaque sommet v tire aléatoirement et uniformément un nombre $rand(v)$ entre 1 et N, un sommet v est alors centre d'une LS_2-synchronisation si et seulement si pour tout sommet w de la boule $B(v, 2)$, on a $rand(v) > rand(w)$.

La formule 4.78 appliquée à $X = B(v, 2)$ donne :

$$Pr(rand(v) > rand(w), \forall w \in B(v,2) \setminus \{v\}) = \frac{1}{N} \sum_{i=2} N \left(\frac{i-1}{N}\right)^{N_2(v)},$$

où $N_2(v)$ est le nombre de sommets dans la boule $B(v, 2)$.

La même technique que pour la preuve de la proposition 14, permet de montrer que

Proposition 16 *La probabilité pour un sommet v d'être centre d'une LS_2-synchronisation est*

$$p_2(v) = \frac{1}{N_2(v)}. \tag{4.98}$$

De cette proposition, on déduit le corollaire suivant :

Corollaire 9 *Soit v un sommet de G, le temps moyen d'attente entre deux LS_2-synchronisations centrées sur le sommet v est égale à $N_2(v)$.*

Si on reconsidère les exemples de la section précédente, on a :

Exemple 4.11 Si G est le graphe cycle de taille $n > 2$, alors, pour tout sommet v, on a :

$$p_2(v) = \frac{1}{5}. \tag{4.99}$$

Exemple 4.12 Si G est le graphe complet de taille $n > 1$, alors, pour tout sommet v, on a :

$$p_1(v) = \frac{1}{n}. \tag{4.100}$$

4.3.2 Nombre moyen de LS_2-synchronisations

Soit $G = (V, E)$ un graphe connexe, $\overline{M_2}(G)$ désigne le nombre moyen de LS_2-synchronisations sur le graphe G, un raisonnement analogue à celui de la section 4.2.2, permet de montrer que :

Proposition 17 *Le nombre moyen de LS_2-synchronisations réalisées par la procédure est*

$$\overline{M_2}(G) = \sum_{v \in V} \frac{1}{N_2(v)}. \tag{4.101}$$

Exemple 4.13 Si G est le cycle de taille $n > 2$, alors

$$\overline{M_2}(G) = \frac{n}{5}. \tag{4.102}$$

Exemple 4.14 Si G est le graphe complet de taille $n > 1$, alors

$$\overline{M_2}(G) = 1. \tag{4.103}$$

Pour minorer $\overline{M_2}(G)$, on a besoin du lemme suivant :

Lemme 7 *Soit $G = (V, E)$ un graphe connexe. On a*
- *$\sum_{v \in V} N_2(v) \leqslant \sum_{v \in V} d(v)^2 + |V|$.*
- *En plus, si G est un arbre, alors cette inégalité devient une égalité.*

Preuve. Soit G un graphe. Commençons par prouver la deuxième affirmation du lemme. Soit G un arbre. On fait une induction sur la taille n. Pour $n = 2$, une simple vérification donne le résultat. Supposons le résultat vrai pour $n \geqslant 2$. Soit $T = (V, E)$ un arbre de taille $n + 1$ et u une feuille de T et $T' = (V', E')$ l'arbre obtenu à partir de T par suppression de u et l'unique arête qui le rattache à l'arbre. Soit δ le degré du sommet père de u dans T'. Si on dénote par $N_2(v)$ le nombre de sommets à distance inférieure ou égale à 2 de v, on a

$$\begin{aligned} \sum_{v\in V} N_2(v) &= \sum_{v\in V'} N_2'(v) + 2\delta + 3 \\ &= \sum_{v\in V'} d(v)^2 - 2(\delta+1) + n + 2\delta + 3 \\ &= \sum_{v\in V} d(v)^2 + n + 1. \end{aligned} \tag{4.104}$$

Si G n'est pas un arbre, on raisonne par récurrence sur le nombre m d'arêtes de E. Étant donné que l'affirmation est valide sur l'arbre couvrant de G, il suffit de montrer que si l'on rajoute une arête non existante $\{u, v\}$, sa contribution totale à la somme $S_1 = \sum_{v\in V} N_2(v)$ n'est pas plus grande que sa contribution à la somme $S_2 = \sum_{v\in V} d(v)^2$.

Pour S_1, on a facilement que $S_1 \leqslant 2d(u) + 2d(w) + 2$. Pour S_2, on peut écrire que $S_2 = 2d(u) + 1 + 2d(w) + 1$.

Ce qui finit la preuve du lemme. □

Retournant au nombre moyen de LS_2-synchronisations, en utilisant le lemme, il vient :

Théorème 10 *Soit $G = (V, E)$ un graphe connexe, de taille $n \geqslant 2$ à $m = |E|$ arêtes. On a*

$$\overline{M_2}(G) \geqslant \frac{n}{\mathcal{D}_2(G) + 1}. \tag{4.105}$$

Preuve. Par le lemme 7, et la définition de $\mathcal{D}_2(G)$, on obtient

$$\frac{1}{n}\sum_{v\in V} N_2(v) \leqslant \mathcal{D}_2(G) + 1. \tag{4.106}$$

Étant donné que

$$\frac{1}{n}\sum_{v\in V} N_2(v) \geqslant \left(\prod_{v\in V} N_2(v)\right)^{\frac{1}{n}} \tag{4.107}$$

on a

$$\mathcal{D}_2(G) + 1 \geqslant \left(\prod_{v\in V} N_2(v)\right)^{\frac{1}{n}}, \tag{4.108}$$

et ainsi,

$$\left(\prod_{v\in V}\frac{1}{N_2(v)}\right)^{\frac{1}{n}} \geqslant \frac{1}{\mathcal{D}_2(G)+1}. \tag{4.109}$$

Encore une fois, en utilisant le fait que

$$\frac{1}{n}\sum_{v\in V}\frac{1}{N_2(v)} \geqslant \left(\prod_{v\in V}\frac{1}{N_2(v)}\right)^{\frac{1}{n}}, \tag{4.110}$$

on obtient le résultat. □

Si on considère le cas des graphes à degrés bornés, le théorème ci-dessus permet de montrer que :

Corollaire 10 *Soit G un graphe dont les sommets sont tous de degré inférieur ou égal à Θ. Alors,*

$$\overline{M_2}(G) \geqslant \frac{n}{\Theta^2+2\theta+1}. \tag{4.111}$$

4.4 Généralisation et remarques

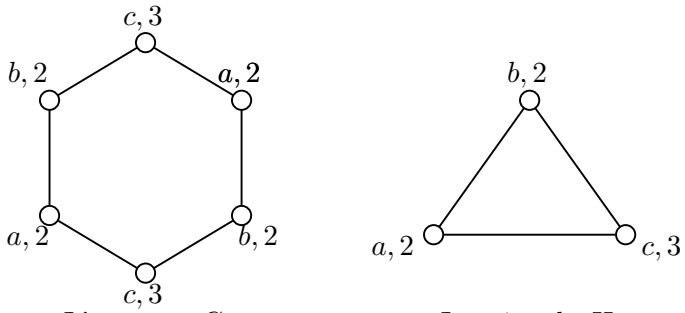
Il est possible d'étendre les algorithmes étudiés ci-dessus pour obtenir des k-synchronisations centrées en des sommets pour tout entier positif k. Un algorithme pour une LS_k-synchronisation est décrit comme suit : chaque sommet v tire uniformément et aléatoirement un entier $rand(v)$ de l'ensemble $\{1,\cdots,N\}$, où N est un entier suffisamment grand. Il y a une k-synchronisation centrée en un sommet v si et seulement si $rand(v) \geqslant rand(w)$, pour tout $w \in B(v,k)\setminus\{v\}$. On peut alors montrer que le nombre moyen des LS_k-synchronisations réalisées par un tel algorithme est minoré par

$$\frac{n}{\sum_{v\in V} N_k(v)}, \text{ où } N_k(v) \text{ est le cardinal de } B(v,k). \tag{4.112}$$

Soit $G=(V,E)$ un graphe quelconque. Supposons que chaque sommet v choisit un entier aléatoire. Sur une distance inférieure ou égale à 2, on peut détecter une coïncidence, c'est-à-dire si deux sommets ont tiré le même nombre. Cette affirmation n'est pas vraie pour une distance $k>2$, en effet, nous avons :

Proposition 18 *Il n'existe pas d'algorithme déterministe pour détecter les coïncidences dans une boules $B(v,k)$ de centre v et de rayon $k \geqslant 3$.*

Preuve. Par l'absurde, supposons qu'un tel algorithme existe. Considérons un hexagone G et un triangle H dont les sommets sont étiquetés comme indiqué dans la figure 4.9. Soit ϕ le morphisme de G vers H qui associe au sommet de G étiqueté x, $(x = a, b, c)$, le sommet portant la même étiquette x dans H. Supposons que chaque



L'exagone G Le triangle H

FIGURE 4.9 – Le graphe G est un revêtement du graphe H.

sommet choisit un entier comme indiqué sur la figure 4.9. Donc, chaque entier apparaît deux fois dans G (il y a coïncidence) et une fois dans H (il n'y a pas de coïncidence).

On définit sur G et sur H deux suites de pas $(G_i)_{i\geqslant 0}$ et $(H_i)_{i\geqslant 0}$ vérifiant la propriété $\mathcal{R}$ suivante : les états des sommets étiquetés $x = a, b, c$ dans G_i sont les mêmes que ceux des sommets étiquetés $x = a, b, c$ dans H_i.
Les graphes sont anonymes et initialement, tous les sommets dans G et H sont dans un même état q_0 . Soient G_0 et H_0 les graphes G et H dans cet état initial. Il est aisé de voir que G_0 et H_0 vérifient la propriété $\mathcal{R}$.

Soient $(G_i)_{i\geqslant 0}$ et $(H_i)_{i\geqslant 0}$ les graphes G et H avec des états vérifiant $\mathcal{R}$, et soit P une étape dans H_i.

— Si P est une communication sur l'arête e, alors la communication est possible sur les deux arêtes de $\phi^{-1}(e)$. On exécute la communication dans G_i, on obtient G_{i+1} et H_{i+1} vérifiant la propriété $\mathcal{R}$.
— Si dans P le sommet x $(x \in \{a, b, c\})$ dans H_i change son état (le nouvel état de x est q) alors les sommets dans $\phi^{-1}(x)$ changent leurs états de q. Ainsi, on obtient G_{i+1} et H_{i+1} qui vérifient la propriété $\mathcal{R}$.

À présent, si un sommet v de H est dans un état qui indique qu'il n'y pas de coïncidence, les sommets de $\phi^{-1}(v)$ seront étiquetés par un état qui leur indiquera qu'il n'y a pas de coïncidence. On obtient ainsi une contradiction et la proposition est prouvée. □

Chapitre 5

Efficacité des algorithmes de synchronisations

L'objectif de ce chapitre est de fournir une mesure du degré d'efficacité des algorithmes probabilistes permettant de résoudre des problèmes de synchronisations locales. Intuitivement, la performance de tels algorithmes sera définie comme le rapport entre ce qu'ils réalisent en moyenne et ce qui serait réalisé dans un cadre idéal. La définition prend tout son intérêt, quand on sait que le cas idéal est assez souvent NP-difficile voir irréalisable.

5.1 Efficacité de l'algorithme du rendez-vous

Comme première application, de cette notion, on va s'intéresser à l'algorithme du rendez-vous. On verra que cette procédure réalise une bonne performance dans le cas des graphes connexes peu denses (arbres), et une performance médiocre pour les graphes denses.

Considérons, pour commencer, les exemples suivants :

Exemple 5.1 Si G est le graphe complet de taille $n \geqslant 2$, alors

$$\overline{M}(G) = \binom{n}{2} \frac{1}{(n-1)^2} = \frac{n}{2(n-1)}. \tag{5.1}$$

Exemple 5.2 Si G est une étoile de taille $n \geqslant 2$, alors,

$$\overline{M}(G) = (n-1)\frac{1}{n-1} = 1. \tag{5.2}$$

Dans les deux cas, le nombre moyen de rendez-vous est très faible. En effet, dans les cas des graphes complets, le couplage maximal est de taille $\lfloor n/2 \rfloor$, ce qui est très grand comparé au nombre moyen de rendez-vous, si n est suffisamment grand. Dans le deuxième exemple, l'algorithme donne un nombre moyen qui est le même que la taille du couplage maximal. Ces remarques donnent lieu à la définition suivante :

Définition 7 Soit $\mathcal{A}$ un algorithme probabiliste pour les synchronisations de type rendez-vous fonctionnant sur les graphes. Son *efficacité* sur un graphe quelconque G est le rapport :

$$\Delta_{\mathcal{A}}(G) = \frac{\overline{M}_{\mathcal{A}}(G)}{K(G)}, \tag{5.3}$$

où $\overline{M}_{\mathcal{A}}(G)$ est le nombre moyen de rendez-vous une fois l'algorithme $\mathcal{A}$ appliqué à G et $K(G)$ est la taille d'un couplage maximal de G.

L'objectif de cette section n'est pas d'étudier l'efficacité de l'algorithme pour des classes de graphes, mais plutôt de fournir une mesure universelle de l'efficacité pour des algorithmes similaires applicables pour des problèmes n'admettant pas de solution déterministe.

Une conséquence immédiate de la définition est que l'algorithme proposé n'est pas efficace pour la classe des graphes complets. Une déduction "logique" de ce fait est que l'algorithme n'est pas efficace pour les graphes denses, cependant, il semble que l'on ne puisse pas faire mieux si l'on se restreint aux messages à un seul bit.

Jusqu'ici on a considéré deux cas extrêmes, quelle l'efficacité pour des classes intermédiaires de graphes ? on n'a pas de réponse générale. Une classe de graphes pas trop denses est celle des arbres. Le reste de la section a pour objectif de fournir une borne inférieure de l'efficacité de l'algorithme sur les arbres.

Soit $G = (V, E)$ un graphe avec $|V| = n$ et $|E| = m$, et soit M sa matrice d'incidence, [63]. Pour une énumération des sommets $v_1, \cdots, v_n$, et des arêtes $e_1, \cdots, e_m$, elle est définie par :

$$M[i,j] = \begin{cases} 0 & \text{si } v_i \text{ n'est pas extrémité de } e_j \\ 1 & \text{si } v_i \text{ est une extrémité de } e_j, \end{cases} \tag{5.4}$$

pour $0 \leqslant i \leqslant n$ and $0 \leqslant j \leqslant m$.

Un sous-ensemble F de E est un couplage de G si et seulement si son indicateur, le m-vecteur colonne x

$$x[j] = \begin{cases} 0 & \text{si } x_j \in F \\ 1 & \text{si } x_j \notin F, \end{cases} \tag{5.5}$$

pour $0 \leqslant j \leqslant m$, satisfait $Mx \leqslant \mathbf{1}_n, x \in \{0,1\}^m$, où $\mathbf{1}_n$ est le n-vecteur colonne $(1,...,1)^T$.

Donc la taille maximale d'un couplage de G, $K(G)$ est la valeur optimale de la fonction objective du programme linéaire entier suivant [63] :

$$\begin{array}{lll} & \text{maximiser :} & < \mathbf{1}_m, x > \\ (5.6) & \text{sous les contraintes :} & Mx \leqslant \mathbf{1}_n \\ & & x \in \{0,1\}^m. \end{array}$$

Ainsi, la valeur de $K(G)$ ne peut être plus grande que la valeur de la fonction objective du même programme si on enlève la contrainte $x \in \{0,1\}^m$. Cependant, cette valeur est bornée par la valeur de la fonction objective pour toute solution réalisable du problème dual suivant (théorème faible de la dualité [63]).

$$\begin{array}{lll} & \text{minimiser :} & < \mathbf{1}_n, y > \\ (5.7) & \text{sous les contraintes :} & M^T y \geqslant \mathbf{1}_m \\ & & y \in \mathbb{R}^n, \quad y \geqslant 0 \end{array}$$

À présent, si on rajoute la contrainte $y \in \{0,1\}^n$, la solution optimale sera la sommet-couverture minimale de G, voir [63]. Il en résulte que $K(G)$ est borné par la taille de toute sommet-couverture de G.

On va maintenant énoncer le théorème principal de cette section :

Théorème 11 *L'efficacité $\Delta_{RV}(T)$ de l'algorithme probabiliste RV proposé sur tout arbre T est strictement supérieure à $\frac{1}{3}$.*

Preuve. Par comparaison avec la valeur de la fonction objective du programme dual, il suffit de montrer que l'arbre T admet une sommet-couverture de taille inférieure à $3\overline{M}(T)$. Par induction sur la taille n de l'arbre, on suppose que c'est vrai pour tout arbre de taille inférieure à n, et on va montrer que c'est vrai pour un arbre T de taille n. D'autre part, sachant que le théorème est vérifié pour tout graphe étoile, on supposera que le diamètre de T est supérieur à 2. On doit alors montrer que T admet une sommet-couverture de taille inférieure à $3\overline{M}(T)$.

On suppose que l'arbre est enraciné en l'un de ses centres. Or le diamètre est supérieur à 2, donc il existe deux sommets a et a' tels que a' est le père de a, tous les sommets voisins de a sont des feuilles et a n'est pas une feuille de T. Soient d et d' leurs degrés respectifs.

Soit T' l'arbre induit par suppression de a et de ses fils de l'arbre enraciné T. Une sommet-couverture de T peut être obtenue par ajout de a à une sommet-couverture de T'. Ainsi, si le théorème est vérifié pour T', on doit prouver que

$\overline{M}(T) - \overline{M}(T') \geqslant \frac{1}{3}$.
un calcul simple permet d'obtenir la différence. En effet, on a

$$\overline{M}(T) - \overline{M}(T') = \frac{d-1}{d} + \frac{1}{dd'} - \sum_{i=1}^{d'-1}[\frac{1}{(d'-1)\delta_i} - \frac{1}{d'\delta_i}], \tag{5.8}$$

où les δ_i sont les degrés des sommets voisins de a.

Or, on peut voir que

$$\sum_{i=1}^{d'-1}[\frac{1}{(d'-1)\delta_i} - \frac{1}{d'\delta_i}] < \frac{1}{d'} \tag{5.9}$$

ainsi,

$$\overline{M}(T) - \overline{M}(T') \geqslant \frac{d-1}{d} + \frac{1}{dd'} - \frac{1}{d'} = (1 - \frac{1}{d})(1 - \frac{1}{d'}). \tag{5.10}$$

À présent, si au moins un des deux degrés d et d' est supérieur à 2, la différence sera au moins égale à $\frac{1}{3}$. Sinon $d = d' = 1$, et la différence (1) se réduit à

$$\frac{1}{2} + \frac{1}{4} - \frac{1}{2\delta} \tag{5.11}$$

où δ est le degré de l'unique sommet voisin de a' différent de a. Si $\delta \geqslant 2$, la différence est alors supérieure ou égale à $\frac{1}{2}$ et la preuve est finie, sinon, l'arbre T se réduit à 4 sommet et une simple vérification prouve le théorème. □

Remarque. Pour le graphe de la figure 5.1, l'efficacité est inférieure à $\frac{1}{2}$. On ne connaît pas de borne plus grande que $\frac{1}{3}$.

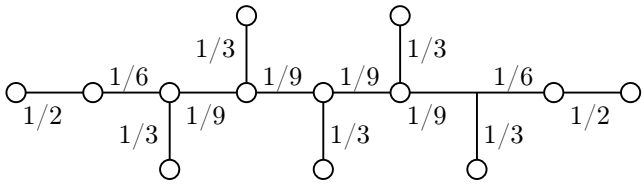


FIGURE 5.1 – Un graphe G tel que $\overline{M}(G) = 31/9 < K(G)/2 = 7/2$.

5.2 Efficacité de l'algorithme des LS_1-synchronisations

Soit $G = (V, E)$ un graphe connexe. Soit $\alpha(G)$ la taille d'un ensemble de sommets indépendants contenus dans G, voir [63]. Le calcul de ce nombre dans des graphes

quelconques est NP-difficile. C'est le nombre d'étoiles sommets-disjointes, et c'est en même temps le nombre maximal de LS_1-synchronisations que l'on puisse réaliser, sachant que deux LS_1-synchronisations peuvent avoir lieu en parallèle si et seulement si elles ont lieu sur deux étoiles arêtes disjointes.

De le même façon, on définit l'efficacité d'un algorithme probabiliste pour réaliser des LS_1-synchronisations :

Définition 8 Soit $\mathcal{A}$ un algorithme probabiliste quelconque permettant de réaliser des LS_1-synchronisations. Son *efficacité* $\Delta_{\mathcal{A}}(G)$ sur un graphe G est le rapport :

$$\Delta_{\mathcal{A}}(G) = \frac{\overline{M}_{1\mathcal{A}}(G)}{\alpha(G)}, \tag{5.12}$$

où $\overline{M}_{1\mathcal{A}}(G)$ est le nombre moyen de LS_1-synchronisations réalisées par $\mathcal{A}$.

Comme pour les RV-synchronisations, c'est le rapport entre le nombre moyen réalisé par l'algorithme $\mathcal{A}$ et le nombre réalisé par un algorithme "idéal" (lequel ne peut être conçu de manière distribuée).

On énonce alors le théorème principal de la section :

Théorème 12 *L'efficacité de l'algorithme des LS_1-synchronisations sur un arbre quelconque T est strictement supérieur à $\frac{1}{3}$.*

Preuve. D'après le corollaire 8, et comme T est un arbre, on a

$$\overline{M}_{1\mathcal{A}}(T) > \frac{n}{3}, \tag{5.13}$$

cependant, il est clair que $\alpha(T) < n$, ce qui finit la preuve. □

Remarque. Dans le cas général, l'algorithme peut avoir une efficacité très faible. En effet, soit $\varepsilon > 0$, il existe un graphe G_ε tel que $\Delta_{\mathcal{A}}(G) < \varepsilon$. Considérons le graphe G_n à $2n$ sommets $v_1, ..., v_n, u_1, ..., u_n$ construits comme suit. L'ensemble $\{u_1, ..., u_n\}$ forme une clique dans G_n, et il y a une arête $\{u_i, v_j\}$ pour tout $1 \leqslant i \leqslant n$ et tout $1 \leqslant j \leqslant n$. Ainsi $v_1, ..., v_n$ sont indépendants et $\alpha(G_n) = n$. D'un autre coté,

$$M_1(G_n) = \frac{n}{2n} + \frac{n}{n+1}. \tag{5.14}$$

Quand $n \to \infty$,

$$\Delta_{LC_1}(G_n) \to 0, \tag{5.15}$$

et ne peut être minoré par un réel positif uniforme pour tous les graphes.

5.3 Efficacité de l'algorithme des LS_2-synchronisations

Soit $G = (V, E)$ un graphe connexe. Soit $\beta(G)$ la taille d'une couverture maximale du graphe G par des étoiles sommets disjointes, c'est le nombre maximal de boules disjointes de rayon 1 couvrant le graphe. C'est aussi le nombre maximum de LS_2-synchronisations réalisables sur le graphe, sachant que deux LS_2-synchronisations peuvent avoir lieu en parallèle sur deux sommets disjoints v et v' si et seulement si la distance entre v et v' est supérieure ou égale à 3. On définit alors l'efficacité d'un algorithme probabiliste $\mathcal{B}$ permettant de réaliser des LS_2-synchronisations par :

Définition 9 Soit $\mathcal{A}$ un algorithme probabiliste quelconque permettant de réaliser des LS_2-synchronisations. Son *efficacité* $\Lambda_{\mathcal{B}}(G)$ sur un graphe G est définie par le rapport :

$$\Lambda_{\mathcal{B}}(G) = \frac{\overline{M}_{2\mathcal{B}}(G)}{\beta(G)}, \tag{5.16}$$

où $\overline{M}_{2\mathcal{B}}(G)$ est le nombre moyen de LS_2-synchronisations réalisées par $\mathcal{B}$.

La borne inférieure de cette efficacité est différente des bornes pour les deux autres algorithmes. En effet :

Théorème 13 *L'efficacité de l'algorithme des LS_2-synchronisations sur un arbre quelconque T est strictement supérieur à $\frac{1}{4}$.*

Preuve. Pour prouver le théorème, on doit montrer que $\overline{M_2}(T) > \frac{1}{4}\beta(T)$. On va donc utiliser une technique similaire à celle de la preuve du théorème 12.

On utilise une induction sur la taille de T. Le théorème est vrai pour les étoiles. Nous supposons que c'est vrai pour les arbres de taille inférieure à n. Supposons que T est un arbre de taille n. Supposons aussi que T ne soit pas une étoile, et que T soit enraciné en l'un de ses sommets.

Soit a un sommet de T de degré $d \geqslant 2$. Soit S l'arbre obtenu à partir de T par suppression de a et toutes les feuilles voisines et arêtes incidentes.

Par induction, on a $\overline{M_2}(S) > \frac{1}{4}\beta(S)$ et $\beta(T) \leqslant \beta(S) + 1$, il suffit de prouver que $\overline{M_2}(T) - \overline{M_2}(S) \geqslant \frac{1}{4}$.

Un calcul simple de l'effet de l'ajout de a et ses fils à la somme définissant $\overline{M_2}$, permet d'obtenir une borne inférieure de $\overline{M_2}(T) - \overline{M_2}(S)$.

En effet, on a

$$\overline{M_2}(T) - \overline{M_2}(S) \geqslant \frac{d-1}{d+1} + \frac{1}{d+d'} - \frac{1}{(d'+1)(d+d'+1)} - \frac{d'-1}{d'(d'+1)}. \tag{5.17}$$

En prenant en compte le fait que $d \geqslant 2$ et $d' \geqslant 2$, si on développe la dernière expression, un calcul permet de montrer que

$$\overline{M_2}(T) - \overline{M_2}(S) - \frac{1}{4} > 0. \tag{5.18}$$

Le théorème en résulte. □

Remarque. Le théorème précédent montre que la borne inférieure est $\frac{1}{4}$; cependant, on ne connaît pas d'arbre T pour lequel l'efficacité soit inférieure à $\frac{1}{3}$.

5.4 Conclusion et perspectives

Dans cette partie, nous avons proposé et analysé différents algorithmes probabilistes pour synchroniser des sommets voisins et permettre l'implémentation des calculs locaux dans les graphes. Pour chacun des algorithmes proposés, nous avons établi un certain nombre de propriétés et de résultats :
Pour l'algorithme du rendez-vous, nous avons vu que c'est un algorithme de type Las Vegas en établissant que la probabilité de succès (au moins un rendez-vous dans le graphe) est minorée par $1 - e^{-1/2}$. Pour le nombre moyen de rendez-vous dans le graphe, nous avons établi que c'est le graphe complet qui minimise ce nombre, et dans l'étude asymptotique, nous avons vu que la probabilité de succès pour le graphe complet vaut $1 - e^{-1/2}$. Il reste à prouver ou à rejeter la conjecture suivante :

Conjecture 1 *Si n est fixe, alors sur tous les graphes connexes de taille n, c'est le graphes complet qui minimise la probabilité de succès.*

Pour prouver cette conjecture, nous pensons qu'il suffit de partir d'un graphe connexe quelconque de taille n, et de trouver une "bonne" stratégie de rajout de nouvelles arêtes de manière à ne pas augmenter la probabilité de succès. En effet, on ne connaît qu'un seul graphe (figure 4.4 de H. Austinat et V. Diekert) pour lequel le rajout d'une nouvelle arête est susceptible d'augmenter la probabilité de succès, et dans ce graphe, la seule fois où cette augmentation peut arriver est lorsque l'on rajoute l'arête entre les deux sommets 1 et 2.

Les résultats expérimentaux de la section 4.1.7 montrent que ce n'est pas les arbres chaînes qui maximisent le nombre moyen de rendez-vous mais une autre classe d'arbres ; Il faudrait donc arriver à caractériser cette classe d'arbres.

Pour les algorithmes des LS_1- et LS_2-synchronisations, nous avons vu que la probabilité d'au moins une synchronisation de ce type dans le graphe est égale à 1. Nous avons aussi établi des bornes inférieures pour le nombre moyen des LS_1- et LS_2-synchronisations.

Le dernier chapitre de cette partie, introduit une nouvelle mesure de l'efficacité pour les algorithmes probabilistes autorisant des synchronisations locales. Cette mesure est le rapport entre le nombre moyen de synchronisations réalisées par l'algorithme et le nombre de synchronisations que réalise un algorithme idéal.

Pour l'algorithme de rendez-vous et celui de LS_1-synchronisation, nous savons que dans le cas des arbres, cette efficacité est minorée par 1/3. Pour l'algorithme de LS_2-synchronisation, nous avons établi que, toujours dans le cas des arbres, cette efficacité est minorée par 1/4. Cependant, nous ne connaissons pas d'arbre tel que l'efficacité de cet algorithme soit inférieure à 1/3. Nous pensons donc que :

Conjecture 2 *L'efficacité de l'algorithme de LS_2-synchronisation sur un arbre quelconque T est supérieure à* 1/3.

Troisième partie

Analyse d'un algorithme d'élection dans les arbres

Chapitre 6

Problème d'élection

Dans cette partie, nous nous intéressons au problème d'élection dans les arbres. Certains résultats préliminaires à cette étude ont été abordés dans [56].

Le problème d'élection est un paradigme de l'algorithmique distribuée. L'élection dans un système distribué se caractérise par une suite de transformations du système le faisant passer d'une configuration initiale où tous les processus sont dans l'état *candidat*, à une configuration finale où un seul processus est dans l'état *élu* et tous les autres dans l'état *battu* [40].

6.1 Définitions

Rappelons d'abord quelques définitions. La plupart des notions utilisées sont issues du livre de G. Tel [73], ainsi que de [42].

Le modèle utilisé est le modèle standard d'un réseau de communication point-à-point représenté par un graphe. Le réseau est un réseau de processeurs ou d'ordinateurs. Dans la suite, on utilisera le terme processeur pour désigner un nœud du réseau.

Algorithmes en vagues

On considère un réseau de processeurs asynchrones. L'ensemble des processeurs est noté $\mathbb{P}$. Un calcul C dans un tel réseau est une collection d'événements partiellement ordonnés, par une relation de précédence $\preceq$. Chaque processeur $p \in \mathbb{P}$ a son propre ensemble d'événements $C_p \subset C$ correspondant au calcul qu'il effectue. Chaque calcul C contient un événement spécial lui permettant de prendre une décision, on notera cet événement *décision*.

Un *algorithme en vagues* échange un nombre fini de messages et prend alors une décision, qui dépend de certains événements propres à chaque processeur. Formellement, un algorithme en vagues est un algorithme distribué satisfaisant les trois

conditions suivantes :

— *Terminaison* : chaque calcul est fini, c'est-à-dire, $\forall C,\ |C| < \infty$.

— *Décision* : chaque calcul contient au moins un événement *décision*, c'est-à-dire, $\forall C,\ \exists e \in C$ tel que e soit un événement *décision*.

— *Dépendance* : dans chaque calcul, chaque événement *décision* est précédé d'un événement local à chaque processeur, c'est-à-dire,

$$\forall C,\ \forall e \in C,\ (e \text{ est un événement } \textit{décision}\ ) \Rightarrow \forall q \in \mathbb{P},\ \exists f \in C_q \text{ tel que } f \preceq e.$$

Dans l'ensemble des processeurs, on distingue les *initiateurs* des *non-initiateurs*. Un initiateur est un processeur capable de lancer spontanément l'exécution de l'algorithme en fonction d'un ensemble de conditions internes. Un non-initiateur ne peut commencer l'exécution de l'algorithme que s'il reçoit un message.

Plus d'informations sur les algorithmes en vagues peuvent être trouvées dans [73].

Réécriture de graphes et algorithmes distribués

Un système distribué peut être codé par un graphe étiqueté : les sommets correspondent aux processeurs, les arêtes aux liens de communication et les étiquettes associées codent l'état du processeur ou du lien de communication.

Une règle de calcul dans le système s'exprime alors sous la forme d'une modification locale des étiquettes (réécriture). Ce formalisme offre des outils pour prouver certaines propriétés des algorithmes distribués [8, 57], pour comprendre ce que l'on peut faire ou les limites de ces calculs.

L'exemple 3.1 du chapitre 3 réalise le codage d'un algorithme calculant un arbre couvrant.

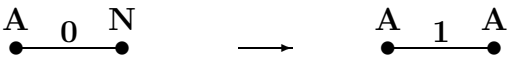
Les systèmes de réécriture ont été introduits par Billaud, Lafon, Métivier et Sopena dans [8], et ont été étudiés dans [42, 43]. Suivant [8, 42, 43], un *système de réécriture de graphe* (GRS) est un triplet $\mathcal{R} = (\mathcal{L}, \mathcal{I}, \mathcal{P})$ où $\mathcal{L}$ est un ensemble d'*étiquettes*, $\mathcal{I}$ un sous-ensemble de $\mathcal{L}$ qui est l'ensemble des *étiquettes initiales* et $\mathcal{P}$ un ensemble de *règles de réécritures*.

Une application d'une règle de réécriture dans un graphe (ou un réseau) donné consiste en :

1. trouver dans le graphe un sous-graphe isomorphe à la partie gauche de la règle (appelé *occurrence* de la règle),
2. modifier les étiquettes comme décrit par la partie droite de la règle.

Le graphe est dit *irréductible* si on ne trouve aucun sous-graphe isomorphe à une partie gauche d'une des règles.

Exemple 6.1 L'algorithme donné dans l'exemple 3.1 du chapitre 3 peut être codé par le système de réécriture $\mathcal{R}_1 = (\mathcal{L}_1, \mathcal{I}_1, \mathcal{P}_1)$ avec $\mathcal{L}_1 = \{\mathbf{N}, \mathbf{A}, \mathbf{0}, \mathbf{1}\}, \mathcal{I}_1 = \{\mathbf{N}, \mathbf{A}, \mathbf{0}\}$, et $\mathcal{P}_1 = \{R\}$, où R est la règle suivante :

R : 

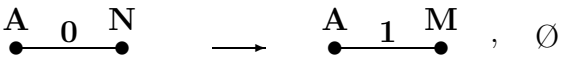
Une règle avec *contexte interdit* est une règle de réécriture de la forme :

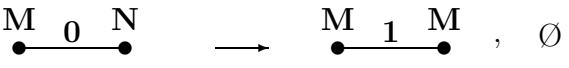
R : $\mathbf{G} \longrightarrow \mathbf{D}$, $\mathbf{I}_1,\mathbf{I}_2,\cdots,\mathbf{I}_k$

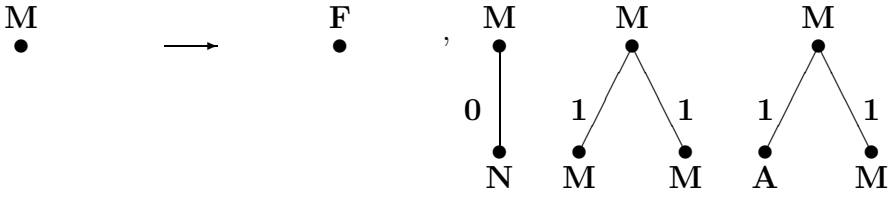
telle que **G** peut être réécrit en **D** si **G** n'est pas un dans un des contextes $\mathbf{I}_1,\mathbf{I}_2,\cdots,\mathbf{I}_k$.

Un système de réécriture *avec contexte interdit* (FCGRS) est un GRS $\mathcal{R} = (\mathcal{L}, \mathcal{I}, \mathcal{P})$ où $\mathcal{P}$ est une ensemble de règles avec contexte interdit.

Exemple 6.2 Soit $\mathcal{R}_2 = (\mathcal{L}_2, \mathcal{I}_2, \mathcal{P}_2)$ le FCGRS défini par $\mathcal{L}_2 = \{\mathbf{N}, \mathbf{A}, \mathbf{M}, \mathbf{F}, \mathbf{0}, \mathbf{1}\}$, $\mathcal{I}_2 = \{\mathbf{N}, \mathbf{A}, \mathbf{0}\}$, $\mathcal{P}_2 = \{R_1, R_2, R_3\}$ où R_1, R_2 et R_3 sont les règles de réécriture avec contexte interdit suivantes :

R_1 : 

R_2 : 

R_3 : 

Les règles R_1 et R_2 n'ont pas de contextes interdits alors que la règle R_3 signifie qu'un sommet étiqueté **M** peut être réétiqueté **F** s'il n'a pas de voisin étiqueté **N** et au plus un voisin étiqueté **A** ou **M**.

Soit alors $G = (V, E)$ un graphe quelconque, avec un seul sommet étiqueté **A**, les autres étiquetés **N** et les arêtes **0**. Le système de réécriture $\mathcal{R}_2$ calcule un arbre couvrant de G de manière distribuée.
Lorsque le graphe est irréductible, l'ensemble des arêtes marquées **1** forme un arbre couvrant du graphe [42].

La figure 6.1 donne un exemple d'exécution de ce système sur un graphe.

Un résumé sur les systèmes de réécriture de graphes et leur usage pour le codage et l'étude des algorithmes distribués peut être trouvé dans [42].

6.2 Problème d'élection

Le problème d'élection a été introduit pour la première fois par LeLann [41]. Dans une configuration initiale du réseau, tous les processeurs sont dans un état *candidat*;

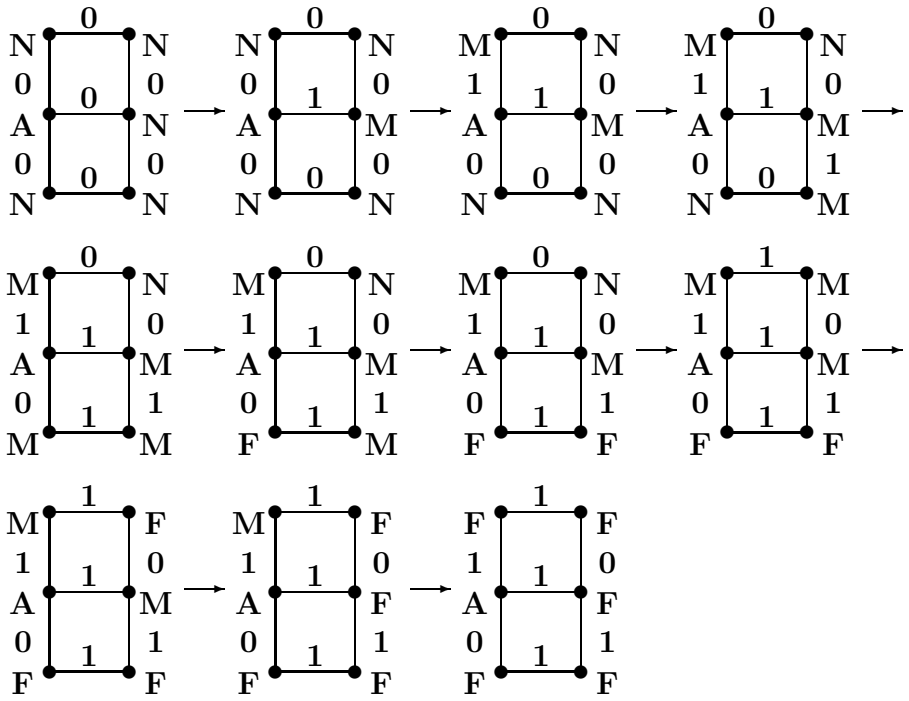


FIGURE 6.1 – Une application du système $\mathcal{R}_2$.

après un certain nombre de transformations, on obtient une configuration finale où un processeur, et un seul, est dans l'état *élu* (ou *leader*) et les autres processeurs dans l'état *battu* (ou *non-leader*), [40].

Ce problème est classique dans le domaine des algorithmes distribués ; l'existence d'une solution et sa complexité dépendent des hypothèses faites sur le système, [1, 26, 47, 43]. Dans la section suivante, on donne quelques exemples d'algorithmes.

6.2.1 Quelques algorithmes d'élection

Élection dans un réseau anneau

Le problème de l'élection dans les réseaux en anneau a été beaucoup étudié et plusieurs solutions ont été proposées [41, 11, 37, 60, 16]. Ici nous présentons un algorithme dû à Chang et Roberts [1][11].

L'anneau est supposé être unidirectionnel, asynchrone, les processeurs n'ont aucune connaissance globale sur le réseau, chaque processeur a une identité unique et l'ensemble des identités est totalement ordonné.

L'algorithme se déroule en trois étapes :

1) chaque processeur P_i envoie un jeton contenant son identité id_i à son suivant. Si un processeur P_j reçoit un jeton contenant une identité id_i telle que $id_j > id_i$,

1. L'algorithme a été initialement proposé par Le Lanne dans [41]. La version que nous présentons ici est une version améliorée de cet algorithme, version due à Chang et Roberts [11].

alors il supprime ce jeton de l'anneau. Ainsi, tous les jetons qui transitent dans l'anneau sont supprimés à l'exception du jeton de plus grande identité.

2) le processeur qui reçoit son propre jeton sait qu'il est le processeur élu. En effet, seul le processeur d'identité maximale est dans ce cas, puisque les autres ont été supprimés.

3) le processeur élu envoie un message contenant son identité et une indication de fin de l'élection. Après $n-1$ messages, les autres processeurs savent quel est le processeur élu.

L'étude de l'algorithme et la preuve de sa validité peuvent être trouvées dans [61].

Élection dans un réseau à topologie quelconque

Dans un réseau à topologie quelconque, le problème d'élection et le problème de recherche d'arbre couvrant sont équivalents. L'algorithme que nous décrivons dans ce paragraphe est emprunté au livre [3] de H. Attiya et J. Welch et illustre bien ce propos. Chaque processeur génére un jeton dans lequel il met son identité et commence un calcul d'arbre couvrant du réseau en utilisant un parcours en profondeur. Quand deux arbres se rencontrent en un processeur, celui-ci se joint à l'arbre couvrant dont l'identité (contenue dans le jeton) est la plus grande. Le processeur ayant réussi à calculer son propre arbre couvrant (dont il est la racine) est le sommet élu. Il envoie, via cet arbre, un message à tous les autres processeurs afin de les informer du résultat.

La spécification, ainsi que l'étude de cet algorithme peuvent être trouvées dans [3].

6.2.2 L'algorithme d'élection étudié

Dans cette section, nous présentons un algorithme d'élection pour les réseaux anonymes. C'est un algorithme qui peut être utilisé si la topologie du réseau est un arbre ou plus généralement, dans tout réseau où un arbre couvrant est disponible. C'est un algorithme en vagues, il peut être codé par un système de réécriture de graphes.

Nous commençons par rappeler un algorithme dit *algorithmme d'arbre* présenté dans [73].C'est un algorithme en vagues dont les initialisateurs sont les feuilles de l'arbre. Si un processeur reçoit un message via chacun des ports le reliant à ses voisins sauf un, il envoie un message à travers le lien restant. Si un processeur reçoit un message à travers tous les liens, alors il prend une décision (voir l'algorithme 5).

Remarque. Dans l'algorithme 5, seuls deux sommets (voisins) arrivent à l'étape de décision, les autres sommets attendent la réception d'un message et s'arrêtent au point **x** de l'algorithme, [73].

Algorithme 5: L'algorithme arbre

var $rec_p[q]$: booléen pour tout $q \in voisins_p$ (* initialisé à *faux* *);
début
  tant que $\#\{q : \neg rec_p[q]\} > 1$ **faire**
    début
      $recevoir(jeton)$ de q;
      $rec_p[q] \leftarrow vrai$;
    fin
  (* Maintenant, il y a *un* q_0 tel que $rec_p[q_0]$ est faux *)
  $envoyer(jeton)$ à q_0 dont $\neg rec_p[q_0]$;
  x :$recevoir(jeton)$ de q_0;
  $rec_p[q_0] \leftarrow vrai$;
  décider;
fin

À la fin de l'exécution de l'algorithme les deux sommets restants doivent décider. Cette décision peut être implémentée de manière à réaliser une élection. En effet, il suffit d'implémenter l'événement *décider* comme exprimé dans l'algorithme 6.

Algorithme 6: Une implémentation de l'événement *décider* pour obtenir une élection

début
  $a_p \leftarrow$*aléatoire()* (* tirer un nombre aléatoire *);
  $envoyer(a_p)$ à q_0;
  $recevoir(a_{q_0})$;
  si $a_p > a_{q_0}$ **alors** $état_p \leftarrow$ **Élu**;
  sinon $état_p \leftarrow$ **Perdant**;
fin

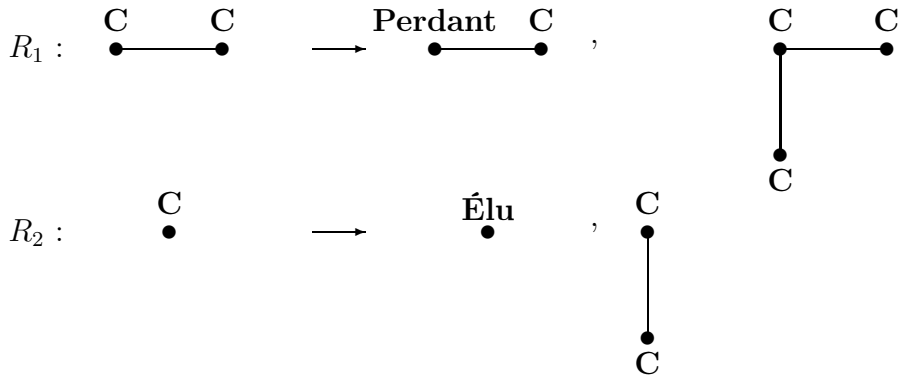
Les deux sommets atteignant le point **x** tirent chacun un nombre aléatoire, ils échangent les nombres qu'ils ont tiré et celui qui a tiré la plus grande valeur est le sommet élu. Le tirage est supposé se faire dans un ensemble suffisamment grand pour que la probabilité que les deux sommets tirent deux valeurs identiques soit nulle.

L'algorithme peut être décrit par un système de réécriture de graphe avec contexte interdit FCGRS $\mathcal{R} = (\mathcal{L}, \mathcal{I}, \mathcal{P})$, avec $\mathcal{L} = \{$**C**, **Perdant,Élu**$\}$, $\mathcal{I} = \{$**C**$\}$ et $\mathcal{P} = \{R_1, R_2\}$, où les deux règles R_1 et R_2 sont définies comme suit :

R_1 **:** un sommet x, étiqueté **C**, qui est adjacent à exactement un sommet étiqueté **C**, peut être réétiqueté avec **Perdant**,

R_2 **:** si un sommet étiqueté **C** n'a aucun sommet étiqueté **C** dans son voisinage, alors il peut être réétiqueté avec **Élu**.

Ou encore plus formellement :



Proposition 19 *Si le système $\mathcal{R}$ est appliqué à un arbre ayant n sommets, initialement étiquetés* **C***, alors, après n pas de réécriture, on obtient un graphe irréductible avec un sommet unique étiqueté* **Élu***, les autres sommets étant étiquetés* **Perdant***.*

Preuve. Pour prouver la proposition, il suffit de montrer la terminaison du système de réécriture ainsi que sa validité. Chaque réécriture supprime une étiquette **C** et la remplace par un étiquette **Perdant**, donc après au plus n applications des règles R_1 ou R_2, le graphe est irréductible.

Pour prouver sa validité, il suffit de montrer l'invariant suivant : à chaque application de la règle R_1, l'ensemble des sommets étiquetés **C** est un arbre, et à la fin, on a un sommet étiqueté **Élu**. Cette propriété est vérifiée au début et tant que l'arbre résiduel contient au moins deux sommets, on applique la règle R_1. Cependant, l'application de cette règle ne se fait que sur des feuilles de l'arbre, c'est-à-dire qu'à chaque réécriture, les sommets supprimés sont des feuilles de l'arbre, donc la propriété est conservée. Enfin, une fois l'arbre réduit à un seul sommet, on applique la règle R_2, et ce sommet est le sommet élu. □

Il est facile de voir que l'algorithme ainsi décrit est un algorithme en vagues dont les initiateurs sont les feuilles de l'arbre. En effet, la proposition 19 assure la terminaison en un temps fini de l'algorithme. L'existence d'un sommet étiqueté **Élu** à la fin de l'algorithme est l'événement de *décision*. Quand l'algorithme fournit un sommet étiqueté **Élu**, c'est-à-dire l'événement de décision, tous les sommets ont fait au moins un calcul local, d'où la *dépendance*.

L'objectif de cette étude est de calculer la probabilité d'élection pour chaque sommet d'un arbre donné. Nous démontrons en particulier, que sous certaines hypothèses, les sommets médians sont les sommets qui ont la probabilité la plus élevée d'être élus.

Nous fournissons aussi des implémentations possibles de cet algorithme suivant différentes considérations.

Chapitre 7

Analyse de l'algorithme

Dans ce chapitre, on considère un arbre $T = (V, E)$ de taille n. D'après la proposition 19, si le système de réécriture $\mathcal{R}$ est appliqué à T, (tous les sommets sont initialement étiquetés **C**), après n applications de la règle de réécriture, on obtient un arbre irréductible dont un seul sommet est étiqueté **Élu**. Le processus d'application des règles de réécriture peut alors être vu comme une suite de suppressions successives des feuilles de l'arbre, la dernière feuille restante étant le sommet élu.

À chaque étape, il y a au moins deux possibilités, si l'on suppose $n \geqslant 2$. L'objectif est ici d'évaluer la probabilité d'être élu pour chaque sommet, lorsqu'on affecte des probabilités aux choix non déterministes. Comme dans les investigations similaires, on peut développer plusieurs points de vue, tous justifiables suivant la méthode choisie pour l'implémentation particulière de l'algorithme. À titre d'exemple, on peut considérer les hypothèses de suppressions suivantes :

(i) Toutes les suites d'enlèvements sont de même probabilité.

(ii) À chaque étape, toutes les feuilles ont la même chance d'être supprimées, et cela indépendamment de tous les événements précédents.

On note alors pour chaque sommet x de l'arbre T, $p_x(T)$ la probabilité pour x d'être élu selon la première considération **(i)**, et $q_x(T)$ la probabilité pour x d'être élu selon la deuxième considération **(ii)**.

7.1 Analyse suivant la considération (i)

Soit x un sommet quelconque de V, la probabilité $p_x(T)$ (ou simplement p_x) d'être élu pour x est le rapport du nombre de suites se terminant sur x sur le nombre total de suites.

Soit $e(T)$ le nombre de suites de suppressions successives aboutissant à une élection dans T, et soit $e_x(T)$ le nombre de suites de suppressions aboutissant à l'élection

de x. On a alors

$$p_x(T) = \begin{cases} 1 & \text{si } T \text{ est réduit au sommet } x \\ \sum_{T'} \frac{e(T')}{e(T)} p_x(T') & \text{sinon.} \end{cases} \tag{7.1}$$

7.1.1 Ordre partiel des facteurs

Pour un arbre donné T de taille n, l'ensemble $\mathcal{T}$ des arbres facteurs de T, muni de la relation binaire $\sqsubseteq$, " facteur de ", est un ensemble (partiellement) ordonné. On voit aisément que $(\mathcal{T}, \sqsubseteq)$ est un treillis gradué (voir [71], chap. 3).
Soit C_u^v le nombre de facteurs saturés de u à v, pour $u, v \in \mathcal{T}$, c'est-à-dire le nombre de suites d'enlèvements de feuilles qui, appliqués au facteur u, le réduisent au facteur v.

En utilisant cette notation, la probabilité d'élection pour x, selon le premier point de vue, se transcrit de la manière suivante :

$$p_x(T) = \frac{C_T^x}{C_T^{\emptyset}}. \tag{7.2}$$

Par ailleurs, on peut utiliser la fonction ζ sur l'algèbre d'incidence $\mathcal{T}$ sur $\mathbf{R}$ pour calculer C_u^v (voir [71]). La fonction ζ est donnée par :

$$\zeta(u, v) = \begin{cases} 1 & \text{si } u \sqsubseteq v \\ 0 & \text{sinon.} \end{cases} \tag{7.3}$$

La probabilité $p_x(T)$ devient :

$$p_x(T) = \frac{(\zeta - 1)^{n-1}(x, T)}{(\zeta - 1)^n(\emptyset, T)}. \tag{7.4}$$

Exemple 7.1 Soit T l'arbre de la figure 7.1, $(\mathcal{T}, \subseteq)$ peut être représenté par son diagramme de Hasse de la figure 7.2. Ce diagramme permet de compter les chaînes saturées entre deux sommets du diagramme. Nous avons :

$$C_T^1 = 2, \quad C_T^2 = 8, \quad C_T^3 = 12, \quad C_T^4 = C_T^5 = 3, \quad C_T^{\emptyset} = 28. \tag{7.5}$$

D'où :

$$p_1 = \frac{1}{14}, \quad p_2 = \frac{2}{7}, \quad p_3 = \frac{3}{7}, \quad p_4 = p_5 = \frac{3}{28}. \tag{7.6}$$

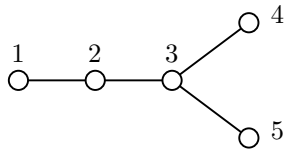


FIGURE 7.1 – Arbre de l'exemple.

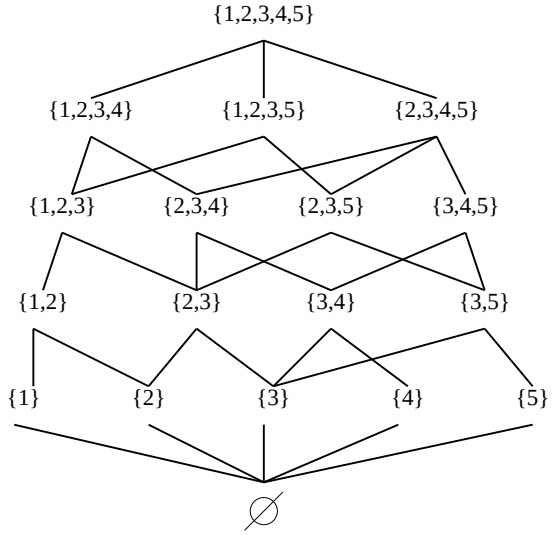


FIGURE 7.2 – Diagramme de Hasse de l'exemple.

7.1.2 Forme matricielle

Il est aisé d'introduire la matrice carrée A dont les lignes et les colonnes sont identifiées avec les facteurs de T. Soient u et v deux facteurs de T ; l'élément A_u^v, dans la $u^{\text{ième}}$ ligne et $v^{\text{ième}}$ colonne de A est donné par :

$$A_u^v = \begin{cases} 1 & \text{si } v \sqsubset v \\ 0 & \text{sinon.} \end{cases} \tag{7.7}$$

Il est clair que le nombre de chaînes de longueur k d'un facteur v à un facteur u est $(A^k)_u^v$. Nous avons en particulier :

$$p_x(T) = \frac{(A^{n-1})_T^x}{(A^n)_T^{\emptyset}}. \tag{7.8}$$

7.1.3 Relation avec les monoïdes de commutation

Soit T un arbre de taille n, dont les sommets sont identifiés avec les entiers $1, 2, ..., n$. On peut faire correspondre à T un monoïde de commutation $S = (V, \theta)$ comme suit. L'alphabet V est l'ensemble $\{1, 2, ..., n\}$ et $(i, j) \in \theta$, $i, j \in V$, $i \neq j$, si et seulement si i et j ne sont pas adjacents. L'arbre T est donc le graphe de conflit de S (voir le chapitre 1).

À chaque suite d'enlèvements de feuilles, aboutissant à l'élection du sommet $x \in V$, correspond un mot $w = i_1 i_2 ... i_{n-1} x \in V^n$, où i_k est la $k^{\text{ième}}$ feuille enlevée. Appelons, dans ce qui suit, ce mot *représentation*. La représentation d'une élection est donc un mot $w = i_1, i_2 ... i_{n-1} i_n$ de longueur n sur V, ayant exactement une occurrence de chaque lettre, tel que, pour tout $k = 1, ..., n$, le mot $i_k ... i_n$ corresponde à un arbre facteur de T. Soit $W(x)$ l'ensemble des représentations qui aboutissent à l'élection de x.

La proposition suivante fournit une méthode de comptage en terme de cardinalité de traces dans un système de commutation voir [13, 14]. On peut donc écrire $p_x(T)$ en termes de cardinalités des traces dans le monoïde de commutation.

Proposition 20 *$W(x)$ est une trace dans le système de commutation S. Le nombre de suites, aboutissant à l'élection de x est la cardinalité de la trace dans laquelle se trouve le mot w, w étant la représentation d'une suite pour l'élection de x.*

Preuve. Pour un sommet x donné, soit $w = i_1 i_2 ... i_{n-1} x \in V^n$ la représentation d'une suite pour l'élection de x. Nous avons à démontrer que la trace $[w]$ contient toutes les représentations correspondant à l'élection de x et uniquement celles-ci.

(i) Considérons un mot $u = j_1 j_2 ... j_{n-1} x$, représentant une suite pour l'élection de x. Pour prouver que $u \in [w]$, il suffit de montrer que, pour tout couple (k, l) de sommets adjacents dans T, $\pi_{\{k,l\}}(u) = \pi_{\{k,l\}}(w)$, où $\pi_{\{k,l\}}$ est la projection sur l'alphabet$\{k, l\}$, voir [14] pour la preuve. Les lettres k et l doivent avoir les mêmes positions relatives dans u et v, sinon soient $w = w_1 k w_2 l w_3 x$ et $u = u_1 l u_2 k u_3 x$. Mais cela impliquerait qu'il existe un chemin de l à x, sans passer par k, et un chemin de k à x sans passer par l. Ceci est impossible, puisque, k et l étant adjacents, on déduirait qu'il y a deux chemins de k à x dans T.

(ii) Soit, par ailleurs, u un mot congruent à w. Nous allons prouver que $u \in W(x)$, c'est-à-dire u représente aussi une suite correspondant à l'élection de x. Nous pouvons postuler que u est obtenu de w en permutant des lettres qui ne sont pas adjacentes dans T. Puisque x est précédée d'une lettre qui ne commute pas avec elle dans w, u doit se terminer aussi par une occurrence de x. Soient maintenant $w = w_1 k l w_2 x$ et $u = w_1 l k w_2 x$, avec k et l non adjacents. Le sommet k est une feuille dans le facteur $k l w_2 x$ et le sommet l dans $l w_2 x$.

Puisque k et l ne sont pas adjacents, il en résulte que l est une feuille dans lkw_2x et k dans kw_2x. Par conséquent, u est aussi la représentation d'une suite pour l'élection de x.

□

7.1.4 Une propriété combinatoire

Dans cette section, nous présentons une propriété combinatoire des probabilités d'élection dans un arbre selon la considération **(i)**.

Soit $T = (V, E)$ un arbre quelconque, T étant biparti, il admet une bi-coloration, c'est-à-dire que ses sommets peuvent être coloriés utilisant deux couleurs et de manière à ce que deux sommets voisins dans T n'ait pas la même couleur. La proposition suivante affirme que si l'on dispose d'une bi-coloration de T, alors pour la considération **(i)**, le sommet élu a la même probabilité d'être d'une couleur ou de l'autre.

Proposition 21 *Soit $T = (V, E)$ un arbre de taille $n \geqslant 2$. Étant donnée une bi-coloration $V = V_{Noir} \cup V_{Blanc}$, on a*

$$\sum_{x \in V_{Noir}} p_x(T) = \sum_{x \in V_{Blanc}} p_x(T). \tag{7.9}$$

Preuve. Par induction sur la taille n de l'arbre T, notons $e(T)$ le nombre de suites de suppressions aboutissant à une élection dans T. La distributions vérifie la relation de récurrence suivante :
si $n = 2$ alors

$$\sum_{x \in V_{Blanc}} p_x(T) = \sum_{x \in V_{Noir}} p_x(T) = 1/2$$

sinon

$$p_x(T) = \sum_{T'} \frac{e(T')}{e(T)} p_x(T'). \tag{7.10}$$

La somme étant faite sur tous les sous arbres de T obtenus par suppression d'une feuille.

On suppose la proposition vraie pour tout arbre de taille n et on considère un arbre T de taille $n + 1$. On a :

$$\begin{aligned}
\textstyle\sum_{x \in V_{Noir}} p_x(T) &= \textstyle\sum_{x \in V_{Noir}} \sum_{T'} \frac{e(T')}{e(T)} p_x(T') \\
&= \textstyle\sum_{T'} \frac{e(T')}{e(T)} \sum_{x \in V_{Noir}} p_x(T') \\
&= \textstyle\sum_{T'} \frac{e(T')}{e(T)} \sum_{x \in V_{Blanc}} p_x(T') \\
&= \textstyle\sum_{x \in V_{Blanc}} \sum_{T'} \frac{e(T')}{e(T)} p_x(T') \\
&= \textstyle\sum_{x \in V_{Blanc}} p_x(T)
\end{aligned}$$

□

7.1.5 Calcul effectif de p

Nous introduisons ici une méthode simple pour calculer la distribution p. Elle est fondée sur la comparaison des probabilités d'élection pour deux sommets voisins. Ces comparaisons, par ailleurs, aboutissent à une caractérisation simple d'un ou de deux sommets ayant la probabilité la plus élevée d'être élu(s).

Dans cette section $T = (V, E)$ est un arbre fixé de taille $n \geqslant 2$.

Pour un couple (x, y) de sommets adjacents, désignons par $N_x(x, y)$ la cardinalité de la composante connexe dans laquelle se trouve x, lorsqu'on supprime l'arête xy ; c'est le nombre de sommets qui se trouvent du côté de x par rapport à son voisin y. Un sommet x est dit *dominant*, si pour tout sommet adjacent y, on a $N_x(x, y) \geqslant N_y(x, y)$. Il est facile de voir que tout arbre possède un ou deux sommets dominants.

Dans la figure suivante (figure 7.3), on a $N_x(x, y) = 5$ et $N_y(x, y) = 3$.

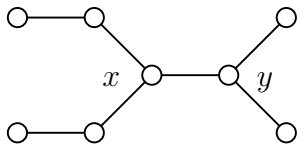


FIGURE 7.3 –

L'algorithme (séquentiel) ci-dessous (algorithme 7) détermine un sommet dominant dans un arbre. On associe à tout sommet un poids initialement égal à 1. L'algorithme supprime de proche en proche une feuille de poids minimal, en augmentant le poids du père du poids du sommet supprimé. Le dernier sommet est un sommet dominant. Le sommet dominant dans l'exemple de la figure 7.5 est le sommet 3.

Un autre algorithme distribué cette fois-ci peut être utilisé pour déterminer le sommet dominant (qui permet à un sommet de se rendre compte qu'il est le sommet dominant). C'est un algorithme en vague et il est donné par l'algorithme 8.

Lemme 8 *Pour tout couple (x, y) de sommets adjacents dans T, nous avons :*

$$\frac{C_T^{xy}}{C_T^{y}} = \frac{N_x(xy)}{n-1} \tag{7.11}$$

Preuve. La suppression de l'arête xy divise T en deux composantes connexes T_x et T_y de tailles $N_x(xy)$ et $N_y(xy)$ respectivement.
Puisque chaque sommet différent de x dans T_x commute avec tout sommet différent

Algorithme 7: Un algorithme séquentiel pour déterminer le sommet médian

```
début
    pour tout x ∈ V faire
        poids(x) := 1;
    tant que Card(V) > 1 faire
        choisir une feuille a de poids minimal;
        V := V \ {a};
        pour le sommet b adjacent à a faire
            poids(b) := poids(b) + poids(a);
            E := E \ {ab};
fin
```

de y dans T_y, le numérateur du premier membre de 7.11 (en vertu d'une légère généralisation de la proposition 20) est le nombre de mots, finissant avec xy, obtenus par un shuffle d'un mot $w_1 \in W_1(x)$ et un mot $w_2 \in W_2(y)$, où $W_1(x)$ (resp. $W_2(y)$) est l'ensemble des mots représentant l'élection de x (resp. y) dans T_x (resp. T_y). Le dénominateur est le nombre des mêmes mots sujets à la condition de se terminer par une occurrence de y (au lieu de xy).

Nous avons donc :

$$C_T^{xy} = Card(W_1(x))Card(W_2(y))\frac{(n-2)!}{(N_x(xy)-1)!(N_y(xy)-1)!} \tag{7.12}$$

et

$$C_T^{y} = Card(W_1(x))Card(W_2(y))\frac{(n-1)!}{N_x(xy)!(N_y(xy)-1)!}. \tag{7.13}$$

Le lemme se déduit facilement de 7.12 et 7.13. □

La proposition suivante fournit une technique simple pour le calcul de la distribution p en comparant les probabilités d'élection pour les sommets voisins.

Proposition 22 *Pour tout couple (x, y) de sommets adjacents, nous avons :*

$$\frac{p_x}{p_y} = \frac{N_x(xy)}{N_y(xy)}. \tag{7.14}$$

Preuve. Le lemme 8 calcule C_T^y (et de façon similaire C_T^x) en fonction de C_T^{xy} et $N_x(xy)$. Par ailleurs, le rapport $\frac{p_x}{p_y}$ est le même que $\frac{C_T^x}{C_T^y}$, d'où la proposition. □

On déduit facilement de cette proposition :

Algorithme 8: Un algorithme distribué pour calculer le sommet médian.

var $rec_p[q]$: booléen pour tout $q \in voisins_p$ (* initialisé à *faux* *);
w_p **: poids de** p **(* initialisé à 1 *);**
$poids_p[q]$: entier pour tout $q \in voisins_p$ (* initialisé à 0 *);
début
  tant que $\#\{q : \neg rec_p[q]\} > 1$ **faire**
    début
      $recevoir(w_q)$ de q ; $rec_p[q] \leftarrow vrai$; $w_p \leftarrow w_p + w_q$ **;**
      $poids_p[q] \leftarrow w_q$;
    fin
  (* Maintenant, il y a *un* q_0 tel que $rec_p[q_0]$ est faux *)
  $envoyer(w_p)$ à q_0 dont $\neg rec_p[q_0]$;
  x :$recevoir(w_{q_0})$ de q_0 ; $rec_p[q_0] \leftarrow vrai$;
  (* *décider* *)
  si $w_p \geqslant 2\max\{poids_p[q]\}$ **alors** Se déclarer médian;
  sinon $envoyer(w_p)$ à q_0 tel que $poids_p[q_0] = \max\{poids_p[q]\}$;
fin

Corollaire 11 *Dans tout arbre, les sommets dominants ont la probabilité p la plus élevée d'être élus.*

Il existe de nombreuses caractérisations des sommets médians [34, 39, 54, 77]. Ici nous utilisons la suivante [77] :

Proposition 23 *Soit $T = (V, E)$ un arbre ; un sommet v est médian, si et seulement si, pour tous ses voisins w, on a $N_w(vw) \leqslant \frac{1}{2}Card(V)$.*

On en déduit :

Proposition 24 *Dans un arbre, ce sont les sommets médians qui ont la probabilité p la plus élevée d'être élus.*

Preuve. Dans l'algorithme donné ci-dessus, on supprime d'abord les voisins du sommet dominant puis celui-ci. Par conséquent, lorsqu'un sommet est supprimé, son poids est inférieur à $\frac{1}{2}n = \frac{1}{2}Card(V)$.
Réciproquement, un sommet médian ne peut être supprimé avant l'un des ses voisins b non médian, sinon on aurait $N_a(ab) < \frac{1}{2}n$ et $N_b(ab) > \frac{1}{2}n$. On en déduit alors qu'un sommet est médian si et seulement si, il est dominant. La proposition est alors une conséquence du corollaire 11. □

La proposition 22 permet de donner une expression explicite des probabilités p_x.

Soit x un sommet quelconque de l'arbre T. Pour toute arête e, posons :

$$\theta_x(e) = \frac{n - N_x(e)}{N_x(e)}. \tag{7.15}$$

$\theta_x(e)$ est donc le rapport entre les tailles des composantes connexes engendrées par la suppression de e, le numérateur étant la taille de la composante dans laquelle se trouve x et le dénominateur celle de l'autre composante. Nous avons :

Proposition 25 *Pour tout sommet x de T, p_x s'écrit :*

$$p_x = \left[\sum_{y \in V} \prod_{e \in c(x,y)} \theta_x(e) \right]^{-1}, \tag{7.16}$$

$c(x, y)$ étant le chemin unique liant x à y dans T.

Preuve. Soit y un sommet quelconque. Il existe un chemin unique $x_0 = x, x_1, ..., x_k = y,\ k \geqslant 0$, de x à y. La proposition 25 permet d'exprimer successivement $p_{x_1}, ..., p_{x_k}$ en fonction de p_x. Nous avons :

$$p_y = p_x \prod_{e \in \{x_0, ..., x_k\}} \theta_x(e). \tag{7.17}$$

La proposition est alors établie par le fait que la somme des probabilités vaut 1. □

7.1.6 Algorithme d'élection et chaînes de Markov

Soit $T = (V, E)$ un arbre de taille $n \geqslant 2$. Nous introduisons deux chaînes de Markov $X^{(t)}$ et $Y^{(t)}$ sur V, considéré comme l'ensemble des états. Elles sont données par les matrices de transitions P et Q, comme suit.
Soient x et y deux sommets quelconques de V. On pose :

$$P_x^y = \begin{cases} \frac{N_y(xy)}{n-1} & \text{si } x \text{ et } y \text{ sont adjacents} \\ 0 & \text{sinon,} \end{cases} \tag{7.18}$$

et

$$Q_x^y = \begin{cases} \frac{N_y(xy)}{n} & \text{si } x \text{ et } y \text{ sont adjacents} \\ \frac{1}{n} & \text{si } x = y \\ 0 & \text{sinon.} \end{cases} \tag{7.19}$$

Il est clair que P et Q sont des matrices stochastiques.

Ces deux matrices, pour l'arbre donné dans la figure 7.5 sont :

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/2 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{7.20}$$

$$Q = \begin{pmatrix} 1/5 & 4/5 & 0 & 0 & 0 \\ 1/5 & 1/5 & 3/5 & 0 & 0 \\ 0 & 2/5 & 1/5 & 1/5 & 1/5 \\ 0 & 0 & 4/5 & 1/5 & 0 \\ 0 & 0 & 4/5 & 0 & 1/5 \end{pmatrix} \tag{7.21}$$

Proposition 26 *Dans tout arbre $T = (V, E)$, le vecteur $p = (p_x, x \in V)$ de distribution de probabilités d'élection est le vecteur propre de P par rapport à la valeur propre 1, c'est-à-dire p est la distribution stationnaire de $X^{(t)}$.*

Preuve. Nous avons :

$$C_T^x = \sum_{y \text{ adjacent à } x} C_T^{xy}. \tag{7.22}$$

En utilisant le lemme 8 pour calculer C_T^{xy}, nous obtenons :

$$C_T^x = \sum_{y \text{ adjacent à } x} \frac{N_y(xy)}{n-1} C_T^y; \tag{7.23}$$

ce qui montre que p est la distribution stationnaire pour la chaîne de Markov $X^{(t)}$. □

Proposition 27 *Pour tout arbre T, le vecteur p est le vecteur propre de Q par rapport à la valeur propre 1, c'est-à-dire que p est la distribution stationnaire de $Y^{(t)}$.*

Preuve. Il suffit de multiplier les membres de 7.23 par $(n-1)$ et de les diviser ensuite par n. □

On remarque que les chaînes $X^{(t)}$ et $Y^{(t)}$ sont irréductibles, la seconde ayant l'avantage d'être aussi apériodique. On a alors, en vertu de la proposition 27, le théorème suivant :

Théorème 14 *Supposons qu'une particule se déplace, aux instants $t = 1, 2, ...$, d'un sommet de l'arbre à un autre sommet voisin (ou à ce même sommet) avec une probabilité donnée par la matrice Q. La probabilité pour que la particule se trouve sur un sommet donné x, à l'instant t, lorsque $t \to \infty$, est égale à la probabilité p_x d'élection pour ce sommet, et ceci, quelque soit la position initiale de la particule.*

7.2 Analyse suivant la considération (ii)

Soit $T = (V, E)$ un arbre quelconque et x un sommet de V. Selon la considération **(ii)**, à chaque étape, toutes les feuilles ont la même chance d'être supprimées, et cela indépendamment de tous les événements précédents. Par conséquent, la probabilité $q_x(t)$ (ou simplement q_x) d'être élu pour un sommet x est donnée par :

$$q_x(T) = \begin{cases} 1 & \text{si } T \text{ est réduit au sommet } x \\ \sum_{T'} \frac{1}{k} q_x(T') & \text{sinon,} \end{cases} \tag{7.24}$$

k étant le nombre de feuilles dans T. La somme s'étend à tous les arbres T' facteurs de T obtenus en supprimant une feuille $y \neq x$.

Exemple 7.2 Reconsidérons l'exemple de la figure 7.1, la distribution $q_i(T), i = 1, ..., 5$, se calcule inductivement, ou en évaluant les transitions dans le diagramme de Hasse (figure 7.4).

On obtient une distribution tout à fait différente de la distribution obtenue avec la considération **(i)** :

$$q_1 = \frac{1}{12}, \quad q_2 = \frac{11}{36}, \quad q_3 = \frac{5}{12}, \quad q_4 = q_5 = \frac{7}{72}. \tag{7.25}$$

Soient $u, v \in \mathcal{T}$, et soit s une suite d'enlèvements de feuilles $s_1, s_2, \cdots, s_k$, qui, appliquées au facteur u, le réduisent au facteur v, on définit le poids de s, et on note $w(s)$ la quantité :

$$w(s) = \frac{1}{f_S(s_1) f_S(s_2) \cdots f_S(s_k)},$$

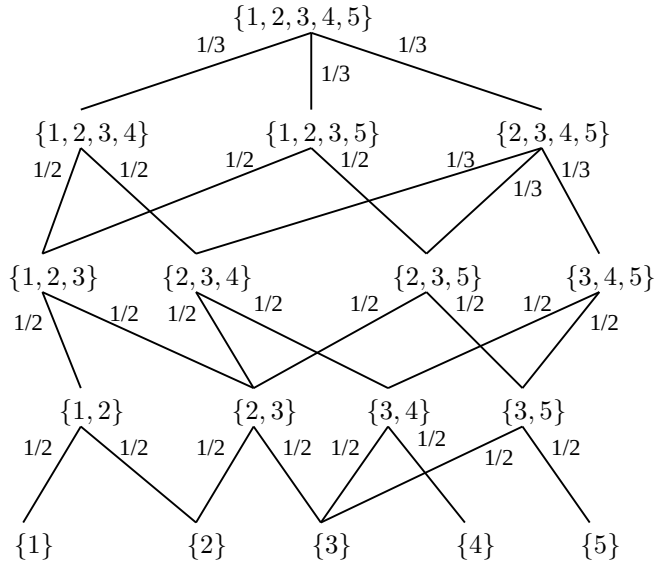


FIGURE 7.4 – Diagramme de Hasse valué de l'exemple.

où $f_S(s_i)$ est le nombre de feuilles de l'arbre s_i, pour tout i, S étant l'arbre résiduel ; $S(u, v)$ désigne l'ensemble des enlèvements, on a :

$$q_x(T) = \sum_{s \in S(T,x)} w(s). \tag{7.26}$$

Remarque. La proposition 24 devient fausse pour la distribution q. Considérons en effet l'arbre de la figure 7.5.

On calcule pour cet arbre la distribution de probabilité q inductivement. Nous obtenons en particulier $q_4 = \frac{697553}{2160000} = 0,322941203...$ et $q_5 = \frac{18971}{72000} = 0,2634866111....$

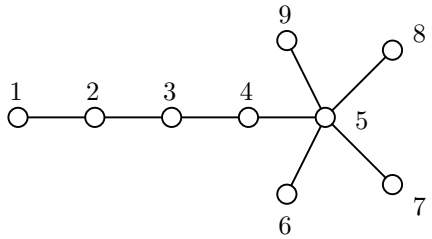


FIGURE 7.5 – Un arbre dans lequel le sommet dominant n'a pas la probabilité q la plus élevée d'être élu.

Le sommet étiqueté 5 est le sommet dominant, sa probabilité q d'être élu est cependant, inférieure à celle du sommet étiqueté 4. Nous ne savons pas s'il y a une caractérisation simple du sommet ayant la plus grande probabilité d'être élu, par

rapport à la distribution q.

7.2.1 Forme matricielle

Pour calculer la probabilité $q_x(T)$ suivant la considération **(ii)**, on peut introduire aussi la matrice B dont les lignes et les colonnes sont identifiées par les facteurs de T. Si u et v sont deux facteurs de T, alors l'élément B_u^v, dans la $u^{\text{ième}}$ ligne et $v^{\text{ième}}$ colonne de B est donné par :

$$B_u^v = \begin{cases} \frac{1}{f(u)} & \text{si } v \text{ est obtenu de } u \text{ par suppression d'une feuille} \\ 0 & \text{sinon.} \end{cases} \tag{7.27}$$

où $f(u)$ est le nombre de feuilles dans u.

Il est alors aisé de voir que la probabilité pour un sommet x d'être élu selon la considération **(ii)** est donnée par :

$$q_x(T) = (B^{n-1})_T^x. \tag{7.28}$$

7.2.2 Une propriété combinatoire

Dans cette section, nous allons voir que la propriété combinatoire déjà établie pour la distribution p (proposition 21) est aussi valide pour la distribution q.

Soit $T = (V, E)$ un arbre quelconque. Supposons que l'on dispose d'une bi-coloration de T, alors pour la considération **(ii)**, le sommet élu a la même probabilité d'être d'une couleur ou de l'autre.

Proposition 28 *Soit $T = (V, E)$ un arbre de taille $n \geqslant 2$. Étant donnée une bi-coloration $V = V_{Noir} \cup V_{Blanc}$, on a*

$$\sum_{x \in V_{Noir}} q_x(T) = \sum_{x \in V_{Blanc}} q_x(T). \tag{7.29}$$

Preuve. Par induction sur la taille n de l'arbre T, notons par $f(T)$ le nombre de feuilles de l'arbre T et remarquons que la distribution q vérifie la relation de récurrence suivante :
si $n = 2$ alors

$$\sum_{x \in V_{Blanc}} q_x(T) = \sum_{x \in V_{Noir}} q_x(T) = 1/2$$

sinon

$$q_x(T) = \sum_{T'} \frac{1}{f(T)} q_x(T') \tag{7.30}$$

La somme étant faite sur tous les sous arbres de T obtenus par suppression d'une feuille.

On suppose la proposition vraie pour tout arbre de taille n et on considère un arbre T de taille $n+1$. On a :

$$\begin{aligned}
\sum_{x\in V_{Noir}} q_x(T) &= \sum_{x\in V_{Noir}} \sum_{T'} \frac{1}{f(T)} q_x(T') \\
&= \sum_{T'} \frac{1}{f(T)} \sum_{x\in V_{Noir}} q_x(T') \\
&= \sum_{T'} \frac{1}{f(T)} \sum_{x\in V_{Blanc}} q_x(T') \\
&= \sum_{x\in V_{Blanc}} \sum_{T'} \frac{1}{f(T)} q_x(T') \\
&= \sum_{x\in V_{Blanc}} q_x(T)
\end{aligned}$$

□

7.2.3 Une implémentation de l'algorithme

Cette implémentation est fondée sur un tirage aléatoire. Soit x un sommet quelconque de l'arbre T. Si x est une feuille, alors elle tire une valeur entière finie représentant sa durée de vie : on suppose que le tirage se fait suivant une loi exponentielle.

À chaque étape, c'est-à-dire à chaque top de l'horloge locale, cette valeur est diminuée de 1. Lorsqu'elle devient égale à 0, la feuille change son étiquette en **Perdant** et envoie un message à son père.

Initialement, chaque sommet p dispose d'un couple *($état_p$,v_p)* initialisé à $(\mathbf{N}, \mathbf{I})$. La variable $état_p$ correspond à l'état de p : supprimé (étiquette **Perdant**) ou non (étiquette **N**) ou encore élu (étiquette **Élu**) ; la variable v_p représente la durée de vie de p, initialement indéfinie (égale à **I**).

L'évolution du couple *($état_p$,v_p)* est décrite par l'algorithme 9. La fonction *uniforme()* est une fonction aléatoire permettant de faire un tirage uniforme et indépendant d'un nombre réel de l'intervalle $[0, 1]$.
La fonction *aléatoire()* est implémentée de façon à retourner un nombre réel suivant une loi exponentielle. Elle peut être implémentée en utilisant la fonction *uniforme()* comme indiqué par l'algorithme 10 qui est une variante d'un algorithme dû à Von Neumann-Knuth-Yao, [40].
Cet algorithme est une adaptation de l'algorithme arbre de la section 6.2.2 du chapitre précédent. C'est un algorithme en vague dont les initiateurs sont les feuilles.
Un sommet p qui est une feuille $\#\{q : \neg rec_p[q]\} = 1$ se génère une durée de vie v_p =*aléatoire()*, à chaque top horloge du sommet, cette durée est réduite, quand $v_p = 0$, p change son $état_p$ en **Perdant** et envoie un message à son père (le seul sommet q tel que $rec_p[q] = faux$).

Algorithme 9: Algorithme exécuté par chaque processeur

```
var  rec_p[q] : booléen pour tout q ∈ voisins_p (* initialisé à faux *);
     v_p : entier (* initialisé à I *);
     état_p : (N,Perdant,Élu) (* initialisé à N *);
début
  tant que #{q : ¬rec_p[q]} > 1 faire
    début
      recevoir(jeton) de q;
      rec_p[q] ← vrai;
    fin
  tant que v_p ≠ 0 faire
    si v_p =I alors v_p ← aléatoire();
  (* Maintenant, il y a un q_0 tel que rec_p[q_0] est faux *)
  état_p ←Perdant;
  envoyer(jeton) à q_0 dont ¬rec_p[q_0];
  x :recevoir(jeton) de q_0;
  rec_p[q_0] ← vrai;
  (* décider *)
  x_p ← uniforme();
  envoyer(x_p) à q_0;
  recevoir(x_{q_0}) de q_0;
  si x_p > x_{q_0} alors état_p ← Élu;
  sinon état_p ← Perdant
fin
```

Un sommet interne $\#\{q : \neg rec_p[q]\} > 1$ commence par recevoir les messages envoyés par les feuilles dont il est le père et qui ont changé d'état ; une fois devenu feuille (il n'y plus qu'un sommet q_0 tel que $rec_p[q_0] = faux$), il initialise l'algorithme d'élection.
D'après la discussion sur l'algorithme d'arbre, il y a juste deux sommets de l'arbre qui franchissent le point **x**, donc qui arrivent à la phase de décision. Les deux sommets sont voisins, chacun tire alors un nombre réel et l'envoie à son voisin, et celui des deux qui a tiré la plus grande valeur est **Élu**, l'autre est **Perdant**.

On a alors la proposition suivante :

Proposition 29 *Si l'algorithme 9 est exécuté par les sommets d'un arbre T, alors la probabilité pour un sommet quelconque d'être élu est donnée par la distribution q.*

Preuve. Il suffit de montrer qu'à chaque étape, toutes les feuilles de l'arbre ont la même probabilité d'être réétiquetées en **Perdant**, (ce qui correspond à la suppression des feuilles dans la considération **(ii)** représentant la distribution q).

Algorithme 10: Fonction *aléatoire(0,1)*

```
début
    var T, n : entier;
        u, v, X, Y : réel;
    T ← 0;
    répéter
        T ← T + 1;  n ← 1;
        u ← uniforme(0, 1);  v ← uniforme(0, 1);
        Y ← u;
        tant que u > v faire
            u ← v;
            v ← uniforme(0, 1);
            n ← n + 1;
    jusqu'à n impair;
    X ← T − 1 + Y;
    retourner(X);
fin
```

Or, on a la propriété suivante : si X est une variable aléatoire suivant une loi exponentielle, alors

$$\forall t, t_0,\ Pr(X > t + t_0 | X > t_0) = Pr(X > t) = e^{-t}. \tag{7.31}$$

La durée de vie pour chaque feuille étant une variable aléatoire suivant une loi exponentielle, toutes les feuilles ont la même probabilité d'être supprimées à chaque instant. Ce qui montre la propriété. □

Une autre implémentation :

Cette implémentation est basée sur un tirage de type pile ou face. Soit x un sommet quelconque de l'arbre T. Si x est une feuille alors il tire une valeur de l'intervalle $[0, 1]$; si cette valeur est inférieure à $\frac{1}{2}$ alors il change son étiquette en **Perdant**, sinon il garde son étiquette.

L'algorithme est exécuté par chaque sommet et à chaque étape, 0, 1 ou plusieurs feuilles peuvent changer d'étiquettes en même temps. Dans la suite, on verra que le temps moyen pour obtenir une élection est n. Cependant, son analyse probabiliste, notamment la réponse à la question : quel est le sommet qui a la probabilité la plus élevée d'être élu est un problème ouvert.

Chaque processeur p dispose d'un couple $(\acute{e}tat_p, v_p)$, où $\acute{e}tat_p$ est une variable prenant ses valeurs dans l'ensemble d'étiquettes {**N**,**Perdant**,**Élu**}, et v_p une variable prenant ses valeurs dans l'intervalle $[0, 1]$.

Initialement, les variables $\acute{e}tat_p$ et v_p sont respectivement égales à **N** et **I** (indéfini). Leur évolution est décrite par l'algorithme 11.

Algorithme 11: Algorithme exécuté par chaque processeur

```
var  rec_p[q] : booléen pour tout q ∈ voisins_p (* initialisé à faux *);
     v_p : entier (* initialisé à I *);
     état_p : (N,Perdant,Élu) (* initialisé à N *);
début
  tant que #{q : ¬rec_p[q]} > 1 faire
    début
      recevoir(jeton) de q;
      rec_p[q] ← vrai;
    fin
  v_p ← uniforme();
  tant que v_p < 1/2 faire
    v_p ← uniforme();
  (* Maintenant, il y a un q_0 tel que rec_p[q_0] est faux *)
  état_p ←Perdant;
  envoyer(jeton) à q_0 dont ¬rec_p[q_0];
  x :recevoir(jeton) de q_0;
  rec_p[q_0] ← vrai;
  (* décider *)
  x_p ← uniforme();
  envoyer(x_p) à q_0;
  recevoir(x_{q_0}) de q_0;
  si x_p > x_{q_0} alors état_p ← Élu;
  sinon état_p ← Perdant
fin
```

L'algorithme est une adaptation de l'algorithme arbre de la section 6.2.2. C'est un algorithme en vagues dont les initiateurs sont les feuilles.

Une feuille tire un nombre aléatoire de l'intervalle $[0, 1]$. Si ce nombre est supérieur à $1/2$, elle change son étiquette en **Perdant**, sinon elle recommence. Quand elle est réétiquetée **Perdant**, la feuille envoie un jeton à son père.

Un sommet interne reçoit les messages des feuilles qui lui sont rattachées, et quand il ne reste qu'un seul sommet duquel il n'a pas reçu de message, il devient une feuille et continue l'algorithme.

À la fin de l'algorithme, deux sommets ont franchi le point **x**, et accèdent à l'étape de décision.

On a alors le résultat suivant :

Proposition 30 *L'espérance mathématique du temps d'attente pour obtenir une élection dans un arbre de taille n est n.*

Preuve. Soit $T = (V, E)$ un arbre de taille n et pour tout i, notons par N_i (resp. f_i) le nombre de sommets (resp. de feuilles) de l'arbre à l'étape i. On a alors

$$\begin{aligned}
\mathbb{E}(N_{i+1}|N_i) &= \tfrac{1}{2^{f_i}} \sum_{k=0}^{f_i} \binom{f_i}{k} (N_i - k) \\
&= \tfrac{1}{2^{f_i}} N_i \sum_{k=0}^{f_i} \binom{f_i}{k} - \tfrac{1}{2^{f_i}} \sum_{k=0}^{f_i} \binom{f_i}{k} k \\
&= N_i - \tfrac{1}{2^{f_i}} \sum_{k=0}^{f_i} \binom{f_i}{k} k \\
&= N_i - \tfrac{f_i}{2}.
\end{aligned}$$

Or, un arbre de taille $\geqslant 2$ a toujours au moins deux feuilles, donc

$$\mathbb{E}(N_{i+1}|N_i) \leqslant N_i - 1,$$

et, pour tout k, on a :

$$\mathbb{E}(N_k|N_0) \leqslant n - k.(\text{ car } N_0 = n).$$

Après $k = n - 2$ étapes d'exécution de l'algorithme, on obtient

$$\mathbb{E}(N_{n-2}) \leqslant 2.$$

On en déduit alors qu'en moyenne, après $n - 1$ étapes, on obtient une élection dans l'arbre. □

7.3 Conclusion et perspectives

L'algorithme proposé dans cette partie permet de réaliser une élection dans un arbre ou dans tout graphe où un arbre couvrant est disponible. Son étude a donné lieu à différentes considérations autant sur le plan de l'analyse probabiliste que sur le plan de l'implémentation.

Pour l'analyse probabiliste, nous avons vu qu'il y a deux façons de considérer le problème :
(i) Toutes les suites de suppressions de feuilles ont la même probabilité (distribution p). Nous avons pu établir quelques propriétés de cette distribution, nous avons notamment pu montrer que c'est le (ou les) sommet(s) médian(s) qui a (ont) la probabilité la plus élevée d'être élu(s). Cependant nous n'avons pas pu fournir une implémentation distribuée de cette distribution de probabilité.
(ii) À chaque étape, toutes les feuilles ont la même probabilité d'être supprimées (distribution q). La relation de récurrence que vérifie cette distribution nous a permis d'obtenir quelques propriétés de cette distribution sans pour autant nous permettre de répondre à la question : quel est le sommet qui a la probabilité q la plus élevée d'être élu ?. Cette question reste donc un problème ouvert.

Cependant, en utilisant un algorithme de tirage aléatoire d'une variable suivant une loi exponentielle, nous avons pu donner une implémentation distribuée de cette distribution.

Pour les deux distributions, et suite à des calculs empiriques, nous avons observé que la différence entre les deux distributions est négligeable par rapport à $1/n$, n étant le nombre de sommets de l'arbre considéré. Cette propriété reste à prouver, ce qui donnerait un algorithme approchant la distribution p.

Enfin, observant qu'à chaque étape, dans la considération **(ii)**, une seule feuille à la fois disparaît, nous avons proposé une autre considération implémentée par l'algorithme 11. À chaque étape, alors 0,1, ou plusieurs feuilles peuvent être supprimées mais l'analyse de cet algorithme est encore un problème ouvert.

Quatrième partie

Routage compact et adaptatif

Chapitre 8

Routage par intervalles

Le routage dans les réseaux de processeurs est un problème classique des systèmes distribués. Pour chaque couple source-destination, il s'agit de donner un chemin entre la source et la destination. Dans un cadre distribué, une façon de coder de tels chemins est de stocker une table locale à chaque routeur lui permettant ainsi d'acheminer tout message lui parvenant.

8.1 Routage et tables de routage

Dans un système parallèle ou distribué, la communication entre les processeurs est assurée par un réseau d'interconnexion qui constitue l'ensemble des liens physiques. Sur chaque processeur est installé un *routeur*, qui peut être vu comme un co-processeur chargé d'acheminer les messages entre les processeurs. Les routeurs mettent en œuvre de façon distribuée un algorithme de routage qui spécifie le chemin à suivre pour se rendre d'un processeur à un autre. Il est décrit par une *fonction de routage*.

Recevant un message sur un de ses ports, un routeur doit être capable de décider vers quel port réémettre le message. Une décision qu'il doit prendre en fonction des informations concernant le destinataire, contenues en général dans l'en-tête du message, et des informations locales au routeur, contenues dans sa mémoire. Cette mémoire est organisée sous forme de *table de routage* qui est stockée localement sur chaque routeur. C'est une table qui doit contenir, pour chaque processeur du réseau, un numéro de port via lequel un message qui lui est destiné pourrait être acheminé. Une façon simple d'organiser une table est d'associer à chaque numéro de processeur le numéro du port qui peut le desservir. Ainsi, sur l'exemple de la figure 8.1, est représenté un réseau dont les routeurs, et les ports sont numérotés. Chaque routeur détient une table *locale* de routage. Sur cette table, sont représentés les numéros des autres routeurs et les numéros des ports qui peuvent les desservir.

Par exemple, un message envoyé par le routeur 6 en destination du routeur 2 est d'abord routé via le port 3, puis une fois sur le routeur 1, il est routé via le port 1.

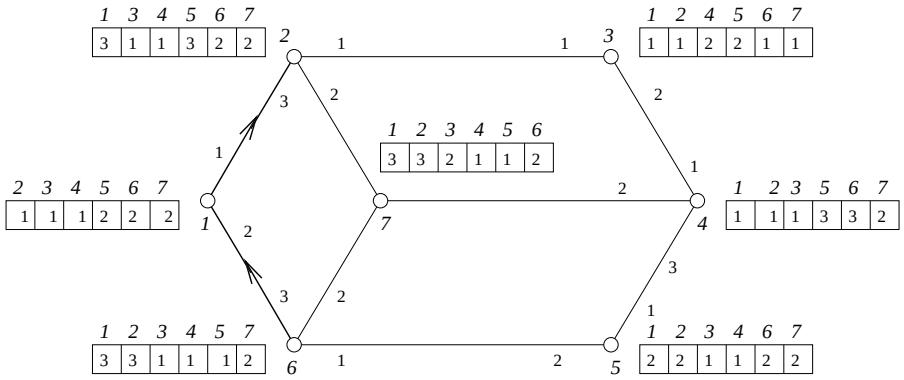


FIGURE 8.1 – Tables de routage pour un réseau.

Cependant, une telle méthode est beaucoup trop coûteuse en espace mémoire. Elle nécessite pour chaque processeur x ayant d voisins, et se trouvant dans un réseau à n processeurs, $O(n \log d)$ bits pour stocker la table de routage. Pour les réseaux de très grande taille, ou de taille variable et grandissant comme dans les réseaux dynamiques, cette méthode n'est pas envisageable. D'où l'intérêt de recourir à d'autres méthodes pour réduire la taille mémoire nécessaire à chaque routeur.

Plusieurs méthodes ont été introduites pour réduire la taille mémoire du routeur, dont le routage par préfixe [5], le routage booléen [21], et *le routage par intervalles* technique à laquelle nous allons nous intéresser ici.

Le routage par intervalles a été introduit pour la première fois par Santoro et Khatib dans [68]. Depuis, de nombreux travaux ont été menés pour étudier cette méthode et de nombreux résultats ont vu le jour.

Cependant, tous les travaux qui ont été menés sur cette méthode ont été fait dans un cadre *déterministe*[1], et peu de chose a été dit dans le cadre *adaptatif*, d'où l'intérêt de ce travail.

Il consiste à étudier la possibilité d'étendre cette technique au cas adaptatif, c'est-à-dire lorsque plusieurs chemins peuvent être utilisés en fonction de la charge du réseau ou de tout autre paramètre.

1. Ici, le mot déterministe n'a pas la même signification que dans les parties précédentes, en effet, le non déterminisme ici signifie avoir le choix entre plusieurs possibilités et non pas l'absence du hasard.

8.2 Routage par intervalles

Tous les réseaux de processeurs ou d'ordinateurs considérés dans la suite sont modélisés par un graphe fini et connexe $G = (V, E)$, V étant l'ensemble des sommets qui représentent les processeurs ou les routeurs, et E l'ensemble des arêtes représentant les liens physiques entre les processeurs.

Notons, pour tout $x \in V$, $out(x)$ l'ensemble des arêtes d'origine x dans G et par $\deg(x)$ le cardinal de cet ensemble, c'est-à-dire le degré de x. Finalement, on note par $\delta(G)$ le degré minimum de G, ainsi, $\delta(G) = \min\{\deg(u) \mid u \in V\}$.

Nous avons vu que le routage par intervalles fût introduit pour réduire la taille des tables de routage. A chaque arête de $out(x)$ est associé un ensemble de destinations codé par un ou plusieurs intervalles d'entiers consécutifs, chaque destination ne pouvant appartenir qu'à un seul lien. Déterminer le port par lequel un message doit être routé revient à déterminer l'intervalle auquel le numéro de son destinataire appartient. Dans la figure 8.2, nous avons représenté le codage du routage par intervalles pour l'exemple de la figure 8.1.

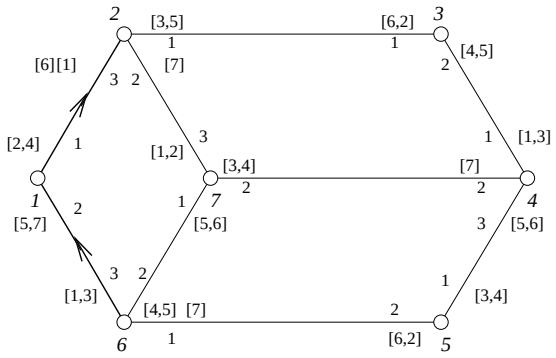


FIGURE 8.2 – Routage par intervalles dans un réseau.

Pour router un message du routeur 6 à destination du routeur 2, on voit que 2 appartient à l'intervalle $[1, 3]$, donc le message sera routé via le port numéro 3, une fois au routeur 1, on voit que 2 appartient à l'intervalle $[2, 4]$, le message sera donc routé via le port numéro 1.

Dans les travaux portant sur le routage par intervalles, il a été montré que $O(kd \log n/k)$ bits suffisent pour coder localement la table de routage, n étant l'ordre du graphe G, d son degré maximum et k le nombre maximum d'intervalles affectés à chacune des arêtes de G.

Les travaux qui ont été effectués sur cette méthode de routage ont consisté à essayer de trouver la valeur minimale de k pour obtenir un routage suivant les plus courts chemins, étant donné que n et d sont en général fixés.

Ces travaux furent menés dans un cadre déterministe : pour chaque paire source-destination, il n'existe qu'un seul chemin possible. Le chemin de routage est donc

entièrement déterminé par ces seuls intervalles.

8.3 Tables de routage adaptatives

L'étude des routages par intervalles adaptatifs (ou non déterministes) a été suggérée dans [74]. Les fonctions de routage dites *adaptatives* permettent de faire varier le chemin de routage. Une destination peut se trouver maintenant dans plusieurs intervalles.

Soit α un entier $\geqslant 1$ quelconque. On définit une *table de routage α-adaptative* comme étant une table de routage dans laquelle chaque destination apparaît dans exactement α sortie distinctes.

Un *étiquetage α-adaptatif à k intervalles*, ou k-ILS$_\alpha$, est une table de routage α-adaptative telle que pour chaque port, l'ensemble des étiquettes des destinations pouvant être atteintes en utilisant ce port peut être regroupé dans au plus k intervalles.
Un ILS$_\alpha$ est *valide* si pour tout couple (u, v) de source-destination $u \neq v$, il existe dans u (et tous les autres sommets intermédiaires) au moins un port parmi les α possibles qui construit un chemin vers v. Un ILS$_\alpha$ valide est dit *routage par intervalles α-adaptatif*, on le notera alors IRS$_\alpha$.

Dans la figure 8.3, nous avons représenté un routage 2-adaptatif sur le réseau de la figure 8.1. À chaque routeur, on peut vérifier que l'on a deux ports possibles pour atteindre toute destination. En plus, pour chaque couple source-destination, on a au moins un chemin valide pour aller de la source à la destination.

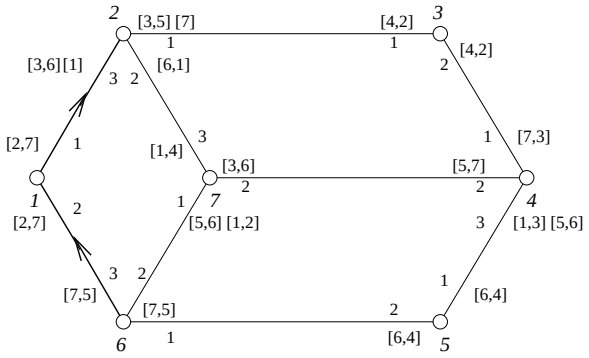


FIGURE 8.3 – Routage par intervalles 2-adaptatif dans un réseau.

La définition implique l'adaptativité d'un routage vue que à chaque étape, le routeur peut sélectionner la prochaine arête du chemin parmi les α possibilités. Ceci donne plusieurs chemins de routage, mais non nécessairement des chemins disjoints, ce qui aurait nécessité une condition supplémentaire sur la connéctivité du graphe. Le

routeur sait que au moins un chemin le connecte à la destination. Les autres chemins, dits de *déflection*, sont utilisés en fonction de la charge du réseau ou dépend de tout autre paramètre. Le cas $\alpha = 1$ correspond au cas classique : le cas déterministe.

Bien sûr, dans la pratique, pour une réelle implémentation du protocole de routage, une fonction de *sélection* choisit un port parmi l'ensemble valide. L'adaptativité implique la nécessité de stocker dans chaque routeur un total de $O(kd\log(n/k))+|S|$ bits pour l'information de routage, où $|S|$ est le nombre de bits nécessaires pour coder la fonction de sélection S. Par exemple, une méthode de sélection peut être le choix aléatoire parmi toutes les permutations des chemins possibles retournés par le routeur si plusieurs messages arrivent en même temps sur le routeur.

8.4 Définitions formelles et notations

Un intervalle $[a, b]$ de l'ensemble $\{1, \cdots, n\}$ est l'ensemble des entiers consécutifs (modulo n) compris entre a et b. Par exemple, l'intervalle $[7, 2]$ de $\{1, 2, \cdots, 9\}$ est l'ensemble $\{7, 8, 9, 1, 2\}$. Si $a \leqslant b$, on dira que $[a, b]$ est un intervalle *linéaire*, et sinon que $[a, b]$ est *cyclique*.

Une *fonction de routage* dans un graphe $G = (V, E)$ est une fonction qui, pour chaque paire source-destination, fournit un ou plusieurs chemins (suites d'arcs de G), pour aller de la source à la destination.

L'approche étant une approche distribuée, chaque routeur a sa propre fonction locale, de telle sorte que pour chaque sommet intermidiare v, (source incluse), et pour chaque destination w, on ne peut fournir que le prochain arc à utiliser pour router de u à w. Chaque arête incidente à un sommet v porte un numéro local (numéro de port), de 1 à $\deg(v)$. En v, pour chaque destination w, on spécifie un (ou plusieurs) numéro d'arc. De la même façon, on peut inverser la représentation en listant l'ensemble de sommets qui utilisent le même arc (ou même numéro d'arc).

Dans cette approche, une table de routage dans un graphe à n sommets est une paire $(\mathcal{L}, \mathcal{I})$ de fonctions dans laquelle : $\mathcal{L} : V \to \{1, \ldots, n\}$ est un étiquetage des sommets et $\mathcal{I} : E \to 2^{\mathcal{L}(V)}$ un étiquetage des arcs tel que pour tout arc $(u, v) \in E$, $\mathcal{L}(w) \in \mathcal{I}(u, v) \subset \{1, \ldots, n\}$ si et seulement si le chemin de u vers w utilise l'arc (u, v).

En plus, étant donné un entier $\alpha \geqslant 1$, une paire $(\mathcal{L}, \mathcal{I})$ est un étiquetage par intervalles α-adaptatif, ILS_α, si pour tous $u, w \in V$, $w \neq u$, l'ensemble $\{(u, v) \in E \mid \mathcal{L}(w) \in \mathcal{I}(u, v)\}$ est de cardinalité α. Un ILS_α *valide* est appelé IRS_α (schéma de routage par intervalles α-adaptatif) et répond à la condition de connéctivité : pour tous $u, w \in V$, $w \neq u$, il existe une suite $\rho = (v_1, \ldots, v_t)$ de sommets telle que $v_1 = u$ et $v_t = w$, et pour chaque $i \in \{1, \ldots, t-1\}$, $\mathcal{L}(w) \in \mathcal{I}(v_i, v_{i+1})$. La suite ρ est un chemin de routage de u à w, et peut ne pas être un chemin simple de G.

Un IRS_α de *plus court chemin* est un IRS_α dans lequel, pour chaque paire (u, w),

il existe un chemin ρ reliant u à w qui est un plus court chemin dans G.

Rappelons que l'on n'impose que l'existence d'au moins un plus court chemin ρ, même si plusieurs chemins sont représentés par l'étiquetage.

Les étiquetages représentant plusieurs ou tous les plus courts chemins sont appelés respectivement *multi-path* et *overall* IRS, et ont été étudiés dans [22, 20, 64].

Remarque. Une conséquence directe de la définition ci-dessus, est que seuls les graphes dont le degré minimum est au moins α supportent un ILS_α et ainsi un IRS_α. Une variante de la définition précédente pour remédier à ce problème consiste à imposer $|\{(u,v) \in E \mid \mathcal{L}(w) \in \mathcal{I}(u,v)\}| = \min\{\alpha, \deg(u)\}$. Même si tous les résultats que nous présentons sont valides pour les deux définitions, pour simplifier, seule la première définition sera considérée dans la suite.

Définition 10 (Compacité) La *compacité* d'un ILS_α $(\mathcal{L}, \mathcal{I})$ est le plus petit entier k tel que tout ensemble $\mathcal{I}(u,v)$ peut être représenté comme l'union d'au plus k intervalles d'entiers consécutifs de $\{1, \ldots, n\}$, (1 et n étant considérés comme consécutifs). De tels ILS_α et IRS_α sont notés respectivement k-ILS_α et k-IRS_α.

Remarque. Pour $\alpha = 1$, les définitions correspondent aux définitions standards ILS/IRS introduites par [68, 75]. Pour simplifier, on notera IRS pour IRS_1. Les étiquetages que l'on considère dans ce chapitre sont supposés être *stricts* : on impose que $\mathcal{L}(u) \notin \mathcal{I}(u,v)$, pour tout $(u,v) \in E$.

Chapitre 9

Routage par intervalles adaptatif

Dans ce chapitre, on ne s'intéresse pas au codage de la fonction de sélection S, mais plutôt au paramètre k. En effet, le codage de la fonction de sélection peut dépendre de la stratégie choisie pour optimiser le trafic : les liens peuvent être choisis de manière aléatoire, ou sélectionnés en fonction d'un historique dépendant de la charge du réseau, ou encore prédits à partir d'une quelconque autre stratégie arbitraire.

9.1 Un schéma d'étiquetage général

Dans cette section, nous montrons que tout graphe G supporte un 1-IRS$_\alpha$ pour tout $\alpha \leqslant \delta(G)$. Ce résultat peut être vu comme une généralisation du schéma d'étiquetage de [68] (tout graphe admet un 1-IRS), et sera l'outil principal pour le reste du chapitre.
Pour cela, nous avons besoin du théorème qui affirme que chaque k-IRS$_\alpha$ peut être transformé en un k-IRS$_{\alpha+1}$ du moment que le degré minimal du graphe est supérieur à $\alpha + 1$. De plus, la transformation conserve tous les chemins représentés dans le k-IRS$_\alpha$ d'origine.

Théorème 15 *Soit $(\mathcal{L}, \mathcal{I}_A)$ un k-IRS$_\alpha$ quelconque dans un graphe $G = (V, E)$ de taille n, avec $\delta(G) \geqslant \alpha+1$, et soit $Y \subseteq E$ tel que tout sommet x a au plus un voisin y de sorte que $(x, y) \in Y$. Alors, $(\mathcal{L}, \mathcal{I}_A)$ peut être transformé en un k-IRS$_{\alpha+1}$ dans G, $(\mathcal{L}, \mathcal{I}_B)$, de sorte que tous les chemins représentés par $(\mathcal{L}, \mathcal{I}_A)$ sont préservés dans $(\mathcal{L}, \mathcal{I}_B)$, et pour tout $(x, y) \in Y$, $\mathcal{I}_B(x, y) = [1, n] \setminus \{\mathcal{L}(x)\}$.*

Preuve. Définissons la procédure suivante qui prend en entrées $(\mathcal{L}, \mathcal{I}_A)$ et Y, et retourne $(\mathcal{L}, \mathcal{I}_B)$. La figure 9.1 représente une application de cette procédure à un sommet du graphe.

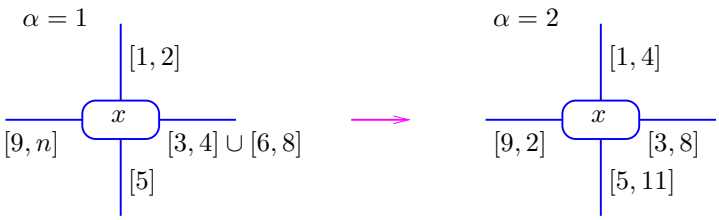


FIGURE 9.1 – Une application de la transformation R à un sommet : au départ chaque destination apparaît 1 fois ($\alpha = 2$). À l'arrivée, chaque destination apparaît 2 fois ($\alpha = 2$).

Pour tout sommet x faire (en parallèle si possible) :

1. Pour tout $(x, y) \in E$, poser $\mathcal{I}_B(x, y) = \mathcal{I}_A(x, y)$.
2. Poser $R = [1, n] \setminus \{\mathcal{L}(x)\}$.
3. Soit (x, y) l'unique arc de Y (si y n'existe pas aller en 4),
   Poser $\mathcal{I}_B(x, y) = R$, et mettre à jour $R = R \setminus \mathcal{I}_A(x, y)$.
4. Tant que $R \neq \emptyset$ faire :
   (a) Trouver y et z tels que $(x, y) \in E$, $z - 1 \pmod n \in \mathcal{I}_B(x, y)$, et $z \in R \setminus \mathcal{I}_B(x, y)$.
   (b) Trouver $y' \neq y$ tel que $(x, y') \in E$, et $z \in \mathcal{I}_B(x, y')$. Soit $[a, b]$ le plus grand intervalle tel que $[a, b] \subseteq \mathcal{I}_B(x, y')$ et $z \in [a, b]$.
   (c) Mettre à jour $R = R \setminus ([a, b] \setminus \mathcal{I}_B(x, y))$.
   (d) Mettre à jour $\mathcal{I}_B(x, y) = \mathcal{I}_B(x, y) \cup [a, b]$.

Soit x un sommet quelconque. Montrons que pour tout i, à la i^{ime} instruction 4, pour tout $z \neq \mathcal{L}(x)$, l'ensemble R vérifie la propriété :

P_i : R contient au plus $n - i$ étiquettes, et si $z \in R$ alors soit z apparaît dans α ensembles $\mathcal{I}_B$, soit il apparaît dans $\alpha + 1$ ensembles $\mathcal{I}_B$,

c'est-à-dire que à la i^{me} itération, R correspond à l'ensemble des étiquettes qui n'ont pas encore été traitées.

Par induction sur i : à la première exécution, dans l'instruction 4, R est l'ensemble de toutes les étiquettes (excepté $\mathcal{L}(x)$),et $\mathcal{I}_B$ est initialisé à $\mathcal{I}_A$. Ainsi, P_1 est vraie. À présent, supposons que la propriété est vraie pour les premières i itérations. Pour montrer que P_{i+1} est vraie, montrons d'abord que l'instruction 4a est faisable, c'est à dire que le couple (y, z) existe : premièrement, si $i = 1$, c'est-à-dire, $R = [1, n] \setminus \{\mathcal{L}(x)\}$, alors, il suffit de choisir un y tel que $\mathcal{I}_A(x, y) \neq R$ (Il existe, car sinon, toute étiquette $\neq \mathcal{L}(x)$ doit apparaître dans au moins $\delta(G) \geqslant \alpha + 1$ ensembles $\mathcal{I}_A$), et de choisir un $z \notin \mathcal{I}_A(x, y)$ (ainsi $z \in R$) de sorte que $z-1$ (mod n) $\in \mathcal{I}_A(x, y)$.

Pour $i > 1$, un couple (y, z) existe sinon, z et $z-1 \pmod n$ doivent apparaître dans le même nombre d'ensembles $\mathcal{I}_B$, ce qui est impossible pour tout $z \in R$ avec $0 < |R| \leqslant n - i < n - 1$ vue que à ce point R vérifie la propriété P_i. Donc, pour tout i, l'instruction 4a est faisable.
Les instructions 4b, 4c, et 4d sont faisables aussi. On remarque que, dans l'instruction 4c, $|R|$ diminue par au moins un élément : $[a, b]$ contient au moins z. Ainsi, P_{i+1} est vérifiée.

Donc, à la fin de la dernière itération ℓ, R est vide et par la propriété P_ℓ, toutes les étiquettes apparaissent dans $\alpha + 1$ ensembles $\mathcal{I}_B$. En prenant l'union dans l'instruction 4d, on peut garantir que $\mathcal{I}_B(x, y) \subseteq \mathcal{I}_A(x, y)$, et préserver ainsi les chemins. Il en résulte que $(\mathcal{L}, \mathcal{I}_B)$ est un $\text{ILS}_{\alpha+1}$ valide. En plus, dans l'instruction 4c, puisque $z - 1$ et z sont consécutives, le nombre minimum d'intervalles nécessaires pour représenter $\mathcal{I}_B(x, y)$ est au plus celui de $\mathcal{I}_A(x, y)$. Donc, $(\mathcal{L}, \mathcal{I}_B)$ a une compacité au plus k, et par l'instruction 3, tous les arcs de Y ont l'intervalle $[1, n] \setminus \{\mathcal{L}(x)\}$, ce qui termine la preuve. □

Remarque. On ne précise pas la complexité temps de l'algorithme ci-dessus, car il dépend de la structure de données du ILS (celui qui donne la plus faible complexité en temps ne donne pas nécessairement la meilleure complexité en espace). Néanmoins, utilisant une représentation naïve du codage des intervalles du ILS, (c'est-à-dire avec $O(kd \log n)$ bits), ce temps est inférieur à $O(n^4)$, mais peut facilement être réduit à $O(|E|k\alpha)$ en utilisant des structures de données plus efficaces.

Soit G un graphe quelconque et T un arbre couvrant de G. Santoro et Khatib dans [68] donnent une construction d'un 1-IRS_1 basé sur un parcours en profondeur de T. Appliquant le théorème 15 inductivement sur α, on obtient :

Corollaire 12 *Tout graphe G tel que $\delta(G) \geqslant \alpha$, supporte un* 1-IRS_α.

Pour $\alpha = 1$, tout graphe supporte un k-IRS avec $k \leqslant n/2$. Pour $\alpha > 1$ nous montrons que $k \leqslant n/\alpha$. Plus précisement :

Théorème 16 *Soit $(\mathcal{L}, \mathcal{I})$ un $k\text{-IRS}_1$ quelconque sur un graphe G de taille n, et soit $\alpha \leqslant \delta(G)$. Alors, G supporte un $k'\text{-IRS}_\alpha$ tel que tous les chemins de $(\mathcal{L}, \mathcal{I})$ sont préservés, et tel que $k' \leqslant \min\{k, n/\alpha\}$.*

Preuve. Le théorème est trivial pour $\alpha = 1$.
Supposons que $\alpha \geqslant 2$. On construit un ensemble Y composé des arcs ayant le plus grand nombre d'intervalles, pour chaque sommet. Après une première application du théorème 15, on obtient un $k\text{-IRS}_2$ pour G, avec le même ensemble de chemins, et où le nombre d'intervalles affectés aux arcs de Y est réduit à 1.

On peut réappliquer le théorème 15 avec un nouvel ensemble Y encore composé des arcs ayant le plus grand nombre d'intervalles, qui est donc au plus le second plus grand dans $(\mathcal{L}, \mathcal{I})$. Enfin, après un total de $\alpha - 1$ applications du théorème 15 (ce qui est faisable puisque $\alpha \leqslant \delta(G)$), on obtient un k'-IRS$_\alpha$ avec le même ensemble de chemins et où le nombre maximum k', d'intervalles assignés aux arcs est borné par k et aussi par le α^{ime} plus grand nombre d'intervalles affectés à un arc dans $(\mathcal{L}, \mathcal{I})$.

Soit x un sommet de degré d dans G, et soient $k_1, \ldots, k_d$ les nombres d'intervalles des ensembles $\mathcal{I}(x, y)$ (le k-IRS$_1$ définit dans G) pour tous les voisins y de x. En plus, supposons que $k_1 \geqslant \cdots \geqslant k_d$. Pour tout k-IRS$_1$, on a $\sum_{i=1}^{\alpha} k_i \leqslant n$, et ainsi le plus petit terme de la somme, k_α, est au plus n/α. Comme affirmé avant, $k' \leqslant \min\{k, k_\alpha\}$, ce qui finit la preuve. □

Remarque. Le théorème 16 peut être légèrement amélioré à $k' \leqslant \min\{k, (n - \delta(G))/\alpha + 1\}$ si tous les arcs incidents à chaque sommet sont étiquetés avec des étiquettes non vides (dans ce cas on a $\sum_{i=1}^{\alpha} k_i \leqslant n - (\delta(G) - \alpha)$). Cette affirmation a lieu dans le cas des tables de routage de plus courts chemins.

9.2 Étiquetage de plus courts chemins

Dans cette section, on s'intéresse aux IRS$_\alpha$ pour lesquels il existe au moins un plus court chemin pour chaque couple de sommets. D'après le théorème 15, plusieurs graphes peuvent être identifiés comme supportant un k-IRS$_\alpha$ de plus court chemin. Par exemple, les grilles, l'hypercube, le graphe complet, le cycle, les arbres, les graphes planaires, les graphes d'intervalles, etc., ont un 1-IRS de plus court chemin, et ainsi des 1-IRS$_\alpha$ de plus court chemin. Les familles de graphes ayant des $O(1)$-IRS de plus court chemin incluent les tores, les k-arbres avec une constante k, les graphes planaires dont le nombre de faces est constant, etc. (Voir [28] pour un état de l'art complet).

Pour tout graphe G, on note IRS$_\alpha(G)$ la *α-compacité* de G qui est :

$$\text{IRS}_\alpha(G) = \min\{k \mid G \text{ a un } k\text{-IRS}_\alpha \text{ de plus court chemin }\}.$$

9.2.1 Comparaison entre IRS et IRS$_\alpha$

Grâce au théorème 15, on a $\text{IRS}_{\alpha+1}(G) \leqslant \text{IRS}_\alpha(G) \leqslant \cdots \leqslant \text{IRS}_1(G)$. Il n'est pas difficile de vérifier qu'il existe des graphes qui supportent un 1-IRS$_2$ de plus court chemin, comme le graphe de Petersen, ou les wheel-graphes mais qui n'ont pas de $1 - \text{IRS}_1$ de plus courts chemins. Le prochain résultat montre que la différence entre la 1- et α-compacité d'un graphe peut être arbitrairement grande, et suggère que la classe des graphes avec une α-compacité bornée, avec $\alpha > 1$, est assez grande.

Théorème 17 *Pour tout entier $\delta \geqslant 0$, il existe un graphe G à $2^{\delta+3}$ sommets tels que* $\mathrm{IRS}_1(G) \geqslant 2^\delta$ *et* $\mathrm{IRS}_2(G) \leqslant 2\delta + 4$.

Preuve. On utilise la construction donnée par Gavoille-Guévremont dans [29] qui montre que $\mathrm{IRS}_1(G) \geqslant n/8$ pour quelques graphes de taille n avec n puissance de deux.

On rappelle ici la construction :

Pour une $p \times q$ matrice booléenne $M = (M_{i,j})$, soit $G_M = (V_M, E_M)$ le graphe tel que :

1) $V_M = \{v_1, \ldots, v_p\} \cup \{a_1, \ldots, a_q\} \cup \{b_1, \ldots, b_q\}$;
2) $\{x, y\} \in E_M$ si et seulement si $(x = a_j$ et $y = b_j)$, $(x = b_j$ et $y = v_i$ et $M_{i,j} = 1)$, ou $(x = a_j$ et $y = v_i$ et $M_{i,j} = 0)$.

Nous avons $n = |V_M| = p + 2q$. Intuitivement, G_M est un graphe à deux niveaux. Le premier niveau est composé des arêtes de type $\{a_j, b_j\}$, et le second en des v_i (stable) qui sont connectés à a_j ou b_j selon que $M_{i,j} = 0$ ou 1.

Pour toute matrice booléenne M, on note par $\overline{M}$ la matrice M avec chaque bit complimenté. En plus, si $M = (\,X\,Y\,)$, où X et Y sont deux matrices de même dimension, on note $\chi(M) = (\,Y\,X\,)$, c'est-à-dire, la matrice obtenue de M en échangeant les colonnes de X avec celles de Y. On considère une matrice spécifique M_δ, $\delta \geqslant 0$, définie par induction. La construction de M_δ est résumées par l'équation 9.1.

$$M_0 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \chi(M_0) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \quad M_{\delta+1} = \begin{pmatrix} M_\delta & M_\delta \\ \chi(M_\delta) & \overline{\chi(M_\delta)} \end{pmatrix} \tag{9.1}$$

La figure 9.2 représente la matrice M_0 ainsi que le graphe G_{M_0} lui correspondant.

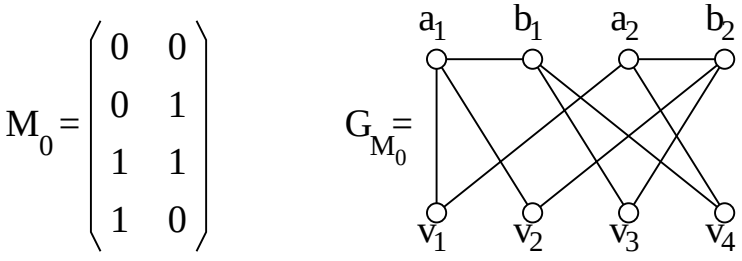


FIGURE 9.2 –

Il est montré dans [29] que $\mathrm{IRS}(G_{M_\delta}) \geqslant 2^\delta$ (intuitivement, quelque soit l'étiquetage des sommets v_i, l'ensemble $\mathcal{I}(a_j, b_j)$ doit contenir seulement un sous-ensemble

particulier des v_i qui est fabriqué pour être difficile à représenter avec un petit nombre d'intervalles). Dans ce cas, $p = 2^{\delta+2}$ et $q = 2^{\delta+1}$, ainsi $n = 2^{\delta+3}$, ce qui prouve la première partie du théorème 17.

Montrons que $\mathrm{IRS}_2(G_{M_\delta}) \leqslant 2\delta + 4$ pour tout $\delta \geqslant 0$. Pour cela, il suffit de définir un 1-IRS_1 de plus court chemin dans G_{M_δ}, $(\mathcal{L}, \mathcal{I})$, tel que pour tous les arcs $(u, v) \notin \{(a_j, b_j), (b_j, a_j)\}$, $|\mathcal{I}(u, v)| \leqslant 2\delta + 4$. En effet, par le théorème 15, un tel étiquetage peut être transformé en un IRS_2 avec le même ensemble de chemins tel que les arêtes $\{a_j, b_j\}$ consistent en un seul intervalle. Par conséquent, ça prouvera que $\mathrm{IRS}_2(G_{M_\delta}) \leqslant 2\delta + 4$.

Dans cette preuve, nous n'optimisons pas l'étiquetage des sommets. Choisissons un étiquetage arbitraire $\mathcal{L}$. Vu que nous nous soucions pas du nombre d'intervalles sur les arêtes $\{a_j, b_j\}$, définissons B_{M_δ} comme étant le graphe G_{M_δ} où toutes les arêtes $\{a_j, b_j\}$ ont été supprimées. En premier, remarquons que B_{M_δ} est un graphe bipartite $2^{\delta+1}$-régulier. Clairement, B_M est isomorphe à tout graphe $B_{M'}$, où M' est obtenu en complimentant quelques colonnes de M (ce morphisme échange les règles appliquées par quelques arêtes a_j et b_j), ou en permutant quelques colonnes (ce morphisme permute quelques arêtes $\{a_j, b_j\}$). Ainsi, pour simplifier, considérons B_δ le graphe commun isomorphe à B_{M_δ}, $B_{\chi(M_\delta)}$, etc., et soit $V_1(B_\delta)$ l'ensemble de la première partition de sommets de B_δ, les a_j et b_j, et $V_2(B_\delta)$ comme les sommets v_i de B_δ.

Définissons par induction sur δ un étiquetage $(\mathcal{L}, \mathcal{I})$ sur B_δ. $B_{\delta+1}$ consiste en deux copies de B_δ, B_1 et B_2, avec quelques arêtes en plus connectant $V_1(B_1)$ à $V_2(B_2)$, et quelques arêtes connectant $V_1(B_2)$ à $V_2(B_1)$. Soit ψ le morphisme entre $V(B_1)$ et $V(B_2)$, et soit ϕ_i le morphisme entre $V_i(B_1)$ et $V_i(B_2)$ pour $i = 1, 2$. Pour $\delta = 0$, on vérifie que l'on peut trouver $\mathcal{I}$ tel que $|\mathcal{I}(u, v)| \leqslant 4$ pour tous les arcs (u, v) de B_0 (pour un étiquetage arbitraire $\mathcal{L}$). Soit $k = \max |\mathcal{I}(u, v)|$, sur tous les arcs (u, v) de B_δ. Sachant que l'on ne se soucie pas de $\mathcal{L}$, on considère $\mathcal{I}(u, v)$ comme un sous-ensemble de sommets au lieu d'un sous-ensemble d'étiquettes.

On commence par regarder à chaque sommet $v \in V_2(B_1)$. Par induction, $|\mathcal{I}(v, a)| \leqslant k$ pour tous $(v, a) \in E(B_1)$. On remarque que si $(v, a) \in E(B_1)$, alors $(v, \phi_1(a)) \in E(B_{\delta+1}) \setminus E(B_1)$. Posant $\mathcal{I}(v, \phi_1(a)) = \psi(\mathcal{I}(v, a))$ pour tout $(v, a) \in E(B_1)$, on est capable de router de $v \in V_2(B_1)$ vers tous les sommets de $V(B_{\delta+1}) \setminus \{\phi_2(v)\}$. On rajoute $\phi_2(v)$ à tout arc incident à v conduisant à $\phi_2(v)$ par un plus court chemin. On peut vérifier que tous les chemins sont encore les plus courts, et vue que les arêtes (v, a) et $(v, \phi_1(a))$ sont distinctes, $|\mathcal{I}(v, a)| \leqslant k + 1$ pour tous $(v, a) \in E(B_{\delta+1})$.

Pour chaque sommet $a \in V_1(B_1)$. Avec un argument similaire , on peut mettre $\mathcal{I}(a, \phi_2(v)) = \psi(\mathcal{I}(a, v))$ pour tous $(v, a) \in E(B_1)$. On est capable de router de $a \in V_1(B_1)$ à $V(B_{\delta+1}) \setminus \{\phi_1(v), \phi_1(\overline{a})\}$, où $\overline{a}$ est l'unique sommet de $V_1(B_1)$ tel que $\{a, \overline{a}\}$ est une arête de $E(G_{M_\delta})$. On ajoute $\phi_1(a)$ et $\phi_1(\overline{a})$ à chaque arc incident à a permettant un plus court chemin de a. Ainsi, $|\mathcal{I}(a, v)| \leqslant k + 2$ pou tous$(a, v) \in E(B_{\delta+1})$.

Le routage de tout $v \in V_2(B_2)$ et de tout $V_1(B_2)$ est définie de manière similaire vue que le graphe $B_{\delta+1}$ est le même si B_1 et B_2 sont échangés. En total, pour tout arc $(u, v) \in E(B_{\delta+1})$, nous avons $|\mathcal{I}(u, v)| \leqslant k + 2$, qui est au plus $2\delta + 4$ pour B_δ.

On finit la preuve avec le théorème 15 appliqué aux arêtes $\{a_j, b_j\}$. □

9.2.2 Une borne inférieure pour la α-compacité

Dans cette section, nous montrons que la α-compacité d'un graphe quelconque de taille n n'est pas bornée pour $\alpha > 1$. Notons que pour $\alpha = 1$, une borne fine existe.
Dans [30], Gavoille et Peleg ont montré que pour tout graphe G, $\mathrm{IRS}(G) \leqslant n/4 + o(n)$, et qu'il existe un graphe extrême G_0 avec $\mathrm{IRS}(G_0) \geqslant n/4 - o(n)$.

On commence par présenter une borne supérieure générale :

Théorème 18 *Pour tout graphe G de taille n et tout $\alpha \leqslant \delta(G)$,*

$$\mathrm{IRS}_\alpha(G) \leqslant \frac{1}{\alpha}(n - \delta(G)) + 1 \ .$$

Preuve. Il suffit de considérer un k-IRS_1 de plus court chemin quelconque sur G (prenons $k \leqslant n/4 + o(n)$), et d'appliquer le théorème 16 tout en remarquant que tous les arcs ont des étiquettes non vides. □

Nous allons montrer qu'il existe des graphes extrêmes avec une α-compacité au moins $n/\alpha^{O(1)}$. On obtient donc une borne inférieure asymptotique optimale pour la compacité du IRS_α de plus court chemin avec α constant.
Notons que il est assez compliqué de construire "à la main" un graphe G de petite taille avec $\mathrm{IRS}_2(G) > 1$. En effet, on a besoin de montrer pour un tel graphe G, que pour tout étiquetage des sommets, pour tous les choix possibles de plus courts chemins, et principalement, pour tous les chemins de déflection, on ne peut pas coder la table de routage en utilisant un seul intervalle. Le premier contre exemple avec $\mathrm{IRS}_2(G) > 1$ que l'on est capable de construire est un graphe à 10^5 sommets.

Pour la suite, nous avons besoin du résultat important suivant :

Lemme 9 (Gavoille-Perennes [31]) *Pour tout ε tel que $0 < \varepsilon < 1$, pour tout entier n assez grand, et pour tout entier d tel que $3 \leqslant d \leqslant \varepsilon n$, il existe un graphe G_0 de taille n de degré maximum d tel que toute table de routage de plus court chemin sur G_0 a une taille $\Omega(n \log d)$ bits/sommet, nécessaire simultanément en $\Theta(n)$ sommets.*

En réalité, on peut adapter leur construction pour transformer G_0 en un nouveau graphe H_0 de telle sorte que H_0 soit d-régulier, ait $n + O(\varepsilon n)$ sommets, et tel que le lemme 9 reste valide pour H_0.

Théorème 19 *Il existe une constante $c \geqslant 1$ telle que pour tout entier $\alpha \geqslant 2$, et pour tout entier n assez grand, il existe un graphe $\lceil \alpha^c \rceil$-régulier H_0 avec au plus n sommets tel que*

$$\mathrm{IRS}_\alpha(H_0) \;\geqslant\; \frac{1}{4c\alpha^c} n \ .$$

Preuve. Soit H_0 le graphe extrême de degré maximum $d = \lceil \alpha^c \rceil$, c étant une constante > 0 adéquate, pour laquelle toute table de routage de plus court chemin nécessite $c_1 n \log d$ bits à chaque sommet, (c_1 constante > 0 dépendant du lemme 9). Soit x un tel sommet. On suppose que H_0 a été déjà transformé en un graphe d-régulier avec au plus n sommets (réduisant éventuellement d'un facteur constant la valeur de c_1). Ainsi, $\deg(x) = d$.

Considérons dans H_0 un k-IRS$_\alpha$ de plus court chemin quelconque, $(\mathcal{L}, \mathcal{I})$. Cet IRS α-adaptatif est une implémentation dans x d'une table de routage α-adaptative avec plus court chemin. Cette implémentation peut être faite en x avec au plus $d \left\lceil \log \binom{n}{2k} \right\rceil$ bits. En effet, pour les $d = \deg(x)$ ports de x il suffit de stocker au plus k intervalles d'étiquettes. Il y a au plus $\binom{n}{2k}$ façons de choisir k sous-intervalles de $[1, n]$. Donc, un total de $d \left\lceil \log \binom{n}{2k} \right\rceil$ bits pour x.

À présent, il est facile de transformer toute table de routage α-adaptative de plus court chemin en une table de routage 1-adaptative de plus court chemin, c'est-à-dire, une table de routage standard, en rajoutant $n \lceil \log \alpha \rceil$ extra bits/sommet : pour chaque étiquette de destination, on spécifie le port à utiliser pour avoir un plus court chemin, et il y a exactement α ports possibles ; voir la figure 9.3.

Ainsi, H_0 a une table de routage avec plus court chemin de taille au plus

$$d \left\lceil \log \binom{n}{2k} \right\rceil + n \lceil \log \alpha \rceil$$

utilisant une implémentation de le version déterministe de $(\mathcal{L}, \mathcal{I})$. De la borne inférieure universelle, on obtient :

$$d \left\lceil \log \binom{n}{2k} \right\rceil + n \lceil \log \alpha \rceil \;\geqslant\; c_1 n \log d \ . \tag{9.2}$$

Sachant que $d = \lceil \alpha^c \rceil < \alpha^c + 1$, et puisque $\binom{n}{2k} \leqslant \left(\frac{n}{2k}\right)^{2k} < 2^n$, l'équation 9.2 devient

$$\begin{aligned} (\alpha^c + 1) \left(\log \binom{n}{2k} + 1 \right) + n (\log \alpha + 1) \;&\geqslant\; c_1 c\, n \log \alpha \\ \Rightarrow \quad \alpha^c\, 2k \, \log \left(\frac{n}{2k} \right) + n \log \alpha + 3n \;&\geqslant\; c_1 c\, n \log \alpha \ . \end{aligned}$$

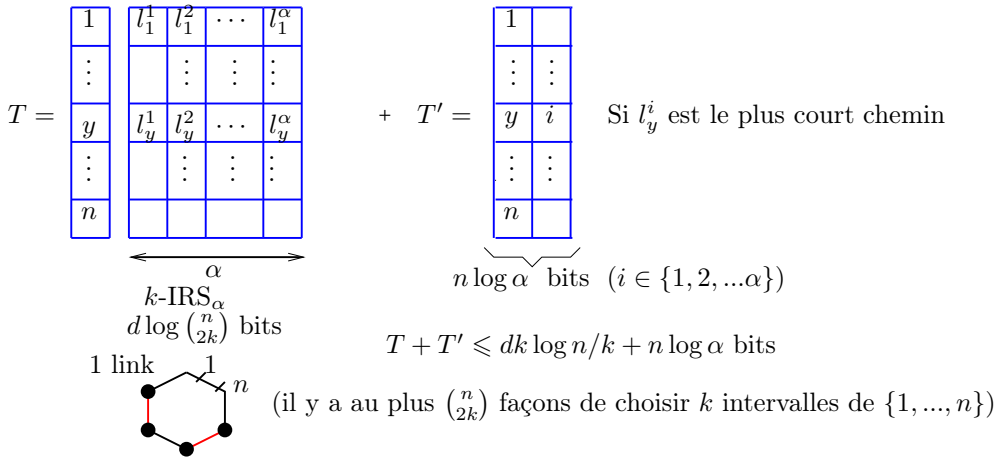


FIGURE 9.3 – Transformation d'une table α-adaptative de plus court chemin en une table de routage 1-adaptative de plus court chemin.

Définissons t tel que $2k = n/t$. On a

$$\alpha^c \cdot \frac{\log t}{t} \geqslant \left(c_1 c - 1 - \frac{3}{\log \alpha} \right) \log \alpha \ . \tag{9.3}$$

Soit $c_2 = c_1 c - 1 - 3/\log \alpha$. Puisque $\alpha \geqslant 2$, $c_2 \geqslant c_1 c - 4$, il suffit alors de choisir $c \geqslant 1$ tel que $c \geqslant 5/c_1$ pour avoir $c_2 \log \alpha \geqslant \log \alpha$. On obtient

$$(9.3) \quad \Rightarrow \quad \alpha^c \geqslant \frac{t \log \alpha}{\log t} \tag{9.4}$$

$$\Rightarrow \quad c \log \alpha \geqslant \log (t \log \alpha) - \log \log t \ . \tag{9.5}$$

Si $t \leqslant 4$, alors $k \geqslant n/8$ et alors le résultat est vérifié pour tout $\alpha \geqslant 2$. En plus, pour tout $t \geqslant 4$, $\log t - \log \log t \geqslant \frac{1}{2} \log t$. Ainsi, $\log(t \log \alpha) \geqslant \frac{1}{2} \log t$, puisque $\log \alpha \geqslant 1$. Ainsi (9.5) devient $c \log \alpha \geqslant \frac{1}{2} \log t$. Et (9.4) devient

$$\alpha^c \geqslant \frac{t \log \alpha}{2c \log \alpha} \quad \Rightarrow \quad t \leqslant 2c\alpha^c \ .$$

Il s'en suit que $k \geqslant n/(4c\alpha^c)$, ce qui finit la preuve. □

9.3 Conclusion et perspectives

Dans cette dernière partie, consacrée au routage compact et adaptatif, nous avons proposé une extension de la technique de routage par intervalles très étudiée dans le cadre déterministe, au cas adaptatif ; la fonction de routage devant alors retourner pour tout couple source-destination, plusieurs chemins de routage.

Le définition que nous avons adoptée, pour une fonction de routage valide, est celle qui consiste à imposer l'existence, pour tout couple de sommets (u, v) d'un chemin pour aller de u à v, les autres chemins étant des chemins de deflection. En effet, cette définition n'impose pas plus de contraintes sur la connectivité du graphe.

Nous avons alors établi le corollaire 12 de la section 9.1 qui montre que tout graphe à degré minimum $\delta(G) \geqslant \alpha$ supporte un routage par intervalles α-adaptatif dont la compacité est $k = 1$.

La même remarque pour le routage de plus court chemin, nous avons étudier le cas où la fonction de codage code au moins un plus court chemin. Nous avons vu que la différence entre la 1- et la α-compacité peut être arbitrairement grande (théorème 17).

Enfin, les deux derniers théorèmes du chapitre 9, établissent une borne inférieure et une borne supérieure pour la α-compacité en général.

Comme perspectives de cette étude, il serait intéressant d'étudier tous ces paramètres pour des classes particulières de graphes telles que les graphes planaires, les hypercubes, les tores, etc.

Une autre perspective serait de changer les définitions des fonctions de routage par intervalles α-adaptatives, en codant cette fois-ci au moins β plus courts chemins. La formulation du problème serait alors : étant donné n, deux entiers positifs α et β, trouver la valeur minimum de la compacité $k = f_{\alpha,\beta}(n)$, c'est-à-dire, trouver le nombre minimum d'intervalles assignés aux arêtes afin de coder des fonctions de routage α-adaptatives codant au moins β plus courts chemins.

Bibliographie

[1] D. Angluin, *Local and global properties in networks of processors*, in Proceedings of the 12*th* Symposium on Theory of Computing, 1980, pp. 82–93.

[2] D. Arques, J. Françon, M. T. Guichet, and P. Guichet, *Comparison of algorithms controlling concurrent access to a database : a combinatorial approach*, in Automata, languages and programming (Rennes, 1986), Springer, Berlin, 1986, pp. 11–20.

[3] H. Attiya and J. Welch, *Distibuted computing, Fundamentals, Simulations and Advenced Topics*, McGraw-Hill, 1998.

[4] H. Austinat and V. Diekert, communication privée.

[5] E. Baker, R. Tan, and J. van Leeuwen, *Prefix routing schemes in dynamic networks*, Tech. Rep. RUU-CS-90-10, Dept. of Computer Science, Utrecht University, 1990.

[6] C. Berge, *Graphes et Hypergraphes*, Bordas, 1973.

[7] R. Bergeron, G. Labelle, and P. Leroux, *Combinatorial Species and Tree-like Structures,*, Cambridge University Press, 1998.

[8] M. Billaud, P. Lafon, Y. Métivier, and E. Sopena, *Graphe rewriting systems with priorities*, in Proceedings of the 5*th* International Workshop on Graph-Theoretic Concepts in Computer Science WG'89, vol. Lecture Notes in Computer Science vol 411, 1989, pp. 94–106.

[9] j. E. Burns and J. Pachl, *Uniform self-stabilizing rings*, ACM Trans. Program. Lang. Syst., 11 (1989), pp. 330–344.

[10] P. Cartier and D. Foata, *Problèmes combinatoires de commutation et réarrangements*, Lecture Notes in Mathematics, 85 (1969).

[11] E. J.-H. Chang and R. Roberts, *An improved algorithm for decentralized extrema finding in circular arrangements of processes.*, Commun. ACM, 22 (1979), pp. 281–283.

[12] A. Comtet and S. Nechaev, *Random operator approach for enumeration in braid groups*, J.Phys.A : Math. Gen., 31 (1998), pp. 5609–5630.

[13] R. Cori and Y. Métivier, *Rational subsets of some partially abelian monoids*, Theoritical Computer Science, 35 (1985), pp. 169–190.

[14] R. Cori and D. Perrin, *Sur la reconaissabilité dans les monoïdes partiallement commutatifs*, RAIRO, Info.Th., 19 (1985).

[15] J. Debois and S. Nechaev, *Statistics of reduced words in locally free and braid groups : Abstract studies and applications to ballistic growth model*, J.Phys.A : Math. Gen., 31 (1998), pp. 2767–2789.

[16] D. Dolev, M. Klawe, and M. Rodeh, *An $o(n \log n)$ undirectional distributed algorithm for extrema-finding in a cicle*, J. Algorithms, 3 (1982), pp. 245–260.

[17] W. Feller, *An Introduction to Probability Theory and Its Application, Volume I*, John Wiley and Sons, 1960.

[18] ——, *An Introduction to Probability Theory and Its Application, Volume II*, John Wiley and Sons, 1966.

[19] P. Flajolet and A. Odlyzko, *Random mapping statistics*, in Advances in Cryptology, vol. Lecture Notes in Computer Science vol 434, Springer, 1990, pp. 329–354.

[20] M. Flammini, G. Gambosi, U. Nanni, and R. B. Tan, *Characterization results of all shortest paths interval routing schemes*, in 5^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO), L. Gargano and D. Peleg, eds., 1998, pp. 201–213.

[21] M. Flammini, G. Gambosi, and S. Salomone, *Boolean routing*, in 7^{th} International Workshop on Distributed Algorithms (WDAG), vol. 725 Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 219–233.

[22] ——, *Interval labeling schemes for chordal rings*, in 1^{st} International Colloquium on Structural Information & Communication Complexity (SIROCCO), P. Flocchini, B. Mans, and N. Santoro, eds., Carleton University Press, May 1994, pp. 111–124.

[23] M. Flé, *Serialization of concurrent programs*, JCSS, 38 (1989), pp. 474–493.

[24] M. Flé and G. Roucairol, *Maximal serializability of iterated transactions*, Theoritical Computer Science, 38 (1985), pp. 1–16.

[25] H. Furstenberg, *Algebraic functions over finite fields*, Journal of Algebra, 7 (1957), pp. 271–277.

[26] H. Garcia-Molina, *Election in distributed computing systems*, IEEE Trans. Comput., C31,1 (1984), pp. 48–59.

[27] S. Gaubert and D. Hong, *Series expansions of lyapunov exponents and forgetful monoids*, Tech. Rep. 3971, INRIA Roquencourt, July 2000.

[28] C. Gavoille, *A survey on interval routing*, Theoretical Computer Science, 245 (1999).

[29] C. Gavoille and E. Guévremont, *Worst case bounds for shortest path interval routing*, Journal of Algorithms, 27 (1998), pp. 1–25.

[30] C. Gavoille and D. Peleg, *The compactness of interval routing*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 459–473.

[31] C. Gavoille and S. Pérennès, *Memory requirement for routing in distributed networks*, in 15^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM PRESS, ed., May 1996, pp. 125–133.

[32] C. Gavoille and A. Zemmari, *The compactness of adaptive routing tables*, in 7^{th} International Colloquium on Structural Information & Communication Complexity (SIROCCO'00), M. Flammini, E. Nardelli, G. Proietti, and P. Spirakis, eds., Carleton Scientific, June 2000, pp. 127–140.

[33] D. Geniet, R. Shott, and L. Thimonier, *A markovian concurrency measure*, RAIRO, Inform. Theor. Appli, 4 (1996), pp. 295–304.

[34] O. Gerstel and S. Zaks, *A new characterization of tree medians with applications to distributed algorithms*, tech. rep., "DAIMI PB-364", Computer science Department, Aarhus University, 1991.

[35] R. Gupta, S. A. Smolka, and S. Bhaskar, *On randomization in sequential and distributed algorithms*, ACM Comput. surv., 26, 1 (1994), pp. 7–86.

[36] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, *Probabilistic Methods for Algorithmic Discrete Mathematics*, Springer, 1998.

[37] D. S. Hirschberg and J. B. Sinclair, *Decentralized extrema-finding in circular configurations of processes*, Commun. ACM, 23 (1980), pp. 627–628.

[38] A. Itai and M. Rodeh, *Symmetry breaking in distributed networks*, Information and Computation, 88(1) (1990), pp. 60–87.

[39] E. Korach, D. Rotem, and N. Santoro, *Distributed algorithms for finding centers and medians in networks*, ACM Trans. on Programming Languages ans Systems 6, No 3 (1984), pp. 381–401.

[40] C. Lavault, *Evaluation des algorithmes distribués : analyse, complexité, méthodes*, Hermes, 1995.

[41] G. LeLann, *Distributed systems : Towards a formal approach*, in Proc. Information Processing'77, B. Gilchrist (ed.), North-Holland, 1977, pp. 155–160.

[42] I. Litovsky, Y. Métivier, and E. Sopena, *Graph relabelling and distributed algorithms*, in Handbook of graph grammars and computing by graph transformation, H. Ehrig, H. Kreowski, U. Montanari, and G. Rosenberg, eds., vol. III, World Scientific, 1999, pp. 1–56.

[43] I. Litovsky, Y. Métivier, and W. Zielonka, *The power and the limitations of local computations on graphs and networks*, in Proceedings of the 8*th* International Workshop on Graph-Theoretic Concepts in Computer Science WG'92, vol. Lecture Notes in Computer Science vol 657, 1993, pp. 333–345.

[44] M. Loève, *Probability Theory*, D. Van Nostrand Company, Inc., 1955.

[45] N. Lynch, *A hundred impossibility proofs for distributed computing*, in proceedings of the eighth annual ACM symposium on Principles of Distributed Computing, 1989, pp. 1–28.

[46] A. Mazurkiewicz, *Concurrent program schemes and their interpretations*, DAIMI, BP78, Aarhus University, (1977).

[47] ——, *Solvability of the asynchronous ranking problem*, Inf. Processing Letters, 28 (1988), pp. 221–224.

[48] A. Mazurkiewicz, *Distributed enumeration*, Inf. Processing Letters, (1997), pp. 233–239.

[49] Y. Métivier, *On recognizable subsets of free partially commutative monoids*, Theoritical Computer Science, 58 (1988), pp. 201–208.

[50] Y. Métivier, N. Saheb, and A. Zemmari, *On the efficiency of randomized local synchronzations*, Research Report RR-1240-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, 2000.

[51] ——, *Randomized rendezvous*, Research Report RR-1228-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, January 2000.

[52] ——, *Randomized rendezvous*, in proc. of the International Colloquium on Mathematics and Computer Science : Algorithms, Trees, Combinatorics and Probabilities, D. Gardy and A. Mokaddem, eds., Birkhauser Trends in Mathematics Series, 2000, pp. 183–194.

[53] Y. Métivier and P.-A. Wacrenier, *A distributed algorithm for computing a spanning tree in anonymous t-prime graphs*, Research Report RR-1229-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, January 2000.

[54] S. Mitchel, *Another characterization pf the centroid of a tree*, Discrete Mathematics, 24 (1978), pp. 277–280.

[55] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[56] Y. Métivier and N. Saheb, *Probabilistic analysis of an election algorithm in a tree*, in Trees in Algebra and Programming-CAPP'94, 1994, pp. 234–245.

[57] Y. Métivier and E. Sopena, *Graph relabelling syatems : A general overview*, Computers and Artificial Intelligence, No. 2 (1997), pp. 167–185.

[58] Y. Métivier and G. Tel, *Termination detection and universal graph reconstruction*, in Proceedings of the 7*th* International Colloquium on Structural Information and Communication Complexity (SIROCC 2000), 2000, pp. 237–251.

[59] D. Perrin, *Words over partially commutative alphabet, in combinatorial algorithms on words*, NATO ASI Series, F12 (1986), pp. 329–340.

[60] G. L. Peterson, $o(n \log n)$ *undirected algorithm for the circular extrema problem*, ACM Trans. Program. Lang. Syst., 4 (1982), pp. 758–762.

[61] M. Raynal, *Algorithmes Distribués et Protocoles*, Eyrolles, 1985.

[62] M. Robson, communication privée.

[63] K. H. Rosen, *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, 2000.

[64] P. Ružička and D. Štefankovič, *On the complexity of multi-dimensional interval routing schemes*, Theoretical Computer Science, 245 (1999).

[65] N. Saheb, *Concurrency measure in commutation monoids*, Discrete Applied Mathematics, 24 (1989), pp. 223–236.

[66] N. Saheb and A. Zemmari, *Methods for computing the concurrency degree*, Research Report RR-1210-98, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, June 1998.

[67] N. Saheb and A. Zemmari, *Methods for computing the concurrency degree of commutation monoids*, in Proc. of the 12th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'00), D. Krob, A. Mikhalev, and A. Mikhalev, eds., Springer-Verlag, 2000, pp. 731–742.

[68] N. Santoro and R. Khatib, *Labelling and implicit routing in networks*, The Computer Journal, 28 (1985), pp. 5–8.

[69] R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*, Addison Wesley, 1996.

[70] G. Siu and A. Zemmari, *Algorithmes distribués, problème du rendez-vous*, Rapport de stage, ENS de Lyon, France, 1999.

[71] R. P. Stanley, *Enumerative combinatorics*, Wads worth and Brooks cole, 1986.

[72] B. Szymanski, Y. Shy, and N. Prywes, *Terminating iterative solutions of simultaneous equations in distributed message passing systems*, in Proceedings of the 4th Symposium on Principles of Distributed Computing, 1985, pp. 287–292.

[73] G. Tel, *Introduction to Distributed Algorithms*, Cambridge University Press, 2000.

[74] J. van Leeuwen and R. B. Tan, *Computer networks with compact routing tables*, in The Book of L, G. Rozemberg and A. Salomaa, eds., Springer-Verlag, 1986, pp. 259–273.

[75] ——, *Interval routing*, The Computer Journal, 30 (1987), pp. 298–307.

[76] G. Viennot, *Heaps of pieces : Basic definitions and combinatorial lemmas*, in Colloque de Combinatoire Enumérative, UQAM, Montreal, ed., 1985.

[77] B. Zelinka, *Medians and peripherians of trees*, Arch. Math., Brno, (1968), pp. 87–95.

Contribution à l'analyse d'algorithmes distribués

Résumé : La première partie de cette thèse est consacrée à l'étude du degré de parallélisme des monoïdes de commutation modélisant les exécutions distribuées des algorithmes. Après une présentation du modèle et des différents résultats déjà établis, nous donnons des méthodes pour calculer ce degré, l'outil principal utilisé étant les marches aléatoires et les chaînes de Markov. La deuxième partie s'intéresse au problème des synchronisations dans les réseaux anonymes. Des travaux ultérieurs ont montré que sous quelques hypothèses, on ne peut résoudre ce problème de manière déterministe, nous proposons donc et analysons des algorithmes probabilistes résolvant ce problème, nous étudions également leur efficacité. La troisième partie est consacrée à l'étude d'un algorithme d'élection dans un réseau en arbre ou dans tout réseau où un arbre couvrant est disponible. Nous montrons que sous quelques hypothèses, le(s) sommet(s) médian(a) a (ont) la probabilité la plus élevée d'être élu(s), et nous donnons quelques implémentations possibles de cet algorithme. Dans la dernière partie, nous nous intéressons à l'étude de la taille mémoire nécessaire pour coder les tables de routage adaptatives dans un réseau de processeurs. Les principaux résultats de cette partie concernent la compacité de ces tables. En effet, nous montrons que tout réseau supporte un routage par intervalle α-adaptatif de compacité 1. Si on impose au moins un plus court chemin, nous donnons une borne inférieure pour la compacité et, enfin, nous montrons que la différence entre la compacité dans le cas déterministe et la compacité dans le cas adaptatif peut être très grande.

Mots-clés : algorithmes distribués, monoïdes de commutations, réseaux anonymes, synchronisations, algorithmes probabilistes, élection, routage par intervalles.

Contribution to the analysis of distributed algorithms

Abstract : The first part of this thesis is devoted to the study of the concurrency degree of commutation monoids, which model the distributed execution of algorithms. We start by the presentation of the model and recall some known results, and we give methods to compute this quantity. The main tools are random walks and Markov chains. In the second part, we study the synchronzations problem in anonymous networks. By previous works, we know that, under the assumption of some hypotheses, we can not solve the problem using determinstic algorithms, thus we have no choice but to use probabilistic ones. We propose and analyse probabilistic algorithms which allow us to solve this problem, and we study their efficiency. In the third part, we present and analyse a probabilistic algorithm for election in tree networks or networks where a spanning tree is available. We show that under the assumption of some hypotheses, the median(s) vertex (vertices) have the highest probabilty to be elected, and we give some possible implementations for this algorithm. In the last part, we are interested on the space complexity necessary to code adaptive routing tables in a network of processors. The main results of this part concern the compactness of such tables, indeed, we show that any network supports an α-adaptive interval routing scheme of compactness 1. If we add the constraint that at least one shortest path is coded, we give a lower bound of this compactness, and we show that the difference between the compactness in the deterministic case and adaptive case can be too large.

Keywords : distributed algorithms, commutation monoids, anonymous networks, synchronizations, probabilistic algorithms, election, interval routing.

Thèse en informatique, préparée au LaBRI (Université Bordeaux I, 351 cours de la Libération, 33405 Talence).