Analyse de quelques algorithmes probabilistes à délais aléatoires

Résumé :

Dans la première partie de cette étude, nous proposons et analysons des algorithmes probabilistes d'élection uniforme dans des graphes de types arbres, les $k-$arbres et les polyominoïdes. Ces algorithmes utilisent des durées de vie aléatoires associées aux sommets découverts (sommets feuilles ou simpliciaux). Ces durées sont des variables aléatoires indépendantes et sont localement engendrées au fur et à mesure que les sommets sont découverts.

Dans la seconde partie, nous analysons un algorithme probabiliste de synchronisation pour le problème de rendez-vous avec agendas dynamiques. L'objectif est de trouver un couplage maximal dans un graphe donné. Ensuite, nous proposons et étudions un modèle de diffusion à délai aléatoire pour la transmission d'un message dans un réseau.

Finalement, dans la dernière partie, nous exposons les outils utilisés pour implémenter la simulation des algorithmes distribués.

Discipline : Informatique

Mots-Clés : Algorithmes distribués, algorithmes probabilistes, élection, $k-$arbres, polyominoïdes, couplage, rendez-vous, diffusion, simulation.

Analysis of some probabilistic algorithms with random delays

Abstract :

In the first part of this study, we propose and analyze a probabilistic algorithms of uniform election in graphs of structures of the trees type, $k-$trees and polyominoids. These algorithms use random delay associated to discovered vertices (leaf vertices or simplicial vertices). These delays are independent random variables and are locally generated as and when the vertices are discovered.

In the second part, we analyze a probabilistic algorithm of synchronization for the problem of rendezvous with dynamic agendas. The goal is to find a maximal matching in a given graph. Then, we propose and study a model of diffusion with random delay for the transmission of a message in a network.

Finally, in the last part, we expose the tools used to implement the simulation of the distributed algorithms.

Discipline : Computer-Science

Keywords : Distributed algorithms, probabilistic algorithms, election, $k-$trees, polyominoids, matching, rendezvous, diffusion, simulation.

LaBRI,
Université Bordeaux 1,
351 cours de la Libération,
33405 Talence Cedex (FRANCE).

N° d'ordre : 3323

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Abdelaaziz EL HIBAOUI**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

ANALYSE DE QUELQUES ALGORITHMES PROBABILISTES À DÉLAIS ALÉATOIRES

Soutenue le : Lundi 11 décembre 2006

Sous la direction :

Nasser Saheb-Djahromi Maître de conférences HDR

Après avis des rapporteurs :

Danièle Gardy ..... Professeur

Christian Lavault ..... Professeur

Devant la commission d'examen composée de :

François Dufour	Professeur	Président
Danièle Gardy	Professeur	Rapporteur
Christian Lavault	Professeur	Rapporteur
Yves Métivier	Professeur	Examinateur
Mike Robson	Professeur	Examinateur
Nasser Saheb	Maître de conférences HDR	Directeur de thèse

– 2006 –

Analyse des algorithmes distribués probabilistes

Résumé : Depuis longtemps, la recherche en algorithmique a été développée en étroite liaison avec la combinatoire. Les deux disciplines se sont mutuellement enrichies : beaucoup d'algorithmes utilisent des constructions combinatoires classiques et, réciproquement, la réalisation d'algorithmes complexes et les problèmes d'analyse associés ont donné lieu à des problèmes combinatoires nouveaux. Par ailleurs, les algorithmes probabilistes sont conçus pour résoudre les problèmes qui n'admettent pas de solutions déterministes efficaces, ou pour améliorer sensiblement la complexité en moyenne et/ou au pire d'algorithmes polynomiaux classiques.

Les travaux de cette thèse s'intègrent dans cette continuité et présentent l'évolution et l'analyse des performances des algorithmes probabilistes entièrement distribués et décentralisés et ce dans des systèmes distribués anonymes, asynchrones et de topologies différentes.

En effet, dans la seconde partie de cette thèse, nous proposons et étudions des algorithmes d'élection probabiliste uniforme dans des structures de type arbres, $k-$arbres, et *polyominoïdes*. Ces algorithmes sont basés sur l'utilisation d'un délai aléatoire associé à tout sommet supprimable (les sommets feuilles ou les sommets simpliciaux). Ces délais sont indépendants et peuvent être localement générés par des sommets au fur et à mesure qu'ils deviennent supprimables. Pour chacune des classes de graphes citées ci-dessus, notre résultat principal est l'introduction d'un algorithme d'élection totalement équitable, c'est-à-dire que quelque soit l'endroit où un sommet est placé dans un graphe, il a la même probabilité d'être élu que tous les autres sommets. Nous avons trouvé une borne supérieure H_{n-1} pour l'espérance mathématique de la durée d'élection dans les graphes à n sommets. Cette borne pourrait représenter quelques intérêts théoriques (elle peut, par exemple, exprimer une mesure de la complexité en temps).

Nous analysons également dans la troisième partie, deux algorithmes probabilistes : un algorithme de rendez-vous avec agendas dynamiques et un autre de diffusion avec un délai aléatoire de transmission.

Le but du premier algorithme est de trouver un couplage dans un graphe d'une manière distribuée. Cet algorithme est basé aussi sur des délais aléatoires. Nous allons d'abord montrer que dans un tour de son exécution, il donne *en espérance* un nombre de rendez-vous supérieur à celui fourni par l'algorithme "$MSZ-$algorithme" étudié dans [MSZ03a]. Puis nous analysons le nombre de rendez-vous pour quelques classes de graphes (graphes complets, chaînes, étoiles, double-étoiles). Nous allons voir ensuite que $\overline{R}(G)$, l'espérance du nombre de rendez-vous de l'algorithme appliqué à un graphe G, n'est pas monotone lors d'une insertion d'une arête ou d'une extension (ajout d'un sommet avec certaines arêtes). De plus, nous calculons l'espérance asymptotique du nombre de rendez-vous dans des graphes aléatoires d'Erdös et Rényi de degré moyen $c > 0$ ($G(n,p)$ et $G(n,m)$), avec $m = cn/2$. D'autre part, nous définissons l'efficacité, pour un graphe G donné, comme le rapport $\overline{R}(G)/K(G)$, où $\overline{R}(G)$ est l'espérance de nombre de rendez-vous et $K(G)$ est le cardinal maximal de couplage dans G. Nous allons montrer que lorsque notre algorithme est appliqué aux graphes généraux, la borne inférieure de l'efficacité est serrée , elle vaut au moins $1/2$. A la fin de cette étude, nous allons prouver que dans une famille de graphe appelée *mille-pattes*, l'efficacité est approximativement 0.7690397520.

En ce qui concerne le second algorithme, le modèle sur lequel il est basé est un modèle de diffusion avec un délai aléatoire de transmission. En effet, nous considérons des variations où les arêtes peuvent avoir différents taux de transmission. Le délai aléatoire de transmission considéré est le temps nécessaire pour transiter un message entre deux noeuds adjacents dans le réseau. Nous supposons que ce temps

est une variable aléatoire qui suit une loi exponentielle. Notre problème consiste à trouver une borne supérieure de l'espérance du temps nécessaire pour accomplir la diffusion sous l'hypothèse que toutes les arêtes ont un délai aléatoire de transmission. Cette étude est en cours de développement et nous cherchons des résultats généraux en lien avec des applications aux problèmes de diffusion dans des réseaux de différentes topologies.

Finalement, dans la dernière partie, nous exposons les bases fondamentales que nous avons utilisées pour implémenter un outil de simulation des algorithmes distribués. Cet outil, nous a permis de faire des expérimentations sur tous les algorithmes présentés dans cette thèse et les résultats obtenus corroborent nos résultats théoriques.

Mots-clés : Algorithmes distribués, algorithmes probabilistes, élection, $k-$arbres, polyominoïdes, couplage, rendez-vous, diffusion, simulation.

Conférences internationales avec comité de sélection et publication des actes

[1] A. EL HIBAOUI and N. SAHEB "Broadcast with random delay transmission", GASCOM, Septembre 2006.

[2] A. EL HIBAOUI, N. SAHEB and A. ZEMMARI, "A Uniform Probabilistic Election Algorithm in k-Trees", IMACS : 17th IMACS World Congress : Scientific Computation, Applied Mathematics and Simulation, Juillet 2005.

[3] A. EL HIBAOUI, N. SAHEB and A. ZEMMARI, "Polyominoids and Uniform Election", FPSAC : 17th International Conference on Formal Power Series and Algebraic Combinatorics, Juin 2005.

Rapports internes

[4] A. El Hibaoui, Y. Métivier, M. Robson, N. Saheb-Djahromi and A. Zemmari, "Analysis of a Randomized Dynamic Timetable Handshake Algorithm", *Technical Report N. RR-1402-06*, LaBRI - Université de Bordeaux 1, Mai 2006.

[5] A. EL HIBAOUI, N. SAHEB and A. ZEMMARI, "A Uniform Probabilistic Election Algorithm in k-Trees", *Technical Report N. RR-1327-04*, LaBRI - Université de Bordeaux 1, Juin 2004.

[6] A. EL HIBAOUI, N. SAHEB and A. ZEMMARI, "Polyominoids and Uniform Election", *Technical Report N. RR-1337-04*, LaBRI - Université de Bordeaux 1, Novembre 2004.

LaBRI,
Université Bordeaux 1,
351 cours de la Libération,
33405 Talence Cedex (FRANCE).

À mes parents pour leur amour et leur soutien
je leur dois beaucoup

Remerciements

Je n'ai pas vu le temps passer durant ces quatre années de thèse ! Je tiens à remercier ici tous ceux qui m'ont aidé, soutenu et encouragé pendant ma thèse.

Mes premiers remerciements vont bien entendu à mon directeur Nasser Saheb, pour m'avoir encadré durant ces quatre années de thèse, avec toute la disponibilité et la patience dont il a su faire preuve à mon égard et également pour les discussions scientifiques que nous avons eu tout au long de ma thèse. Et évidement, je n'oublie pas un merci tout particulier à Akka Zemmari.

Je suis également reconnaissante envers tous les membres de mon jury, sans qui ce document ne serait qu'un morceau de papier sans grande valeur.

J'ai profité pendant ces quatre années de l'environnement autant sympathique que scientifique du LaBRI, qui m'a permis de réaliser cette thèse dans les meilleures conditions, et cela grâce à tous ses membres, thésards et autres, qui font du LaBRI un lieu où il fait bon travailler. Mes remerciements les plus sincères à Sofian Maabout, Mohamed Mosbah, et Kaninda Musumbu de leur soutient et de leur encouragement. Egalement, je tiens à remercier pas mal d'amis pour l'ambiance détendue et sympathique de ces années. Merci donc à collègue de bureau Houda Anoun d'avoir relue quasiment toute ma thèse alors qu'elle était entrain de préparer la sienne, un deuxième merci pour elle aussi pour son rôle de psy :), merci à Mustapha Diouf d'avoir eu le courage de partager avec nous le bureau et pour toutes les discussions très enrichissantes qui allaient avec, à Ismail pour ses conseils, ses corrections, et pour plein d'autres choses encore dont je continue à me rappeler toujours, et à Mohamed Bouklit pour son écoute et son ouverture. Concernant mes collègues du LaBRI, en dehors de ce que je viens de citer, je remercie en premier lieu Affif Sellami pour sa gentillesse et son énergie. Merci à Olivier Bernardi et à Eric Fusy, je n'oublierai jamais les jours suaves que nous avons passé ensemble à Marseille, Dijon et en Sicile. Je remercie également les autres collègues de ma promotion, Bilel Derbel, Badr Benmammar, Rui Chen, Pascal Ochem, Selvain Pelat-Alloin, Daniel Goncalves, Mickael Montassier, Lionel Carminati, ... et ceux des autres promotions, Brahim Hamid, Nader Mbarek, Jérémie Chalopin, Pierre Hana, Rodrigue Ossamy. Je souhaite à tous mes collègues, thésards et docteurs, une bonne continuation et tous mes voeux de réussite.

Sur le plan plus personnel, je tiens tellement à remercier mes parents pour m'avoir permis d'en arriver là, pour leur soutien et leur encouragement. Une pensée toute particulière à ma chère et adorable épouse Amal pour sa présence à mes côtés (souvent à quelques Milliers de kilomètres près mais toujours dans un petit coin de mon écran d'ordinateur).

Et pour finir un merci très particulier au personnel administratif du LaBRI, un grand merci à Philipe biais, Ida Nerestan, et Cathy Roubineau qui m'ont beaucoup touché à coeur par leurs comportements et par la qualité de leurs services.

Table des matières

Table des matières

Table des figures

Liste des Algorithmes

Chapitre 1

Introduction

Les instructions d'un programme peuvent s'effectuer de manière séquentielle, parallèle ou distribuée. En effet, une machine ou *architecture séquentielle* permet le traitement d'une seule opération à la fois. Ainsi, pour augmenter la capacité de calcul, on peut regrouper un ensemble de machines pour permettre le traitement de plusieurs opérations en parallèle. L'interconnexion de machines est appelée *réseau Informatique*. Un réseau est caractérisé par sa taille, sa topologie et son accès.

Le terme *système distribué* est de plus en plus souvent utilisé en informatique, et prend selon les auteurs des significations qui diffèrent de manière conséquente. Ce qui gère toutes les ressources connectées au réseau de manière transparente est un système distribué au sens de [Mul89]. Tanenbaum [TvR85] le définit comme étant un ensemble d'ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent. Lamport le définit comme un système qui nous empêche de travailler quand une machine dont nous n'avons jamais entendu parler tombe en panne.

Les progrès technologiques des ordinateurs et le haut débit permettent à différentes applications sur des ordinateurs distincts (et distants) de coopérer pour effectuer des tâches coordonnées. La possibilité de connecter un ensemble de machines en réseau local introduit la notion de partage de ressources : les utilisateurs découvrent alors les avantages de partager une imprimante, des disques, etc... Cette notion entraîne alors d'autres besoins. Les systèmes distribués recouvrent actuellement un champ très vaste. Citons, par exemple, les clusters de machines, les grilles de calculs parallèles ou distribués, les systèmes à objets, le déploiement d'applications Web, le stockage d'informations dans des réseaux de pairs...

La problématique qui reste présente est *comment concevoir des algorithmes afin de réussir à faire collaborer toutes ces entités pour accomplir une tâche globale ?* Par exemple : comment gérer les ressources de manière globale sans pour autant utiliser des algorithmes centralisés qui posent problème en cas de tolérance aux pannes ? Comment gérer la cohérence et la synchronisation entre les machines du réseau : les informations doivent rester cohérentes sur tout le système, ce qui suppose, en cas de modification d'un état local, de diffuser la modification à toutes les autres entités. Ce problème se pose notamment pour la gestion du temps global.

Un *algorithme distribué* $\mathcal{A}$ est un algorithme qui s'exécute sur plus d'une machine ou processeur. En d'autres termes, un algorithme distribué $\mathcal{A}$ sur un système distribué $\mathcal{S}$ n'est autre que la caractérisation des transitions locales à réaliser séquentiellement par chaque processus de $\mathcal{S}$ à la réception d'un message.

Le choix du meilleur algorithme pour une tâche particulière peut être un processus compliqué, qui exige souvent des analyses mathématiques sophistiquées. La partie de l'informatique qui comprend l'étude de telles questions est appelée *analyse des algorithmes*.

Une étude théorique aide à bien comprendre la structure du problème pour en dégager les propriétés qui seront la base de l'algorithme. Ce n'est pas toujours suffisant pour obtenir une bonne efficacité (complexité théorique). Une étude algorithmique peut s'avérer nécessaire. L'existence et l'expression de ces algorithmes dépendent des hypothèses techniques sur les réseaux comme par exemple le mode de communication (par message ou par registre), la synchronisation partielle de processus voisins, l'anonymat des processeurs. Ce cadre rend difficile la lisibilité de certains algorithmes, il ne permet pas toujours de comprendre leur puissance et leurs limites et de faire des preuves mathématiques de leurs propriétés.

Concernant un panorama plus général sur l'algorithmique distribuée, nous pourrons nous reporter à [Lyn96, Tel91, Tel94, Tel00] et en particulier à [Lav95] pour une revue assez détaillée des différentes hypothèses et différents modèles que l'on peut utiliser pour décrire un algorithme distribué. Les ouvrages cités ci-dessus donnent une bonne vue d'ensemble sur l'étude des algorithmes distribués.

Dans cette thèse, la notion de système distribué englobe tous les systèmes composés de plusieurs entités autonomes (processus, ordinateurs, etc.) communiquant chacune avec ses voisins au moyen de canaux de communication ou de variables partagées. Chaque entité évolue alors en fonction de son état propre et de celui de ses voisins.

Nous abordons, dans ce document, certains aspects des thèmes suivants en liaison avec les algorithmes distribués :

Algorithmes probabilistes d'élection uniforme.

Un des paradigmes de l'informatique distribuée est l'*élection* où un processus unique du systèmes distribué $\mathcal{S}$ est privilégié grâce à un choix distribué de tous les processus coopérants de $\mathcal{S}$. L'élection dans un réseau consiste à atteindre une configuration dans laquelle un seul et unique processus est dans un état prédéterminé, *élu*, et tous les autres dans un état prédéterminé différent du précédent, *battu*.

Le problème de l'élection est à la base des principaux mécanismes de contrôle utilisés dans les systèmes distribués. Ainsi, on peut se poser, par exemple, les questions suivantes : comment choisir une machine pour gérer les conflits nécessairement rencontrés lorsque des utilisateurs partagent une ressource (l'exclusion mutuelle). Un autre problème qui peut se poser est lorsque cette machine tombe en panne : comment désigner sans l'intervention des utilisateurs une machine qui

prendra le relais de la machine défectueuse ? Plus précisément, comment rendre les machines autonomes ? Il devient par exemple difficile de choisir un machine s'elles ont les mêmes propriétés et qu'elles sont indiscernables.

De nombreux algorithmes pour élire un sommet dans un graphe sont connus depuis les années soixante-dix [Lan77]. Il est, par ailleurs, connu que certaines classes de graphes n'admettent pas d'algorithme d'élection distribuée [Ang80]. Dans certaines classes, seuls les algorithmes probabilistes permettent de contourner cette difficulté en donnant une solution partielle très acceptable d'un point de vue pratique [DFJ+98, MR95, GSB94]. D'autre part, certains algorithmes distribués permettent d'obtenir un objet appartenant à un ensemble donné (par exemple, étant donné un graphe G, on peut sous certaines hypothèses, calculer un arbre recouvrant T). Il s'agit alors de déterminer la probabilité d'obtenir un objet particulier. Ainsi, la répartition de probabilité d'être élu pour les sommets d'un graphe dépend de l'implémentation *distribuée* de l'algorithme.

L'utilisation de l'aspect aléatoire en informatique distribuée est une technique puissante qui a été fructueuse dans divers domaines. En effet, Il existe plusieurs situations où l'utilisation d'algorithmes probabilistes peut aider à trouver une solution. Par exemple dans les problèmes suivants : l'élection d'un chef dans les réseaux fortement symétriques [IR90, BSV+96, YK99], l'exclusion mutuelle [Dij74, Rab82], ou l'orientation d'anneau [IJ93], il est connu qu'aucun algorithme déterministe ne peut fournir une solution, alors qu'il y a des algorithmes probabilistes simples et élégants.

Dans d'autres situations, comme le routage [Lei92, Val82], les algorithmes probabilistes fournissent une meilleure performance que les déterministes.

Le concept d'équité ou de l'équitabilité (en anglais *fairness*) est une propriété importante, voir [DIM95]. Nous pouvons par exemple utiliser cette propriété pour garantir que toutes les machines, qui collaborent pour faire un calcul complexe, ont la même quantité de données à traiter. Ici, nous nous intéresserons essentiellement à l'élection uniforme sur un réseau. Cependant, nous employons la propriété d'équité dans son sens strict : tous les processeurs ont la même probabilité d'être élus, de sorte que lorsque l'algorithme est réitéré plusieurs fois, les ressources soient allouées "équitablement".

Le concept *d'élection uniforme* que nous abordons dans la première partie de cette thèse fait l'objet du travail de Métivier, Saheb et Zemmari dans [MSZ03b]. Dans ce travail, les auteurs présentent un modèle d'élection uniforme comme une instance d'un modèle général probabiliste basé sur l'algorithme d'Angluin [Ang80]. Ce concept consiste à attribuer la même chance d'être élu à tout "sommet", quelque soit sa position dans le "graphe". Par ailleurs, nous proposons et nous étudions des algorithmes distribués probabilistes *d'élection uniforme* sur différentes topologies de réseaux (arbres, $k-$arbres et polyominoïdes).

Ces algorithmes probabilistes sont fondés sur l'élimination successive des sommets actifs du graphe d'entrée jusqu'à ce que ce graphe soit réduit à un seul sommet. Ce dernier sera considéré comme le sommet élu.

L'étude du modèle uniforme met en jeu quelques outils élémentaires ou avancés de la théorie des probabilités tels que des techniques d'évaluation de la somme et du maximum pour des variables aléatoires indépendantes. Le postulat central du temps d'attente exponentiellement distribués séparant des événements locaux, liés au processus de Poisson, laisse croire que c'est une hypothèse raisonnable dans un tel contexte.

Les processus d'élection proposés dans cette thèse peuvent être modélisés par des processus de Markov irréversibles en temps continu. Le comportement d'un tel processus est généralement réglé par un système d'équations différentielles admettant une condition initiale. L'analyse du modèle général peut être théoriquement réalisée en terme de solution du système. En réalité, la probabilité d'être élu n'est que la probabilité d'absorption. Malheureusement, tous les systèmes présentés n'admettent pas toujours une solution explicite, excepté dans des cas très simples.

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.

Nous sommes concernés par l'établissement de communications par des signaux de synchronisation ou d'une manière équivalente par un ordonnancement distribué local qui trouve des couplages entre paires de processus afin d'établir des communications. Cette implémentation sera réalisée par l'établissement des rendez-vous entre les processus. Ainsi, nous allons proposer et analyser un algorithme de *rendez-vous avec agendas dynamiques*. Nous allons également mesurer son efficacité et faire une étude comparative avec celui proposé dans [MSZ03a].

Cette thèse est organisée comme suit :

Partie I. *Introduction aux probabilités et aux systèmes distribués.*

Dans cette partie nous introduisons les notions nécessaires dans la suite de cette thèse. Le chapitre 2 donne une introduction à la théorie des graphes et des probabilités. Ensuite le chapitre 3 introduit les modèles des systèmes distribués et présente une partie de la théorie qui y est rattachée.

Partie II. *Algorithmes probabilistes d'élection uniforme.*

Cette partie contient toutes mes contributions effectuées lors de cette thèse concernant les algorithmes distribués probabilistes d'élection. Les résultats qui y sont présentés ont été obtenus en collaboration avec Nasser Saheb et Akka Zemmari.

Chapitre 4 : Ce chapitre est le fruit des travaux de Métivier, Saheb et Zemmari dans [MSZ03b]. Nous concevons et analysons un algorithme probabiliste d'élection par une seule passe dans les arbres basé sur le résultat d'Angluin dans [Ang80]. La méthode

d'analyser le processus d'élection sur un arbre est un ensemble d'éliminations distribuées qui enlève les feuilles, une par une, ramenant l'arbre à un seul sommet, appelé le *chef* (ou le sommet *élu*). Toute feuille dont la durée de vie est expirée est enlevée. Nous supposons que les durées de vie sont des variables aléatoires indépendantes assignées aux sommets selon une certaine distribution calculable localement lorsqu'ils deviendront des sommets feuilles. Par la suite, nous présentons un exemple particulier où ces paramètres sont calculés de façon à donner la même chance d'être élus à tous les sommets de l'arbre. La section 4.5 de ce chapitre étudie un paramètre intéressant qui peut être considéré comme une mesure de la complexité en temps de l'élection. En effet, il s'agit de la durée de l'élection, ce paramètre n'est autre que le temps d'absorption pour un processus markovien irreversible en temps continu.

Chapitre 5 : Dans ce chapitre, nous étudions l'élection uniforme sur les $k-$arbre. Ce travail est un prolongement de l'étude du chapitre précédent sur les arbres (ou les $1-$arbres). Un $k-$arbre est défini récursivement de la manière suivante : la clique avec k sommets est un $k-$arbre ; étant donné un $k-$arbre G avec n sommets, un $k-$arbre avec $(n+1)$ sommets est construit en ajoutant un nouveau sommet v et en liant ce sommet à une $k-$clique de G. Le sommet v dont le voisinage $N(v)$ est une $k-$clique est appelé sommet simplicial. Là aussi, nous affectons à chaque sommet simplicial une variable aléatoire qui représente sa durée de vie. Tout sommet simplicial dont la durée de vie est expirée est supprimé. Dans un $k-$arbre, le problème l'élection devient plus difficile à traiter que dans le cas de l'arbre. En effet, le choix uniforme d'une $(k+1)-$clique n'aboutit pas en général à une élection uniforme sur l'ensemble des sommets, car chaque sommet du $k-$arbre appartient à un nombre indéterminé de cliques.

Le processus d'élection distribuée dans un $k-$arbre est une suite d'éliminations qui supprime les sommets simpliciaux dont les durées de vie sont achevées. Dans les $k-$arbres, ce processus d'élection est composé de deux phases : La première est exécutée une seule fois, elle permet de découvrir, selon des règles particulières que nous allons développer dans la section 5.3.1, un arbre couvrant sur le $k-$arbre en question. Dans cette phase, tous les sommets mémorisent les liens qui appartiennent à cet arbre couvrant pour s'en servir dans la phase qui suit. Dans la seconde phase, le processus d'élection se transforme en un processus d'élection dans son arbre couvrant. Ainsi, un sommet simplicial dans un $k-$arbre G est une feuille dans l'arbre couvrant de G produit par la première phase du processus d'élection. Ce dernier sera modélisé, dans la section 5.5, par un processus markovien en temps continu dans lequel tous les sommets ont la même chance d'être élus.

Chapitre 6 : Dans ce chapitre, nous introduisons une famille de graphes appelés polyominoïdes. Cette famille combine entre la structure des arbres et celle des polyominos.

Nous fournissons un ensemble de règles produisant la classe de tous les polyominoïdes dans la section 6.3. La construction est totalement distribuée et la famille des polyominoïdes peut être engendrée par une grammaire algébrique[1].

[1] grammaire hors-contexte

L'algorithme d'élection sera décrit dans la section 6.4. Cet algorithme est basé sur des *calculs locaux*. L'élection se fait en utilisant une seule passe et les règles de réécriture que nous définissons permettent d'élire les sommets à travers un arbre couvrant standard.

Le résultat principal de la section 6.5 est l'uniformité de l'élection dans les polyominoïdes. Pareillement et dans cette section, nous allons modéliser le processus d'élection par un processus markovien en temps continu.

Partie III. *Synchronisation*.

Chapitre 7 : Dans ce chapitre, nous présentons et étudions un algorithme de rendez-vous basé sur des délais aléatoires. Cet algorithme peut également être considéré comme un algorithme distribué probabiliste pour trouver le couplage maximal. Tout processus du réseau génère une variable aléatoire uniforme dans l'intervalle réel $[0, 1]$ pour chacun de ses processus voisins. Le nombre produit est censé être un temps possible pour avoir un rendez-vous, si les deux processus sont disponibles à ce moment-là. Initialement, le nombre des temps potentiellement possibles proposés par les processus est deux fois le nombre de liens entre eux. Toutes les fois que l'horloge atteint le plus petit temps généré, il y aura un rendez-vous entre le processus qui propose ce temps et le processus sollicité et tous les deux annulent toutes autres données de leurs agendas. Les voisins mettent, ainsi, à jour leurs agendas en supprimant les temps qu'ils ont proposé initialement à ces deux processus. L'algorithme continue pour les processus restants jusqu'à ce que le temps 1 ait expiré.

Nous étudions l'espérance du nombre de rendez-vous qui est la moyenne de la taille du couplage obtenu. Nous donnons une formule récursive qui pourrait être employée pour calculer, en général, ce nombre et nous donnons aussi les formes closes ou les valeurs asymptotiques pour quelques familles intéressantes de graphes (arbres, graphes aléatoires, mille-pattes,...).

Notre algorithme est plus efficace que ceux que nous connaissons dans la mesure où l'espérance du nombre de rendez-vous par tour est plus grande.

Chapitre 8 : Dans ce chapitre, nous considérons une approche de diffusion par inondation pour diffuser des messages à travers le réseau. La diffusion commence par le nœud source disséminant un message parmi ses voisins. Toutes les fois qu'un nœud reçoit le message pour la première fois, il envoie une copie à tous ses voisins à l'exception du noeud duquel il a reçu le message. Le processus de diffusion s'arête lorsque tous les nœuds du réseau ont reçu au moins une copie du message initial à diffuser. Le problème de diffusion est également lié au problème de la propagation des rumeurs ou des virus dans des réseaux sociaux. Un nœud serait *contaminé* s'il connaît le message initial diffusé. Nous nous sommes intéressés au problème de diffusion dans un réseau avec un temps de transmission aléatoire sur chaque lien du réseau. En effet, le temps que prend un message pour traverser un lien de réseau varie d'un message à l'autre. Notre problème est de trouver une borne supérieure de l'espérance mathématique du temps nécessaire pour accomplir la diffusion

sous l'hypothèse que chaque lien a un délai aléatoire de transmission.

Partie IV. *Simulation*.

Cette partie est consacrée à l'implémentation des comportements des systèmes distribués et des algorithmes distribués qui s'excutent sur une architecture distribuée. Un seul chapitre constitue cette partie.

Chapitre 9 : Ce chapitre est composé comme suit : Dans la section 9.1, nous exposons les outils existants. Dans la section 9.2 nous donnons une description de l'architecture générale de notre simulateur. Finalement, dans la section 9.3, nous détaillons la conception et la réalisation de la plate-forme de ce simulateur.

Les résultats présentés aux chapitres 5, 6, 7 et 8 ont respectivement fait l'objet des publications [HSZ05a], [HSZ05b], [HMR+06] et [HSD06].

Première partie

Introduction aux probabilités et aux algorithmes distribués

Chapitre 2

Preliminaires

Sommaire

Dans ce chapitre, nous commençons par rappeler certaines définitions élémentaires sur les graphes et introduire les notations que l'on retrouvera fréquemment. Nous présentons ensuite comment un algorithme distribué peut être modélisé au moyen de réétiquetage de graphes. Finalement, comme nous nous intéresserons dans le reste du document aux algorithmes probabilistes, il est nécessaire de présenter les modèles stochastiques qui seront utilisés pour décrire les algorithmes probabilistes que nous allons étudier.

2.1 Notions de la théorie des graphes

Dans cette section, nous rappelons quelques notions fondamentales de la théorie des graphes. Les définitions sont empruntées du livre de C. Berge [Ber85] et celui de Rosen [Ros00].

2.1.1 Définitions élémentaires

Un *graphe orienté* G est un couple (V, A), où V est un ensemble fini d'éléments $\{v_1, v_2, \cdots, v_n\}$ appelés *sommets*, et A est un ensemble de couples d'éléments distincts du produit cartésien $V \times V$, appelés *arcs*.

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

Un graphe valué noté $G = (V, E, w)$, où V et E sont définis comme ci-dessus et la valuation est une fonction $w : E \to \mathbb{R}$ qui, à toute arête $e \in E$, associe sa valuation (ou poids) $w(e)$.

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

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 $E_1 \cap E_2 = \emptyset$.

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 $N_G(v)$ l'ensemble des voisins de v dans le graphe G. Le degré $deg_G(v)$ du sommet v est le cardinal de $N_G(v)$.

2.1.2 Chemins et connexités

Soit G un graphe non orienté.

Un *chemin* P de v_1 à v_i dans G est une suite $\sigma = v_1, e_1, v_2, e_2, ..., e_{i-1}, v_i$ de sommets et d'arêtes, telle que, pour $1 \leq j < i$, e_j soit incidente avec v_j et v_{j+1}.

La *longueur* d'un chemin est égale au nombre d'arêtes parcourues. La longueur du chemin ci-dessus est $i - 1$.

Lorsque les extrémités d'un chemin sont identiques, $v_1 = v_i$, alors ce chemin est appelé *cycle*.

Deux sommets v et v' sont dits *connexes* ou *liés*, s'il existe un chemin de v à v'.

Un graphe est *connexe* si deux sommets quelconques x et y sont connexes.

Un *sous-graphe* est une partie plus petite d'un graphe.

Un *sous-graphe induit* est un sous-ensemble de sommets avec toutes les arêtes du graphe reliant les sommets de ce sous-ensemble.

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

Un *sous-graphe couvrant* du graphe G est un sous-graphe connexe obtenu par suppression d'arêtes tel que pour chaque paire de sommets, il existe un chemin reliant ces deux sommets.

2.1.3 Arbres et forêts

Un *arbre* T est un graphe connexe sans cycles.

Un sommet de degré inférieur ou égal à 1 s'appelle *feuille*. Dans le cas des arbres de taille $n \geq 2$, les feuilles sont de degré 1.

Si v est une feuille dans un arbre T et u est le sommet voisin de v, alors u est le *père* de v ou v est un *fils* de u.

Si l'arbre est réduit à deux sommets alors chacun d'eux est fils et père de l'autre sommet.

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

Dans la suite, nous exposons quelques concepts propres aux graphes orientés. Une *arborescence A* (ou *arbre enraciné*) est un arbre dans lequel un sommet distingué r est considéré comme la *racine* et les arêtes (ou les arcs) sont orientés de telle manière qu'il existe un chemin (dans A) de la racine vers tous les sommets.

Si A est une arborescence alors pour un sommet donné v de A le sous-graphe de A induit par tous les sommets de A, qui sont accessible à v, est le *sous-arbre de A enraciné en v*.

Un *arbre couvrant* d'un graphe G non orienté est un graphe A tel que

– A couvre G.

– A est un arbre.

Tout graphe connexe admet un arbre couvrant.

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.

2.1.4 Classes particulières de graphes

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

Un graphe G est *biparti* si V peut être partitionné en deux classes, $V(G) = V_1 \cup V_2$ telles que aucun arc de V ne relie deux sommets de V_1 ou deux sommets de V_2.

Un graphe biparti 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* d'ordre n est le graphe $K_{1,n-1}$.

2.1.5 Boules

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

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

On omettra de mettre G pour N_G, deg_G, et B_G dès que le graphe sera compris dans le contexte.

2.1.6 Divers paramètres d'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 :

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

2.2 Systèmes de réécriture de graphe

2.2.1 Graphes étiquetés

Les réécritures de graphes [LMS99] constituent un prolongement des travaux menés par P. Rosenstiehl et al. [FHR72] dans les années 70, par D. Angluin [Ang80] dans les années 80 et

plus récemment par Yamashita et Kameda [YK96, KY96] concernant les propriétés locales et les propriétés globales dans les réseaux et les graphes.

Dans tout ce qui suit, nous considérons seulement les graphes connexes simples où des sommets et des arêtes sont étiquetés avec des étiquettes appartenant à un alphabet L. Cet alphabet peut être fini ou infini. Soit $\mathcal{L}$ un ensemble dont les éléments sont appelés *états*. Un *graphe $\mathcal{L}$-étiqueté* est un couple (G, λ) où G est un graphe et λ est une fonction de $V(G) \cup E(G)$ vers $\mathcal{L}$.
Le graphe G est appelé le *graphe sous-jacent*, et λ est une fonction d'étiquetage de G.

La classe des graphes étiquetés sur un alphabet fini L sera notée $\mathcal{G}_L$.

Soit (G, λ) et (G', λ') deux graphes étiquetés. (G, λ) est un sous-graphe étiqueté de (G', λ') , noté par $(G, \lambda) \subseteq (G', \lambda')$, si G est un sous-graphe de G' et λ est la restriction de λ' à $V(G) \cup E(G)$.

L'application $\varphi\colon V(G) \to V(G')$ est un homomorphisme du graphe étiqueté de (G, λ) vers (G', λ'), si φ est un homomorphisme de G vers G' préservant les étiquettes, i.e., pour tous sommets $v, w \in V(G)$, $\lambda'(\varphi(v)) = \lambda(v)$, et $\lambda'(\{\varphi(v), \varphi(w)\}) = \lambda(\{v, w\})$.
φ est un isomorphisme du graphe étiqueté s'il est bijectif.

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

2.2.2 Systèmes de réécriture de graphe

Nous décrivons dans cette section les notions formelles des systèmes de réécriture de graphe. Une *règle de réécriture* est un triplet $R = (G_R, \lambda_R, \lambda'_R)$ tel que (G_R, λ_R) et (G_R, λ'_R) sont deux graphes étiquetés.

Un *système de réécriture de graphe* (SRG) est un triplet $\mathcal{R} = (\mathcal{L}, I, P)$ où $\mathcal{L}$ est un ensemble d'états, I est un sous-ensemble de $\mathcal{L}$ appelé l'ensemble des *états initiaux* et P un ensemble fini de règles de réécriture.

Une *$\mathcal{R}$-étape de réécriture* est un quintuplet $(G, \lambda, R, \phi, \lambda')$ tel que R est une règle de réécriture et ϕ est à la fois une occurrence de (G_R, λ_R) dans (G, λ) et une occurrence de (G_R, λ'_R) dans (G, λ').

Une étape de réécriture est notée $(G, \lambda) \to_{R,\phi} (G, \lambda')$.

Une *séquence de $\mathcal{R}-$réécriture* est un uplet $(G,\lambda_0,R_0,\varphi_0,\lambda_1,R_1,\varphi_1,\lambda_2,\ldots,\lambda_{n-1},R_{n-1},\varphi_{n-1},\lambda_n)$ tel que pour tout i, $0 \leq i < n$, $(G, \lambda_i, R_i, \varphi_i, \lambda_{i+1})$ est une étape de $\mathcal{R}-$réécriture.
L'existence de telles séquences de réécriture sera notée par $(G, \lambda_0) \xrightarrow[\mathcal{R}]{*} (G, \lambda_n)$.

Le calcul s'arrête quand l'étiquetage du graphe est tel qu'aucune règle de réécriture ne peut être appliquée.

Un graphe étiqueté (G, λ) est alors dit *$\mathcal{R}-$irréductible* s'il n'existe aucune occurrence de

(G_R, λ_R) dans (G, λ) pour toute règle de réécriture R de P.

Pour tout graphe étiqueté (G, λ) dans $\mathcal{G}_I$ nous notons par $Irred_{\mathcal{R}}((G, \lambda))$ l'ensemble de tous les graphes étiquetés $\mathcal{R}-$irréductible (G, λ') tels que $(G, \lambda) \xrightarrow[\mathcal{R}]{*} (G, \lambda')$. Intuitivement parlant, l'ensemble des $Irred_{\mathcal{R}}((G, \lambda))$ contient tous les étiquetages finaux qui sont obtenus à partir des graphes $I-$étiquetés par application des règles de réécriture de P et peuvent être vu comme un ensemble de tous les résultats possibles des calculs codés par le système $\mathcal{R}$. Nous nous intéressons à l'étude de la probabilité d'obtenir une solution particulière parmi un ensemble de solutions possibles.

Un système de réécriture est défini par la donnée d'un ensemble fini de telles règles et par une structure de contrôle local : ce contrôle local autorise ou bien interdit l'application de certaines règles. Ainsi nous allons définir des réécritures contrôlées par "les contextes interdits" : nous ne pouvons appliquer une règle que si dans l'environnement immédiat de l'application certains contextes sont absents. Le comportement du réseau est défini par un étiquetage initial et par une suite de pas de calcul.

Deux étapes de réécriture sont dites *indépendantes* si elles sont exécutées dans des sous-graphes disjoints. Dans ce cas, ces deux étapes peuvent être appliquées dans n'importe quel ordre et même simultanément. Ces règles locales de réécriture sont appliquées jusqu'à ce que la "stabilité" soit atteinte, c'est-à-dire aucune règle n'est applicable. La configuration résultante est dite de *forme normale*. En effet, une exécution de l'algorithme est ainsi déterminée par une succession de règles de $\mathcal{R}$ appliquée en une boule de rayon fixe.

Définition 2.2.1. Un système de réécriture de graphe $\mathcal{R}$ est dit noethérien s'il n'existe pas de séquence infinie de $\mathcal{R}$-réécriture de graphe tel qu'initialement, l'étiquetage du graphe satisfait l'ensemble des états initiaux [LMS99].

2.3 Notions de la théorie des probabilités

La modélisation mathématique pour étudier un problème peut être issue du domaine des probabilités. Notre but n'est évidemment pas de donner une introduction relativement complète à la théorie des probabilités, mais de présenter les outils et méthodes bien connus dont nous aurons besoin dans la suite de cette thèse. Les références en la matière ne manquent pas. Parmi elles nous pouvons citer les ouvrages [Fel50, Fel66].

Dans certains systèmes, nous sommes amenés à introduire la notion de probabilités, et ce pour plusieurs raisons. Elle permet tout d'abord de modéliser de façon probabiliste certains comportements réels après une observation statistique, comme par exemple sur des canaux non fiables, la probabilité qu'un message soit perdu.

Les probabilités peuvent également être introduites pour des raisons d'efficacité. L'ajout de probabilités est même parfois nécessaire. En effet, il existe des problèmes pour lesquels il a été prouvé qu'il n'existe pas de solution déterministe. C'est souvent le cas dans des systèmes symétriques où les probabilités sont requises pour rompre la symétrie. Un exemple bien connu

est celui du dîner des philosophes [Dij72], pour lequel Lehmann et Rabin [LR81] ont montré qu'il n'existe pas de solution déterministe, symétrique et totalement distribuée. Ils donnent alors une solution probabiliste à ce problème [LR94]. Cet algorithme permet de résoudre un problème d'allocation de ressource insoluble dans le cadre déterministe. Informellement, N philosophes sont assis autour d'une table, une baguette étant disposée entre deux philosophes "voisins". Pour manger, un philosophe a besoin de tenir simultanément ses deux baguettes, à savoir celle partagée avec son voisin de gauche et celle partagée avec son voisin de droite. Le but de l'algorithme étant d'assurer que si un philosophe a faim, alors en un temps fini un philosophe (pas nécessairement le même) va manger.

Les notations utilisées en probabilité, $Pr(E)$, $\mathbb{E}(X)$ et $var(X)$, sont les notations usuelles respectives de la probabilité d'un événement E, de l'espérance et de la variance d'une variable alétoire (v.a.) X.

2.3.1 Variables aléatoires et lois de probabilités

Une *v.a. (variable aléatoire)* est une application mesurable définie sur l'ensemble des résultats possibles d'une expérience aléatoire, telle qu'il soit possible de déterminer la probabilité qu'elle prenne une valeur donnée ou qu'elle prenne une valeur dans un intervalle donné. Cette localisation conduit à associer à toute v.a. une loi de probabilité sur $\mathbb{R}$.

Les variables aléatoires $X_1, \ldots, X_n$ sont dites indépendantes si pour tout n-uplet $(A_1, \ldots, A_n)$ de boréliens de $\mathbb{R}$, les évènements "$X_1 \in A_1, \ldots, X_n \in A_n$" sont indépendants. Une suite (X_n) de variables aléatoires indépendantes est telle que pour tout n les variables aléatoires $(X_1, \ldots, X_n)$ sont indépendantes.

Nous appelons *loi* de la v.a. X la loi de probabilité P_X sur $\mathbb{R}$, définie pour tout borélien A de $\mathbb{R}$ par :

$$P_X(A) = Pr(X \in A) \ .$$

Variables aléatoires discrètes

Nous disons qu'une v.a. est discrète si elle ne prend qu'un nombre fini ou dénombrable de valeurs :

$$X \in \{x_k \,,\ k \in K \subset \mathbb{N}\} \ .$$

Dans le cas où une v.a. est discrète, la loi de la v.a. X est la loi de probabilité sur l'ensemble des valeurs possibles de X qui affecte la probabilité $Pr(X = x_k)$ au singleton $\{x_k\}$.

Les lois discrètes les plus courantes sont : la loi uniforme, la loi de Bernoulli et la loi binomiale.

La loi uniforme sur un ensemble fini est la loi des "tirages au hasard" dans cet ensemble, ou équiprobabilité. Elle donne la même probabilité $1/n$ à tous les éléments de l'ensemble, avec n est le cardinal de cet ensemble.

Variables aléatoires continues

Soit X une v.a. à valeurs dans $\mathbb{R}$ et f_X une densité de probabilité sur $\mathbb{R}$. Nous disons que X est une v.a. continue de densité f_X si pour tout intervalle I de $\mathbb{R}$ nous avons :

$$Pr(X \in I) = \int_I f_X(x)\, dx \ .$$

La loi de la v.a. X est la loi continue sur $\mathbb{R}$, de densité f_X.

Les lois continues les plus fréquentes sont : la loi uniforme, la loi exponentielle, la loi normale, la loi gamma et la loi student.

2.3.2 Fonction de répartition

La fonction de répartition d'une v.a. X à valeurs dans $\mathbb{R}$ (ou plus exactement de sa loi) est la fonction F_X, de $\mathbb{R}$ dans l'intervalle réel $[0, 1]$, qui à $x \in \mathbb{R}$ associe :

$$F_X = Pr(X \leq x) \ .$$

La fonction de répartition d'une variable aléatoire continue est la primitive de la densité qui s'annule en $-\infty$:

$$F_X(x) = Pr(X \leq x) = \int_{-\infty}^{x} f_X(t)\, dt \ .$$

C'est une fonction continue sur $\mathbb{R}$. En tout point x où f_X est continue, F_X est dérivable et :

$$\frac{dF_X(x)}{dx} = f_X(x) \ .$$

Le théorème suivant donne une interprétation du produit de convolution de deux fonctions de répartition.

Théorème 2.3.1. *[Fel66] Soient X et Y deux variables aléatoires indépendantes de fonctions de répartition F et G respectivement. Alors*

$$Pr(X + Y \leq t) = \int_{-\infty}^{+\infty} G(t - x)F\{dx\}.$$

2.3.3 Processus et chaînes de Markov

Pour décrire un ensemble de phénomènes où le hasard intervient indépendamment de notre volonté, on introduit la notion de processus stochastique.

Un processus stochastique est une famille de variables aléatoires X_t :

$$\{X_t, t \in T\}$$

où t représente souvent le temps. Les valeurs de T peuvent être continues ou discrètes.

Une classe particulière de ces processus a été définie dans les années 30 par le mathématicien russe Markov.

Nous apellerons *état* une valeur du processus stochastique prise à un instant t, et *transition* le passage d'un état à un autre.

Un processus est markovien, si pour tout instant u, pour toute valeur $X_u = x$ donnée, la probabilité pour que le processus prenne la valeur y à un instant quelconque t ultérieur ($t > u$), ne dépend pas des valeurs prises par le processus avant l'instant u. Le processus est *sans mémoire.*

Les chaînes de Markov sont des processus markoviens qui évoluent dans le temps "discret" selon des transitions probabilistes. Elles vont donc décrire un ensemble de phénomènes aléatoires selon le principe des processus stochastiques.

Soit $(X_t)_{t\in\mathbb{N}}$ une suite de variables aléatoires à valeur dans $\mathbb{N}$. On dit que (X_t) est une chaîne de Markov si :

$$Pr(X_{t+1} = x_{t+1}|X_t = x_t, \cdots, X_0 = x_0) = Pr(X_{t+1} = x_{t+1}|X_t = x_t)$$

Ce qui signifie que X_{t+1} ne dépend que de X_t. L'indice est en général interprété comme le temps, ce qui nous précise que la probabilité sur la valeur de X à l'instant $t + 1$ ne dépend que de sa valeur à l'instant t. Ceci est appelé la propriété de Markov. De plus la chaîne de Markov sera dite homogène si la probabilité $Pr(X_{t+1} = x_{t+1}|X_t = x_t)$ ne dépend pas de t.

Pour nos problèmes, X_t caractérise le système à l'instant t : cela peut être le graphe résiduel (sommets et arêtes non encore supprimés). L'état initial ($t = 0$) est le graphe tout entier.

Une chaîne de Markov est irréductible si, partant d'un état quelconque et considérant un état quelconque, on a une probabilité non nulle pour qu'un jour, le système passe par cet état final. Alors tous les états communiquent.

Chapitre 3

Généralités sur les algorithmes distribués probabilistes

Sommaire

Les probabilités sont de plus en plus utilisées dans la conception et l'analyse des systèmes logiciels et matériels informatiques. L'introduction des tirages aléatoires dans les algorithmes concurrents et distribués permet de résoudre certains problèmes insolubles dans le cadre déterministe et de réduire la complexité de nombreux autres.

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

Des considérations générales sur les algorithmes distribués probabilistes peuvent être trouvées dans [Tel00]. Dans [MR95] et dans [Lav95] les auteurs présentent quelques techniques utilisées pour concevoir et analyser ces algorithmes.

Nous nous intéressons aux algorithmes probabilistes qui permettent d'obtenir un objet appartenant à un ensemble donné, il s'agit alors de déterminer la probabilité d'obtenir un objet particulier. Par exemple, étant donné un graphe G nous pouvons, sous certaines hypothèses, calculer un arbre recouvrant T.

3.1 Modèles et définitions

Nous considérons le modèle standard des réseaux de communication point-à-point [Tel00, Lav95]. Un réseau est décrit par un graphe orienté, connexe et fini $G = (V, E)$. Les sommets représentent les nœuds ou les processus du réseau, les arcs représentent les liens de communication entre les processus. Tout processus possède une mémoire locale non partagée de capacité bornée et au moins un processeur. Il ne peut communiquer directement qu'avec ses voisins. Aussi, il est capable de faire la distinction entre ses portes. Les identités (numéros IP sur Internet) des autres processus, et en particulier celles de ses voisins, lui sont inconnues (*orientation locale*). Il est fiable : il ne se produit aucune défaillance, ni sur les sommets, ni sur les liens de communication.

Les processus communiquent dans un réseau uniquement par messages. L'échange de messages se fait sans perte ni altération ni modification. Il constitue le coût d'exécution dominant d'un algorithme sur un réseau. La manière dont les échanges entre processus s'effectuent détermine le type de réseau.

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 processus 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 processus envoie un message à un autre processus 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.

Dans tous les algorithmes que nous présentons dans les chapitres qui suivent, les communications sont considérées locales et simultanées entre un processus et tous ses voisins, à l'exception du chapitre 8 où un message met un temps aléatoire pour traverser le lien, ce temps est supposé suivre une loi exponentielle.

Réseau synchrone, réseau asynchrone

Un réseau est *synchrone* s'il existe une horloge globale[1] et que toutes les opérations de calculs se font simultanément : à chaque top d'horloge, les processus effectuent une action. En effet, le temps est divisé en un nombre infini $0, 1, 2, \cdots$ d'*unités de temps*. Les processus commencent une exécution à l'instant $t = 0$. À chaque unité de temps t, un processus 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 son état (c'est-à-dire effectue un calcul quelconque) et envoie éventuellement des messages à ses voisins. Le nouvel état du processus p et les messages envoyés dépendent de son état à l'instant $t - 1$ et

[1] commune à tous les processus

des messages qu'il a reçu.

Un réseau *asynchrone* est un réseau sans horloge globale, chaque processus a sa propre horloge, et peut évoluer dans le temps indépendamment de l'évolution des autres processus. En effet, aucune supposition n'est faite au niveau du temps d'exécution d'une procédure par un processus ou au niveau du temps d'envoi et de la réception d'un message.

Réseau anonyme

Tout processus est caractérisé par son identificateur propre ou identité appartenant à un ensemble quelconque (non vide, fini ou non), muni d'un ordre total (adresse Internet IP ou adresse MAC par exemple). Un réseau de processus est dit *anonyme* si les processus ne connaissent pas leurs identités (*symétrie totale*).

3.2 Quelques problématiques de l'algorithmique distribuée

Il apparaît que certains problèmes naturels sur les réseaux distribués sont difficiles à résoudre et n'admettent parfois aucune solution. Nous allons en fournir quelques exemples et voir comment leur "difficulté" peut être modulée. Il y a en particulier les problèmes de type "rupture de symétrie" dont le problème d'élection est le plus significatif : étant donné une certaine situation initiale, est-il possible de distinguer un nœud unique [IR90, BSV+96].

Alors, il apparaît que dans ce cas, si le réseau a de plus une topologie "symétrique" le problème n'a pas de solution. Nous y reviendrons dans la section suivante. La difficulté de ces problèmes peut être modulée en fonction de connaissances supplémentaires sur la structure du réseau que peuvent avoir les processus.

Par ailleurs et dans le reste de ce document, nous sommes amenés à introduire et à analyser des algorithmes probabilistes et à analyser la probabilité d'obtention d'une certaine solution. Néanmoins, l'existence d'algorithmes permettant de résoudre certains problèmes dépend des hypothèses faites sur le réseau ou de la connaissance que chaque processus a sur le réseau, par exemple : réseau synchrone ou asynchrone, réseau anonyme ou non, connaissance de la topologie, de la taille ou une borne de celle-ci.

3.2.1 Symétrie et impossibilté de l'élection

Depuis le travail original d'Angluin [Ang80], il est connu que certains problèmes comme l'élection n'ont parfois aucune solution pour des raisons de symétrie "interne" au réseau. Un réseau G est "symétrique" s'il existe un autre graphe H, de taille inférieure, tel que G est en quelque sorte constitué de copies de H entrelacées de telle manière qu'un processus ne puisse, à l'aide des informations locales auxquelles il a accès, savoir si le réseau sous-jacent sur lequel il s'exécute est G ou H (il y a une application injective de G dans H). Dans cette configuration,

nous disons que G est un revêtement de H. Un algorithme distribué étant constitué essentiellement d'échanges d'information entre un sommet et ses voisins, il ne peut rassembler que des informations locales, et par conséquent, son exécution sur H peut être similaire et indistinguable, sous un certain sens, à une exécution sur G. Malheureusement, la condition d'Angluin est nécessaire, mais pas suffisante. En effet, dans [BSV+96] les auteurs montrent qu'il existe des graphes qui n'admettent pas un recouvrement, mais pour lesquels il n'est pas possible de trouver un d'algorithme d'élection du chef.

D'ailleurs, si nous cherchons à élire, nous pouvons éventuellement obtenir un seul sommet *élu* sur H, mais G étant constitué de copies de H, il existe une exécution de l'algorithme sur G, similaire à celle sur H, qui s'achèvera avec de multiples sommets *élus*. L'algorithme ne pourra donc élire systématiquement sur G. Par conséquent, le problème de l'élection n'a pas de solution sur G.

Par exemple, un certain nombre de problèmes sont plus "faciles" à résoudre si l'on connaît le nombre de sommets dans le réseau.

Il s'avère que le genre d'asymétrie qui peut être détecté d'une façon distribuée dépend du modèle de calcul utilisé.

FIG. 1 – Le graphe K_2

Par exemple, considérons le graphe de la Fig. 1 (graphe très symétrique).

Initialement les deux sommets sont dans le même état et ils exécutent le même algorithme. Cependant, si l'un d'eux décide d'envoyer un message, forcément, la même décision sera prise par l'autre sommet. Ainsi, les deux sommets s'échangent mutuellement des messages, puis chacun d'eux calcule son nouvel état. Les nouveaux états sont identiques puisque le contenu du message reçu est similaire à celui envoyé au voisin. Par conséquent, les sommets évoluent de la même manière et en aucun temps, ils seront dans des états différents.

3.2.2 Détection de la terminaison

Une autre problématique qui se pose dans l'algorithmique distribuée concerne la détection de la terminaison. Dans le cas séquentiel, lorsqu'un algorithme s'exécute, il lui est facile de constater s'il a terminé sa tâche ou pas. Dans le cas distribué c'est plus délicat. En effet, même si, localement, un processus peut constater qu'il n'a aucune instruction à effectuer, du fait qu'il fonctionne en interaction avec le reste du réseau, il ne peut prédire si, dans le futur, il va recevoir, ou non, un message qui devra provoquer une réaction de sa part. Il ne peut a priori pas cesser d'attendre des messages éventuels provenant de ses voisins. Un observateur extérieur peut constater que, globalement, la tâche à réaliser est achevée, le calcul distribué est terminé : tous les sommets sont dans un état où il n'ont et n'auront plus rien à faire. Une telle terminaison est définie comme implicite : le calcul est effectivement terminé mais aucun sommet ne le *sait*. Nous pouvons également prescrire, cela paraît raisonnable et est même indispensable dans certain cas,

que la terminaison soit détectée localement par un, ou tous les sommets. Nous pouvons exiger la détection de la terminaison locale (le sommet sait qu'il a atteint son état final, qu'il n'effectuera plus aucune opération) ou la terminaison globale (un sommet sait que tous les autres sommets ont atteint leur état final ; il s'agit en général du sommet effectuant le dernier pas de calcul).

3.2.3 Tolérance aux pannes

La capacité d'un système à fonctionner malgré une défaillance d'une de ses composantes est appelée tolérance aux pannes (parfois nommée tolérance aux fautes, en anglais *fault tolerance*).

En présence de pannes, un problème tel que celui du consensus peut se résoudre tout simplement dans un système synchrone. Par contre, le résultat classique de Fischer, Lynch et Paterson [FLP85] montre que, dans un système asynchrone, il est impossible à résoudre, même en se restreignant à des pannes définitives de processus.

Puisqu'il est impossible d'empêcher totalement les pannes, une solution consiste à ce que : lorsqu'une des ressources tombe en panne, les autres ressources prennent le relais afin de laisser le temps aux administrateurs du système de remédier à l'avarie.

L'élection de leader ultime peut être considéré comme un détecteur de défaillances. Dans [DGFHR01], les auteurs ont obtenu un algorithme d'élection de leader ultime efficace. Il s'agit en fait du détecteur de défaillances le plus faible pour résoudre le consensus.

3.3 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.

Un des paradigmes de l'informatique distribuée est le bris de la symétrie des processus en octroyant à l'un d'entre eux un privilège ; être leur *chef*. Il est bien connu qu'une fois qu'un chef est trouvé, la plupart des autres propriétés du réseau puissent être calculées facilement. Par exemple, le chef peut initier le calcul d'un un arbre couvrant, calculer le nombre de sommets dans le réseau, assigner à chacun d'eux une identité unique, et. Réciproquement, si chaque processus a une identité unique, le processus dont la plus petite (ou plus grande) identité peut être désigné pour être le chef.

Sur le plan théorique, le problème d'élection dans un réseau synchrone ou asynchrone est

d'ailleurs, par essence, sans solution exacte lorsque les identités des processus sont indiscernables ; il est alors impossible de briser la symétrie structurelle des processus du système. Le papier original dans le secteur est dû à Angluin [Ang80](cf. sous section 3.2.1).

Ainsi, le problème ne peut être résolu par un algorithme simplement déterministe. La symétrie est impossible à briser et le problème est sans solution. Seuls des algorithmes probabilistes permettent de contourner cette difficulté en donnant une solution partielle très acceptable au sens probabiliste à ce genre de problèmes.

Election dans les anneaux

L'élection sur des anneaux a déjà donné lieu à quantité de recherches, tant dans le cas où les processus sont tous distingués par leurs identités ("anneaux avec identités") que dans celui où ils sont sans identités, indiscernables ("anneaux anonymes"). Nous savons également différencier les classes d'anneaux selon des critères pertinents du point de vue de la complexité en communication des algorithmes, i.e. anneaux synchrones ou asynchrones, uni- ou bidirectionnels, mais aussi anneaux avec ou sans information structurelle ("sens général de la direction", connaissance exacte ou approchée de la taille de l'anneau, etc.) : tous types de connaissance à même d'améliorer considérablement la complexité en communication de bien des algorithmes distribués.

Nous avons le théorème suivant dû à Angluin [Ang80] et Itai et al [IR90] :

Théorème 3.3.1. *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 processus.*

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).

Afin de briser la symétrie dans un anneau anonyme dont la taille est connue, Itai et Rodeh [IR90] supposent que chaque processus a accès à une suite infinie de nombres réels représentant les probabilités. Évidemment, ceci est fait pour obtenir des preuves plus propres.

Caractérisations des algorithmes probabilistes

A cause du "hasard", de tels algorithmes peuvent fournir de fausses solutions ou encore ne pas se terminer du tout. Par conséquent, si la probabilité que de tels événements se produisent est faible, il suffit d'exécuter l'algorithme plusieurs fois et nous sommes "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, noté $\mathbb{E}$, est bornée et,
  – retournent une solution correcte.

  Formellement, soit $\mathcal{A}$ un algorithme qui traite un problème P et soit $c(\mathcal{A})$ à sa complexité. Nous cherchons en général a avoir $\mathbb{E}[c(\mathcal{A})] < \infty$ (et aussi à le minimiser).

- Algorithmes de type *Monte Carlo* : Le nom de ces algorithmes fait allusion aux jeux de hasard pratiqués à Monte-Carlo. ce sont des algorithmes probabilistes qui

  – se terminent en un temps fini et,
  – retournent une solution correcte avec probabilité $\geq 1/3$.

  Le véritable développement des méthodes de Monte-Carlo s'est effectué, sous l'impulsion de John von Neumann et Ulam notamment, lors de la Seconde Guerre mondiale et des recherches sur la fabrication de la bombe atomique. Notamment, ils ont utilisé ces méthodes probabilistes pour résoudre des équations aux dérivées partielles.

Nous pouvons noter que le résultat d'un algorithme distribué est un élément de l'ensemble S des solutions possibles ; ceci donne naturellement naissance à des questions du type : étant donné un élément e de S, calculer la probabilité pour que le résultat de l'algorithme distribué soit e. Par ailleurs, dans le chapitre qui suit nous donnerons un modèle général pour caractériser la répartition de probabilités d'être élu pour les sommets d'un arbre.

Deuxième partie

Algorithmes probabilistes d'élection uniforme

Chapitre 4

Élection uniforme probabiliste dans les arbres

Sommaire

Ce chapitre est organisé comme suit. La section 4.1, présente quelques résultats connus et mentionne l'objectif de cette étude. La section qui suit est consacrée à la présentation du modèle probabiliste général d'élection dans un arbre. Nous donnons une description formelle de l'algorithme qui sera l'objet principal de notre analyse. Cet algorithme est un algorithme probabiliste d'élection par une seule passe qui utilise un paramètre localement calculable guidant aléatoirement le processus d'élimination. Certaines propriétés de ce modèle d'élection seront exposées dans la section 4.3. Par ailleurs, nous allons voir que le processus d'élection pourra être modélisé par des chaînes de Markov en temps continu. La section 4.4 traite un cas particulier du modèle général, appelé modèle uniforme, dans lequel tous les sommets de l'arbre ont la même chance d'être élus. Finalement, la section 4.5 étudie la durée d'élection dans le modèle uniforme pour des exemples particuliers d'arbres.

4.1 Introduction

Le problème de l'élection est de choisir exactement un élément dans l'ensemble de processeurs. Ainsi, à partir d'une configuration où tous les processeurs sont dans le même état, nous

devons obtenir une configuration où exactement un processeur est dans l'état *chef* et tous les autres processeurs sont dans l'état *perdant*. Le chef peut être employé plus tard pour prendre des décisions ou pour centraliser des informations.

Ce problème a été posé la première fois par Le Lann [Lan77] et depuis, de nombreux algorithmes pour élire un sommet dans un graphe sont connus [Ray88, Tel00]. Il a été résolu sous plusieurs hypothèses différentes : le graphe peut être un anneau, un arbre, un graphe complet ou en général un graphe connexe. Le système peut être synchrone ou asynchrone. Le graphe peut être orienté ou non orienté. Les processeurs peuvent ou ne peuvent pas connaître la taille du graphe ou une borne de celle-ci. Les communications peuvent être synchrones ou asynchrones.

Il est, par ailleurs, connu que certaines classes de graphes n'admettent pas d'algorithmes d'élection distribuée déterministes [God02]. Par conséquent, les algorithmes probabilistes sont apparus pour donner une solution exacte ou approchée à cette problématique.

Dans le cas des arbres, un processus d'élection peut être vu simplement comme suite de suppression successive de *feuilles*, ramenant le graphe à un sommet unique. Ce processus a été présenté la première fois par Angluin ([Ang80] Théorème 4.4) où elle a mentionné qu'il existe une ligne de processeurs (d'états finaux) qui établit de manière déterministe un centre dans chaque arbre. Une méthode simple pour découvrir le centre d'un arbre est de supprimer à chaque tour les feuilles de l'arbre, ainsi s'il reste un seul sommet avant la disparition totale de l'arbre, ce sommet sera l'unique centre. S'il en reste deux, l'arbre aura deux centres.

Un *ordre d'élimination* peut être vu comme un processus aléatoire où la probabilité d'élire un sommet dépend de la façon dont nous présentons le choix de la disparition des feuilles. Dans [MS94] les auteurs considèrent deux approches élémentaires. La première est fondée sur l'hypothèse que *toutes* les suites du processus d'élection ont la *même* probabilité (la **p**-distribution). La seconde suppose que, à chaque étape, toutes *les feuilles* ont la *même* probabilité d'enlèvement (la **q**-distribution). Dans la première approche le calcul des probabilités impliquées est transformé en un problème combinatoire intéressant, qui fournit une simple caractérisation des probabilités recherchées sur l'ensemble de sommets. Cependant, la difficulté réside dans le fait qu'aucun processus d'élimination probabiliste, par une simple passe en utilisant un calcul local, n'est disponible pour mettre en application l'approche sans faire un calcul préliminaire sur l'arbre. La deuxième approche peut être implémentée d'une manière distribuée et la probabilité d'être élu est calculée inductivement. Son inconvénient est que la distribution de probabilité engendrée, étant à caractère rigide, n'admet aucune caractérisation simple.

Le but principal de ce chapitre est d'étudier les chances d'élection pour les sommets d'un arbre. Ainsi, nous présentons et analysons un algorithme probabiliste fondé sur l'élimination successive des feuilles d'un arbre jusqu'à ce que cet arbre soit réduit à une seule feuille. Cette dernière sera considérée comme le sommet élu. Ainsi, la répartition de probabilités d'être élu pour les sommets de l'arbre dépend de l'implémentation *distribuée* de l'algorithme.

Le concept d'équité ou de l'équitabilité (en anglais *fairness*) est une propriété importante, voir [DIM95]. Nous pouvons par exemple utiliser cette propriété pour garantir que tous les processus (machines), qui collaborent pour faire un calcul complexe, ont la même quantité de données à

traiter.

Dans la seconde partie de cette étude, nous nous intéresserons essentiellement à l'élection uniforme sur un réseau ayant une topologie d'arbre. Ainsi, nous employons la propriété d'équité dans son sens strict : tous les processus ont la même probabilité d'être élus, de sorte que lorsque l'algorithme est réitéré plusieurs fois, les ressources sont probablement allouées équitablement.

4.2 Modèle général et processus d'élection

Le modèle de communication est un réseau de communication point-à-point qui est représenté comme un simple graphe connexe non orienté, où les sommets représentent des processeurs et deux sommets sont liés par une arête si les processeurs correspondants ont une liaison directe. Ce modèle suit les modèles standards pour les systèmes distribués donnés dans [Pel00, Tel00].

Dans tout ce qui suit nous supposons que :

- Le graphe est un arbre.
- Les processeurs communiquent par envoi de messages, et chaque processeur sait par quel canal il reçoit un message.
- Le réseau est anonyme ; des identités uniques ne sont pas disponibles pour distinguer les processeurs.
- L'algorithme présenté ici est synchrone et basé sur un temporisateur : les processeurs ont accès à une horloge physique comme donné dans [Tel00] page 87 :

  1. nous supposons que le temporisateur est une période continue globale auquel les processeurs ont accès : une variable à valeurs réelles dont la valeur augmente continuellement dans temps,
  2. nous faisons une hypothèse concernant le temps global, c'est que chaque événement a lieu à un certain temps et que l'événement lui-même est instantané (de durée nulle).

L'élection probabiliste présentée ici est basée sur le travail d'Angluin [Ang80]. Un arbre de taille n est initialement donné. Le sommet élu est le sommet qui survit à une suite de $(n-1)$ élimination de feuilles. Dans l'algorithme d'Angluin, un paramètre localement calculable a été introduit pour guider le processus d'élimination.

Description de l'algorithme

Dans ce qui suit, nous employons la description de l'algorithme introduit par Tel (cf. [Tel00] page 192, Algorithme 6.3).

Dans l'algorithme 1 ci-dessus, la fonction $attente-\text{exponentielle}(\lambda(v))$ produit un temps d'attente aléatoire exponentiellement distribué de paramètre $\lambda(v)$; ainsi le temps d'attente n'est autre

Algorithme 1: Élection probabiliste dans un arbre.

var $Neigh_v$: ensemble de sommets (* les voisins de v *) ;
$rec_v[u]$ pour chaque $u \in Neigh_v$: booléen **init** faux ;
(* $rec_v[u]$ est vrai si v a reçu un message de u *) ;
$\lambda(v)$: le paramètre avec la même valeur initiale pour tous les sommets ;
(* Dans le cas uniforme, initialement $\lambda(v) := 1$ *) ;

début
Tant que(#{u : $rec_v[u]$ est faux}> 1)
début
recevoir <**tok**, $\lambda(u)$ > de u ;
$\lambda(v) := F(\lambda(v), \lambda(u))$;
(* F est n'importe quelle fonction ; dans le cas uniforme nous prenons
$F(\lambda(v), \lambda(u)) = \lambda(v) + \lambda(u)$ *) ;
$rec_v[u] := vrai$
fin
(* Maintenant, il y a *un* u_0 avec $rec_v[u_0]$ est faux ; le sommet v est une feuille*)
attente−exponentielle ($\lambda(v)$) ;
si (un message <**tok**, p > est arrivé)
début
$state_v := élu$;
fin
sinon
début
$state_v := perdant$;
envoyer <**tok**, $\lambda(v)$ > à u_0 avec $rec_v[u_0]$ est faux
fin
fin

que :

$$\frac{-1}{\lambda(v)} ln(x),$$

où x est une variable aléatoire (v.a.) uniforme à valeurs réelles dans l'intervalle [0, 1].

L'algorithme est lancé par les feuilles à l'instant 0. Tout sommet reste en attente de messages de ses voisins jusqu'à ce qu'il reçoive des messages de tous ses voisins sauf un. Tout sommet v qui devient une feuille a une durée de vie exponentiellement distribuée de paramètre $\lambda(v)$ déterminée au moment où v est une feuille.

Une feuille v dont la durée de vie expire sans qu'elle ait reçu de message, disparaît avec son arête incidente : son état devient *perdant*; elle envoie à son père un message avec la valeur $\lambda(v)$. Le paramètre $\lambda(v)$ du sommet v qui reçoit un message <**tok**,α > avec une valeur α devient égal à $F(\lambda(v), \alpha)$.

Un sommet est élu s'il reçoit de tous ses voisins des messages avant la fin de sa durée de vie. Le temps est continu de sorte que deux messages ne soient jamais simultanément envoyés ; par conséquent, le sommet élu est le seul sommet ayant reçu un message de tous ses voisins (voir [Tel00] page 193).

Selon le choix de F, nous obtenons différentes distributions des chances d'être élu. D'ailleurs, le processus d'élection est modelé par un processus de mort en temps continu comme suit. Chaque sommet v, qui devient une feuille, a une durée de vie exponentiellement distribuée de paramètre $\lambda(v)$, censé être déterminé au moment où v est une feuille. Cette prétention est équivalente à celle que la probabilité de mort de v dans l'intervalle de temps $[t, t + h]$ est $h\lambda(v) + o(h)$, lorsque $h \to 0$, à tout instant t, et ceci indépendamment de ce qui se passe ailleurs et de ce qui s'est produit dans le passé.

Une feuille dont la durée de vie s'est écoulée disparaît avec l'arête incidente. Le processus continue dans l'arbre résiduel jusqu'à ce que l'arbre soit réduit à un seul sommet. Ce modèle général laisse le choix de $\lambda(v)$ ouvert. Tout ce qui est exigé est que le paramètre $\lambda(v)$ doit être localement déterminé quand v devient une feuille ; il peut être calculé au fur et à mesure que les sommets sont supprimés. L'idée intuitive derrière ce modèle est de contrôler les chances d'élection par une λ-assignation : si $\lambda(v)$ augmente alors la probabilité d'élire v diminue.

Si le paramètre λ est le même pour n'importe quel sommet de l'arbre, alors à chaque feuille supprimée est assignée la même probabilité comme dans la deuxième approche du travail de [MS94] et, en conséquence, la probabilité d'être élu est une **q**-distribution. En conséquence, le modèle actuel prolonge la deuxième approche présentée dans [MS94] (concernant la distribution de probabilité générée). Cependant, il est possible d'assigner différents taux de mortalité $\lambda(v)$ aux sommets selon leurs positions initiales dans l'arbre. Il est clair que les différentes considérations mènent ainsi à différentes distributions de probabilités dans le processus d'élection.

Une question qui surgit normale sera : *comment assigner λ pour donner la même chance d'être élu à tous les sommets d'un arbre T ?*

Cette tâche semble *a priori* difficile, puisque les sommets "intérieurs" d'un arbre sont plus

couverts que les sommets "périphériques".

Le résultat principal, le théorème 4.4.1, stipule que cet exemple particulier du modèle général, appelé *modèle uniforme*, peut être réalisé en affectant à $\lambda(v)$ la taille du sous-arbre de T constitué de la racine v.

Cette tâche peut être effectuée d'une manière distribuée comme suit. Initialement, tous les sommets de T ont le même poids 1 : $w(v) = 1$. Pour une feuille v, le paramètre $\lambda(v)$ est égal à son poids : $\lambda(v) = w(v)$. Quand une feuille est enlevée son père récupère son poids, ce qui revient à dire que le poids du père est incrémenté du poids du fils enlevé $F(\lambda(v), w(v)) := \lambda(v) + w(v)$.

4.3 Propriétés simples du modèle général

Une fois qu'un sommet v devient une feuille, sa mort se produit selon un processus markovien avec un paramètre $\lambda(v)$ donné, qui est censé être déterminé par quelques règles spécifiques que nous détaillons dans la suite. Chaque sommet feuille v de l'arbre résiduel, se comporte *indépendamment* et, sous l'hypothèse qu'il est vivant au temps t, la probabilité de sa mort dans l'intervalle de temps $[t, t + h]$ est donnée par $\lambda(v)h + o(h)$, quand $h \to 0$. Le paramètre $\lambda(v)$ est référé comme le *taux de mortalité* de v. La quantité $1/\lambda(v)$ est l'espérance mathématique de la durée de vie du sommet v (dès l'instant où celui-ci devient une feuille). Il convient de noter que $\lambda(v)$ *n'est pas* un paramètre fixé et assigné initialement à v ; il peut être calculé en fonction des valeurs transmises à v (valeurs transmises par les sommets adjacents disparus). En particulier, si v a plusieurs sommets voisins au moment où v devient une feuille, alors seulement un de ses voisins est encore en vie. Ceci signifie que $\lambda(v)$ est calculé étape-par-étape, il ne dépend que des sommets voisins disparus.

Proposition 4.3.1. *[KT81] Soit v un sommet, d'un arbre T, devenant une feuille à un certain instant t. Un taux de mort $\lambda(v)$ est assigné à v à cet instant. Soit $L(v)$ la durée de vie de v (dès cet instant). Alors nous avons :*

(i) *$L(v)$ est une v.a. (variable aléatoire) exponentiellement distribuée de paramètre $\lambda(v)$:*

$$Pr(L(v) > x) = e^{-\lambda(v)x}, \quad \forall x \geq 0. \tag{2}$$

(ii) *$L(v)$ a l'espérance mathématique $\frac{1}{\lambda(v)}$.*

(iii) *$L(v)$ satisfait la propriété de "sans mémoire" :*

$$Pr(L(v) > x + y \mid L(v) > x) = Pr(L(v) > y) = e^{-\lambda(v)y}, \ \forall x, y \geq 0. \tag{3}$$

Preuve.

□

À un certain instant t, l'arbre T de taille n est réduit à un arbre facteur T' de taille n', $2 \leq n' \leq n$, dont les feuilles $v_1, \ldots, v_k, k \geq 2$, ont respectivement les taux de mort suivants $\lambda(v_1), \ldots, \lambda(v_k)$. Soit $L(v_i)$ une v.a. dénotant la durée de vie de v_i, $1 \leq i \leq k$.

Lemme 4.3.1. *Le taux de mort de l'arbre T' est $\lambda(T') = \sum_{i=1}^{k} \lambda(v_i)$.*

Ce lemme signifie que, puisque les $L(v_i)$ sont totalement indépendantes, la probabilité d'une mort dans $[t, t+h]$ est $h \sum_{i=1}^{k} \lambda(v_i) + o(h)$, quand $h \to 0$. Par conséquent, $\lambda(T')$ est considéré comme le taux de parallélisme sur l'arbre facteur T'. Cet arbre a la durée de vie $L(T')$ (dès le moment où il est apparu jusqu'au moment où une de ses feuilles disparaît) $L(T')$ est un v.a. définie par : $L(T') = \min_{1\leq i\leq k} L(v_i)$, puisque la réduction (suppression) suivante sera faite à l'instant $t + \min_{1\leq i\leq k} L(v_i)$. Cette v.a. est caractérisée par la fonction de distribution :

$$Pr(L(T') \leq x) = 1 - Pr(L(T') > x) = 1 - e^{-x\lambda(T')}, \forall x \geq 0. \tag{4}$$

Ainsi $L(T')$ est une v.a. exponentiellement distribuée avec une espérance mathématique égale à $\frac{1}{\lambda(T')}$.

Afin de calculer la probabilité $q_t(v)$ d'élection d'un sommet v dans un arbre T, nous devons additionner les probabilités des transitions en commençant par l'arbre initial T et en se ramenant à la fin à un sommet irréductible v selon le processus aléatoire irréversible présenté. Reconsidérons le cas de l'arbre facteur T' et identifions *l'état du processus* à un certain instant t par l'arbre facteur résiduel à cet instant, la prochaine configuration sera un des arbres facteurs $T_i = T' \setminus \{v_i\}$ obtenus en supprimant une feuille v_i et son arête incidente, pour $i = 1, \ldots, k$. Chacun de ces "états prochains" a une probabilité donnée par :

Proposition 4.3.2. *Soit T' l'état du processus d'élection à un certain instant t. Soient $v_1, \ldots, v_k, k \geq 2$, les feuilles de T'. Alors l'état du processus sera un des arbres T_i obtenus à partir de T' par la suppression de la feuille v_i et de son arête incidente, avec la probabilité de transition :*

$$p(T', T_i) = \frac{\lambda(v_i)}{\lambda(T')} = \frac{\lambda(v_i)}{\sum_{1\leq j\leq k} \lambda(v_j)}, \quad 1 \leq i \leq k. \tag{5}$$

Preuve. Selon la proposition précédente les variables aléatoires $L(v_i)$ sont sans mémoire et, en conséquence, la distribution de la durée de vie supplémentaire à un instant donné t, sujet à l'état de survie à cet instant, est identique à la distribution initiale de la durée de vie. D'autre part les $L(v_i), 1 \leq i \leq k$ sont indépendantes et ainsi,

$$\begin{aligned} p(T', T_i) &= Pr\Big(\min\{L(v_1), ..., L(v_k)\} = L(v_i)\Big) \\ &= Pr\Big(L(v_i) \leq L(v_j) \mid 1 \leq j \leq k\Big) \\ &= \int_0^{+\infty} \lambda(v_i) e^{-\lambda(v_i)x_i} \Bigg(\prod_{\substack{j=1 \\ j\neq i}}^{k} \int_{x_i}^{+\infty} \lambda(v_j) e^{-\lambda(v_j)x_j} dx_j \Bigg) dx_i \\ &= \frac{\lambda(v_i)}{\sum\limits_{1\leq j\leq k} \lambda(v_j)}. \end{aligned}$$

□

Soit T un arbre de taille $n \geq 2$. Considérons la suite d'arbres facteurs, $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \rangle$,

avec $\sigma_1 = T$, tel que chaque σ_i est obtenue à partir de σ_{i-1} par une transition (c'est-à-dire la suppression d'une feuille), pour $2 \leq i \leq n-1$. Il est claire que la probabilité d'une telle suite est définie par

$$p(\sigma) = \prod_{i=2}^{n-1} p(\sigma_{i-1}, \sigma_i).$$

Dénotons par $\Omega(T)$ l'ensemble de toutes ces suites, la probabilité d'être élu pour un sommet donné v est :

$$Pr(\text{élu}(v)) = \sum_{\sigma \in \Omega(T), \sigma_{n-1} = \{v\}} p(\sigma).$$

Pour une $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \rangle \in \Omega(T)$ donnée, sa durée $D(\sigma)$ est la somme des durées de vie des feuilles disparues dans σ. L'espérance mathématique de la durée de processus d'élection, notée $\overline{D}$, est

$$\overline{D} = \sum_{\sigma \in \Omega(T)} p(\sigma) \mathbb{E}(D(\sigma)),$$

où $\mathbb{E}(D(\sigma))$ est l'espérance mathématique de $D(\sigma)$. Dans la section 4.5, nous donnons la durée d'élection pour un exemple particulier du modèle dans le cas des chaînes et des étoiles.

Le processus d'élection dans T est un processus de Markov en temps continu, où l'ensemble des états est l'ensemble des arbres facteurs de T, l'état initial est T et les transitions sont définies par la suppression de feuilles avec des probabilités assignées aux feuilles comme définies ci-dessus. Selon ce point de vue, la proposition suivante permet de représenter mathématiquement tous les paramètres qui semblent être intéressants.

Proposition 4.3.3. *Soit T' un arbre facteur. Soit $P_{T'}(t)$ dénote la probabilité que l'état de l'élection au temps t est T'. Nous avons :*

(i) $\dfrac{dP_T(t)}{dt} = -\lambda(T) P_T(t),$

(ii) *pour tout arbre facteur $T' \neq T$ de taille au moins 2,*

$$\frac{dP_{T'}(t)}{dt} = -\lambda(T') P_{T'}(t) + \sum_{v} \lambda(v) P_{T' \cup \{v\}}(t),$$

où la sommation est effectuée à tous les sommets v adjacents à T' et qui n'appartiennent pas à T', et

(iii) *pour tous les sommets v,* $\dfrac{dP_{\{v\}}(t)}{dt} = \displaystyle\sum_{v' \text{ adjacent à } v} \lambda(v') P_{\{v,v'\}}(t),$

avec la condition initiale $P_T(0) = 1$.

Preuve. Considérons l'évolution du processus dans l'intervalle $[t, t+h]$. Soit T' un arbre facteur de T. Calculons la probabilité de se trouver dans l'état T à l'instant $t+h$.

– Pour $T' \neq T$ et non réduit à une feuille, nous avons :

$$P_{T'}(t+h) = \sum_v P_{T'\cup\{v\}}(t)\pi_{T'\cup\{v\},T'}(h) + P_{T'}(t)\pi_{T',T'}(h) + o(h),$$

où $\pi_{S,T'}(h)$ est la probabilité d'une transition (directe) de S à T' dans un intervalle de temps de longueur h ; la sommation est effectuée à tous les sommets v adjacents à T' et qui ne lui appartiennent pas. Nous obtenons :

$$P_{T'}(t+h) = h\sum_v \lambda(v)P_{T'\cup\{v\}}(t) + P_{T'}(t)[1-\lambda(T')] + o(h).$$

Par conséquent,

$$\frac{P_{T'}(t+h) - P_{T'}(t)}{h} = -\lambda(T')P_{T'}(t) + \sum_v \lambda(v)P_{T'\cup\{v\}}(t) + \frac{o(h)}{h}.$$

Ceci prouve **(ii)**.

– Pour prouver **(i)**, nous remarquons que dans le cas de $P_T(t+h)$, la somme ci-dessus ($\sum_v \dots$) du côté droit disparaît, puisque T n'a aucun arbre facteur qui le précède. Ceci établit **(i)**.

– Pour montrer **(iii)**, il suffit de remarquer que l'état $(\{v\}, \emptyset)$ ou simplement $\{v\}$, est absorbant et, ainsi, $\pi_{\{v\},\{v\}}(h) = 1$. Donc, dans $P_{\{v\}}(t+h)$, le terme négative disparaît. Un calcul simple donne **(iii)**.

□

La dernière proposition est mathématiquement pertinente, puisqu'elle fournit une solution générale pour la probabilité d'être élu et de la durée d'élection en termes de probabilité des états donnés comme solution d'un système d'équations différentielles. Ce genre d'équations est souvent difficile à résoudre, voire même insolvable. Les cas où nous pouvons espérer trouver des solutions sont des cas très simples. Pourtant, même dans ces cas, la résolution nécessite l'emploi de plusieurs techniques pour faire un calcul efficace.

Remarque 4.3.1. Dans le cas où tous les sommets ont le même paramètre assigné λ, la probabilité d'être élu est de type **q**-distribution donnée dans [MS94], puisque selon la proposition 4.3.2, toutes les transitions d'un arbre facteur T' donné ont la même probabilité.

4.4 Modèle d'élection uniforme

Dans la suite, nous étudions un exemple particulier du modèle général. Soit $T = (V, E)$ un arbre. Initialement, tous les sommets ont le même *poids* 1 : $w(v) = 1, \forall v \in V$. Lorsqu'une feuille disparaît, son père récupère son poids, l'ajoutant à son poids courant. Au moment où un sommet v devient une feuille dans un arbre résiduel T', son poids est le nombre de sommets disparus de

ses côtés plus son poids initial ($w_{init}(v) = 1$). La durée de vie du sommet v, selon ce modèle, est une v.a. ayant une distribution exponentielle de paramètre égal à son poids ; $\lambda(v) = w(v)$.

Comme nous allons voir, cette stratégie mène à une élection probabiliste *totalement* "équitable" : dans un arbre de taille n tous les sommets ont la même probabilité $\frac{1}{n}$ d'être élus. Pour illustrer ce qui précède, considérons un exemple.

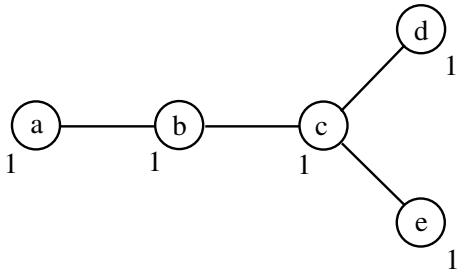


FIG. 2 – Arbre T de taille 5.

Exemple 4.4.1. Soit T l'arbre de la figure 2. Initialement, tous les sommets ont le même poids 1.

Il y a deux suites de transition menant au choix du sommet étiqueté a. Ces suites peuvent être identifiées par les ordres correspondants, $\langle e, d, c, b\rangle$ et $\langle d, e, c, b\rangle$, de suppression de feuilles, Fig 3.

Considérons l'étude du cas de la première suite. Au commencement, il y a 3 feuilles de poids 1 chacune et la probabilité de suppression du sommet $e-$étiqueté est ainsi $\frac{1}{3}$ (proposition 4.3.2). Dans l'arbre réduit il y a maintenant deux feuilles de poids 1 chacune, et la probabilité d'enlever le sommet $d-$étiqueté est $\frac{1}{2}$. Maintenant, dans l'arbre facteur constitué de sommets étiquetés a, b, et c, la feuille $a-$étiquetée (ou tout simplement la feuille a) est de poids 1 alors que la feuille c est de poids 3. Puisque le paramètre λ d'une feuille n'est que son poids, selon la proposition 4.3.2, l'élimination du c est 3 fois plus rapide que celle de a ; elle est donc égale à $\frac{3}{4}$. Le même argument appliqué à l'arbre facteur composé de a et b, assigne la probabilité $\frac{4}{5}$ à l'enlèvement du sommet b. Par conséquent, la probabilité de la première suite est :

$$Pr(\langle e, d, c, b\rangle) = \frac{1}{3}\frac{1}{2}\frac{3}{4}\frac{4}{5} = \frac{1}{10}.$$

Clairement, la deuxième suite $\langle d, e, c, b\rangle$ a la même probabilité que la première, en raison de la symétrie de d et e, et donc la probabilité d'élire le sommet $a-$étiqueté est :

$$Pr(\text{élu}(a)) = Pr(\langle e, d, c, b\rangle) + Pr(\langle d, e, c, b\rangle) = \frac{1}{5}.$$

Un calcul semblable prouve que les autres sommets de l'arbre ont la même probabilité $\frac{1}{5}$ d'être élus.

La méthode de calcul présentée dans l'exemple est une méthode directe, elle devient plus lourde lorsqu'il s'agit d'un graphe de taille plus grande. En effet, il s'avère très difficile d'énumérer le nombre de suites et par ailleurs de calculer la probabilité de chacune d'elles.

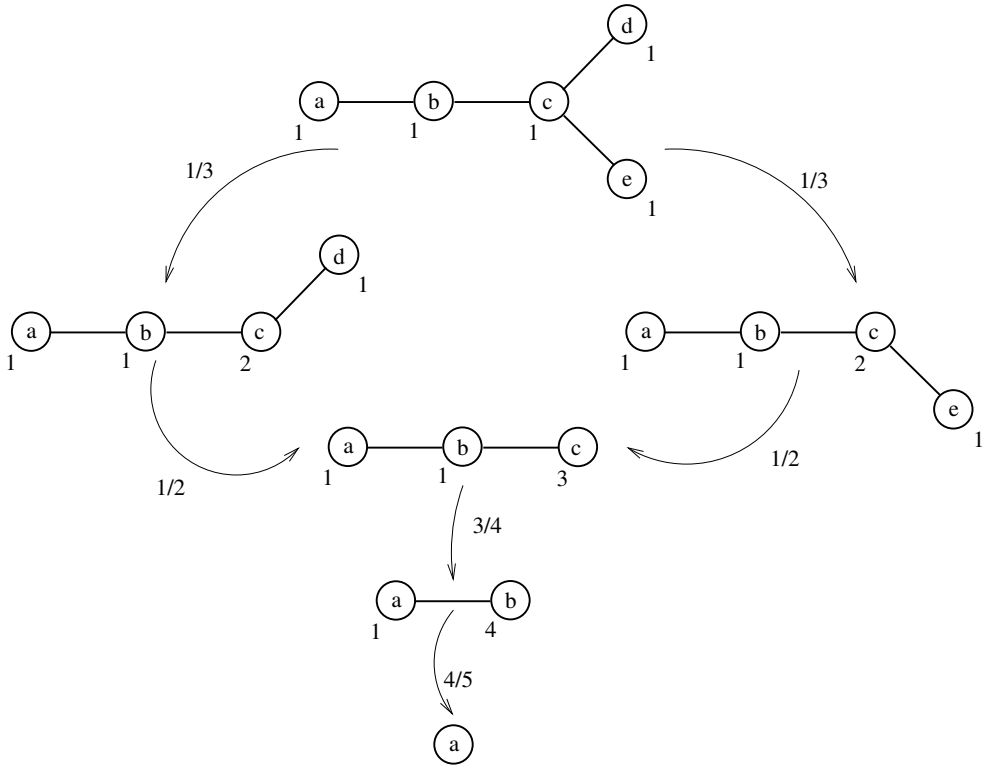


FIG. 3 – Suites d'élection du sommet étiqueté a.

Par conséquent, pour prouver l'affirmation principale d'uniformité, nous introduisons une légère modification du modèle présenté dans cette section. Il se peut que la preuve suivante ne soit pas la plus courte. Mais elle a cependant l'avantage de traduire le modèle dans une variante sur les arbres orientés.

Rappelons ici qu'une *forêt* est un ensemble de paires d'arborescences disjointes. Dans une arborescence ou dans une forêt une feuille désigne un sommet sans fils (successeur).

Soit $F = (V, A)$ une forêt d'arborescences. Nous considérons un processus d'élection probabiliste sur la forêt F comme suit.

À tout moment de l'intervalle $[t, t + h]$, toute feuille v peut disparaître avec une probabilité proportionnelle à son poids (initialement tous les sommets sont de poids 1). Si une feuille qui devrait disparaître a un père (c'est-à-dire un sommet prédécesseur) alors elle communique son poids à son père. Le père augmente son poids par celui du fils disparu. Le processus continue sur la forêt réduite, obtenue à partir de F par la suppression de la feuille qui devrait disparaître avec son arc incident (s'il existe), jusqu'à la disparition totale de la forêt.

Puisque les graphes considérés ne sont plus connexes, nous employons de préférence des termes tels que le processus d'élimination (ou *le processus de mort*), et le sommet *gagnant* est le dernier sommet qui survit.

Exemple 4.4.2. Considérons la forêt de la Fig. 4 et calculons la probabilité que x batte tous les autres sommets de la forêt.

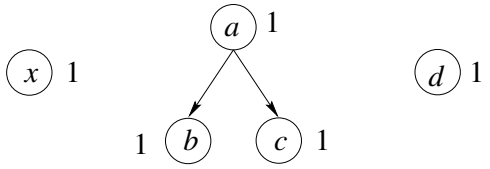


FIG. 4 – Forêt d'arborescences.

Cet événement est la conséquence d'une suite de suppression parmi les huit suites de suppression de feuilles, à savoir : $\langle b, c, a, d\rangle$, $\langle b, c, d, a\rangle$, $\langle b, d, c, a\rangle$, $\langle d, b, c, a\rangle$, avec la possibilité d'échanger b et c, c-à-d., les quatre dernières suites sont obtenues en permutant les positions de b et c dans les quatre suites précédentes. Considérons la première suite $\langle b, c, a, d\rangle$. Étant donné qu'initialement il y a 4 feuilles de poids 1 chacune, la suppression de b est de probabilité $\frac{1}{4}$. Dans la forêt réduite la probabilité de l'élimination de c est $\frac{1}{3}$. Maintenant, dans la forêt restante il y a 3 feuilles ; x et d ont un poids de 1 tandis que a est de poids 3. Par conséquent, la probabilité de suppression de a est $\frac{3}{5}$. Il reste finalement x et d de poids 1 chacun et la probabilité pour que x survit plus que le sommet d devient $\frac{1}{2}$. La probabilité de cette suite est ainsi $\frac{1}{4}\frac{1}{3}\frac{3}{5}\frac{1}{2} = \frac{1}{40}$. Les sept autres probabilités sont calculées, donnant de la même manière, attribuant ainsi à tout sommet x la probabilité $\frac{1}{5}$ de battre les autres sommets.

Comme nous le verrons plus loin, ce nouveau processus de mort, lié au modèle uniforme d'élection, a l'avantage d'admettre un raisonnement inductif. Nous pouvons écrire un système d'équations différentielles qui représente l'état du processus d'élection comme nous l'avons fait

dans la proposition 4.3.3 pour le modèle général sur des arbres. Cependant, nous ne continuerons pas dans cette direction dans le présent chapitre, mais nous reviendrons sur cet aspect dans le chapitre qui suit sur les $k-$arbres, où un arbre n'est autre qu'un $1-$arbre. Dans la suite de cette étude, nous cherchons à calculer la distribution du temps d'attente pour la disparition d'une forêt. Pour atteindre cet objectif, nous montrons que la probabilité qu'une forêt donnée soit encore en vie ne dépend que de sa propre taille et non de sa structure. Nous dérivons finalement la propriété d'uniformité du modèle présenté dans cette section.

Soit une forêt F, notons par $W(F)$ le temps nécessaire à cette forêt pour disparaître ; c'est une v.a. à valeurs réelles positives. Pour deux forêts F_1 et F_2, F_1 *survit* plus que F_2 (ou F_1 *bat* F_2) si $W(F_1) > W(F_2)$.

La proposition suivante est pertinente, elle affirme que la distribution de $W(F)$ dépend seulement de la taille de F.

Proposition 4.4.1. *Soit F une forêt de taille n alors la fonction de distribution $G_F(t)$ de la v.a. $W(F)$ est donnée par :*

$$G_F(t) = Pr(W(F) \leq t) = (1 - e^{-t})^n, \quad \forall t \geq 0. \tag{6}$$

Preuve. Par récurrence sur n. Si F est réduit à un sommet, alors la proposition est vraie (la durée de vie pour un seul sommet est une v.a. exponentiellement distribuée de paramètre 1). Supposons que la proposition est vraie pour une forêt de taille inférieure à n et prouvons qu'elle reste vraie pour une forêt F de taille n ($n \geq 2$).

(i) Supposons que F est constituée de forêts $F_1, F_2, \cdots, F_k$ avec $k \geq 2$. Soit $n = n_1 + n_2 + \cdots + n_k$, où n_i est la taille de la forêt F_i, $1 \leq i \leq k$.Dans ce cas $W(F_i)$, $1 \leq i \leq k$, sont des v.a. mutuellement indépendantes et ainsi par hypothèse de récurrence, nous avons :

$$\begin{aligned} Pr(W(F) \leq t) &= \prod_{i=1}^{k} Pr(W(F_i) \leq t) \\ &= \prod_{i=1}^{k} (1 - e^{-t})^{n_i} \\ &= (1 - e^{-t})^n. \end{aligned}$$

(ii) Autrement, F est de taille n. Elle se compose d'une racine unique r et des arborescences $A_1, A_2, \cdots, A_k$. Soit F' une forêt composée de $A_1, A_2, \cdots, A_k$. Ainsi, F' est de taille $n-1$ et par hypothèse d'induction, $W(F')$ a une fonction de distribution $(1 - e^{-t})^{n-1}$. Toutefois, $W(F)$ est la somme de deux v.a. indépendantes : $W(F')$ et la durée de vie de la racine r. Cette dernière est une v.a. exponentielle de paramètre n (poids de r). Donc, $W(F)$ a une fonction de distribution (voir [Fel66], p. 142. Théorème 2) donnée par :

$$G_F(t) = \int_0^t G_{F'}(t - x) d(1 - e^{-nx}).$$

Par conséquent,

$$\begin{aligned} G_F(t) &= \int_0^t n[1-e^{-(t-x)}]^{n-1}e^{-nx}dx \\ &= \int_0^t n[e^{-x}-e^{-t}]^{n-1}e^{-x}dx \\ &= [-(e^{-x}-e^{-t})^n]_{x=0}^{x=t} \\ &= (1-e^{-t})^n. \end{aligned}$$

□

Nous prouvons d'abord quelques résultats "d'uniformité" dans un processus d'élimination sur une forêt. La proposition suivante affirme que dans une forêt, la probabilité qu'un sommet isolé batte tous les autres sommets de la forêt est égale à l'inverse de la taille de la forêt, et cela quelque soit la structure de la forêt.

Proposition 4.4.2. *Soit F une forêt de taille $n-1, n \geq 2$ et x un sommet isolé n'appartenant pas à F. Alors la probabilité que x batte F (dans la forêt F' composée de F et x) est égale à $\frac{1}{n}$.*

Preuve. Par induction sur n. Pour $n = 1$ et $n = 2$, la proposition est évidente. Supposons qu'elle soit vraie pour des forêts de tailles inférieures à $n-1$ et prouvons qu'elle reste vraie pour une forêt de taille $n-1$. Soit F une telle forêt. Nous considérons les deux cas suivants.

(i) F est une arborescence avec une racine r et des sous-arborescences $A_1, ..., A_k$ de tailles $n_1, ..., n_k$ respectivement. Le sommet x batte l'arborescence F si et seulement si :

- x bat $A_1, ..., A_k$. Appelons cet évènement E_1, et
- une fois que $A_1, ..., A_k$ disparaissent (dans ce cas la racine r devient active), x bat r - appelons cet évènement E_2.

Par conséquent, la probabilité $q(x;F)$ que x bat F est donnée par $Pr(E_1)Pr(E_2|E_1)$. Le premier facteur est par induction $\frac{1}{\sum_{i=1}^k n_i+1}$ et le second facteur, selon la proposition 4.3.2 (avec $\lambda(x) = 1$ et $\lambda(r) = n-1$), est

$$\frac{\sum_{j=1}^k n_j + 1}{\sum_{j=1}^k n_j + 2}.$$

Ceci établit $q(x;F) = 1/n$ (puisque $\sum_{j=1}^k n_j + 2 = n$).

(ii) Supposons maintenant que F se compose des arborescences $A_1, ..., A_k$ de tailles $n_1, ..., n_k$ respectivement. Alors x bat F si et seulement s'il bat tous les A_i. Mais les A_i se comportent *indépendamment* et, d'ailleurs, par l'hypothèse d'induction, la probabilité $q(x;A_i)$ que x bat A_i ne dépend que de la taille k_i de A_i. Par conséquent, $q(x;F)$ est pareille que $q(x;F')$; où F' est composée de tous les $n-1$ sommets et l'ensemble *vide* d'arcs. La dernière probabilité est clairement égale à $1/n$.

□

En outre, il est possible d'étendre la proposition à deux forêts concourantes. Soient deux forêts (d'arborescences) F et G de tailles m et n ($m, n \geq 1$) respectivement, nous considérons le processus de suppression de feuilles dans l'union des deux forêts $F \cup G$. Nous disons que F *bat* G si au moins un sommet de F survit encore après la disparition totale de G.

Proposition 4.4.3. *Soient F et G deux forêts comme ci-dessus. Alors la probabilité que F batte G est $\frac{m}{m+n}$.*

Preuve. Si une des forêts concourantes est réduite à un seul sommet, la proposition coïncide avec la proposition 4.4.2. La propositions affirme que pour deux forêts données la probabilité du gain est proportionnelle aux tailles des deux forêts et ne dépend pas de leurs liens (arcs). En particulier, puisque la proposition est valide quand les forêts sont réduites à m et n sommets isolés, la proposition est équivalente à l'affirmation que la probabilité que F bat G est identique si nous enlevons tous les arcs des forêts. Nous pouvons prouver la proposition par une récurrence sur $m + n$. Pour $m + n = 2$, la proposition est bien évidemment vérifiée. Supposons qu'elle soit vraie pour $m + n < h$ et prouvons qu'elle l'est aussi pour $m + n = h$.

Soit F et G deux forêts de tailles m et n, avec $m+n = h$. Si une des deux forêts concourantes a au moins deux arbres, alors la proposition est vraie, et ce parce que chaque composante connexe se comporte *indépendamment* et que la proposition est vérifiée sur les composantes. En outre, selon l'hypothèse de récurrence, la probabilité de l'événement impliqué est la même à chaque fois que des arcs sont supprimés. Par conséquent, sa probabilité est identique à celle des forêts sans arcs, qui est $\frac{m}{h}$.

Maintenant supposons que F et G soient deux arborescences connexes enracinées en r et s respectivement. Si l'une d'elles est réduite à un sommet, la proposition devient évidente. Ainsi, soient F une forêt qui se compose de la racine r et des sous-arborescences $F_1, ..., F_k$ et G de s et des sous-arborescences $G_1, ..., G_l$. Selon l'hypothèse d'induction, la probabilité que $F_1, ..., F_k$ battent G est $\frac{m-1}{m-1+n}$ et la probabilité que $G_1, ..., G_l$ battent F est $\frac{n-1}{m+n-1}$. Par conséquent, la probabilité du "duel" entre r et s est $1 - \left(\frac{m-1}{m-1+n} + \frac{n-1}{m+n-1}\right) = \frac{1}{m+n-1}$.

D'autre part, la probabilité du gain pour r dans un duel contre s est $\frac{n}{m+n}$ (rappelons que la probabilité de l'enlèvement pour un sommet est proportionnelle à son poids, qui est m pour r et n pour s). Or, la probabilité que F batte G est la somme des probabilités de $F_1, ..., F_k$ battant G et le produit de probabilité d'un duel entre r et la probabilité du gain sous la dernière assomption. Par conséquent,

$$\begin{aligned} Pr(F \text{ bat } G) &= \frac{m-1}{m-1+n} + \frac{1}{m+n-1} \times \frac{n}{m+n} \\ &= \frac{m}{m+n}. \end{aligned}$$

□

Remarque 4.4.1. Retournant aux cas des arbres non orientés, nous devons préciser que l'équivalence des propositions précédentes n'a pas lieu pour des arbres. En effet, soit T un arbre de taille n et x un sommet isolé n'appartenant pas à cet arbre, si nous considérons le processus d'élection, comme présenté dans la section (chaque feuille a une durée de vie qui est une v.a. exponentiellement distribuée de paramètre $\lambda(v) = w(v)$), la probabilité d'être élu pour le sommet isolé x *n'est pas* $\frac{1}{n+1}$. En fait, elle est conjecturé être $\frac{2(n-1)}{n^2}$. Il convient de noter que dans ce cas, il est supposé que la durée de vie supplémentaire est générée selon le dernier poids, puisqu'à chaque étape le poids d'une feuille peut changer (une fois qu'elle devient isolée).

Théorème 4.4.1. *Soit T un arbre de taille n et soit v un sommet de T. Si $q_T(v)$ dénote la probabilité d'élire v dans T selon le modèle uniforme, alors $q_T(v) = \frac{1}{n}$.*

Preuve. Si $n = 1$ ou $n = 2$ alors le théorème est vrai. Sinon, supposons que $v_1, \ldots, v_k$ sont les sommets adjacents à v. Soient $A_1, \ldots, A_k$ les arborescences adjacentes enracinées en $v_1, \ldots, v_k$ de tailles $n_1, \ldots, n_k$ respectivement. Alors, le sommet v est battu dans T si et seulement si une des

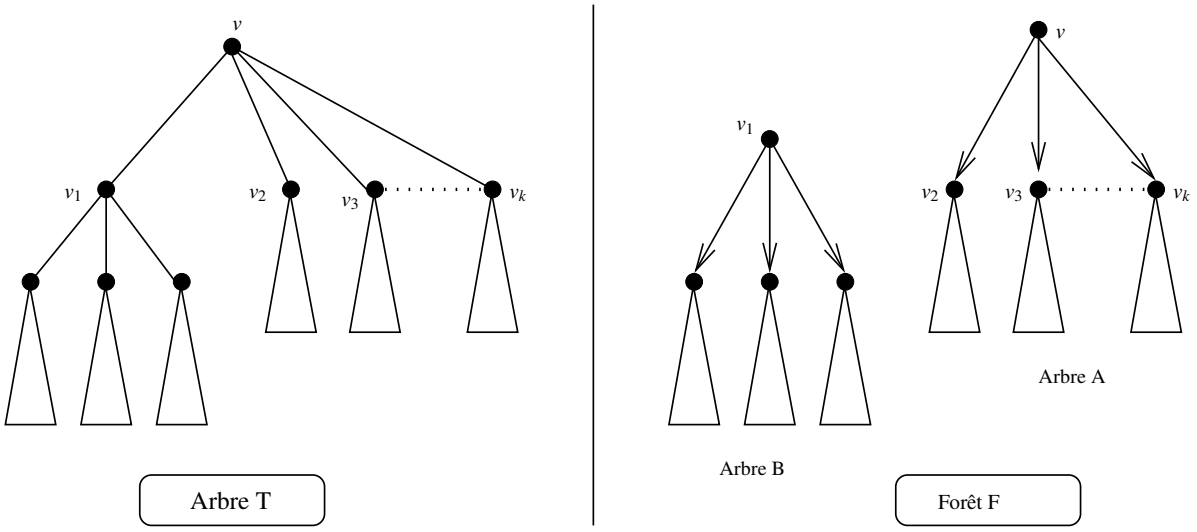


FIG. 5 – Processus ascendant de mort sur une forêt d'arbres enracinés

racines v_i de A_i bat v dans l'arborescence résiduelle enracinée en v les sommets qui ne sont pas dans A_i - appelons cette arborescence B_i - (pour un certain $i \in \{1, \ldots, k\}$). Or, ces événements sont deux à deux disjoints pour des i différents. Par exemple, dans la Fig. 5, la probabilité que le sommet v soit enlevé avant l'arbre facteur du côté de v_1 (c'est-à-dire l'arbre non orienté B) dans le processus d'élection sur T est la même probabilité que v_1 batte v dans la forêt F.
Par conséquent,

$$1 - q_T(v) = \sum_{i=1}^{k} q(v_i; \{A_i, B_i\}). \tag{7}$$

Selon la dernière proposition, $q(v_i; \{A_i, B_i\})$ est égale à $\frac{n_i}{n}$ et, ainsi, $1 - q_T(v) = \frac{n-1}{n}$, puisque $\sum_{i=1}^{k} n_i = n - 1$. □

4.5 Durée d'élection dans le modèle uniforme

Un paramètre intéressant qui vient juste après la distribution de probabilités sur les sommets est la *durée d'élection*. Elle peut être considéré comme une mesure de la complexité en temps. En effet, il s'agit du temps d'absorption pour le processus markovien en temps continu considéré dans la section 4.3.

Théoriquement, la proposition 4.3.3 détermine la probabilité d'être dans un état T' (rappelons que T' est un arbre facteur de l'arbre initial) au temps t. Mais, la problématique qui se pose est que le système d'équations différentielles produisant ces probabilités est tout à fait difficile à résoudre. L'étude semble intéressante puisqu'elle doit être commune à de grandes classes des processus de mort pure.

Nous considérons ici deux exemples très spéciaux du problème d'élection uniforme pour lequel un calcul élémentaire fournit la probabilité d'être dans l'état T'.

Les résultats présentés dans cette section sont simples. En effet, la motivation est la présentation de la conjecture suivante. Bien que, rien ne soit connu actuellement au sujet de la durée d'élection et de son espérance dans les arbres en général, les résultats sur l'espérance mathématique de la durée d'élection dans les chaînes et dans les étoiles suggèrent que :

Conjecture 4.5.1. *La durée d'élection moyenne dans le modèle uniforme est approximativement logarithmique en la taille de l'arbre.*

Remarque 4.5.1. Dans la pratique, nous avons simulé le processus d'élection uniforme dans des arbres dont la structure est plus complexe. Les résultats de la simulation laissent croire que cette conjecture est solide.

4.5.1 Durée d'élection dans une chaîne

Nous reconsidérons le modèle uniforme présenté dans la section précédente. Nous supposons que l'arbre T dans lequel le processus d'élection a lieu est une chaîne de longueur n. Pour un nombre entier positif $m \leq n$, soit $P_m(t)$ la probabilité qu'au temps t la chaîne résiduelle a la taille m. Il est important de noter que cette identification d'états diffère de celle utilisée dans le contexte de la proposition 4.3.3 : l'état du processus au temps t a été initialement défini par l'arbre facteur résiduel alors qu'ici c'est par sa taille. Nous avons :

Théorème 4.5.1. *Sous les hypothèses précédentes, nous avons :*

(i) $P_{n-k}(t) = (k+1)e^{-(k+2)t}(e^t - 1)^k$, *si* $\quad 0 \leq k \leq n-2$ *et*

(ii) $\frac{dP_1(t)}{dt} = (n-1)e^{-nt}(e^t - 1)^{n-2} = n(n-1)e^{-2t}(1 - e^{-t})^{n-2}$.

Preuve. Si nous identifions l'état du processus de Markov avec la taille de la chaîne résiduelle, toute chaîne de taille $n-k$, $1 \leq k \leq n$, est le résultat d'une chaîne de taille $n-k+1$ et produit à son tour une chaîne de taille $n-k-1$. Le même fait se tient pour une chaîne de taille n et pour

une chaîne de taille 1. Il est important de noter que la première chaîne n'a aucune chaîne qui la précède, et qu'il n'y a aucune chaîne qui succède la seconde chaîne.

En utilisant la même technique que dans la proposition 4.3.1, nous obtenons :

(a) $\frac{dP_n(t)}{dt} = -2P_n(t)$,

(b) $\frac{dP_{n-k}(t)}{dt} = -(k+2)P_{n-k}(t) + (k+1)P_{n-(k-1)}(t)$, pour $1 \leq k \leq n$, et

(c) $\frac{dP_1(t)}{dt} = nP_2(t)$,

avec la condition initiale $P_n(0) = 1$. La première équation vérifiant la condition initiale produit $P_n(t) = e^{-2t}$, qui affirment que **(i)** se tient pour $k = 0$. Supposons que la clause **(i)** est vraie pour $k = l$ et prouvons qu'elle reste vraie pour $k = l + 1$ quand $l + 1 \leq n - 2$. Ainsi, d'après **(b)**, nous avons :

$$\frac{dP_{n-(l+1)}(t)}{dt} = -(l+3)P_{n-(l+1)}(t) + (l+2)(l+1)e^{-(l+2)t}(e^t - 1)^l.$$

La dernière équation linéaire se résout facilement, produisant :

$$P_{n-(l+1)}(t) = (l+2)e^{-(l+3)t}(e^t - 1)^{l+1}.$$

Ceci établit **(i)**. Finalement, pour dériver **(ii)**, il suffit de substituer $P_2(t)$ par $(n-1)e^{-nt}(e^t - 1)^{n-2}$ dans **(c)**. □

Soit D_n la durée d'élection dans une chaîne de taille n. Elle est une v.a. à valeurs réelles positives qui sont déterminées par **(c)** dans le théorème précédent. En effet, nous avons :

Corollaire 4.5.1. *La durée d'élection D_n est une v.a. de type Bêta avec la fonction de densité*

$$f(t) = \begin{cases} n(n-1)e^{-2t}(1-e^{-t})^{n-2} & \textit{pour } t \geq 0 \\ 0 & \textit{autrement.} \end{cases}$$

Son espérance mathématique est $H_n - 1 = \sum_{i=2}^{n} \frac{1}{i}$, où H_n est le $n^{ème}$ nombre harmonique.

Preuve. Nous avons $Pr(D_n \leq t) = P_1(t)$ pour $t \geq 0$. Ainsi, D_n a une fonction de distribution qui est la solution de **(ii)** dans le théorème précédent. Elle a, en conséquence, la fonction de densité $f(t)$. En introduisant la substitution $x = e^{-t}$, avec $x \in [0, 1]$, nous obtenons une nouvelle fonction de densité $g(x) = n(n-1)(1-x)^{n-2}x$. C'est un bêta densité (voir [Fel50], Chapitre VI)

$$\beta_{n-1,2}(x) = \frac{\Gamma(n+1)}{\Gamma(n-1)\Gamma(2)}(1-x)^{n-2}x, \ x \in [0, 1].$$

Un calcul standard, mais long, donne l'espérance mathématique de D_n. Une méthode simple pour le calculer est d'additionner les espérances mathématiques des temps de transitions d'une chaîne de taille i à une chaîne de taille $i - 1$, pour $i = n, n-1, \ldots, 2$. Puisque le taux de la transition $i \to i - 1$ est $n - i + 2$, le temps moyen de cette transition est $\frac{1}{n-i+2}$. La somme de ces fractions étant $H_n - 1$. □

4.5.2 Durée d'élection dans une étoile

Comme deuxième exemple à présenter dans cette étude, nous considérons la durée d'élection du modèle uniforme dans une étoile G de taille n. Les étoiles et les chaînes constituent deux exemples extrêmes d'arbres. C'est pourquoi, l'étude de ces deux cas très particuliers semble intéressante.

En utilisant la même notation que dans le cas des chaînes, nous avons :

Théorème 4.5.2. *Pour un entier k donné, $1 \leq k \leq n$, la probabilité $P_k(t)$ d'être dans l'état k (c'est-à-dire une étoile de taille k) est*

$$P_k(t) = \binom{n-1}{k-1}(1-e^{-t})^{n-k}e^{-(k-1)t}.$$

Preuve. Un calcul semblable à celui que nous avons vu dans le cas des chaînes donne :

(i) $\frac{dP_n(t)}{dt} = -(n-1)P_n(t)$,

(ii) pour $2 \leq k \leq n$, $\frac{dP_k(t)}{dt} = -(k-1)P_k(t) + kP_{k+1}(t)$ et

(iii) $\frac{dP_1(t)}{dt} = 2P_2(t)$,

avec la condition initiale $P_n(0) = 1$.

Maintenant, un calcul direct employant une induction décroissante sur k prouve le théorème.

□

Le théorème caractérise en particulier la durée d'élection uniforme D_n dans une étoile de taille n :

Corollaire 4.5.2. *La v.a. D_n a une distribution qui admet la fonction de distribution suivante :*

$$f(t) = \begin{cases} (n-1)(1-e^{-t})^{n-2}e^{-t} & \text{pour } t \geq 0 \\ 0 & \text{autrement.} \end{cases}$$

Son espérance mathématique est $H_n - 1 = \sum_{i=2}^{n} \frac{1}{i}$.

Preuve. Clairement, D_n a une fonction de distribution $P_n(t)$, donnée par le théorème ci-dessus. Par conséquent, elle admet la fonction de densité $f(t)$. La même technique utilisée pour les chaînes peut être appliquée pour obtenir l'espérance mathématique $H_n - 1$. □

Remarque 4.5.2. Il est également possible d'employer une méthode plus directe pour prouver le dernier théorème et son corollaire. En effet, la probabilité d'avoir une étoile de taille k au temps t est égale à la probabilité qu'à cet instant, il y a exactement $n - k$ feuilles de l'étoile initiale qui ont disparues, et puisqu'elles se comportent indépendamment, cet événement a la probabilité

$$\binom{n-1}{n-k}(1 - e^{-t})^{n-k} e^{-(k-1)t},$$

qui est la même expression que celle de $P_k(t)$, donnée par le théorème 4.5.2.

□

Remarque 4.5.3. Nous avons vu dans le cas du processus de mort sur des forêts (avec une élimination ascendante) que le temps d'absorption W est le maximum de n v.a. indépendantes et exponentiellement distribuées. Il est important de noter que W inclut le temps d'attente pour faire disparaître le dernier sommet (ou le sommet élu). Bien que D et W ne soient pas identiques, les cas étudiés donnent les fonctions de distribution pour D qui sont "semblables" à celles de W. Il y a peut-être, en général, une simple caractéristique commune pour la distribution de la durée d'élection.

Chapitre 5

Élection uniforme dans les $k-$arbres

Sommaire

Dans ce chapitre, nous partons de l'algorithme d'élection uniforme dans les arbres et nous avons essayé de généraliser la notion d'élection uniforme dans une classe de graphes plus grande que celle des arbres. Nous nous sommes, ainsi, intéressés aux *$k-$arbres* ; graphes caractérisés par un *ordre d'élimination simplicial*. Comme nous avons vu dans la section 3.2 du chapitre 3, le problème d'élection peut être étudié sous plusieurs considérations. Godard et Métivier dans [GM02] prouvent qu'il n'existe pas d'algorithme d'élection déterministe pour résoudre ce problème dans certaines classes de graphes (à savoir, graphes étiquetés qui ne sont pas minimaux dans la relation de revêtement). Nous présentons et analysons un algorithme distribué probabiliste d'élection uniforme dans un réseau où les processeurs sont organisés en une topologie de $k-$arbre. Cet algorithme est *totalement* équitable dans la mesure où il donne la même chance d'être élu à tout sommet, quelque soit sa position dans le $k-$arbre.

L'organisation de ce chapitre est la suivante. Dans la section 5.2, nous présentons le modèle et les hypothèses utilisés dans cette étude. La section 5.3 donne une description de notre algorithme probabiliste d'élection uniforme. La section 5.5 est consacrée à un processus de Markov

modélisant le processus d'élection probabiliste. Finalement, dans la section 5.6, nous étudions l'espérance mathématique de la durée d'élection.

5.1 Introduction aux $k-$arbres

Les $k-$arbres, présentés pour la première fois par Rose dans [Ros74] sous ce nom, sont des exemples spéciaux de graphes d'élimination parfaite. Dans la littérature, ces graphes correspondent à une généralisation $k-$dimensionnelle naturelle des arbres. Ainsi, les arbres constituent un cas spécial de cette classe de graphes.

5.1.1 Construction d'un $k-$arbre

Soit k un nombre entier positif fixe. Un *$k-$arbre* est défini récursivement sur sa taille[1] n comme suit.

- Un graphe complet K_k de taille k est un $(k-1)-$arbre. Il est aussi appelé une *$k-$clique*.
- Un $k-$arbre de taille $n+1$ est obtenu à partir d'un $k-$arbre de taille n en ajoutant un nouveau sommet et en le reliant à tous les sommets d'une $k-$clique du $k-$arbre.

D'après cette définition, nous remarquons qu'un arbre est un $1-$arbre.

Formellement, un graphe $G = (V, E)$ est un $k-$arbre si et seulement si il existe un ordre total $(x_1, x_2, \ldots, x_n)$ sur V tel que $x_1 x_2 \ldots x_k$ est une clique et pour tout i $(i = k+1, \ldots, n)$ le sommet x_i est de degré k et appartient à une unique $(k+1)-$clique du sous-graphe induit $G_i = G_{\{x_1, x_2, \ldots, x_i\}}$.

Dans la suite du chapitre, nous avons besoin de la définition suivante :

Définition 5.1.1. Dans un $k-$arbre G, un sommet v est appelé *simplicial* si son degré est égal à k.

D'après cette définition, il est facile de remarquer qu'un $k-$arbre de taille $|V| \geq 2$ a au moins deux sommets simpliciaux et que chacun d'eux est lié à une $k-$clique.

Dans le $2-$arbre de la figure 6 les sommets noirs sont simpliciaux ; chacun d'eux est de degré 2.

5.1.2 Quelques propriétés des $k-$arbres

Les $k-$arbres surgissent dans l'étude de quelques systèmes linéaires. Ils ont de nombreuses propriétés. Nous faisons ici un survol rapide de quelques unes de ces propriétés.

Proposition 5.1.1. *Soit $G = (V, E)$ un $k-$arbre de la taille $|V| = n$. Le graphe G a $\frac{k(2n-k-1)}{2}$ arêtes et a au moins deux sommets simpliciaux. De plus, le nombre de ses $(k+1)-$cliques est $n-k$.*

[1] nombre de sommets

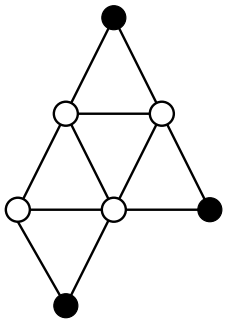


Fig. 6 – Exemple d'un 2−arbre.

Un $k-$arbre est un graphe triangulé [GHP95], c'est-à-dire que pour tout cycle de longueur supérieure à 3, il existe une arête joignant deux sommets non consécutifs de ce cycle. Cette arête est appelée *corde*.

Remarquons tout d'abord que le graphe de la figure 7 n'est pas un graphe triangulé. En effet, ce graphe possède le cycle $[a, b, c, d, e, a]$ qui est sans corde.

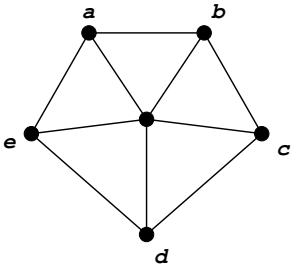


Fig. 7 – Un graphe non triangulé.

Une propriété caractéristique des graphes triangulés est qu'ils sont des graphes à élimination parfaite (*perfect elimination graphs*). Les sommets d'un tel graphe G, de taille n, peuvent être étiquetés de 1 à n de sorte que, pour tout i, le sommet i et ses sommets adjacents forment une clique C_i du sous-graphe G_i induit par les sommets $i, \ldots, n$ (l'ordre $1, \ldots, n$ est alors appelé un ordre d'élimination) Rose [Ros74] reprend la définition précédente des $k-$arbres et il a remarqué qu'ils constituent une classe particulière des graphes à élimination parfaite pour laquelle $|C_i| = k + 1$, pour $i = 1, \ldots, n - k - 1$. Les $k-$arbres ne sont souvent mentionnés dans la littérature que comme des graphes à élimination parfaite. Idury et Schäffer [IS93] ont prouvé que la propriété de planarité des 3−arbres n'est pas vraie en général.

Pour une étude détaillée sur les $k-$arbres, on peut consulter les ouvrages [BLS99, GHP95, GY04, Ros74, Lec02].

5.1.3 Arbres de cliques et graphes de cliques

Un arbre de cliques [Bod93] (ou un arbre de décomposition [GH01]) du graphe $G = (V, E)$ se compose d'un arbre $T = (I, F)$ et d'une famille de sous-ensembles $\{X_i | i \in I\}$ de V tel que :

1. $\bigcup_{i \in I} X_i = V$,
2. pour toute arête $\{u, v\} \in E$, il existe un sommet $i \in I$ tel que $u, v \in X_i$, et
3. pour tout $i, j, k \in I$, si j est sur le chemin de i à k dans I, alors $X_i \cap X_k \subseteq X_j$.

Si G est un $k-$arbre, alors il existe un arbre de cliques de G tel que $\forall i \in I, |X_i| = k + 1$. La figure 8 illustre un $2-$arbre et son arbre de cliques associé.

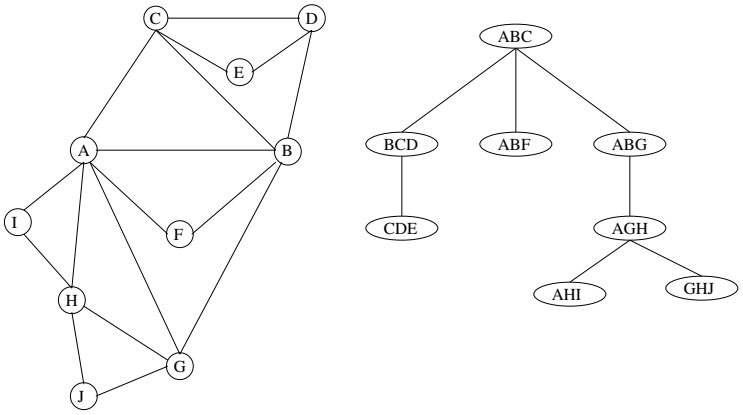


FIG. 8 – Un $2-$arbre et son arbre de cliques associé.

Lemme 5.1.1. *[GH01] Soit* $(\{X_i | i \in I\}, T = (I, F))$ *l'arbre de cliques du* $k-$*arbre* $G = (V, E)$*, et soit* $W \subseteq V$ *une clique dans* G*. Alors, il existe un* $i \in I$ *avec* $W \subseteq X_i$*.*

5.2 Modèle et hypothèses

Dans le modèle d'élection probabiliste dans les arbres vu au chapitre précédent, l'algorithme introduit est basé sur l'utilisation d'un délai aléatoire associé à tout sommet supprimable (les sommets feuilles). Nous avons vu que les délais sont indépendants et peuvent être localement générés par les sommets au fur et à mesure qu'ils deviennent feuilles. Nous avons également présenté un modèle particulier dans lequel chaque sommet a une chance égale à celles des autres sommets de l'arbre d'être élu. Par la suite, nous allons nous en servir du même modèle pour décrire notre algorithme d'élection dans les $k-$arbres.

Nous supposons toujours que les graphes (ou réseaux, selon la description) considérés sont anonymes, c'est-à-dire pour tout sommet du graphe, les identités des autres sommets lui sont inconnues. Ce fait implique que : initialement, dans le graphe, tous les sommets doivent être dans le même état, celui d'être candidat (comme le modèle standard d'élection [Ray88, Tel00]).

L'idée de base de l'algorithme est que les sommets simpliciaux jouent le rôle des feuilles dans l'arbre. Initialement, tous les sommets du graphe sont candidats. Cependant, seuls les sommets simpliciaux sont actifs et peuvent générer une durée de vie alétoire. Un sommet simplicial dont la durée de vie est écoulée disparaît et, de proche en proche, on finit par n'avoir qu'un seul sommet dans le graphe, c'est le sommet élu.

La difficulté sera choisir le sommet auquel sera transmis le poids d'un sommet simplicial qui disparaît.

5.3 Algorithme d'élection dans les $k-$arbres

L'identité d'un sommet est souvent prise comme une adresse pour envoyer des messages aux autres sommets. Dans un graphe anonyme, les identités des sommets ne sont pas disponibles pour les distinguer. Cependant, il est possible que les sommets soient identifiés par l'intermédiaire d'adressage indirect ([Tel00] section 2.4.4), où chaque sommet distingue ses voisins par des noms localement connus de ses canaux (ports).

L'algorithme asynchrone d'élection, présenté dans cette section, est conçu pour des réseaux anonymes ayant une topologie de $k-$arbres. La seule connaissance disponible aux sommets est la taille de la clique de base[2], c'est-à-dire k.

Notre algorithme 2 est divisé en deux tâches : la première, appelée *decouvrirArbre*, sert à calculer un arbre couvrant, tandis que la seconde, *elire*, est une suite aléatoire d'élimination de sommets simpliciaux le long de l'arbre couvrant, laissant à la fin un sommet unique. Il convient de noter que l'algorithme d'élection peut être fait en une seule étape "transversale" sans calcul préliminaire. Cependant, la présente conception de l'algorithme en deux étapes semble plus facile à analyser. Ainsi, l'élection dans un $k-$arbre est transformée en une élection dans l'un de ses arbres couvrants.

Description de l'algorithme

Dans un premier temps, nous étudions les conditions fondamentales de la construction d'un arbre couvrant d'un $k-$arbre $G = (V, E)$. Des poids sont affectés aux sommets. La construction d'un arbre couvrant revient à choisir, parmi tous les arbres de G ceux dont les feuilles coïncident avec les sommets simpliciaux de G.

Notre algorithme est distribué et il s'exécute comme suit. Initialement, tous les sommets du $k-$arbres ont le même poids 1, selon la condition d'anonymat imposée au graphe. Le processus d'élection se comporte comme un processus markovien en temps continu. En effet, à tout instant et avant la fin du processus d'élection, le graphe contient un ensemble d'au moins 2 *sommets périphériques* qui sont tous actifs (ici les sommets *périphériques* sont des sommets simpliciaux[3]).

[2] appelée aussi clique minimale

[3] sommets dont le degré est égal à k

Dès qu'un sommet devient simplicial, il devient actif en générant sa durée de vie qui est une v.a. (variable aléatoire) à distribution exponentielle de paramètre égal à son poids. Ainsi, une fois la durée de vie (durée de suppression) d'un sommet expirée, le sommet est enlevé avec toutes ses arêtes incidentes. À ce moment, un nouveau sommet adjacent au sommet enlevé peut devenir périphérique si son degré devient égal à k. Si c'est le cas, le nouveau sommet collecte le poids du sommet enlevé et l'additionne à son poids courant. Par conséquent, il génère à son tour sa durée de vie qui est une v.a. exponentiellement distribuée ayant son nouveau poids comme paramètre et il devient, ainsi, actif. Sinon, un sommet voisin non périphérique sera choisi uniformément afin de récupèrer le poids du sommet enlevé. Le sommet choisi devra attendre jusqu'à ce qu'il devienne simplicial pour s'activer. Ce processus de suppression continue sur le $k-$arbre résiduel et se termine quand le graphe est réduit à un seul sommet.

5.3.1 Arbre couvrant d'un $k-$arbre

Dans cette section, nous donnons une description d'un algorithme de vagues [Tel00], qui permet de construire un arbre couvrant pour un $k-$arbre donné. Dans un $k-$arbre $G = (V, E)$, les initiateurs de l'algorithme sont les sommets simpliciaux. Tout sommet simplicial v commence à calculer un arbre couvrant partiel auquel il appartient. Rappelons que si G est un $k-$arbre de taille ≥ 2, il contient au moins deux sommets simpliciaux, donc au moins deux arbres couvrants partiels seront calculés.

La construction d'un arbre couvrant du $k-$arbre G se fait progressivement par le maintien d'une forêt couvrante, grâce à la conservation d'une propriété d'absence de cycle entre sous-arbres (*fragments*). Ainsi, la taille de ces fragments augmente et leur nombre diminue au cours du temps, jusqu'à la construction d'un arbre couvrant unique. Cependant, à chaque étape de l'algorithme, lorsqu'un sommet v devient simplicial, il décide après un délai aléatoire quel sommet voisin sera son père dans l'arbre couvrant. Cette décision est prise selon les règles suivantes :

$\mathbf{R}_1$: si $n = k$ ou $n = k + 1$ (G est une clique), alors tout sommet de la clique génère sa durée de vie qui est une v.a. exponentiellement distribuée de paramètre égal à son poids ($w = 1$), sinon,

$\mathbf{R}_2$: si dans le graphe $G' = (V\backslash\{v\}, E\backslash\{(v, v') \mid (v, v') \in E\})$, graphe obtenu par la suppression du sommet v et ses arêtes incidentes, un nouveau sommet unique u dans le voisinage de v devient simplicial après la suppression de v, alors v choisit u,

$\mathbf{R}_3$: autrement, v choisit uniformément au hasard un de ses voisins qui n'est pas encore dans un arbre couvrant.

Avec ces règles nous sommes assurés qu'à la fin, tous les arbres partiels seront fusionnés en un seul arbre. Cet arbre servira pour la deuxième procédure de l'algorithme.

Afin de mieux comprendre le fonctionnement de ces règles, nous donnons un exemple d'exécution.

Exemple 5.3.1. Considérons le 2−arbre de la Fig. 6. Les séquences de graphes de la Fig. 9 montrent les étapes d'une construction d'arbre couvrant. Au début, chaque sommet simplicial commence à calculer un arbre couvant qualifié de partiel. L'utilisation de délais aléatoires assure qu'à chaque étape, un et seul un sommet simplicial choisit son père selon les règles décrites ci-dessus. Par conséquent, à la fin de ce processus, tous les arbres couvrants partiels fusionneront en un seul arbre couvrant.

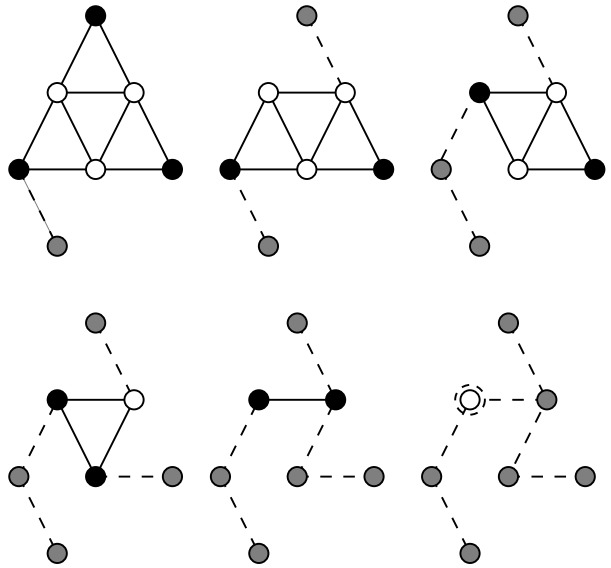


FIG. 9 – Construction d'un arbre couvrant sur le 2−arbre de la Fig. 6 : les sommets noirs sont simpliciaux, les sommets colorés en gris et les arêtes en pointillés sont ceux de l'arbre couvrant.

Remarque 5.3.1. Pour pouvoir guider le processus d'élection dans un $k-$arbre G, les relations $R_i,\ 1 \leq i \leq 3$, doivent produire des arbres couvrants dont les feuilles sont des sommets simpliciaux dans G. L'adoption de telle technique nous facilitera, par la suite, l'analyse de l'algorithme. La figure 10 montre un exemple d'arbre couvrant que nos règles ne génèrent pas. En effet, parmi tous les arbres de recouvrement d'un $k-$arbre G, il est nécessaire de choisir ceux dont les sommets feuilles coïncident avec les sommets simpliciaux de G. Dans cet exemple, le sommet e est une feuille dans l'arbre couvrant T mais il n'est pas simplicial dans G. Par conséquent, comme il y a une "bijection" entre T et G, le sommet e est actif dans T alors qu'il ne l'est pas dans G.

Une conséquence directe de cette remarque est que le choix uniforme d'une $(k+1)-$clique n'aboutit pas en général à une élection uniforme sur l'ensemble des sommets et ce parce que les sommets d'un $k-$arbre n'appartiennent pas au même nombre de cliques.

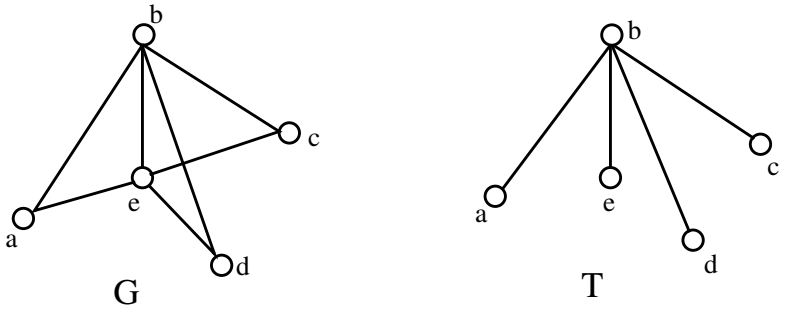


FIG. 10 – Arbre couvrant non généré par les règles R_i

5.3.2 L'élection

Comme le $k-$arbre donné est anonyme, initialement tous les sommets ont un poids $w = 1$ et sont des candidats. Une fois qu'un sommet devient simplicial dans un $k-$arbre, il est une feuille dans son arbre couvrant. Par conséquent, ce sommet génère sa durée de vie qui est une variable aléatoire exponentiellement distribuée dont l'espérance mathématique est $1/w$. Lorsque la durée de vie d'un sommet simplicial est expirée, le sommet n'est plus candidat et l'algorithme le supprime avec toutes ses arêtes incidentes, donnant ainsi son poids à son unique voisin dans l'arbre couvrant produit par la première procédure de notre algorithme. Le processus continue jusqu'à ce que le graphe soit réduit à un sommet unique (l'élu).

Par conséquent, lorsque l'algorithme est réitéré plusieurs fois, tout sommet du $k-$arbre aura probablement la même probabilité d'être choisi par cet algorithme.

5.4 Implémentation de l'algorithme

Soit un $k-$arbre $G = (V, E)$. L'algorithme 2 est exécuté par chaque sommet de G. Ainsi, tout sommet v dispose d'une variable $stat_v$ dont les valeurs sont dans l'ensemble $\{actif, perdu, élu\}$ codant l'état de ce sommet à tout moment du processus d'élection. Il dispose aussi d'un tableau rec_v de taille égale au degré de v. rec_v représente l'état de chaque canal. Tout élement de ce tableau prend une valeur dans l'ensemble des étiquettes $\{actif, couvrant, perdu, vu, supprimé\}$. La variable $\lambda(v)$ est un entier naturel qui sert à collecter les poids des sommets voisins disparus.

Initialement, tout sommet v du graphe G est actif ainsi que tous ses canaux, son poids est de valeur égale à 1.

La première procédure 3−4 est employée pour produire un arbre couvant d'un $k-$arbre. Un sommet interne $\#\{i : rec_v[i] = actif\} > k$ commence par recevoir les messages envoyés par ses voisins simpliciaux dont la durée de vie est achevée. En effet, si le message reçu par un sommet v est de type $\prec$ **get** $\succ$ alors v sait qu'il est interrogé sur son degré, il envoie alors le nombre de ses ports actifs. Dans le cas contraire, le message reçu est de type $\prec tok, b, value \succ$, alors le sommet transmetteur est dans l'arbre couvrant et il a choisit un de ses voisins pour être son successeur

Algorithme 2: Élection probabiliste dans un $k-$arbre.

```
1 : var  canal_v : ensembe des cannaux (* les portes du sommet v *) ;
2 :      rec_v[i] pour tout i ∈ canal_v :{actif, couvrant, perdu, vu, supprimé} init actif ;
3 :      stat_v :{ actif, perdu, élu} init actif.
4 :      λ(v) : le paramètre du sommet v init 1 ;
5 : début
6 :           decouvrirArbre(rec_v, stat_v, λ(v)) ;
7 :           elire(rec_v, stat_v, λ(v)) ;
8 : fin
```

dans cet arbre. Cependant, v est ce sommet successeur si la valeur de la donnée *value* du message reçu est non nulle. La donnée b est à valeur *vrai* lorsque la clique de base est atteinte.

Algorithme 3: Procédure pour découvrir un arbre couvrant

```
decouvrirArbre(rec_v, stat_v, λ(v))
1 : début
2 :    tant que(#{i : rec_v[i] = actif} > k) faire
3 :         recevoir msg du canal j;
4 :       début
           (* j ∈ {i : rec_v[i] = actif} *)

           (* Le sommet v envoie son degré *)
5 :        si ( msg = < get >) alors
6 :           envoyer< tok, #{i : rec_v[i] = actif} > via j ;
7 :        sinon début (* < tok, b, value > message est reçu *)
8 :           si  (b) alors k = k − 1 ;
9 :           si (value !=0) alors
10 :              début λ_v = λ_v + value;
11 :                   rec_v[j] := couvrant ;
12 :              fin
13 :           sinon rec_v[j] := supprimé ;
14 :        fin
15 :      fin
⋮      (* voir la suite dans la page qui suit *)
```

Une fois qu'un arbre couvrant est découvert, les poids de tous les sommets sont remis à la valeur 1 et la deuxième procédure 5 d'élection uniforme dans un arbre sera lancée.

Dans cette procédure, un sommet dont le nombre de liens étiquetés *couvrant* est > 1 continue à recevoir des messages de ses voisins de l'arbre. Une fois qu'il est une feuille, il devient "inactif" pendant un temps généré selon une distribution exponentielle, produit par la procédure *attente-exponentielle*. Ce temps d'attente est la durée de vie associée à ce sommet, il est une v.a. exponentiellement distribuée de paramètre égal à $\lambda(v)$. Maintenant, si après l'expiration de

Algorithme 4: Procédure pour découvrir un arbre couvrant (suite).

(* Maintenant, il y'a k canaux actifs ; le sommet v est simplicial*)
16 : *attente*−exponentielle ($\lambda(v)$) ;
17 : **si** (#{i : $rec_v[i]$ = actif}= 1) **et** (recevoir<**get**> via l) **alors**
18 : **début**
19 : $rec_v[l]$:= couvrant ;
20 : **pout tout** (j : $rec_v[j] = couvrant$) **faire** envoyer<**ok**> via j ;
21 : **fin**
22 : **sinon début**
23 : **pour tout** (j : $rec_v[j] = actif$) **faire** envoyer<**get**> via j;
(* Le sommet v veut récupérer les degrés de ses voisins *)
24 : **pour tout** (j : $rec_v[j] = actif$) **faire**
25 : **début**
26 : recevoir<**tok**,d_a > via j ;
27 : **si** ($d_a = k + 1$) **alors** $rec_v[j]$:= vu ;
28 : **fin**
29 : **si** (#{i : $rec_v[i]$ = vu}>= 1) **alors**
30 : **début** choisir $q \in \{i : rec_v[i] = vu\}$
31 : envoyer<**tok**,(#{i :$rec_v[i]$= vu}>1),$\lambda(v)$ > via q;
32 : $rec_v[q]$:= couvrant ;
33 : **pour tout** $j \in \{j : rec_v[j] = vu$ **ou** $rec_v[j] = actif\}$ **do**
34 : **début**
35 : envoyer<**tok**,(#{i :$rec_v[i] = vu\} > 1), 0 >$ **via** j;
36 : $rec_v[j]$:= supprimé ;
37 : **fin**
38 : **fin**
39 : **sinon début**
40 : choisir $q \in \{i : rec_v[i] = actif\}$;
41 : envoyer< **tok**, $faux, \lambda(v) >$ via q;
42 : $rec_v[q]$:= couvrant ;
43 : **Pour tout** $j \in \{j : rec_v[j] = vu$ **ou** $rec_v[j] = actif\}$**faire**
44 : **début**
45 : envoyer< **tok**, $faux, 0 >$ **via** i ;
46 : $rec_v[j]$:= supprimé ;
47 : **fin**
48 : **fin**
49 : recevoir<**ok**> de j ;
($*j \in \{i : rec_v[i] = couvrant\}*$)
50 : **Pour tout** $l \in \{i : rec_v[i] = couvrant,\ i \neq j\}$ envoyer<**ok**> via l ;
51 : **fin**
52 : **fin**

ce temps aucun message n'est reçu alors le sommet sera supprimé avec son arête incidente et son poids sera envoyé à son unique successeur dans l'arbre couvrant. Sinon, ce sommet est le sommet élu.

Algorithme 5: Procédure d'élection probabiliste dans un arbre couvrant.

elire(rec_v, $stat_v$, $\lambda(v)$)
1 : **début**
(* Dans le cas uniforme, initialement $\lambda(v) := 1$ *) ;
2 : **tant que**(#{u : $rec_v[u]$ = courant}> 1)
3 : **début**
4 : recevoir <**tok**, $\lambda(u)$ > de u ;
5 : $\lambda(v) := \lambda(v) + \lambda(u)$;
6 : $rec_v[u]$:= perdu ;
7 : **fin**
8 : **fin**
(* Maintenant, il y a un canal u_0 avec $rec_v[u_0]$ est couvrant.
Le sommet v est une feuille dans l'arbre couvrant*)
9 : attente-exponentielle ($\lambda(v)$) ;
10 : **si** (un message <**tok**,p > est arrivé)
11 : $state_v$:= élu ;
12 : **sinon début**
13 : $state_v$:= perdu ;
14 : envoyer <**tok**,$\lambda(v)$ > via u_0 où $rec_v[u_0]$ = couvrant ;
15 : **fin**
16 : **fin**

Si $n = k$ ou $n = k + 1$ (le graphe G est une clique), chaque sommet génère sa durée de vie selon la règle $\mathbf{R}_1$. *Le sommet qui survit sera élu.*

5.5 Modèle de processus de Markov

Un algorithme d'élection uniforme dans un $k-$arbre $G = (V, E)$ peut essentiellement être vu comme un processus continu irréversible de Markov en temps continu comme suit.

L'ensemble d'états est l'ensemble de tous les *$k-$arbres facteurs* de G (i.e. tout $k-$arbre obtenu par un ordre d'éliminations simpliciales).

Soient t un instant donné et G' l'état du système à cet instant. Un sommet v de G' est *actif* si et seulement s'il est simplicial dans G'.

Tout sommet actif v a une durée de vie $L(v)$ qui est une v.a. exponentiellement distribuée de paramètre $\lambda(v) = w(v)$:

$$Pr(L(v) > x) = e^{-\lambda(v)x}, \ \forall x \geq 0.$$

Cette expression traduit le fait que : à tout intervalle de temps $[t, t + h]$, la probabilité de mort du sommet actif v est $\lambda(v)h + o(h)$, quand $h \to 0$, et ceci indépendamment de ce qui se

passe ailleurs avec les autres sommets du $k-$arbre et de ce qui s'est produit dans le passé. Cette hypothèse est en parfait accord avec la distribution de l'algorithme. Le processus aléatoire est une variante des processus de mort pure qui sont à leur tour des exemples spéciaux du processus de Markov en temps continu (voir [Fel50], Chapitre XVII).

L'état initial est un $k-$arbre G donné. Les singletons sont les états absorbants. Pour un état donné G', soit $\lambda(G') = \sum_{i=1}^{m} \lambda(v_i)$, où $v_1, \ldots, v_m$ sont des sommets simpliciaux dans G'. Si $L(G')$ dénote la durée de vie de G' (c'est-à-dire le temps de rester dans l'état G'), alors $L(G')$ est égale à la valeur minimale des variables aléatoires indépendantes associées aux sommets $(v_i)_{i=1,\ldots,m}$ et, en conséquence, elle a une distribution exponentielle de paramètre $\lambda(G')$. Les probabilités de transition peuvent être définies comme suit. Soit G' l'état du processus d'élection à un instant t donné. Soient $v_1, \ldots, v_m, m \geq 2$, des sommets simpliciaux dans G'. Alors, l'état du processus d'élection est un des $k-$arbres facteurs G_i obtenu à partir de G' en supprimant le sommet simplicial v_i et ses arêtes incidentes. La probabilité de transition s'exprime ainsi par :

$$p(G', G_i) = \frac{\lambda(v_i)}{\lambda(G')} = \frac{\lambda(v_i)}{\sum_{1 \leq j \leq m} \lambda(v_j)}, \quad 1 \leq i \leq m. \tag{8}$$

Dorénavant, nous supposons que $G = (V, E)$ est un $k-$arbre de taille $n \geq 2$ et que $T = (V, F)$ est un arbre couvrant de G produit par les règles R_i vues dans la section 5.3.1.

Remarques 5.5.1.

- Dans le reste de ce chapitre, lorsque nous citons le terme d'arbre couvrant cela signifie tout arbre couvrant obtenu par les règles R_i vues dans la section 5.3.1.

- La suppression d'un sommet simplicial dans un $k-$arbre G revient à supprimer une feuille dans son arbre couvrant T.

La proposition suivante détermine la probabilité de la distribution des états de l'élection à tout instant t.

Proposition 5.5.1. *Soit G' un $k-$arbre facteur et soit $P_{G'}(t)$ l'expression qui dénote la probabilité que l'état de l'élection à l'instant t soit G'. Nous avons :*

(i) $\dfrac{dP_G(t)}{dt} = -\lambda(T)P_G(t),$

(ii) *Pour tout $k-$arbre facteur $G' \neq G$ de taille au moins 2,*

$$\frac{dP_{G'}(t)}{dt} = -\lambda(G')(t)P_{G'}(t) + \sum_{v} \lambda(v)P_R(t),$$

avec $R = G' \cup (\{v\}, \{\{v, u\},\ \forall u \in G' \text{ et adjacent à } v \text{ dans } G\})$,

où la sommation est effectuée pour tous les sommets v adjacents à G′ dans G et qui n'appartiennent pas à G′, et

(iii) $$\frac{dP_{(\{v\},\emptyset)}(t)}{dt} = \sum_{u \text{ adjacent à } v \text{ dans } G} \lambda(u) P_{(\{v,u\},\{\{v,u\}\})}(t),$$

avec la condition initiale $P_G(0) = 1$.

Preuve. Cette proposition peut être prouvée en appliquant directement la méthode fournie dans [KT81]. Ainsi, en analysant la probabilité de la transition à partir des états voisins vers l'état T' dans l'intervalle du temps $[t, t + h]$ (voir [Fel50], Chapitre XVII, Section 5) en termes d'état de l'élection à l'instant t, nous arrivons au système d'équations différentielles ci-dessus.

Une preuve identique a été proposée pour la proposition 5.5.1 du chapitre précédent sur l'élection uniforme dans les arbres.

□

Cette proposition caractérise la probabilité de distribution des états à un moment donné t. En particulier, elle nous permet de calculer les probabilités d'absorption.

Exemple 5.5.1. Soit G le 2−arbre de la Fig. 6 et soit T un arbre couvrant construit par la première étape de l'algorithme appliquée à G. Rappelons qu'initialement tous les sommets ont le même poids $w = 1$.

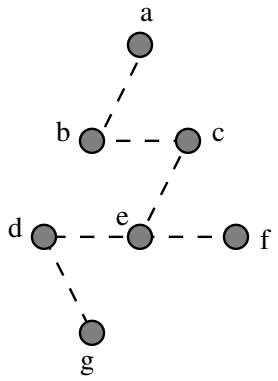


Fig. 11 – Un arbre couvrant du 2−arbre donné dans la Fig. 6

Puisque la solution du système différentiel de la proposition précédente est difficile à trouver, nous faisons un calcul direct semblable à celui vu dans l'exemple 4.4.1 du chapitre précédent pour calculer les probabilités d'être élu pour tout sommet de ce 2−arbre.

Il y a trois ordres d'éliminations menant au choix du sommet étiqueté a. Ces trois suites peuvent être identifiées par les ordres éliminations de sommets simpliciaux correspondants. $\langle f, g, d, e, c, b\rangle$, $\langle g, f, d, e, c, b\rangle$ et $\langle g, d, f, e, c, b\rangle$.

Considérons la première suite. Initialement, il y a 3 sommets simpliciaux de poids 1. Ces sommets simpliciaux sont feuilles dans l'arbre couvrant T, ainsi la probabilité que le sommet f sera supprimé est $\frac{1}{3}$, le sommet est supprimé avec toutes ses arêtes incidentes. Dans le $k-$arbre réduit, il y a maintenant deux sommets simpliciaux de poids 1 et donc la probabilité de suppression de g est $\frac{1}{2}$. Dans le $k-$arbre facteur constitué par les sommet a, b, c, e, d le sommet simplicial a est de poids 1 et d a un poids 2. Maintenant, la probabilité de suppression de d est 2 fois la probabilité de suppression de a et est donc $\frac{2}{3}$. C'es à dire que la suppression de d est 2 fois plus probable que celle de a. Le même argument appliqué au $k-$arbre de facteur composé par les sommets simpliciaux a et e, assigne la probabilité $\frac{4}{5}$ à la suppression de e. De même, c est supprimé avec une probabilité égale à $\frac{5}{6}$ et finalement, la probabilité d'enlever b devient $\frac{6}{7}$.

Par conséquent, la probabilité de la première suite est $\frac{1}{3}\frac{1}{2}\frac{2}{3}\frac{4}{5}\frac{5}{6}\frac{6}{7} = \frac{4}{63}$. Un calcul semblable assigne la probabilité $\frac{2}{63}$ à la deuxième suite $\langle g, f, d, e, c, b\rangle$ et $\frac{1}{21}$ à la la troisième $\langle g, d, f, e, c, b\rangle$. Finalement, nous déduisons que le sommet a est choisi avec une probabilité égale à $\frac{1}{7}$.

Nonobstant le nombre de suites qui devient plus grand (le calcul est plus lourd à faire), l'utilisation de la même technique appliquée aux autres sommets donne pour chaque sommet la probabilité $\frac{1}{7}$ d'être élu.

Une approche calculatoire qui apparaît très simple et efficace est la simulation. Cette technique abandonne entièrement toute analyse mathématique et se limite à une "imitation" du système étudié. Le modèle de simulation fournit des informations quantitatives là où les méthodes mathématiques échouent.

La table 1 donne les résultats de la simulation de l'algorithme sur le $2-$arbre étudié dans cet exemple.

Sommets	Probabilités
a	0.142799
b	0.142947
c	0.142790
d	0.142970
e	0.142892
f	0.142870
g	0.142732

TAB. 1 – Résultats de simulation d'exécution de l'algorithme d'élection uniforme sur le $2-$arbre de la Fig. 6. La taille de la simulation est 1 000 000.

Les résultats de la simulation obtenus confirment que tous les sommets ont la même probabilité d'être élus.

Par ailleurs, afin de prouver analytiquement l'affirmation principale de ce chapitre sur l'uniformité, nous réempruntons l'utilisation des concepts de graphes orientés (voir chapitre 4 section 4.3). Dans la section qui suit, le processus d'élection sera modélisé par un processus ascendant de mort sur une forêt d'arbres enracinés.

5.5.1 Processus de mort d'une forêt

Les arborescences peuvent être employées pour calculer, de manière simple, les probabilités d'absorption.

Dans cette section, nous considérons des forêts d'arborescences. Soit F une forêt de taille n, un processus de mort sur F est représenté comme suit. Au commencement, tous les sommets de F sont de poids 1. Chaque feuille (ou sommet devenant feuille) a une durée de vie exponentiellement distribuée avec un paramètre égal à son poids. Une fois la durée de vie d'une feuille expirée, elle est enlevée avec son unique arête incidente. Si la feuille enlevée a un père, alors le père récupère le poids du fils enlevé et le processus continue jusqu'à ce que tous les sommets disparaissent. Soit $W(F)$ le temps nécessaire pour la disparition totale de la forêt F ; c'est une v.a. réelle positive.

Le modèle actuel n'est pas vraiment différent du modèle de processus d'élection et toutes les propriétés de ce dernier sont vérifiées, si nous remplaçons le concept de feuille dans un arbre par celui de feuille dans une arborescence.

5.5.2 Preuve du théorème principal

Le lemme suivant établie la relation entre le processus d'élection probabiliste dans un $k-$arbre et le processus de mort dans une forêt.

Lemme 5.5.1. *Considérons un $k-$arbre G et un arbre couvrant T de G. Soit v un sommet de G. Supposons que les sommets adjacents à v soient $v_1, \ldots, v_k$. Soit F une forêt constituée de deux arborescences A et B, obtenues par la suppression de l'arête $\{v, v_1\}$, ayants v et v_1 respectivement comme racines. Alors, la probabilité que v soit supprimé avant l'arbre facteur du coté de v_1 (c'est-à-dire l'arbre non orienté B) dans le processus d'élection sur G suivant T est équivalente à la probabilité que v_1 batte v dans F.*

Preuve. Les événements pour lesquels les probabilités doivent être calculées peuvent être identifiés comme des ordres d'élimination de feuilles :

Soit $\sigma = \langle f_1, \ldots, f_m \rangle$ un ordre d'élimination, où f_i, $1 \leq i \leq m$ sont des feuilles ou des sommets qui deviennent feuilles dans T (respectivement dans F) après la suppression de certains sommets dans une suite d'elimination. Nous avons $f_m = v$ et v_1 ne figure pas dans σ.

Il est facile de voir que tout ordre d'élimination qui satisfait les conditions précédentes dans T, les satisfait aussi dans F et vice-versa. En outre, la probabilité de l'ordre σ est donnée selon (8) par :

$$P(\sigma) = \prod_{1 \leq i \leq m} q_i,$$

avec

$$q_i = \frac{\lambda(f_i)}{\lambda(T_i)},$$

où T_i est l'arbre résiduel (respectivement arborescence) juste avant la suppression de la feuille f_i.

A chaque étape de suppression de feuilles selon σ, T et F ont le même ensemble de feuilles et, par conséquent, les quantités impliquées dans T correspondent parfaitement à celles qui le sont dans F. □

Dans la suite, nous fixons dans un premier temps un arbre couvrant dans G puis et nous cherchons à calculer la probabilité d'élire un sommet dans G suivant T.

Proposition 5.5.2. *[MSZ03b] Soit $q_T(v)$ la probabilité d'être élu pour un sommet v dans un arbre couvrant T d'un $k-$arbre G de taille n. Nous avons $q_T(v) = \frac{1}{n}$.*

Théorème 5.5.1. *L'algorithme d'élection dans un $k-$arbre est totalement équitable, c'est-à-dire la probabilité d'être élu est la même pour tout sommet.*

Preuve. Si $n \leq k+1$, le théorème est évident (voir la règle R_1). Nous supposons maintenant que $n > k+1$.

Soit $p_G(v)$ la probabilité d'élire le sommet v dans un $k-$arbre G de taille n. Nous notons par $\mathcal{S}$ l'ensemble des arbres couvrants de G générés par les règles vues dans la section 5.3.1, et notons par $\pi(T)$ la probabilité que l'algorithme produit dans sa première étape d'exécution l'arbre couvrant $T \in \mathcal{S}$, en vertu de la proposition 5.5.2, nous avons :

$$\begin{aligned} p_G(v) &= \sum_{T \in \mathcal{S}} q_T(v)\pi(T) \\ &= \frac{1}{n} \sum_{T \in \mathcal{S}} \pi(T) \\ &= \frac{1}{n} \end{aligned}$$

□

5.6 Durée d'élection

L'étude de la durée d'élection présente quelques intérêts théoriques. Dans le chapitre précédent, nous avons étudié cette v.a. pour des exemples très particuliers d'arbres. Il a été conjecturé que la valeur de son espérance mathématique est logarithmique en la taille de l'arbre. Notre algorithme fonctionne sur un $k-$arbre en deux phases. La première phase, qui permet de découvrir un arbre couvrant, a une durée linéaire en la taille du graphe. Nous prouvons ici que la seconde phase est d'espérance bornée par $ln(n)$.

Dans la suite, la durée d'élection est le temps simulé dans la deuxième étape de l'algorithme d'élection. Pour plus de précision, étant donné un $k-$arbre G, la *durée d'élection* sur G, notée $D(G)$, est le temps d'absorption du processus d'élection probabiliste sur G. Nous avons :

Proposition 5.6.1. *L'espérance mathématique* $\mathbb{E}(D(G))$ *de la durée d'élection est bornée par* $H_{n-1} = \sum_{i=1}^{n-1} \frac{1}{i}$, *où n est la taille du G.*

Preuve. Nous remarquons que pour n'importe quelle forêt F, en vertu de la proposition 4.4.1 (cf. chapitre 4), nous avons :

$$\mathbb{E}(W(F)) = H_n.$$

Dans le reste de la preuve, nous montrons que l'espérance de $D(G)$, conditionnée par l'événement qu'un arbre couvrant T a été choisi et un sommet v est élu le long de T, est bornée par H_{n-1}, pour tout arbre T et pour tout sommet v.

La condition du choix de T et de v, implique que le processus de suppression a lieu dans un processus de mort ascendant sur $A_v(T)$ (rappelons que $A_v(T)$ est l'arborescence obtenu à partir de T en prenant v comme sa racine). Tout ordre d'élimination de feuilles, conditionné par le choix de (T, v) dans le processus d'élection, a la même probabilité que dans le processus de mort sur $A_v(T)$. Cependant, le processus d'élection se termine au moment où T est réduit à v, alors que le processus de mort continue jusqu'à la disparition de v. D'autre part, l'espérance mathématique du temps d'élimination d'une nouvelle feuille est l'inverse de la somme des poids de toutes les feuilles et ce dans le processus d'élection et dans le processus de mort.

Chaque feuille dans le processus de mort est aussi une feuille dans le processus d'élection (pour un ordre d'élimination de feuilles fixé) et, par la suite, la somme des poids dans le processus d'élection est supérieure ou égale à la même somme dans le processus de mort à toute étape. Par conséquent, dans une suite de suppression donnée, l'espérance du temps d'attente entre deux suppressions successives dans une élection ne peut pas être supérieure à celle dans le processus de mort.

Par ailleurs, le fait que dans le processus de mort, le dernier sommet v a une espérance mathématique de sa durée de vie égale à $\frac{1}{n}$ implique que l'espérance mathématique conditionnée de $D(G)$ est inférieure ou égale à $H_{n-1} = H_n - \frac{1}{n}$. □

Chapitre 6

Polyominoïdes et Élection Uniforme

Sommaire

Dans la première partie de ce chapitre, nous présentons une famille de graphes appelée polyominoïde. Les polyominoïdes sont introduits en tant que graphes particuliers non orientés. En outre, nous donnons un ensemble de règles produisant la classe de tous les polyominoïdes. L'essentiel de ce chapitre est consacré à l'analyse d'un algorithme probabiliste d'élection uniforme dans ces graphes. La deuxieme partie du chapitre décrit le fonctionnement de cet algorithme et présente quelques outils nécessaires pour son analyse.

6.1 Introduction

Les réseaux étudiés dans ce chapitre sont anonymes et ils ont une topologie de polyominoïdes. Un polyominoïde est un graphe qui combine les structures d'arbres et celles des polyominos (voir la Fig. 12). Les polyominos ont une histoire ancienne, ils sont apparus au début du $20^{\text{ème}}$ siècle, mais ils ont été popularisés par Golomb [Gol54, Gol96] et Gardner [Gar56, Gar95] dans les années cinquantes dans *Scientific American columns*, "Jeux Mathématiques". Ils ont été également étudiés par des mathématiciens [BM2a, BM96, DDD95, DV84], parce qu'ils constituent des objets combinatoires qui présentent des propriétés intéressantes. Ils ont été aussi le

sujet d'études intensives de certains physiciens, grâce à leur convenance pour modéliser plusieurs phénomènes physiques. Ils sont connus sous le nom d'animaux en physique statistique ([Tem56]). En informatique, leur étude a été motivée dans différents secteurs tels que la conception des réseaux cellulaires VLSI en électronique ([LFNV02, RRP97]) et le traitement d'images ([Bre04]).

La motivation principale derrière l'étude présentée dans ce chapitre est de présenter et d'analyser un algorithme distribué probabiliste d'élection uniforme dans les polyominoïdes. L'algorithme enlève du polyominoïde tout sommet dont la durée de vie est expirée (le graphe restant demeure un polyominoïde). L'analyse de l'algorithme révèle le fait surprenant que, quelque soit l'endroit où le sommet est placé dans le polyominoïde, il a la même probabilité de survivre que les autres, comme dans le cas des arbres et des $k-$arbres que nous avons vu aux chapitres précédents.

Notre algorithme peut être considéré comme une variante probabiliste de l'algorithme distribué introduit dans [LMZ95], où des délais aléatoires sont présentés.

Les réseaux étudiés ici sont asynchrones, les processeurs n'ont pas accès à une horloge globale et tout message envoyé par processeur à un de ses voisins arrive dans un délai fini. Ce délai sera négligeable dans la suite de l'étude afin de simplifier l'analyse. Des étiquettes sont attachées aux sommets et aux arêtes. Elles répresentent leurs états.

Nous considérons des *calculs locaux cellulaires* qui, à chaque étape, peuvent modifier l'état (ou l'étiquette) d'un sommet. En effet, la nouvelle étiquette d'un sommet dépend de sa précédente étiquette (état) et des étiquettes de ses voisins. La nouveauté de notre approche est l'utilisation d'un délai aléatoire pour l'étiquetage ; un sommet ne peut modifier son état que si le délai d'attente associé à ce sommet est fini. Ces délais sont des variables aléatoires exponentielles, définies indépendamment, pour les sommets actifs. Le paramètre d'une variable aléatoire d'un sommet est égal au poids attribué à ce sommet. Le poids est calculé localement en fonction du poids initial et des poids collectés des voisins non élus. Le processus d'étiquetage continue jusqu'à ce que aucune transformation ne soit possible, c'est-à-dire qu'une configuration finale soit atteinte. Dans cette configuration, il y a uniquement un sommet qui a une étiquette différente des autres, ce sommet est considéré comme l'élu.

6.2 Définitions et notations

Dans la littérature, nous trouvons plusieurs définitions concernant les polyominos et les grilles, voir [MS96, Ros00]. Traditionnellement, un polyomino est l'ensemble de cellules situées à l'intérieur d'un polygone orthogonal dessiné sur une grille. Nous définissons les polyominoïdes en tant que graphes finis dont les sommets sont des points de $\mathcal{Z} = \mathbb{Z}\times\mathbb{Z}$, où $\mathbb{Z}$ dénote l'ensemble des entiers relatifs. Ils sont liés par une relation de voisinage.

Nous employons les termes habituels tels que "en haut", "en bas", "à droite" et " à gauche" sur $\mathbb{Z} \times \mathbb{Z}$. Les arêtes sont des liens entre les paires de points, c'est-à-dire l'ensemble de paires de points de formes $\{(x, y), (x + 1, y)\}$ ou $\{(x, y), (x, y + 1)\}$, pour tout $x \in \mathbb{Z}$ et pour tout $y \in \mathbb{Z}$.

On dit que deux sommets $v = (x, y)$ et $v' = (x', y')$ de $\mathcal{Z}$ sont *voisins* si $x = x'$ et $|y - y'| = 1$ ou si $y = y'$ et $|x - x'| = 1$.

Tout sommet d'une arête e est appelé son *extrémité*.

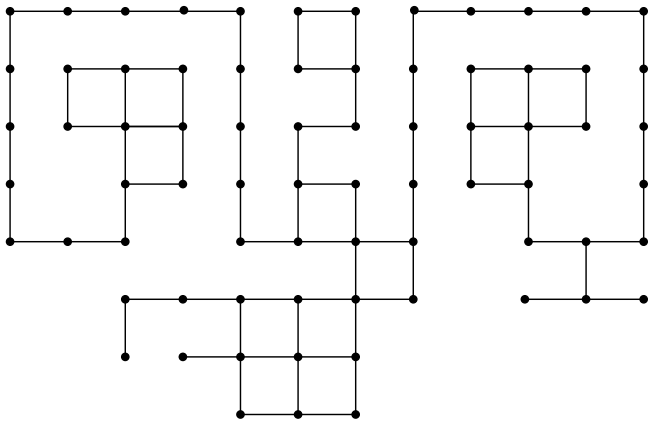


Fig. 12 – Un exemple de polyominoïde

Soit $\mathcal{T}$ l'ensemble de toutes les arêtes dont les extremités sont voisins (selon la définition de voisinage ci-dessous) et soit **U**=$(\mathcal{Z}, \mathcal{T})$ le graphe infini constitué de l'ensemble des sommets $\mathcal{Z}$ et l'ensemble des arêtes $\mathcal{T}$.

Une *cellule* est un sous-graphe de **U**, induit par un ensemble de quatre paires de sommets voisins $\{(x, y), (x + 1, y), (x + 1, y + 1), (x, y + 1)\}$. Ainsi, une cellule correspond à un réseau carré.

Nous rappelons qu'un *chemin* est une suite alternée finie $\sigma = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k$ de $k+1$ sommets et de k arêtes *distinctes* ($k \geq 0$), tel que chaque arête e_i a comme extrémités v_{i-1} et v_i.

La *longueur* d'un chemin σ est le nombre de ses arêtes k. Il convient de noter qu'un chemin peut passer plusieurs fois par un sommet, alors qu'il ne peut emprunter qu'une seule fois une arête.

Pour alléger l'écriture, nous représentons un chemin par les sommets empruntés en omettant ses arêtes, lorsqu'il n'y a pas de danger de confusion, identifiant ainsi σ par la suite des sommets $v_0, v_1, \ldots, v_k$ de manière à ce que toute paire de deux termes successifs v_i et v_{i+1} devrait constituer un ensemble unique (une arête).

Dans un polyominoïde, un *cycle* est un chemin de longueur $k \geq 4$ dans lequel le premier sommet v_0 et le dernier v_k coïncident.

A partir de cette définition, on peut constater que **U** est *biparti*, c'est-à-dire tous ses cycles sont de longueurs paires.

Pour un cycle γ dans un polyominoïde, nous pouvons facilement définir ses sommets intérieurs [Sed92] :

Définition 6.2.1. Soit γ un cycle dans un polyominoïde, un sommet (x, y) est *à l'intérieur* du cycle $\gamma = (x_0, y_0), (x_1, y_1), \ldots, (x_{k-1}, y_{k-1}), (x_0, y_0), ((x_k, y_k) = (x_0, y_0))$ si $(card\{i \mid y = y_i$ et $y \neq y_{i+1}$ et $x \leq x_i\})$ est impair.

Selon cette définition, les sommets de γ sont à l'intérieur du cycle γ.

Pour illustrer cette définition, nous donnons l'exemple suivant. Etant donné le polyominoïde présenté dans la figure 13. Le sommet (x, y) est dans le cycle constitué par les sommets et les arêtes rouges. Dans ce cycle, l'ensemble de sommets $\{(x+1, y), (x+3, y), (x+5, y)\}$ est de cardinal impaire.

□

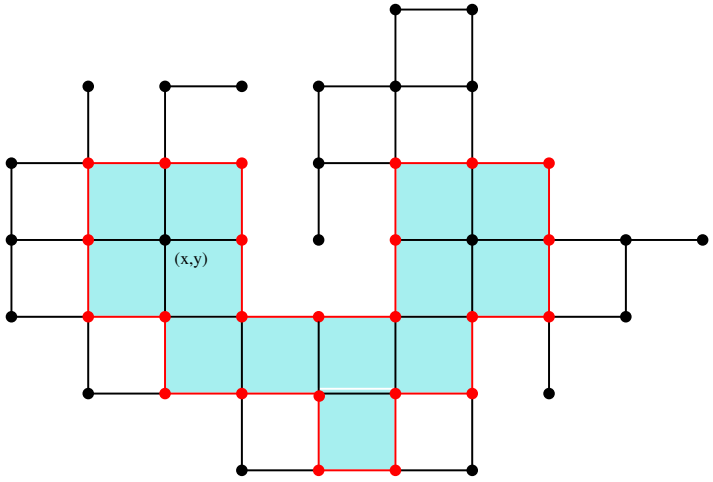


FIG. 13 – Sommet intérieur d'un cycle

Définition 6.2.2. Un *polyominoïde* **P**=(V, E) est un sous-graphe partiel de **U** soumis aux conditions suivantes :

(i) V est fini,

(ii) **P** est connexe et

(iii) **P** ne contient pas de trous, c'est-à-dire pour tout cycle γ dans **P**, les sommets à l'intérieur de γ sont contenus dans V et si deux sommets voisins sont à l'intérieurs de γ, alors l'arête qui relie ces deux sommets est dans E.

La *taille* de **P**=(V, E) est le cardinal de l'ensemble V.

Un pavage (*tiling* en anglais)[BNRR95] est une façon de remplir un espace à l'aide d'un motif répétitif, sans trou ni débordement. Par conséquent, il est facile de voir que la dernière propriété est équivalente à la propriété de pavage de **P** ; l'ensemble des sommets internes de γ et leurs arêtes de liaison constituent un sous-graphe d'une grille.

Définition 6.2.3. Un polyominoïde **Q**=(V_Q, E_Q) est appelé *sous-polyominoïde* du polyominoïde **P** = (V_P, E_P) si $V_Q \subseteq V_P$ et $E_Q \subseteq E_P$ tel que $E_Q = E_P \cap \{\{u, v\} \mid (u, v) \in V_Q^2\}$.

6.3 Construction distribuée d'un polyominoïde

La classe des polyominoïdes peut être définie par récurrence sur le graphe **U**. En outre, la construction peut se faire d'une manière distribuée.

Soit $\mathcal{P}$ l'ensemble des sous-graphes partiels de **U** obtenu par les règles inductives suivantes :

(a) Pour tout $(x, y) \in \mathcal{Z}$, le graphe $\mathbf{P}=(\{(x, y)\}, \emptyset)$ est dans $\mathcal{P}$.

(b) Soit $\mathbf{P}=(V, E) \in \mathcal{P}$. Considérons deux sommets voisins v et v' tels que $v \in V$ et $v' \notin V$, alors $\mathbf{Q}=(V \cup \{v'\}, E \cup \{\{v, v'\}\})$ est dans $\mathcal{P}$.

(c) Soit $\mathbf{P}=(V, E) \in \mathcal{P}$. Supposons que V contienne 4 sommets voisins $v_1 = (x, y)$, $v_2 = (x + 1, y)$, $v_3 = (x + 1, y + 1)$ et $v_4 = (x, y + 1)$ situés sur une cellule dans **U**, tel que trois arêtes de cette cellule sont dans E et la quatrième, appelée e, ne l'est pas, alors $\mathbf{Q}=(V, E \cup \{e\})$ est dans $\mathcal{P}$.

La construction est totalement distribuée et l'application des règles de réécriture [LMS99] nécessite seulement la connaissance des secteurs de voisins qui sont dans une boule de rayon 2. Il s'agit des transformations modifiant l'étiquetage des sous-graphes de rayon 2. De ce fait, la construction locale peut s'exprimer en considérant des transformations affectant un sommet v et l'ensemble des sommets qui sont à une distance 2 de ce sommet (c'est-à-dire ses voisins et les voisins de ses voisins). Ainsi, la famille des polyominoïdes peut être engendrée par une grammaire algébrique similaire à une grammaire hors-contexte.

À ce stade, il est difficile de prouver que l'ensemble $\mathcal{P}$ est la classe de tous les polyominoïdes sur **U**. La proposition suivante montre l'équivalence des deux définitions.

Proposition 6.3.1. *Un sous-graphe partiel* $\mathbf{P}=(V, E)$ *de* **U** *est un polyominoïde si et seulement s'il appartient à* $\mathcal{P}$.

Preuve.

⇒ Soit $\mathbf{P}=(V, E) \in \mathcal{P}$ et prouvons que **P** est un polyominoïde. Il est claire que le graphe défini par la règle (a) ci-dessus est un polyominoïde. Par conséquent, il suffit de prouver que les constructions données par les règles (b) et (c) préservent la structure des polyominoïdes. Supposons que **P** est un polyominoïde et montrons que le sous-graphe $\mathbf{Q}_1$ obtenu par (b) et le sous-graphe $\mathbf{Q}_2$ obtenu par (c) sont aussi des polyominoïdes. Les propriétés de la connexité et de la finité sont évidentes. Nous allons montrer qu'aucun trou n'est créé lors de l'application de la règle (b) ou de la règle (c).

- En appliquant la règle (b), un nouveau sommet v' est ajouté à **P**. Comme v' est de degré 1, il n'y a aucun nouveau cycle dans $\mathbf{Q}_1$ et tous les sommets à l'intérieur d'un cycle dans **P** demeurent à l'intérieur du même cycle dans $\mathbf{Q}_1$. Évidemment, le même fait est vérifié pour toute arête dont les extrémités sont dans $\mathbf{Q}_1$.
- $\mathbf{Q}_2=(V, E')$ est le graphe obtenu à partir d'un polyominoïde $\mathbf{P}=(V, E)$ par une application de (c). Soit v un sommet à l'intérieur d'un cycle γ dans $\mathbf{Q}_2$. Si toutes les arêtes de γ sont dans E, alors v devrait être dans V. Autrement, γ emploie une arête d'une cellule constituée par l'ensemble de sommets $S = \{v_1, v_2, v_3, v_4\}$. Soit $\{v_1, v_2\}$ cette

arête. L'arête $\{v_1, v_2\}$ n'appartient pas à E, de plus nous avons $E' = E \cup \{v_1, v_2\}$. Dans ce cas, il est possible de transformer γ en un autre cycle γ' inclu dans E en évitant v_1 et ceci en empruntant d'autres sommets de S.

$\Leftarrow$ Soit $\mathbf{P}=(V, E)$ un polyominoïde. Nous montrons par une preuve sur le cardinal de l'ensemble V que $\mathbf{P}$ appartient à $\mathcal{P}$.

Si V est de cardinal 1, alors $\mathbf{P}\in \mathcal{P}$. Supposons maintenant que tout graphe $\mathbf{P}$ de taille[1] inférieure ou égale à n est un polyominoïde. Soit $\mathbf{Q} =(V', E')$ un polyominoïde de taille $n + 1$. Si $\mathbf{Q}$ possède un sommet v de degré 1, alors lors de la suppression de v et de son arête incidente, $\mathbf{Q}$ sera transformé en $\mathbf{P}$. Il est claire que $\mathbf{P}$ préserve les propriétés (i)-(iii) et par conséquent, selon l'hypothèse de récurrence, il appartient à $\mathcal{P}$. Une application de la règle (b) permet d'affirmer que $\mathbf{Q}$ est aussi dans $\mathcal{P}$.

Supposons maintenant que tout sommet du polyominoïde $\mathbf{P}=(V, E)$ est de degré supérieur ou égal à 2. Nous avons $|E| - |V| \geq 0$, sinon, $\mathbf{P}$ est un arbre et admet un sommet de degré 1. Nous utilisons maintenant une deuxième récurrence sur $|E| - |V|$. Il est évident de voir que $\mathbf{P}$ admet au moins un cycle. Soit γ un *cycle maximal*[2] dans $\mathbf{P}$. Il est facile de voir que si nous supprimons une arête du cycle γ de $\mathbf{P}$, le graphe résiduel obtenu, noté $\mathbf{R}$, préserve les propriétés (i)-(iii). Ainsi, selon l'hypothèse de récurrence sur $|E| - |V|$, $\mathbf{R} \in \mathcal{P}$.

Une application de la règle (c) sur le polyominoïde $\mathbf{R}$ permet de reconstruire $\mathbf{P}$ en tant que membre de $\mathcal{P}$.

□

6.4 Algorithme d'élection uniforme dans les polyominoïdes

L'algorithme d'élection asynchrone, présenté dans cette section, est conçu pour des réseaux anonymes ayant une topologie de polyominoïdes. Chaque sommet connaît seulement les orientations des arêtes qui le relient à ses voisins, mais ne connaît ni la taille du polyominoïde ni ses propres coordonnées dans le plan. La solution dans le cas général consiste à calculer un arbre couvrant. Ainsi, l'élection est lancée par chaque sommet feuille dans cet arbre.

En utilisant les propriétés des polyominoïdes, nous allons concevoir un algorithme distribué qui permet de choisir uniformément un sommet dans un polyominoïde donné en tant que *leader* (l'élu).

6.4.1 Élection distribuée

L'algorithme d'élection est décrit dans cette sous-section par un système de réécriture de graphe. Les systèmes de réécriture de graphe, ou plus généralement les calculs locaux dans les

[1] nombre de sommets

[2] cycle qui n'est pas à l'intérieur d'un autre.

graphes, sont des modèles puissants qui fournissent des outils généraux pour coder les algorithmes distribués, pour comprendre leur puissance et pour prouver leur validité. On trouvera des survols dans [LMS99, LMZ95, Sel04].

Chaque sommet (resp. arête) possède une étiquette qui représente son état.

Notre algorithme distribué est basé sur le système de réécriture présenté par Litovsky, Métivier et Zielonka dans [LMZ95]. Rappelons qu'une règle de calcul est définie par un graphe connexe et deux étiquetages de ce graphe, donc elle est appliquée localement. Soit la règle de réécriture $\mathcal{R}$ donnée par un graphe connexe H et deux fonctions de marquages de H [LMZ95] : un étiquetage initial λ et un étiquetage final λ'. Soit G un graphe étiqueté. Si G contient un sous-graphe étiqueté isomorphe avec (H, λ), alors en appliquant la règle $\mathcal{R}$, nous changeons l'étiquetage de ce sous-graphe en λ'.

Nous pouvons alors coder notre algorithme distribué par des règles de réécritures locales. En effet, des étiquettes attachées aux sommets et aux arêtes sont localement modifiées, c'est à dire que les sommets et les arêtes appartiennent à un sous-graphe de rayon fixe 2 dans lequel nous pouvons appliquer une de nos règles. La réécriture est exécutée jusqu'à ce qu'on obtienne un graphe *irréductible* où aucune règle n'est applicable.

Initialement, tous les sommets du graphe G ont la même étiquette. Nous cherchons un graphe *noethérien* réécrivant le système tel que lorsque nous obtenons un graphe étiqueté irréductible, après un certain nombre d'étapes de réécritures, il existe une étiquette spéciale attachée à exactement un seul sommet ; ce sommet sera considéré comme l'élu.

Dans la suite, nous utilisons un système de réécriture de graphe enrichi par un délai aléatoire. En effet, une règle ne peut être appliquée que si le délai correspondant est expiré. Nous étudions des calculs locaux dans un polyominoïde dont les réécritures ne sont qu'une illustration. Ces calculs sont formalisés comme des modifications des étiquettes (ou des états) associées aux sommets et aux arêtes du graphe. Ces modifications sont décidées localement : la nouvelle étiquette attachée à un sommet v (resp. à une arête e) ne dépend que du sous-graphe induit par les sommets qui sont dans la boule de rayon 2. Le système de réécriture de graphes appliqué ici utilise des contextes interdits [LMS92, God02, Sel04]. L'idée est d'empêcher l'application d'une règle de réécriture toutes les fois où les occurrences correspondantes sont "inclues" dans certaines configurations spéciales, appelées contextes. Ainsi, une règle peut être appliquée que s'elle ne se produit pas dans un contexte interdit donné et que le delai asssocié est expiré.

Une *règle de réécriture avec contextes interdits* est un quadruplet $R = (G_R, \lambda_R, \lambda'_R, F_R)$ tel que $(G_R, \lambda_R, \lambda'_R)$ est une règle de réécriture et F_R est un ensemble fini de contextes de (G_R, λ_R). Une règle de réécriture peut être appliquée dans un sous-graphe si et seulement si ce sous-graphe n'est pas inclus dans une occurrence de l'un de ses contextes interdits [LMS95, LMS99].

$$R : \{F_C ; Left \longrightarrow Right\}$$

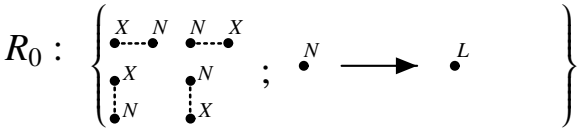
Soient $\mathbf{P}$=(V, E) un polyominoïde et $\mathcal{L} = \{N, A, B, L\}$ l'ensemble des étiquettes utilisées. L'étiquette N encode l'état neutre, A encode l'état actif, B encode l'état battu (ou perdant) et L encode l'état élu (leader).

Initialement, tout sommet possède un poids $w = 1$ et une étiquette N, on dit qu'il est $N-$étiqueté. Nous notons par X tout état différent de B, i.e. $X \in \mathcal{L} \setminus \{B\}$.

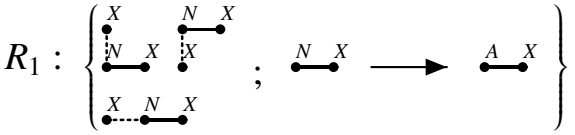
L'algorithme s'exécute sur un polyominoïde $\mathbf{P}$ de la manière distibuée suivante : tout sommet v de $\mathbf{P}$, $N-$étiqueté, décide localement s'il est actif ou non selon les règles d'activation ci-dessous.

Nous représentons les règles d'activation des sommets par R_i et les règles de transmission de poids par R_i'.

R_0 : Si le degré de v est nul ($deg(v) = 0$), alors l'élection est terminée et v est le sommet élu. Il est important de noté que ce sommet est considéré comme un sommet *actif*.

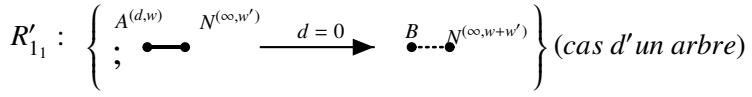


R_1 : Si le degré de v est 1, alors v devient actif et génère une durée de vie qui est une v.a. exponentielle de paramètre égal à son poids. Une fois sa durée de vie expirée, il disparaît avec l'arête incidente et son voisin récupère son poids.

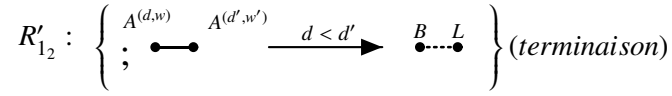


Le sommet voisin u de v, ayant récupéré le poids de ce dernier, est soit dans l'état actif, soit dans l'état neutre. Dans ces deux cas, nous avons :

– Si u est neutre alors au moment où il récupère le poids de v, il décide localement s'il devient actif dans le graphe résiduel $\mathbf{P}' = (V \setminus \{v\}, E \setminus \{\{v, u\},\ u \in V\})$.

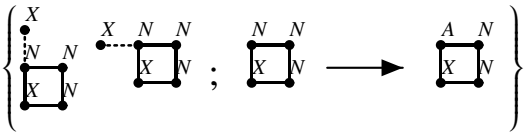


– Sinon, u est actif au moment de la suppression de v alors u devient l'élu, nous nous retrouvons partiellement dans le cas de la règle R_0.

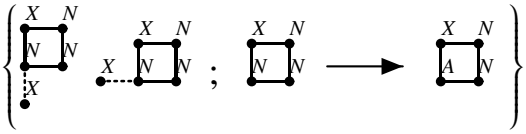


R_2 : Si le degré de v est 2 et qu'il se trouve à gauche (en haut ou en bas) d'une cellule, il devient actif et génère une durée de vie qui est une v.a. exponentielle de paramètre égal à son poids.

R_{2_1} :

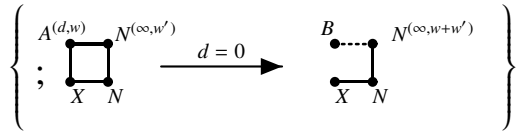


R_{2_2} :



Lorsque sa durée de vie est expirée, il est enlevé avec ses arêtes incidentes et son voisin de droite récupère son poids.

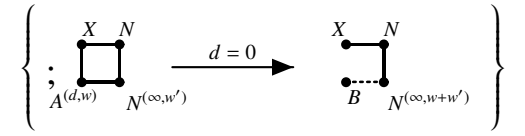
R'_{2_1} :



Pour plus de précision, soit $\{(x,y),(x+1,y),(x,y+1),(x+1,y+1)\}$ une cellule, si le degré du sommet (x,y) est 2 alors (x,y) est actif et une fois que sa durée de vie est expirée, son voisin $(x+1,y)$ collecte son poids.

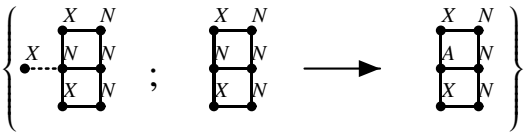
En raison de la symétrie horizontale, nous avons de la même manière, si $deg((x,y+1)) = 2$ alors $(x,y+1)$ est actif et son voisin $(x+1,y+1)$ collecte son poids à sa disparition.

R'_{2_2} :



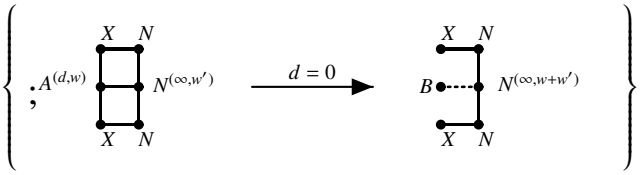
R_3 : Si $deg(v) = 3$, alors si v appartient à deux cellules et qu'il n'existe aucune arête sur son côté gauche, c'est-à-dire v a seulement une arête horizontale et les deux autres sont verticales, alors v devient actif.

R_3 :



Au moment où il disparaît, son voisin de droite récupère son poids.

R'_3 :



Ainsi, soient $\{(x,y),(x+1,y),(x,y+1),(x+1,y+1)\}$ et $\{(x,y),(x+1,y),(x,y-1),(x-1,y-1)\}$ deux cellules, si le degré du sommet (x,y) est 3 alors (x,y) devient actif et lorsque sa durée de vie s'achève, son voisin $(x+1,y)$ récupère son poids .

Lemme 6.4.1. *Le système de réécritures ci-dessus est noethérien [Sel04].*

Dans la suite, nous avons besoin de la définition suivante.

Définition 6.4.1. Un sommet qui appartient à un cycle maximal dans **P** s'appelle *sommet périphérique*.

Lemme 6.4.2. *Soit v un sommet actif de degré* 2 *ou* 3 *dans un polyominoïde* **P**. *Alors v est un sommet périphérique.*

Preuve. Soient **P**=(V,E) un polyominoïde et $v=(x,y)$, $v \in V$, un sommet actif de degré 2 ou 3. Par définition, v est situé dans une cellule, c'est-à-dire à l'intérieur d'un cycle. Soit γ un cycle maximal dans **P** contenant v à son intérieur. Si $v \in \gamma$, alors la preuve est complète. Sinon, il existe un sommet plus proche $u=(x',y)$ de γ tel que $x' < x$. Or, **P** est un polyominoïde et tout cycle γ dans **P** ne contient pas de trous ; les arêtes du segment $[(x',y),(x,y)]$ sont dans E. Ceci ne peut pas être vrai puisque v n'admet aucune arête à sa gauche. □

L'algorithme d'élection proposé ici enlève un sommet actif une fois sa vie expirée. Ainsi, la question que nous pouvons nous poser est : "*comment assurer la continuité du processus de suppression ?*"

Pour répondre à cette question nous devons prouver que le graphe résiduel préserve les propriétés propres aux polyominoïdes.

Proposition 6.4.1. *Soit* **P**=(V,E) *un polyominoïde de taille* ≥ 2 *et soit v un sommet actif dans* **P**. *Le graphe* **P′**$=(V \setminus \{v\}, E \setminus \{\{v,u\}, \forall\, u \in V\})$ *est un polyominoïde.*

Preuve. Soient **P**=(V,E) un polyominoïde de taille ≥ 2, v un sommet actif dans **P** et le graphe **P′**$=(V \setminus \{v\}, E \setminus \{\{v,u\}, \forall\, u \in V\})$. Pour montrer la proposition, nous devons montrer que **P′** est un graphe connexe sans trous.

- Si $deg(v)=1$, alors la suppression de v et de son arête incidente dans **P** n'introduit ni la déconnexion de **P′** ni la création d'un trou dans **P′**.

- Si $deg(v)=2$ ou $deg(v)=3$ alors soient v, v_1, v_2, v_3 quatre sommets rectangulaires d'un polyominoïde **P** tel que v est le sommet actif dont la durée de vie vient d'expirer (cas des régles R_2 et R'_2). Considérons un sommet $u \in V \setminus \{v, v_1, v_2, v_3\}$. Alors, si le sommet u est accessible à un sommet v_i, $1 \leq i \leq 3$, à travers un chemin qui passe par v, alors lorsque v sera supprimé, u restera accessible à v_i par un autre chemin empruntant les sommets $v_{j\neq i,\ j=1,2,3}$. Par conséquent, selon le lemme 6.4.2, v est un sommet périphérique et sa suppression ne crée aucun trou.

□

6.4.2 Arbre couvrant standard

Soit **P**=(V, E) un polyominoïde. Le graphe $T = (V, F)$ qui représente la trace des transmissions de poids sur les arêtes de **P** sera construit d'une façon distribuée comme suit.

– Si $e = \{(x, y), (x + 1, y)\}$ est une arête dans E alors e appartient à F, c'est-à-dire que toute arête horizontale dans E appartient à F :

$$e = \{(x, y), (x + 1, y)\} \in E \Longrightarrow e \in F$$

– Si $e = \{(x, y), (x, y + 1)\} \in E$ et e n'est pas une arête verticale gauche d'une cellule dans **P** alors $e \in F$, c'est-à-dire que toute arête verticale, qui n'est pas une arête gauche d'une cellule, appartient à F :
$e = \{(x, y), (x, y + 1)\} \in E \Longrightarrow e \in F$ ssi $\{(x, y), (x + 1, y), (x + 1, y + 1), (x, y + 1)\}$ n'est pas une cellule dans **P**.

Le graphe T relie tous les sommets de **P** et c'est un arbre puisqu'il est acyclique. Ainsi, nous avons :

Proposition 6.4.2. *Le graphe $T = (V, F)$ décrit ci-dessus est un arbre couvrant du polyominoïde* **P**.

Preuve. Nous pouvons prouver cette proposition par une construction inductive de T sur **P** :

1. Si **P**=$(\{(x, y)\}, \emptyset)$, le polyominoïde se compose de seulement un sommet, alors la proposition est affirmée ($T = (\{(x, y)\}, \emptyset)$).
2. Soit **Q**=(V, E) un polyominoïde et soit $T = (V, F \subseteq E)$ l'arbre couvrant de **Q** obtenu par les règles ci-dessus. Considérons deux sommets voisins v et v' tels que $v \in V$ et $v' \notin V$. Conformément à la règle inductive vue dans la section 6.3, le graphe **P**=$(V \cup \{v'\}, E \cup \{\{v, v'\}\})$ est un polyominoïde. À présent, il reste à prouver que l'arbre $T' = (V \cup \{v'\}, F \cup \{\{v, v'\}\})$ est un arbre couvrant de **P**.
   Aucun cycle n'est créé lorsque la nouvelle arête $\{v, v'\}$ est ajoutée. Ainsi, le graphe joignant l'arbre couvrant de **Q** et l'arbre $(V_A = \{v, v'\}, E_A = \{\{v, v'\}\})$ est un arbre couvrant du polyominoïde **P**.
3. Soit **Q**=(V, E) un polyominoïde et soit $T = (V, F \subseteq E)$ son arbre couvrant. Supposons maintenant que V contienne 4 sommets voisins $v_1 = (x, y)$, $v_2 = (x+1, y)$, $v_3 = (x+1, y+1)$ et $v_4 = (x, y + 1)$ tel que les trois arêtes qui relient ces quatre sommets sont dans E et la quatrième, appelée e, ne l'est pas. Alors conformément à règle inductive de section 6.3, le graphe résiduel **P**=$(V, E \cup \{e\})$ après l'insertion de la nouvelle arête e est un polyominoïde. Néanmoins, il reste à prouver que la transmission des poids s'effectue à travers un arbre couvrant.

Soit C une cellule de $\mathbf{P}$=$(V, E \cup \{e\})$. Soient $e_S = \{v_1, v_2\}$, $e_E = \{v_2, v_3\}$, $e_N = \{v_3, v_4\}$ et $e_W = \{v_1, v_4\}$ les arêtes de C et soit $T' = (V, F')$ le graphe composé par les arêtes sur lesquelles les poids des sommets supprimés sont transmis à leurs successeurs dans $\mathbf{P}$. Alors, nous avons :

- Si $e = e_W$ alors $F' = F$. L'arête ajoutée e est du côté gauche de la cellule C, alors dans ce cas l'arbre couvrant ne change pas.
- Si $e = e_E$ alors
$$F' = F \setminus \{e_W\} \cup \{e_E\}.$$
- Si $e = e_S$ alors
$$F' = F \setminus \{e_W\} \cup \{e_S\}.$$
- Si $e = e_N$ alors
$$F' = F \setminus \{e_W\} \cup \{e_N\}.$$

Nous pouvons facilement voir que le graphe T' est un graphe connexe et, d'ailleurs, aucun cycle n'est engendré avec ces instructions. Par conséquent, T' est un arbre couvrant de $\mathbf{P}$.

□

Remarque 6.4.1. L'arbre couvrant construit par ces règles est unique.

Définition 6.4.2. L'arbre couvrant $T = (V, F)$ est appellé *arbre couvrant standard* du polyominoïde $\mathbf{P}$.

La Figure 14 donne l'arbre couvrant standard du polyominoïde donné dans la Figure 12.

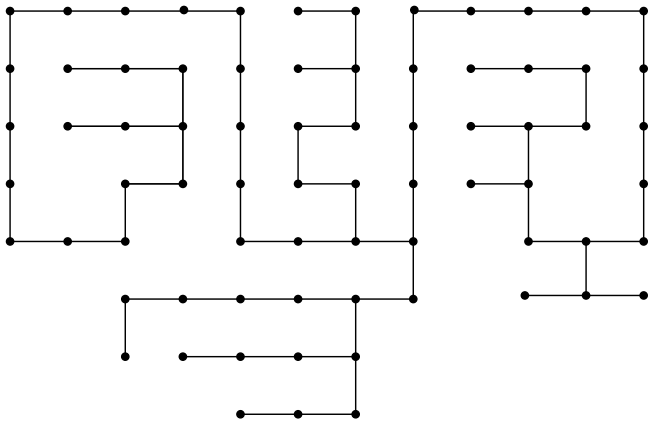


FIG. 14 – Arbre couvrant standard du polyominoïde donné dans la Fig. 12.

Proposition 6.4.3. *Soient* $\mathbf{P}$=(V, E) *un polyominoid et* $T = (V, F)$ *son arbre couvrant standard. Alors, un sommet* $v \in V$ *est actif dans* $\mathbf{P}$ *si et seulement s'il est une feuille dans* T.

Preuve.

⇒ Soit $v \in V$ un sommet actif dans $\mathbf{P}=(V,E)$. Nous allons montrer que v est une feuille dans l'arbre couvrant T de $\mathbf{P}$.

- Si $deg(v) = 1$, alors certainement, v est une feuille dans T.
- Si $deg(v) = 2$, alors v est une extrémité de l'arête gauche d'une cellule contenue dans $\mathbf{P}$. D'ailleurs, étant donné que v est actif et de degré 2, il ne peut pas avoir une arête horizontale incidente à son côté gauche. Les arêtes dont il est extrémité sont d'orientations différentes : une horizontale et une verticale. L'arête horizontale est dans F tandis que la verticale ne l'est pas. Par conséquent, v est une feuille dans T.
- Si $deg(v) = 3$, alors v appartient à deux cellules et il n'admet pas qu'une arête soit incidente de son côté gauche. La seule arête incidente à v inclus dans F est l'arête incidente horizontale qui est à sa droite. Donc, v est une feuille dans T.

⇐ Supposons maintenant que v est une feuille dans T et montrons qu'il est actif dans $\mathbf{P}$.

- Si $deg(v) = 1$, alors v est actif.
- Supposons que $deg(v) = 2$ dans $\mathbf{P}$. Les deux arêtes incidentes à v dans $\mathbf{P}$ ne peuvent être toutes les deux ni horizontales ni verticales, autrement, v ne sera pas une feuille dans T. Dans le cas où elles sont différentes, seulement l'horizontale est dans F et donc, selon les règles de construction de F, la cellule qui contient ces arêtes est dans $\mathbf{P}$. Par conséquent, v est actif.
- Si $deg(v) = 3$, alors avec un raisonnement semblable au cas de $deg(v) = 2$, nous pouvons prouver que seulement une des trois arêtes incidentes à v est dans F : une arête est horizontale et deux autres sont verticales. Les deux arêtes verticales ne sont pas dans F et chacune d'elles appartient à une cellule. L'arête horizontale ne peut être que incidente à v de son coté droit et elle est commune aux deux cellules formées par les verticales. Ainsi, nous nous retrouvons dans le contexte de la règle R_3 où le sommet v est actif.

□

Proposition 6.4.4. *Soient* $\mathbf{P}$ *un polyominoïde de taille* ≥ 2*,* T *son arbre couvrant standard et* v *est un sommet actif dans* $\mathbf{P}$*. Soient* $\mathbf{P}'$ *le polyominoïde résiduel lorsque* v *et ses arêtes incidentes sont supprimés et* T' *son arbre couvrant standard. Alors,* T' *peut être obtenu à partir de* T *par l'élimination de la feuille* v *et de son arête incidente.*

Preuve. Soient $\mathbf{P}$ un polyominoïde de taille ≥ 2 et T son arbre couvrant standard. Il est évident de voir que l'arbre résiduel T', après la suppression de la feuille v et de son arête incidente dans T, est un arbre couvrant du polyominoïde $\mathbf{P}'$. $\mathbf{P}'$ est le polyominoïde obtenu à partir de $\mathbf{P}$ une fois que v et ses arêtes incidentes sont disparus de $\mathbf{P}$. Maintenant, il reste à prouver que T' est l'arbre couvrant standard de $\mathbf{P}'$:

- Manifestement, les arêtes horizontales de $\mathbf{P}'$ sont dans T'.

– Les arêtes verticales de $\mathbf{P}'$ qui ne sont pas situées sur le côté gauche d'une cellule de $\mathbf{P}'$, satisfont la même condition dans $\mathbf{P}$. Elles sont alors dans T. En outre, elles sont dans l'arbre couvrant de $\mathbf{P}'$.

□

D'après les résultats précédent, nous pouvons résumer l'algorithme d'élection probabiliste distribué sur un polyominoïde $\mathbf{P}$ comme suit.

Tant que $\mathbf{P}$ n'est pas réduit à un sommet unique
faire
- tout sommet qui est actif ou devient actif (règles $R_0 - R_3$) génère sa durée de vie selon son poids,
- une fois que la durée de vie d'un sommet actif est expirée, il est enlevé avec ses arêtes incidentes et son voisin dans l'arbre couvrant standard collecte son poids.

fin

6.5 Analyse de l'algorithme

L'algorithme d'élection dans un polyominoïde est vu comme un algorithme d'élection dans son arbre couvrant standard : comme nous venons de le voir dans la proposition 6.4.3, chaque sommet actif dans un polyominoïde est une feuille dans son arbre couvrant standard et les poids de ce sommet dans les deux configurations sont égaux.

Soit $\mathbf{P}=(V,E)$ un polyominoïde. Initialement, tous les sommets ont le même *poids* 1 : $w(v) = 1, \forall v \in V$. Selon les règles vues dans la section 6.4, lorqu'un sommet actif disparaît, son successeur collecte son poids et l'additionne à son poids courant. A l'instant t où un sommet v devient actif dans le polyominoïde résiduel $\mathbf{P}'$, son poids est le nombre de sommets disparus de ses côtés[3]. La durée de vie $L(v)$ du sommet v est une variable aléatoire ayant une distribution exponnentielle de paramètre $\lambda(v) = w(v)$:

$$Pr(L(v) > t) = e^{-\lambda(v)t}, \quad \forall t \geq 0.$$

Nous disons que la mort du sommet actif v se produit selon un processus markovien de paramètre $\lambda(v)$ égal à son poids $w(v)$. Cette propriété est équivalente à : la probabilité de disparition de v dans l'intervalle de temps $[t, t+h]$ est $\lambda(v)h + o(h)$, lorsque $h \to 0$ à tout instant t, et celà indépendamment de ce qui se passe ailleurs et de ce qui s'est produit dans le passé. Le processus aléatoire est une variante du processus de mort pure qui est, à son tour, un exemple spécial du processus de Markov en temps continu.

[3]les sommets des sous arbres dont ses voisins sont les racines.

Théorème 6.5.1. *La stratégie décrite ci-dessous mène à une élection probabiliste* **totalement** *équitable : dans un polyominoïde, tous les sommets ont la même probabilité d'être élus.*

La preuve de ce théorème est très compliquée et ressemble en quelque sorte à celle vue dans le chapitre sur les arbres. Nous présentons, dans la sous-section suivante, quelques résultats préliminaires aidant à faire la preuve du théorème.

6.5.1 Uniformité de l'élection

L'élection probabiliste peut être modélisées mathématiquement par un processus de Markov en temps continu. L'état initial du processus est $\mathbf{P}=(V,E)$ (le polyominoïde en entier). Soit $\mathcal{E}_{\mathbf{P}}$ l'ensemble des états de tous les sous-polyominoïdes $\mathbf{Q}=(U,F)$ de $\mathbf{P}$ qui vérifient la propriété suivante : toutes les fois que deux sommets diagonaux de la forme $((x,y)$ et $(x+1,y-1))$ ou $((x,y)$ et $(x+1,y+1))$ sont dans $\mathbf{Q}$, alors le sommet de droite $(x+1,y)$ est dans U et ce s'il est dans V.

La proposition suivante montre que $\mathcal{E}_{\mathbf{P}}$ est l'ensemble de tous les sous-polyominoïdes qui peuvent être atteint à partir $\mathbf{P}$ par un ordre d'élimination des sommets actifs (rappelons que lorsque un sommet actif est supprimé, toutes ses arêtes incidentes sont aussi supprimées).

Proposition 6.5.1. *Un sous-polyominoïde* $\mathbf{Q}$ *d'un polyominoïde* $\mathbf{P}$ *peut être atteint avec une probabilité positive si et seulement si* $\mathbf{Q}$ *est dans* $\mathcal{E}_{\mathbf{P}}$.

Preuve.

- $\Rightarrow$ Soit $\mathbf{Q}$ un sous-polyominoïde atteint à partir de $\mathbf{P}=(V,E)$ avec une probabilité positive et prouvons que $\mathbf{Q}\in\mathcal{E}_{\mathbf{P}}$. Selon la définition du processus de transition, $\mathbf{Q}$ doit être obtenu à partir de $\mathbf{P}$ par $k-$ordre d'élimination de sommets actifs ($0\leq k<n$) où $|V|=n$. Pour $k=0$, la proposition est vraie. Supposons maintenant qu'elle est vraie pour k et prouvons qu'elle reste vraie pour $k+1$. Ainsi, soit $\mathbf{Q}$ un polyominoïde obtenu à partir d'un certain polyominoïde $\mathbf{R}$ de $\mathbf{P}$ par la suppression d'un sommet actif v et de ses arêtes incidentes. Par hypothèse de récurrence, $\mathbf{R}$ est dans $\mathcal{E}_{\mathbf{P}}$ et puisque aucun sommet de degré ≥ 2 se trouvant au côté droit d'une cellule de $\mathbf{R}$ n'est actif, le sous-polyominoïde résiduel $\mathbf{Q}$ satisfait aussi la condition d'être dans $\mathcal{E}_{\mathbf{P}}$.
- $\Leftarrow$ Soit maintenant $\mathbf{Q}=(U,F)$ un polyominoïde dans $\mathcal{E}_{\mathbf{P}}$. Nous prouvons par une récurrence décroissante sur $m=|U|$ que $\mathbf{P}$ peut être atteint par $n-m$ transitions avec une probabilité positive. Pour $m=n$, $\mathbf{Q}$ est égal à $\mathbf{P}$ et par conséquent, $\mathbf{Q}\in\mathcal{E}_{\mathbf{P}}$. Supposons que $m<n$, nous devons donc montrer qu'il existe un sommet $v\in V\setminus U$, tel que son ajout à U avec un certain nombre d'arêtes qui le relient aux sommets de $\mathbf{Q}$, produit un nouveau polyominoïde $\mathbf{R}$ appartenant à $\mathcal{E}_{\mathbf{P}}$. Or, puisque $m<n$, il existe un sommet $u\in V\setminus U$. Considérons alors un chemin qui relie u à un sommet $s\in U$, soit $v\notin U$ le dernier sommet de ce chemin, alors v a des sommets voisins dans U, voir la Fig. 15.
  - Si v a un seul sommet voisin dans U, alors v est un sommet actif dans $\mathbf{R}$ (son degré est 1 dans $\mathbf{R}$). Ainsi, $\mathbf{R}$ est dans $\mathcal{E}_{\mathbf{P}}$.

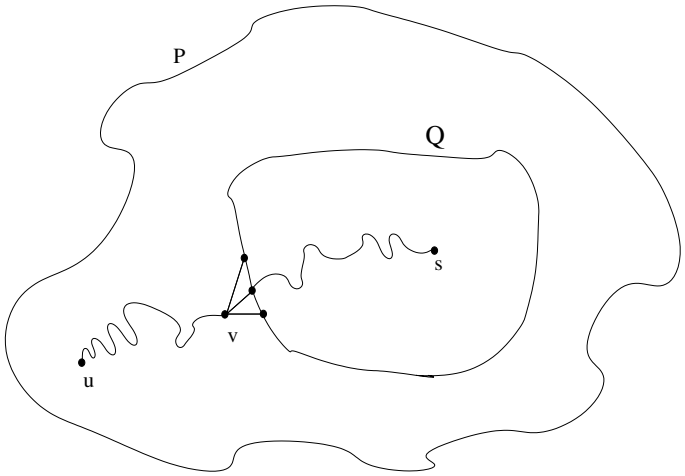


FIG. 15 – Accessibilité de **P** à partir d'un sous-polyominoïde $\mathbf{Q} \in \mathcal{E}_{\mathbf{P}}$

– Sinon, v a deux ou trois sommets voisins dans U. Dans ce cas, soit (x, y) les coordonnées du sommet v. Selon l'hypothèse que **Q** est dans $\mathcal{E}_{\mathbf{P}}$, il n'y pas de sommets voisins de v dans U de la forme $((x, y+1)$ et $(x-1, y))$ ou $((x-1, y)$ et $(x, y-1))$, tels que v soit un sommet situé au côté droit d'une cellule de **R** composée par ces sommets. Par conséquent, v ne peut être qu'au côté gauche d'une ou de deux cellules de **R**. En outre, v est un sommet actif dans **R**. Par conséquent, $\mathbf{R} \in \mathcal{E}_{\mathbf{P}}$.

□

Soit **Q** l'état du système à l'instant t. Selon la structure aléatoire distribuée de l'algorithme, tout sommet actif v de **Q** a une durée de vie exponentiellement distribuée de paramètre égal à son poids.

Ce fait est équivalent au fait que dans l'intervalle de temps $[t, t+\Delta t]$, v peut disparaître avec toutes ses arêtes incidentes avec la probabilité $w(v)\Delta t + o(h)$, quand $h \to 0$, et ceci indépendamment de ce qui se passe ailleurs dans le reste du polyominoïde et aussi indépendamment de ce qu'il s'est produit dans le passé.

Nous pouvons, ainsi, montrer que la probabilité de passage du polyominoïde **Q** au polyominoïde **R**, obtenu par la suppression du sommet actif v et de toutes ses arêtes incidentes, est :

$$P_{(\mathbf{Q},\mathbf{R})} = \frac{w(v)}{\sum_{u \ actif \ dans \ \mathbf{Q}} w(u)} \tag{9}$$

à condition que **Q** ne soit pas réduit à un sommet. Les états absorbants (voir [Fel50]) sont les polyominoïdes réduits à un sommet (sommet élu).

Les propriétés ci-dessous caractérisent le processus d'élimination dans un polyominoïde.

- *Le taux de mort du polyominoïde* $\mathbf{P}$ *est :* $\lambda(\mathbf{P}) = w(\mathbf{P}) = \sum_{u \ actif \ dans \ \mathbf{P}} w(u)$
- *La durée de vie de* $\mathbf{P}$ *:* $L(\mathbf{P}) = min_u\{L(u),\ u \ actif \ dans \ \mathbf{P}\}$ *a la fonction de distribution :*

$$Pr(L(\mathbf{P}) \leq x) = 1 - Pr(L(\mathbf{P}) \geq x) = 1 - e^{-x\lambda(\mathbf{P})},\ \forall x \in \mathbb{R}^+$$

Proposition 6.5.2. *Soit* $\mathbf{Q}$ *un polyominoïde dans* $\mathcal{E}_{\mathbf{P}}$ *et soit* $P_{\mathbf{Q}}(t)$ *la probabilité que l'état de l'élection au temps t est* $\mathbf{Q}$*. Nous avons :*

(i) $\dfrac{dP_{\mathbf{P}}(t)}{dt} = -w(\mathbf{P})P_{\mathbf{P}}(t),$

(ii) pour tout sous-polyominoïde $\mathbf{Q} \neq \mathbf{P}$ *de taille au moins 2 et qui appartient à* $\mathcal{E}_{\mathbf{P}}$*,*

$$\frac{dP_{\mathbf{Q}}(t)}{dt} = -w(\mathbf{Q})P_{\mathbf{Q}}(t) + \sum_{v} w(v)P_{\mathbf{R}}(t),$$

avec $\mathbf{R} = \mathbf{Q} \cup (\{v\}, \{\{v, u\}, u \text{ adjacent à } v \text{ dans } T\})$*, (rappelons que T est l'arbre couvrant standard de* $\mathbf{P}$*)*

où la sommation est effectuée sur tous les sommets v adjacent à $\mathbf{Q}$ *dans T n'appartenant pas à* $\mathbf{Q}$*, et*

(iii) $\dfrac{dP_{(\{v\},\emptyset)}(t)}{dt} = \sum_{u \text{ adjacent à } v \text{ dans } \mathbf{P}} w(u)P_{(\{v,u\},\{\{v,u\}\})}(t),$

avec la condition initiale $P_{\mathbf{P}}(0) = 1$.

Preuve. La preuve de cette proposition est semblable à celle de la proposition 4.3.3 du chapitre 4.

□

La solution du système d'équations différentielles de la proposition 6.5.2 donne une description mathématique de la probabilité d'être dans l'état $\mathbf{Q}$ à l'instant t. Elle caractérise *en principe* la probabilité de distribution des états à un instant donné t. En particulier, elle devrait nous permettre de calculer les probabilités d'absorption [Fel50].

Les propositions 6.4.2-6.5.1 permettent de confirmer que tout ordre de transition sur $\mathcal{E}_{\mathbf{P}}$ peut être simulé, avec la même probabilité, par un ordre de transition sur l'ensemble d'arbres facteurs de l'arbre couvrant standard de $\mathbf{P}$ (rappelons qu'un arbre facteur d'un arbre, cf. chapitre 4 § 4.3, est un sous-arbre obtenu par un ordre d'élimination de feuilles). Ainsi, l'étude du processus d'élection dans un polyominoïde est traduite par une étude d'élection sur son arbre couvrant standard [MSZ03b].

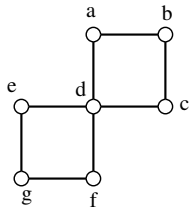


FIG. 16 – Exemple d'étude

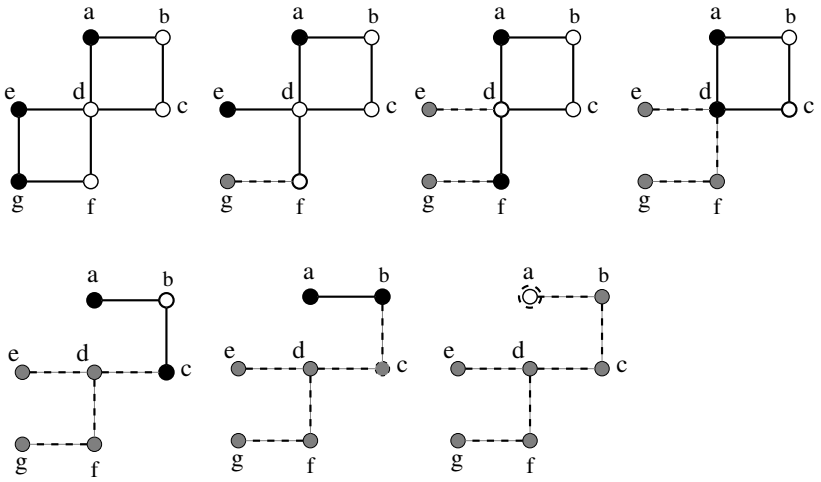


FIG. 17 – Élection dans un polyominoïde : Les sommets colorés en noir sont les sommets actifs, ceux en gris et les arêtes pointillées sont ceux de l'arbre couvrant standard. Le sommet doublement encerclé est l'élu.

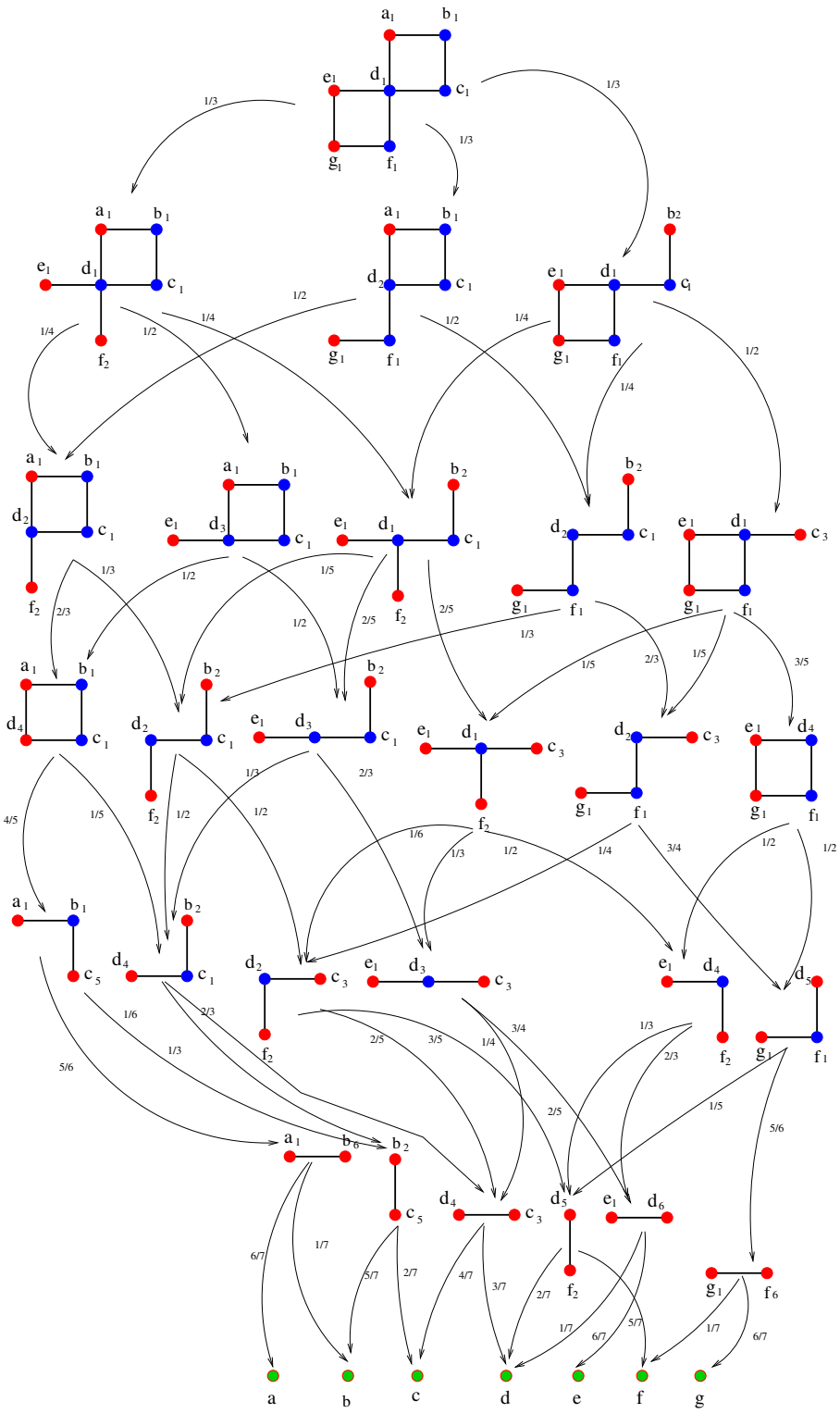


FIG. 18 – Graphe de transitions

Exemple 6.5.1. Considérons le polyominoïde de la Fig. 16. Le graphe de transitions du processus d'élection est donné dans la Fig. 18. Dans ce graphe, les indices en dessous des noms des sommets représentent leurs poids courants. Il existe trois ordres d'élimination qui aboutissent à l'élection du sommet a. La séquence des graphes de la Fig. 17 présente un de ces ordres.

Les étapes du processus d'élimination peuvent aussi être vues comme une construction d'un arbre couvrant, où la racine de cet arbre est le sommet élu.

Initialement, tout sommet actif commence le calcul en générant sa durée de vie. Une fois que la durée de vie d'un sommet actif est expirée, il est enlevé avec toutes ses arêtes incidentes. Dans la Fig. 17, les Arêtes, sur lesquelles les poids sont transmis, sont représentées par les arêtes pointillées, tandis que les autres sont enlevées du graphe. A une étape intermédiaire du processus d'élimination, les arêtes pointillées constituent des arbres couvrant partiels. Par conséquent, nous nous sommes assurés qu'à la fin tous les arbres couvrants partiels seront fusionnés en un seul arbre couvrant. Or, cet arbre est unique, alors il sera considéré comme l'arbre couvrant standard du polyominoïde. Ainsi, l'étude de l'algorithme d'élection dans un polyominoïde est projetée à une étude d'élection dans son arbre couvrant standard.

Nous avons vu dans le modèle de l'arbre qu'initialement tous les sommets ont le même poids 1 et que chaque feuille a une durée de vie qui est une variable aléatoire exponentiellement distribuée avec un paramètre égal à son poids. Au moment où la durée de vie d'une feuille est expirée, elle est enlevée avec son arête incidente et son poids est récupéré par son père. Ce processus de suppression continue jusqu'à ce que l'arbre soit réduit à un seul sommet. Le sommet restant est considéré par la suite comme l'élu. Par conséquent, la probabilité d'élire un sommet v dans un polyominoïde $\mathbf{P}$ est identique à la probabilité d'élection dans son arbre couvrant standard. Celle-ci s'est avérée être égale à $1/n$, où n est la taille de $\mathbf{P}$.

La table suivante représente les résultats empiriques de l'algorithme précédent appliqué sur le polyominoïde donnée dans la Fig. 16. La taille de la simulation est 1 000 000.

Sommets	Probabilités
a	0.142797
b	0.142799
c	0.142840
d	0.142970
e	0.142892
f	0.142980
g	0.142722

TAB. 2 – Résultat de la simulation du polyominoïde de la Fig. 16.

Dans la suite, nous supposons que T est l'arbre couvrant standard d'un polyominoïde $\mathbf{P}$ de taille n. Les feuilles de T sont enlevées suivant un processus aléatoire comme décrit ci-dessus jusqu'à ce que T soit réduit à un sommet unique. Nous devons prouver l'uniformité de l'élection pour tous les sommets de T.

D'abord, nous introduisons une légère modification sur le modèle d'élimination des feuilles dans T. Nous traduisons le modèle en une variante sur les arbres orientés. En effet, pour un sommet v donné, l'unique arbre enraciné en v peut être construit. Ces arbres enracinés peuvent être employés d'une manière simple pour calculer les probabilités d'absorption.

Nous considérons une forêt d'arbres enracinés. Soit F une forêt d'arbres enracinés, nous présentons un processus de mort sur F comme suit. Chaque feuille v a une durée de vie exponentiellement distribuée de paramètre égal à son poids ; au commencement, tous les sommets de F ont un poids de égal à 1. À tout intervalle de temps $[t, t + \Delta t]$, si la durée de vie d'une feuille est expirée, la feuille est enlevée avec son unique arête incidente. Si la feuille disparue possède un père, alors son père prend son poids, et l'ajoute à son poids. Le processus d'élimination de feuilles continue sur la forêt réduite jusqu'à l'extinction totale de la forêt.

Preuve. [Preuve du théorème principal 6.5.1]
Soient F_1 et F_2 deux forêts de taille respectivement n_1 et n_2. Nous pouvons utiliser les résultats des propositions 4.4.1-4.4.3 (cf. chapitre 4) pour faire une preuve au théorème 6.5.1. En effet, la preuve peut s'effectuer en s'appuyant sur le fait qu'il existe une similitude d'exécution entre le processus d'élection sur **P** et celui sur son arbre couvrant standard T_P. Par conséquent, pour tout sommet v du polyominoïde **P**, la probabilité d'élire v dans **P** est égale à la probabilité d'élire ce sommet dans T_P.

□

Troisième partie

Synchronisation

Chapitre 7

Analyse d'un algorithme probabiliste de rendez-vous avec agendas dynamiques

Sommaire

Dans ce chapitre nous présentons et analysons un algorithme de rendez-vous basé sur des délais aléatoires. Cet algorithme peut être considéré comme un algorithme distribué probabiliste pour trouver un couplage maximal. Chaque processus p du réseau génère pour chaque voisin q un nombre aléatoire, choisi uniformément dans l'intervalle réel $[0, 1]$. Ce nombre représente la durée d'attente nécessaire pour obtenir un rendez-vous avec ce voisin.

Initialement, le nombre de rendez-vous proposés par les processus est le double du nombre de liens entre eux (i.e. une proposition pour chaque demi-arête). L'agenda d'un processus est constituée par l'ensemble des temps générés par ce processus pour l'ensemble de ses voisins. Au moment où l'horloge atteint le plus petit temps de l'ensemble des temps générés, il y aura un rendez-vous entre le processus qui propose ce temps et le voisin à qui est assigné ce temps de "retrouvaille". Les deux processus en rendez-vous annulent tous les autres rendez-vous avec

leurs autres voisins. En effet, ces derniers suppriment de leurs agendas les temps qu'ils proposent à ces deux processus en rendez-vous. L'algorithme continue à s'exécuter par les processus restants équipés par des agendas modifiées jusqu'à ce que le temps d'un tour (une unité de temps) soit expiré. L'espérance du nombre de rendez-vous trouvée par cet algorithme, dans un tour, est essentiellement plus grande que celle obtenue dans [MSZ03a]. Par conséquent, notre algorithme distribué de rendez-vous est plus efficace que celui de [MSZ03a].

Ce chapitre est organisé comme suit. Dans la section 7.1 nous présentons le modèle et les hypothèses considérés dans ce chapitre, nous donnons une description informelle de notre algorithme. La section 7.3 concerne le nombre de rendez-vous, l'espérance du nombre de rendez-vous, l'impact d'une insertion de lien et d'une extension et finalement l'espérance du nombre de rendez-vous dans les graphes aléatoires. La section 7.4 étudie l'efficacité de l'algorithme en donnant des bornes pour les graphes en général et, en particulier, une borne plus serrée pour les arbres.

7.1 Modèle et hypothèses d'étude

Nous considérons le modèle du système distribué vu dans le chapitre 3. Les processus ou les sommets sont anonymes et chacun d'eux est capable de distinguer ses portes ; il sait par quelle port un message est reçu ou envoyé. Soit δ une fonction qui numérote les portes, nous supposons que pour chaque sommet v et pour tout voisin u, $\delta_v(u)$ est un entier unique appartenant à $[1, deg(u)]$.

Notre algorithme est synchrone : les processus ont accès à une horloge physique globale comme dans [Tel00] page 87. L'horloge fournit des valeurs réelles qui croissent avec le temps t, nous supposons que :

1. l'horloge est globale et commune à tous les processus,
2. tout événement, qui a lieu à un certain temps, est de durée nulle.

Nous supposons aussi que les communications ne prennent aucun temps, c'est-à-dire que le délai entre la décision d'un sommet et la notification à ses voisins est égal à 0.

7.2 Le problème

Nous sommes concernés par l'établissement de communications par des signaux de synchronisation ou d'une manière équivalente par un ordonnancement distribué local qui trouve des couplages entre paires de processus afin d'établir des communications.

Cette implémentation peut être réalisée par l'établissement des rendez-vous entre les processus. Reif et Spirakis ont décrit dans [RS82] (annexe p. 93) le problème de rendez-vous comme suit : *nous supposons que tout processus a une ressource spéciale appelée* **canal** *qui peut être dans l'état* **ouvert** *ou l'état* **fermé**. *Un rendez-vous entre le processus p et le processus q au temps*

t est une combinaison des états des deux processus au temps t de sorte que les deux canaux de p et q soient ouverts en même temps.

Ainsi nous sommes concernés par des signaux locaux de sorte que tout processus indique à au plus un voisin sa capacité à envoyer ou à recevoir des données.

Le problème du rendez-vous dans un réseau distribué coïncide avec celui du couplage dans un graphe. Rappelons la définition d'un couplage.

Définition 7.2.1. Un *couplage* dans un graphe simple non orienté $G = (V, E)$ est un sous-ensemble d'arêtes $\mathcal{M}$ de E, tel que $\forall (e_1, e_2) \in \mathcal{M}^2, e_1 \cap e_2 \in \{\emptyset, e_1\}$.

Définition 7.2.2. Un couplage de G est *maximum* si aucune arête ne peut être ajoutée sans violer la contrainte de la définition 7.2.1.

Un couplage de cardinalité maximale est un couplage maximal ; l'inverse n'est pas vrai. La cardinalité maximale s'appelle *nombre de couplage*.

7.2.1 L'Algorithme

Nous considérons une variante du protocole de communication décrit dans [FR80] (section 3.1 p. 377-378). Le même genre de messages de commande pour négocier des communications est employé dans [Dem98]. Notre solution est symétrique et entièrement distribuée.

Procédure HS
Tout processus p exécute pendant chaque tour (une unité de temps) les instructions suivantes :

Le processus p génère $t_p(q)$: *une variable aléatoire (v.a.) choisie uniformément dans l'intervalle réel* $[0, 1]$ *pour tout voisin q*;
Le processus p attend jusqu'à ce que l'un des trois événements suivants se produise :

- *p reçoit* 1 *d'un voisin r ; p envoie* 0 *à tous ses autres voisins.*
  (Il y a rendez-vous entre p et r. *)*
- $t = t_p(q)$ *et p n'a reçu aucun message de q*; *p envoie* 1 *à q et* 0 *à tous ses autres voisins.*
  (Il y a rendez-vous entre p et q. *)*
- $t = 1$.
  (Le tour se termine sans participation de p à un rendez-vous.*)*

Losqu'un processus reçoit 0 via une de ses portes, le rendez-vous assigné à cette porte est annulé.

Remarque 7.2.1. Puisque le temps est continu, nous sommes assurés et avec une probabilité 1 que seulement un de ces événements se produira.

7.2.2 Résultats et travaux précédents

Le reste de ce chapitre sera consacré à l'analyse de la procédure HS. Nous commençons d'abord par étudier le *nombre moyen de rendez-vous*, celui-ci n'est que la moyenne de la taille du couplage obtenue. A ce stade, nous donnons une formule récursive qui pourrait être employée pour calculer ce nombre en général, c'est-à-dire pour tout graphe. Nous donnons aussi des formes serrées ou des valeurs asymptotiques pour quelques familles intéressantes de graphes.

Pour les graphes aléatoires de degré moyen constant c, quand la taille tend vers l'infini, nous montrons que la probabilité pour qu'un rendez-vous ait lieu sur une arête donnée tend vers $1/(1+c)$.

Souvent, l'efficacité d'un algorithme n'est connue que de manière asymptotique, c'est-à-dire pour de grandes valeurs du paramètre n, alors que pour une utilisation dans des conditions réalistes de l'algorithme, ce paramètre reste de taille modérée. Nous considérons le *rapport d'efficacité*, qui est le rapport entre l'espérance du nombre de rendez-vous et le nombre de couplage (couplage de cardinalité maximale). Pour des graphes en général ce rapport est au moins $1/2$ et cette borne ne peut pas être plus petite à cette valeur. Pour les arbres nous trouvons une borne inférieure légèrement supérieure à $136/177$ ou environ 0.768, que nous conjecturons être optimale ; une famille de graphes appelée *mille-pattes* a un rapport d'efficacité asymptotique environ 0.769.

Nous comparons notre algorithme avec celui proposé par [RS82] (p. 225) et analysé dans [MSZ03a]. Dans un graphe quelconque, une arête a au moins la même probabilité d'être choisie par le nouvel algorithme, comme c'était le cas avec le précédent. Ceci implique que la taille des couplages donnée par le nouvel algorithme domine celle fournit dans [MSZ03a], à la fois dans l'espérance et dans la distribution. Pour certains graphes tels que les graphes complets, la différence est considérable.

Un algorithme semblable a été présenté par [HKP01] pour traiter le nombre de couplages dans les graphes. Cependant, nos investigations divergent dans leurs buts et leurs analyes.

7.3 Nombre de rendez-vous

L'algorithme commence par générer uniformément $2|E|$ variables aléatoires réelles indépendantes (v.a) dans l'intervalle réel $[0,1]$: une variable aléatoire pour chaque demi arête. Nous supposons que les $(2|E|)!$ permutations sur l'ensemble de ces nombres réels ont la même probabilité. C'est l'hypothèse principale sur laquelle se base la suite de notre analyse.

En effet, pour maintenir cette hypothèse, nous devons tout simplement postuler que, pour chaque arête $e = \{u, v\}$ dans G, l'algorithme produit deux v.a. continues $X_e(u)$ et $X_e(v)$, qui correspondent à $t_u(v)$ et à $t_v(u)$ dans la description de l'algorithme, supposons que ces v.a. associées aux arêtes soient *toutes* indépendantes. Le premier rendez-vous a lieu sur une arête $e = \{u, v\}$, si au moins une des deux v.a. associées, $X_e(u)$ ou $X_e(v)$, est minimale dans le graphe entier. Ainsi, pour le premier rendez-vous sur G, toute arête a la même chance $1/|E|$ d'être choisie.

L'attribution du premier rendez-vous aux sommets u et v sur l'arête $\{u, v\}$, implique que ces deux sommets et leurs arêtes d'incidentes sont enlevées du graphe. L'algorithme continue à s'exécuter sur le graphe résiduel (préservant les générations aléatoires des arêtes restantes) jusqu'à ce qu'aucune arête ne demeure dans l'ensemble des arêtes du graphe. Le *nombre de rendez-vous* dans un tour est tout simplement le nombre total d'arêtes auxquelles des rendez-vous sont assignés. Nous notons par $R(G)$ ce nombre. C'est une v.a. entière. Elle prend la valeur 0 avec probabilité 1 si $E = \emptyset$, la valeur 1 avec probabilité 1 si $|E| = 1$. En général, elle prend une valeur de l'ensemble de toutes les cardinalités des couplages maximaux dans G, avec une certaine probabilité.

Il semble utile de noter le fait suivant qui s'avère facile à prouver :

Fait 7.3.1. *Soit le premier rendez-vous assigné à l'arête $e = \{u, v\}$. Soit $G_e = (V', E')$ le graphe obtenu en enlevant du graphe G les sommets u et v et toutes leurs arêtes incidentes. Alors, toutes les $(2|E'|)!$ permutations des v.a. générées (une v.a. pour chaque demi arête) dans G_e, conditionnées par le fait que $X_e(u)$ ou $X_e(v)$ est minimale dans le graphe entier, ont la même probabilité $1/(2|E'|)!$. Ceci signifie que pour préserver l'hypothèse initiale sur le nouveau graphe, aucune autre génération aléatoire n'est nécessaire, et par conséquent, ils peuvent être identifiés avec les $(2|E'|)!$ valeurs générées restantes.*

Ce fait nous autorise à calculer récursivement la distribution de probabilité de $R(G)$. Ainsi, soit $G = (V, E)$ un graphe. Si $E = \emptyset$, nous avons $R(G) = 0$. Nous pouvons aisément prouver la proposition suivante.

Proposition 7.3.1. *Soit $G = (V, E)$ un graphe avec $n = |V| \geq 2$ et $m = |E| \geq 1$. Nous avons alors*

$$Pr(R(G) = k) = \frac{1}{m} \sum_{e \in E} Pr(R(G_e) = k - 1), \quad \forall k \geq 1. \tag{10}$$

Il convient de noter que pour $m \geq 1$, cet algorithme permet d'avoir au moins 1 rendez-vous avec probabilité 1 dans un tour alors que ce n'est pas le cas dans [MSZ03a].

- Dans le cas d'un graphe complet de taille n, nous avons avec probabilité 1, $R(G) = \lfloor \frac{n}{2} \rfloor$.
- Dans le cas d'un graphe en étoile de taille $n \geq 2$, nous avons avec probabilité 1, $R(G) = 1$.
- Dans le cas d'une double-étoile de taille $n \geq 4$, la Fig. 19, nous avons avec probabilité 1, $R(G) = n/2$.
- Soit G le graphe de la Fig. 20. Un simple calcul donne :
$$Pr(R(G) = 1) = \tfrac{1}{5} \text{ et } Pr(R(G) = 2) = \tfrac{4}{5}.$$

Soit maintenant la v.a. $M_e(G)$. Elle vaut 1 s'il y a rendez-vous sur une arête et vaut 0 sinon et ce pour toute arête $e = \{u, v\}$ du graphe $G = (V, E)$ (dans un tour). La v.a. $R(G)$ peut être écrite comme la somme $\sum_{e \in E} M_e(G)$. D'autre part, nous avons :

Proposition 7.3.2. *Soit $F(e)$ l'ensemble des arêtes de G non incidentes aux extrémités de l'arête e. La v.a. $M_e(G)$ peut être caractérisée récursivement par*

$$Pr(M_e(G) = 1) = \frac{1}{m} + \frac{1}{m} \sum_{f \in F(e)} Pr(M_e(G_f) = 1). \tag{11}$$

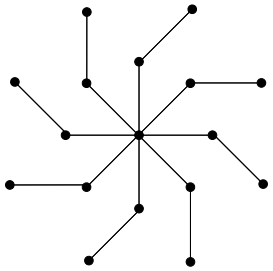


FIG. 19 – Double-étoile

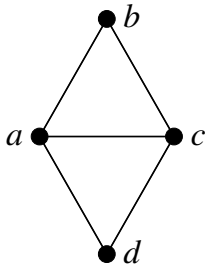


FIG. 20 – Un graphe simple

Dans [MSZ03a], les auteurs proposent et analysent un algorithme probabiliste basé sur le choix uniforme d'un sommet voisin :

MSZ procédure

Chaque sommet exécute en boucle les actions suivantes :

- *le sommet v choisit au hasard un de ses voisins* $c(v)$ *;*
- *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 un rendez-vous entre v et* $c(v)$ *si v reçoit* 1 *de* $c(v)$ **)*

A chaque fois que deux sommets voisins se choisissent mutuellement, il y a rendez-vous entre les deux sommets. La probabilité d'un rendez-vous sur une arête $e = \{u, v\}$ dans un tour en appliquant MSZ-algorithme est égale à $\frac{1}{d(u)d(v)}$, où $d(u)$ et $d(v)$ sont respectivement le degré du sommet u et celui du sommet v.

La proposition suivante montre que notre algorithme est plus efficace que MSZ-algorithme. En effet, la probabilité d'avoir un rendez-vous sur une arête donnée par le nouvel algorithme est supérieure ou égale à la probabilité du même événement dans l'algorithme précédent.

Proposition 7.3.3. *Soit $e = \{u, v\}$ une arête dans $G = (V, E)$. Nous avons*

$$Pr(M_e(G) = 1) \geq \frac{1}{d(u)d(v)}.$$

Preuve. Par induction sur $m = |E|$. La proposition se prouve facilement pour $m = 1$ et $m = 2$. Supposons maintenant qu'elle soit vraie pour les graphes ayants un nombre d'arêtes inférieur à m et montrons qu'elle reste vraie pour un graphe G avec m arêtes. Selon la proposition 7.3.2, la probabilité étudiée peut s'écrire :

$$Pr(M_e(G) = 1) = \frac{1}{m} + \frac{1}{m} \sum_{f \in F(e)} Pr(M_e(G_f) = 1), \tag{12}$$

où G_f est le graphe résiduel après la suppression de e et de toutes les arêtes incidentes à ses extrémités dans G.

Dans la somme, il y a $m - d(u) - d(v) + 1$ termes. Si ce nombre est zéro, alors

$$\frac{1}{m} = \frac{1}{d(u) + d(v) - 1} \geq \frac{1}{d(u)d(v)} \tag{13}$$

et la proposition est prouvée. Sinon, selon l'hypothèse d'induction, chaque terme de la somme est supérieur ou égal à $\frac{1}{d(u)d(v)}$ et, par conséquent :

$$Pr(M_e(G) = 1) \geq \frac{1}{m} + \frac{1}{m}(m - d(u) - d(v) + 1)\frac{1}{d(u)d(v)}. \tag{14}$$

Il est alors facile de montrer que la dernière expression est supérieure ou égale à $\frac{1}{d(u)d(v)}$.

□

En fait dans la majorité des cas, la probabilité d'avoir un rendez-vous sur une arête par le nouvel algorithme, est sensiblement plus grande que celle trouvée dans [MSZ03a]. En effet, il est possible de construire des exemples simples dans lesquels $d(u)d(v)$ est beaucoup plus grand que m et, par conséquent, la borne inférieure grossière $\frac{1}{m}$, pour $Pr(M_e(G) = 1)$, dépasse la valeur $\frac{1}{d(u)d(v)}$.

7.3.1 Espérance mathématique du nombre de rendez-vous

Dans un algorithme distribué probabiliste, un paramètre important qui mesure l'efficacité est l'espérance du nombre d'événements qui peuvent se produire pendant un tour de l'algorithme, voir [MSZ03a]. Dans ce qui suit, nous étudierons ce paramètre pour notre nouvel algorithme.

Notons par $\overline{R}(G)$ l'espérance mathématique du $R(G)$. Nous avons :

$$\overline{R}(G) = \mathbb{E}(R(G)) = \sum_k k.Pr(R(G) = k) = \sum_{e \in E} \mathbb{E}(M_e(G)).$$

($\mathbb{E}$ dénote l'espérance mathématique.)

Nous dérivons facilement de la proposition 7.3.3 le résultat de la proposition suivante.

Proposition 7.3.4. *Pour tout graphe, l'espérance mathématique du nombre de rendez-vous dans MSZ-algorithme n'excède pas celle du nouvel algorithme.*

Considérons quelques exemples particuliers.

1. Graphes complets. Pour le graphe complet K_n de taille n, nous avons $\overline{R}(K_n) = \lfloor \frac{n}{2} \rfloor$. Comparé avec l'espérance du nombre de rendez-vous dans un tour pour l'MSZ-algorithme, qui est $n/2(n-1)$, pour K_n, $n \geq 2$, nous concluons que le nouvel algorithme est beaucoup plus efficace.

2. Etoiles. Si G est un graphe en étoile de taille $n \geq 2$, nous avons $\overline{R}(G) = 1$.

3. Double-étoiles. Pour une double-étoile (la Fig. 19) de taille $n \geq 4$, nous avons $\overline{R}(G) = n/2$.

4. Chaîne. Comme dernier exemple, nous considérons des chaînes et nous calculons l'espérance asymptotique du nombre de rendez-vous pour ces graphes. Soit $G_n = (V_n, E_n)$ une chaîne telle que $|V_n| = n$, et soit $R_n = \overline{R}(G_n)$ l'espérance du nombre de rendez-vous dans G_n. Nous avons le lemme suivant :

Lemme 7.3.1.

1. *L'espérance du nombre rendez-vous dans les chaînes satisfait la récurrence :*

$$\forall n \geq 2,\ R_n = 1 + \frac{2}{n-1} \sum_{i=0}^{n-2} R_i \ \ et\ \ R_0 = R_1 = 0.$$

2. *Si nous notons par $R(z)$ la fonction génératrice ordinaire (FGO) définie par $R(z) = \sum_{n \geq 0} R_n z^n$, alors $R(z) = z(1 - e^{-2z})/(2(1-z)^2)$.*

Preuve. La première clause est triviale. En revanche, pour prouver la seconde, nous dérivons de la clause (i) la récurrence suivante :

$$R_{n+1} = \frac{n-1}{n} R_n + \frac{2}{n} R_{n-1} + \frac{1}{n}. \tag{15}$$

Les séries génératrices permettent bien de résoudre les équations récurrentes *non homogènes à coefficients variables.* Maintenant, une simple transformation de l'équation ci-dessus aux fonctions génératrices ordinaires *FGO* correspondantes nous ramène à l'équation différentielle suivante :

$$z(1-z)R'(z) - (1 - z + z^2)R(z) = \frac{z^2}{1-z}, \tag{16}$$

La résolution de l'équation récurrente (15) revient à résoudre l'équation différentielle sous-jacente (16) qui est moins complexe (les ouvrages [AG05, Lav05] donnent une bonne vue sur la résolution d'équations récurrentes). L'équation différentielle (16) admet pour solution la fonction

$$R(z) = z(1 - e^{-2z})/(2(1 - z)^2), \tag{17}$$

□

Remarque 7.3.1. Cette fonction génératrice est semblable à la fonction génératrice correspondante au problème "Seating Arrangement Problem" proposé et résolu par Flajolet dans [Fla97].

La FGO présentée dans le lemme précédent peut être utilisée pour trouver l'expression explicite du terme R_n. En effet, un simple calcul mène à :

$$R_n = \frac{1}{2}\sum_{i=1}^{n-1} \frac{(-1)^{n+1}2^i}{i!}(n - i). \tag{18}$$

La valeur asymptotique de R_n nous donne une expression simplifiée de l'expression précédente. Ainsi, nous avons :

Proposition 7.3.5. *La valeur asymptotique de R_n est donnée par :*

$$R_n \sim (1 - e^{-2})\frac{n}{2} = 0.432332n \text{ lorsque } n \to \infty. \tag{19}$$

Preuve. Pour calculer la valeur asymptotique de R_n, nous utilisons la méthode de Bender présentée dans [Fla97]. $R(z)$ peut être écrite comme un produit de deux fonctions génératrices $R(z) = a(z)b(z)$ avec $a(z) = 1 - e^{-2z}$ et $b(z) = \frac{z}{2(1-z)^2}$. La fonction génératrice $a(z)$ est partout convergente et la fonction génératrice $b(z)$ a $\beta = 1$ comme rayon de convergence. Par conséquent, $b(z) = \sum_{n\geq 0} \frac{n}{2}z^n$ et $a(z) = \sum_{n\geq 1} -\frac{(-2)^n}{n!}z^n$, et $b(z)$ satisfait

$$\frac{b_{n-1}}{b_n} \to \beta = 1 \text{ quand } n \to \infty, \tag{20}$$

les coefficients du produit $R(z) = a(z)b(z)$ sont donnés par

$$[z^n]R(z) = R_n \sim a(1)b_n = (1 - e^{-2})\frac{n}{2} = 0.432332n \qquad \text{lorsque } n \to \infty. \tag{21}$$

□

Dans MSZ-algorithme, le nombre de rendez-vous dans une chaîne est donné par $(n+1)/4$. Par conséquent, dans le cas des chaînes, ce nouvel algorithme est plus efficace que MSZ-algorithme.

7.3.2 Impact de l'insertion des arêtes

Dans cette section, nous nous intéressons à l'effet de l'ajout ou de la suppression d'arêtes dans un graphe. Pour comprendre cet impact, nous proposons de l'étudier sur un exemple. Nous considérons l'étude de l'espérance du nombre de rendez-vous pour les graphes de la figure 21 : le graphe de base est le graphe G_1, le graphe G_2 est obtenu à partir du G_1 par élimination de l'arête du milieu alors que le graphe G_3 est le graphe obtenu par l'insertion d'une nouvelle arête dans G_1.
Dans le graphe G_1, nous pouvons avoir soit un seul rendez-vous sur l'arête du milieu avec probabilité $1/5$, soit deux rendez-vous sur les arrêtes de face non voisines et comme il y a deux possibilités d'avoir ces deux rendez-vous, chacune d'elle est obtenue avec une probabilité égale à $4/5$.

Un calcul simple donne pour le graphe G_1 :

$$\overline{R}(G_1) = Pr(R(G_1) = 1) + 2 \times Pr(R(G_1) = 2) = 1/5 + 2 \times 4/5 = 9/5$$

et pareillement $\overline{R}(G_2) = \overline{R}(G_3) = 2$.

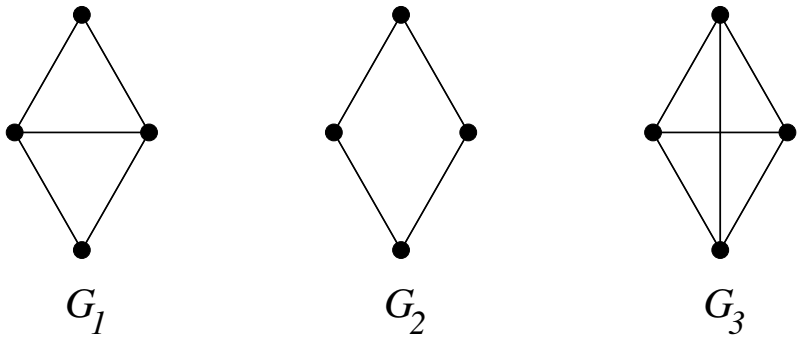


FIG. 21 – Trois graphes connexes ayants le même nombre de sommets.

Ces exemples prouvent que l'impact d'insertion d'arêtes dans un graphe connexe n'a pas un effet monotone sur l'espérance du nombre de rendez-vous.

7.3.3 Impact d'une extension

Dans la section précédente nous avons prouvé que l'insertion d'arêtes dans un graphe peut entraîner un effet négatif sur l'espérance du nombre rendez-vous. La question qui surgit naturellement est la suivante : *"quel sera l'impact d'une extension ?"*, c'est-à-dire, quel sera l'impact sur l'espérance $\overline{R}(G)$ lorsque des nouveaux sommets et des nouvelles arêtes sont ajoutés. Les arêtes ajoutées sont entre les nouveaux sommets et les sommets initiaux.

Bien que l'effet semble être positif, la preuve n'est pas évidente. La proposition suivante montre que l'impact est non négatif et il est au plus 1 pour l'insertion d'un sommet.

Proposition 7.3.6. *Soit $G = (V, E)$ un graphe et G' un graphe obtenu à partir de G par ajout d'un nouveau sommet x et zéro ou plusieurs arêtes entre x et les sommets de G. Nous avons :*

$$\overline{R}(G) \leq \overline{R}(G') \leq \overline{R}(G) + 1.$$

Preuve. Par récurrence sur $n = |V|$. Pour $n = 0$ la proposition est évidente. Supposons qu'elle soit vraie pour des graphes de taille moins que n et prouvons qu'elles reste vraie pour un graphe G de taille n. Soit e la première arête sur laquelle un rendez-vous a eu lieu dans G'. Nous avons deux cas ; selon que x soit une extrémité de e ou non.

Pour prouver cette proposition, il suffit de montrer que les inégalités ci-dessus sur les valeurs d'espérances, conditionnées par le fait que les événements $\mathcal{E}_1$ et $\mathcal{E}_2$ suivants soient maintenus.

1. **Evénement $\mathcal{E}_1$.** Soit $e = \{x, v\}$ pour tout $v \in V$. Nous avons d'une part $\overline{R}(G'/\mathcal{E}_1) = \overline{R}(G'_e) + 1$ et d'autre part formulée par l'hypothèse de récurrence, $\overline{R}(G'_e) \leq \overline{R}(G) \leq \overline{R}(G'_e) + 1$. Ceci donne $\overline{R}(G) \leq \overline{R}(G'/\mathcal{E}_1) \leq \overline{R}(G) + 1$.
2. **Evénement $\mathcal{E}_2$.** Soit e une arête de G. Notons que, sous cette condition, tous les sommets de G ont la même chance $1/|V|$ d'être une extrémité de e. Maintenant, selon l'hypothèse d'induction $\overline{R}(G_e) \leq \overline{R}(G'_e) \leq \overline{R}(G_e) + 1$, conditionnée par le choix d'une arête quelconque $e \in E$. Or, $\overline{R}(G)$ est la valeur moyenne de $\overline{R}(G_e)+1$ et $\overline{R}(G'/\mathcal{E}_2)$ est la moyenne de $\overline{R}(G'_e)+1$, $\forall e \in E$. Par conséquent, $\overline{R}(G) \leq \overline{R}(G'/\mathcal{E}_2) \leq \overline{R}(G) + 1$.

□

7.3.4 Espérance asymptotique du nombre de rendez-vous dans les graphes aléatoires

Nous considérons ici des graphes aléatoires de degré moyen $c > 0$. Nous calculons d'abord la valeur asymptotique de la probabilité d'avoir un rendez-vous sur une arête donnée du graphe. Soit G un graphe aléatoire de taille n dont le degré moyen est c. Dans ce modèle, la probabilité que deux sommets distincts soient liés par une arête est $c/(n-1)$, de sorte que le degré moyen soit c. Ainsi le modèle considéré est semblable au *modèle de probabilités constantes* exposé dans [Bol01], dans lequel le degré moyen demeure borné. Soit $\{a, b\}$ une arête existante entre deux sommets a et b d'un graphe aléatoire G.

Le but principal de l'étude présentée ici est de dériver une expression asymptotique de la probabilité $P(n, c)$ de rendez-vous sur une arête $\{a, b\}$. D'abord nous considérons un modèle simplifié, appelé *modèle d'arbre*, dans lequel cette probabilité, pour une arête $\{a, b\}$, est asymptotiquement proche de celle du graphe aléatoire initial. Dans le modèle d'arbre, le graphe contient une arête $\{a, b\}$ et deux arbres dirigés aléatoires disjoints enracinés en a et b. Dans ces deux arbres, un sommet v choisit son degré sortant de telle sorte à avoir la valeur moyenne $c + o(1)$ et ce en ajoutant une arête à chacun des $n - 2$ nouveaux sommets, indépendamment et avec une

probabilité égale à $c/(n-1)$ pour chacun. Ce modèle est différent du modèle initial du graphe aléatoire par le fait que, tout simplement, tous les nouveaux sommets potentiels sont *distincts*; dans le modèle de graphe en général, les nouveaux sommets potentiels sont tous des sommets du graphe autres que v et qui n'étaient pas déjà reliés à v. Le processus est effectué *seulement* sur un sommet v tel que les arêtes ajoutées en v peuvent influencer sur le résultat de l'algorithme sur l'arête $\{a, b\}$.

Pour simplifier le résultat principal de $P(n, c)$, nous montrons d'abord quelques lemmes.

Lemme 7.3.2. *Avec une probabilité tendant vers* 1 *les arbres construits ci-dessus sont finis.*

Preuve. La probabilité qu'une arête $e = \{a, b\}$ participe à un couplage dépend de la plus petite variable aléatoire $X_e(a)$ et $X_e(b)$ avec une distribution non-uniforme. Pour simplifier la discussion nous définissons une variable aléatoire z_e égale à

$$Pr[min(X_e(a), X_e(b)) < min(Y, Z)],$$

où Y et Z sont des variables aléatoires indépendantes qui ont la même distribution que X_e. z_e est uniformément distribuée sur $[0, 1]$ et entre deux arêtes e et f, e sera choisie si $z_e > z_f$. Supposons que v est à l'extrémité d'une chaîne $v_0 = b, v_1, v_2, ..., v_h = v$, alors l'arête $\{v_{h-1}, v_h\}$ peut influencer sur le résultat en $\{a, b\}$ seulement si elle est choisie de façon à empêcher $\{v_{h-2}, v_{h-1}\}$ d'être pris au rendez-vous, sinon l'arête $\{v_{h-2}, v_{h-1}\}$ aurait été choisie empêchant ainsi $\{v_{h-3}, v_{h-2}\}$, etc. Ceci peut se produire si $z_{\{v_{h-1},v_h\}} > z_{\{v_{h-2},v_{h-1}\}} \cdots > z_{\{v_0,v_1\}}$ qui a une probabilité de $1/h!$. Or l'espérance du nombre des sommets v qui sont à une distance h de a ou b est $2c^h$ et donc la probabilité que l'un d'eux influence l'arête $\{a, b\}$ est au plus $2c^h/h!$, cette quantité tend vers 0. De ce fait, choisissons h suffisamment grand mais qui croît lentement avec n. Soit $h = \ln \ln n$, la troncation de l'arbre à ce niveau donne une probabilité $1 - o(1)$ d'avoir la même probabilité de choisir $\{a, b\}$ comme c'était le cas dans l'arbre non-tronqué.

□

Le lemme suivant donne une limite de la probabilité qu'une arête donnée n'est pas inhibée d'être choisie.

Lemme 7.3.3. *La probabilité P qu'une arête* $\{a, b\}$ *n'est pas inhibée dans le modèle d'arbre tend vers* $1/(1 + c)$ *quand* $n \to \infty$.

Preuve. Nous considérons d'abord des graphes possédant l'arête $\{a, b\}$ et seulement un arbre aléatoire construit selon la description ci-dessus à partir de b. Dans un tel arbre tronqué avec un h suffisamment grand, l'arête $\{a, b\}$ a la probabilité p_h de ne pas être inhibée et pour tout x $(0 \leq x \leq 1)$, $f_h(x)$ est la probabilité conditionnelle que z est plus grand que x sachant que $\{a, b\}$ n'est pas inhibée, où z est la v.a. définie dans la preuve du lemme 7.3.2. Toute arête $\{b, c_i\}$ laissant b a une probabilité p_{h-1} de ne pas être empêchée par une arête en dessus d'elle dans l'arbre et elle a une probabilité $f_{h-1}(x)$ d'avoir son z plus grand que x conditionnée sur cet événement. Par conséquent, si $z_{\{a,b\}} = x$, la probabilité que l'arête $\{a, b\}$ ne soit pas inhibée est $\approx \prod_{i=1}^{n-2}(1 -$

$(cp_{h-1}f_{h-1}(x))/(n-1)$). Ceci nous donne

$$p_h \approx \int_0^1 e^{-cp_{h-1}f_{h-1}(x)}dx$$

et

$$f_h(y) \approx \frac{\int_y^1 e^{-cp_{h-1}f_{h-1}(x)}dx}{p_h}.$$

La limite lorsque $h \to \infty$ donne

$$p = \int_0^1 e^{-cpf(x)}dx$$

$$f(y) = \frac{\int_y^1 e^{-cpf(x)}dx}{p}$$

f satisfait les conditions de bornes $f(0) = 1$ et $f(1) = 0$.

Ces équations admettent comme solution $f(x) = \ln(1 + c - cx)/\ln(1 + c)$ et $p = \ln(1 + c)/c$ de sorte que $e^{-cpf(x)} = 1/(1 + c - cx)$.

Maintenant, nous pouvons considérer la probabilité P que l'arête $\{a, b\}$ ne soit pas empêchée. En mettant en évidence la possibilité qu'elle soit empêchée par une arête incidente à a ou b, nous avons $P = \int_0^1 e^{-2cpf(x)}dx = \int_0^1 (1 + c - cx)^{-2}dx = 1/(1 + c)$.

□

Le résultat principal sur les graphes aléatoires considérés dans cette section est donné par le théorème suivant :

Théorème 7.3.1. *La probabilité $P(n, c)$ qu'un rendez-vous ait lieu sur une arête existante $\{a, b\}$ dans un graphe aléatoire G, avec un degré moyen c et de taille n, tend vers $1/(1 + c)$, lorsque $n \to \infty$.*

Preuve. En vertu des lemmes précédents, il suffit de montrer que les probabilités limites sont les mêmes pour le graphe aléatoire et l'arbre. Dans une boule de rayon h autour de $\{a, b\}$ ces probabilités sont les mêmes dans les deux modèles à moins que deux sommets distincts d'arbres enracinés en a et en b s'avèrent être le même sommet. Puisque l'espérance du nombre de paires d'arbres est $O(1)^h$, qui est $(\ln n)^{O(1)}$, la probabilité que ceci se produise est $(\ln n)^{O(1)}/n = o(1)$, donnant ainsi la probabilité $P + o(1)$ pour le modèle de graphe. Ceci établit le théorème.

□

Soit $\mathcal{R}(n, c)$ l'espérance du nombre de rendez-vous dans le graphe aléatoire considéré ici. Nous avons :

Proposition 7.3.7. *$\mathcal{R}(n, c)$ a la valeur asymptotique $\frac{n}{2}(1 - \frac{1}{1+c})$, quand $n \to \infty$.*

Preuve. Nous pouvons directement montrer cette proposition en utilisant le théorème ci-dessus et le fait que le nombre moyen d'arêtes incidentes à un sommet est c.

□

Nous considérons maintenant un autre modèle de graphe aléatoire avec un nombre d'arêtes déterminé $m = cn/2$ (voir le modèle $\mathcal{G}(n, m)$ dans [Bol01]). La proposition suivante montre qu'en matière de rendez-vous sur des arêtes, les deux modèles de graphes aléatoires sont asymptotiquement équivalents.

Proposition 7.3.8. *Dans le modèle de graphe aléatoire $\mathcal{G}(n, m)$ avec $m = cn/2$, la probabilité $P(n, c)$ qu'une arête donnée ne soit pas inhibée tend vers $1/(1 + c)$.*

Preuve. Nous montrons que le nouveau modèle de graphe aléatoire (en utilisant la même source de nombres aléatoires dans les deux cas) réalisera la même boule de rayon h autour de l'arête $\{a, b\}$ avec probabilité $1 - o(1)$, de sorte que les probabilités limites dans les deux modèles soient identiques. La différence entre les boules de rayon h dans les deux cas est que le modèle de probabilité constante ajoute toujours une arête candidate avec probabilité $c/(n - 1)$, alors que le modèle de nombre d'arêtes fixe ajoute une arête candidate avec probabilité m'/p' où m' est le nombre d'arêtes nécessaires restantes et p' est le nombre de paires de sommets restants ; les paires qui peuvent être considérés.

Nous notons, d'abord, qu'avec une probabilité égale à $1 - o(1)$, le modèle de probabilité constante ajoute un nombre de sommets deux fois plus grand que l'espérance du nombre de paires qui est donc $(\ln n)^{O(1)}$.

Maintenant nous pouvons ignorer les cas qui sont plus que $(\ln n)^{O(1)}$ sommets dans les arbres. Cependant, nous déduisons que chaque arête potentielle est ajoutée avec une probabilité égale à $c/(n - 1)$ ou avec une probabilité entre $cn/(n^2 - n(\ln n)^{O(1)})$ et $c(n - n(\ln n)^{O(1)}/n^2)$ selon le modèle considéré. Ainsi, la différence entre les probabilités dans les deux modèles pour toute arête considérée est $(\ln n)^{O(1)}/n^2$. Pour toutes les arêtes potentielles, la somme de ces différences est $(\ln n)^{O(1)}/n = o(1)$. Par conséquent, la probabilité que les deux processus aléatoires donnent des boules différentes de rayon h est $o(1)$ de sorte que les mêmes probabilités limites de $1/(c+1)$ soient aussi maintenues pour le modèle du nombre d'arêtes fixé.

□

7.4 Efficacité de l'algorithme

Dans la section précédente nous avons comparé le nouvel algorithme à l'algorithme de [MSZ03a] en termes d'espérance du nombre de rendez-vous. Cependant, il est à noter que cette mesure de performance est relative. En effet, considérons le cas simple de graphes en étoiles ; l'espérance du nombre de rendez-vous n'est pas plus que 1, en dépit de la taille de l'étoile. Ceci ne signifie pas que l'algorithme est inefficace, l'étoile ne permet pas d'avoir un nombre de

couplages plus grand.

Nous rappelons l'efficacité de l'algorithme probabiliste de rendez-vous présentée dans [MSZ03a]. Soit $\mathcal{A}$ un algorithme probabiliste quelconque de rendez-vous sur des graphes. Son *efficacité* sur un graphe $G = (V, E)$, avec $E \neq \emptyset$, est le rapport

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

où $\overline{R}_{\mathcal{A}}(G)$ est l'espérance du nombre de rendez-vous dans un tour, toutes les fois que $\mathcal{A}$ est appliqué à G, et $K(G)$ est le nombre de couplage dans G.

Selon cette définition l'efficacité de notre algorithme est 1 pour les graphes complets, les étoiles et les double-étoiles. Elle est asymptotiquement $1 - e^{-2}$ pour les chaînes.

Dans la suite de cette étude de l'efficacité nous cherchons à trouver une borne inférieure générale de l'efficacité de l'algorithme, c'est-à-dire une borne pour la classe de tous les graphes. Cette borne inférieure s'avère être serrés. Dans le cas des arbres (réseaux peu connectés), l'étude suivante prouve que l'algorithme réalise une meilleure performance, ayant une efficacité supérieure à $3/4 = 0.75$.

7.4.1 Borne inférieure de l'efficacité dans le cas général

Rappelons-nous d'abord que l'efficacité de MSZ-algorithme pour un graphe complet K_n est asymptotiquement $1/n$ et donc, cet algorithme n'admet pas une borne inférieure positive sur la classe de tous les graphes. Nous montrons que notre algorithme sur cette classe réalise une efficacité au moins $1/2$.

Proposition 7.4.1. *Pour tout graphe non vide G, $\Delta_{\mathcal{A}}(G) \geq \frac{1}{2}$.*

Preuve. Il suffit de voir que la taille du couplage maximal dans G est au moins la moitié de n'importe quelle autre taille de couplage.

□

Remarque 7.4.1. La preuve directe de la proposition montre qu'avec probabilité 1, le nombre de rendez-vous est au moins la moitié du nombre de couplage (performance idéale).

Nous montrons à présent que la borne inférieure $1/2$ de l'efficacité de l'algorithme sur tous les graphes est la borne la plus serrée.

Proposition 7.4.2. *Il existe une suite de graphes G_n de taille $4n$, telle que l'efficacité sur G_n tend vers $1/2$, quand $n \to \infty$.*

Preuve. Pour prouver ceci, nous considérons un graphe $G(n_1, n_2)$ avec $2n_1 + 2n_2$ sommets dans $V \cup U \cup W \cup X$, où $V = \{v_1, v_2, \cdots, v_{n_1}\}$, $U = \{u_1, u_2, \cdots, u_{n_2}\}$, $W = \{w_1, w_2, \cdots, w_{n_1}\}$ et $X =$

$\{x_1, x_2, \cdots, x_{n_2}\}$. Pour tout $1 \leq i \leq n_1$ et $1 \leq j \leq n_2$ il y a des arêtes $\{v_i, w_i\}$, $\{w_i, u_j\}$ $\{u_j, x_j\}$, voir la Fig. 22.

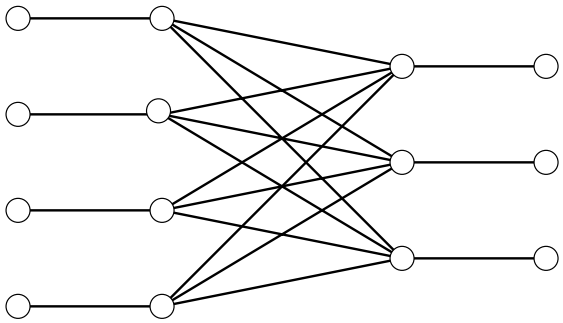


FIG. 22 – Un graphe $G(4, 3)$.

Il est facile de voir que $K(G(n_1, n_2)) = n_1 + n_2$, et si nous dénotons par $\mathbb{E}(n_1, n_2)$ l'espérance du nombre de rendez-vous dans $G(n_1, n_2)$, un calcul direct mène à la formule récurrente suivante :

$$\mathbb{E}(n, 0) = \mathbb{E}(0, n) = n \quad \text{et}$$
$$\mathbb{E}(n_1, n_2) = 1 + \frac{n_1 n_2 \mathbb{E}(n_1 - 1, n_2 - 1) + n_1 \mathbb{E}(n_1 - 1, n_2) + n_2 \mathbb{E}(n_1, n_2 - 1)}{n_1 n_2 + n_1 + n_2}, \quad \text{pour } n_1, n_2 \geq 1.$$

Ainsi, pour $n_1 = n_2$, un simple raisonnement donne :

$$\mathbb{E}(n, n) \leq 1 + \mathbb{E}(n - 1, n - 1) + \frac{2n}{n^2 + 2n}.$$

Par conséquent, $\mathbb{E}(n, n) = n + o(n)$. Maintenant soit $G_n = G(n, n)$, nous avons : $K(G_n) = 2n$ et donc le théorème est prouvé.

□

7.4.2 Borne inférieure de l'efficacité dans le cas des arbres

Nous terminons cette étude en donnant une borne inférieure de l'efficacité de notre algorithme appliqué aux arbres. Pour une classe de graphes appelée mille-pattes, l'algorithme réalise une efficacité légèrement plus grande que la borne inférieure. Nous ignorons s'il existe des graphes pour lesquels l'efficacité s'approche plus de la borne inférieure. Pour simplifier la preuve du théorème principal, nous présentons d'abord quelques lemmes préliminaires et nous avons aussi besoin de quelques définitions.

Définition 7.4.1. Un *C−arbre* est un arbre dans lequel trois sommets distincts x, y et z sont tels que $\{x, y\}$ et $\{x, z\}$ sont des arêtes et y et z sont des feuilles.

Pour toute arête $e = \{x, y\}$ dans un arbre, soit $t(e)$ le nombre d'arbres non-vides de la forêt obtenue suite à la suppression de l'arête e et de toutes les autres arêtes incidentes à x ou y. Pour

un arbre T, $Forests(T)$ dénote le multi-ensemble de toutes les occurrences d'arbres non-vides de toutes les forêts résultantes après la suppression d'une arête $e = \{x, y\}$ et des autres arêtes incidentes à ses extrémités x et y.

Lemme 7.4.1. *Si T est un $C-$arbre de m arêtes, la somme de $t(e)$ sur toutes les arêtes e dans T est au moins $2m - 4 + \sum_{v \in H} d(v)(d(v) - 1)/6$ où H est l'ensemble de tous les sommets de T de degrés strictement supérieur à 3.*

Preuve. Tout $C-$arbre peut être construit en commençant par une chaîne de trois arêtes et en ajoutant à plusieurs reprises de nouvelles arêtes de l'une des trois manières suivantes :

1. Ajouter une nouvelle arête liant une feuille existante v à un nouveau sommet.
2. Ajouter une nouvelle arête qui relie un sommet v de degré 2 à un nouveau sommet.
3. Ajouter une chaîne de longueur 2 à un sommet existant v de degré au moins 2.

Dans le second cas, il n'y a aucune arête entre v et une feuille tandis que dans le dernier cas, il y en a au plus une. La conclusion du lemme est vraie pour la chaîne initiale ($m = 3$ et $\sum(t = 2) = 2m - 4$). Nous montrerons par récurrence que le lemme reste vrai pour tout le procédé de construction que nous avons décrit.

1. La nouvelle arête e est telle que $t(e) = 1$. H et t de toute arête existante sont inchangés à l'exception des arêtes de distance 1 de v pour lesquelles t est incrémenté de 1. Ainsi, la somme $\sum t$ augmente au moins de 2.
2. La nouvelle arête e est telle que $t(e) = 2$. H et tous les autres t sont inchangés. Encore dans ce cas, la somme $\sum t$ est augmentée de 2.
3. Soit D le degré de v après l'ajout. Les deux arêtes de la chaîne ajoutée ont des t égaux à 1 et $D - 1$ ou $D - 2$ arêtes. Les $D - 1$ arêtes qui étaient adjacentes à v ont leurs t incrémentés par 1. Par conséquent, $\sum t$ est augmentée d'au moins $2D - 2$. Si $D = 3$, $\sum t$ est augmentée de 4 et H reste inchangé ; si $D = 4$, $\sum t$ est augmentée de 6 et v entre dans H avec $d(v)(d(v)-1)/6 = 2$ sinon, v reste dans H avec $d(v)(d(v)-1)/6$ augmenté de $(D-1)/3$. Dans tous les cas l'augmentation de $\sum t$ est au moins 4 (pour les deux arêtes ajoutées) plus l'augmentation de $\sum_{v \in H} d(v)(d(v) - 1)/6$.

□

Lemme 7.4.2. *Le nombre d'occurrences dans $Forests(T)$ de sous-arbres isomorphes à une $3-$chaîne (sommets a, b, c, d et arêtes $\{a, b\}, \{b, c\}, \{c, d\}$) est au plus le nombre d'occurrences d'arbres T avec $K(T) = 1$ plus $\sum_{v \in H} d(v)(d(v) - 1)$.*

Preuve. Considérons une $3-$chaîne a, b, c, d ; elle est liée au reste de l'arbre par l'intermédiaire d'une arête qui peut être soit $\{a, x\}$ soit $\{b, x\}$. Dans chaque cas, il y a deux arêtes dont la suppression (avec leurs arêtes incidentes) laisse un arbre c, d ou b, c, d avec $K(T) = 1$. Cette $3-$chaîne est produite (comme un élément de la forêt) seulement par la suppression de toute autre arête incidente à x. Soit $d(x)$ le degré de x. Si $d(x) \leq 3$, il y a au plus deux occurrences de cette chaîne dans $Forests(T)$. Autrement, pour les $d(x)$ arêtes incidentes à x, chaque arête produit au plus $d(x) - 1$ occurrences de $3-$chaînes dans $Forests(T)$, soit pour un total de

$d(x)(d(x)-1)$. Maintenant, une somme sur tous les sommets x liés aux occurrences de la 3−chaîne nous donne le résultat. □

Lemme 7.4.3. *Tout arbre T avec $K(T) \geq 3$ a au moins quatre occurrences dans $Forests(T)$ d'arbres T avec $K(T) = 1$.*

Théorème 7.4.1. *Le cardinal moyen des couplages trouvés par l'algorithme pour un arbre T (autre qu'une 3−chaîne) est au moins $cK(T)$ où $K(T)$ est le nombre de couplages de T et $c = 136/177 \approx 0.76836$.*

Preuve. Nous montrerons par récurrence que pour un arbre T ayant $K(T) = m$, $\overline{R}(T) \geq cm + d$ où $d = 9/59$, à l'exception d'une 3−chaîne T pour laquelle nous savons que $\overline{R}(T) = 5/3$ et $K(T) = 2$; d'ailleurs nous savons que si $K(T) = 1$, alors $\overline{R}(T) = 1$. Considérons un couplage maximal $\mathcal{M}_{max}$ sur T et le sous-arbre minimal S de T contenant toutes les arêtes de $\mathcal{M}_{max}$. Notons que S est un C−arbre. Il suffit maintenant de montrer notre affirmation pour S puisque nous savons que $\overline{R}(S) \leq \overline{R}(T)$ et $K(S) = K(T)$. Soit $m = K(S)$; nous savons que S contient au moins $2m - 1$ arêtes. Nous disons qu'une arête est *mauvaise* si elle ne peut pas participer à un couplage maximal ; une arête est mauvaise si et seulement si elle est incidente aux deux arêtes de $\mathcal{M}_{max}$ de sorte qu'il y ait au plus $m - 1$ arêtes mauvaises. Nous écrivons m_b et M respectivement pour le nombre de mauvaises arêtes et le nombre de toutes les arêtes ; $M \geq 2m - 1$. Le choix d'une mauvaise arête d'un couplage laisse une forêt F avec $K(F)$ égal à $K(S) - 2$. Ainsi, par induction, nous avons pour chaque forêt F composée de t arbres avec $\sum_{t\in F} K(t) = k$ l'inégalité suivante $\overline{R}(F) \geq ck + td + (1 - c - d)t_1 + (5/3 - 2c - d)t_2$ où t_1 est le nombre d'arbres t dans F avec $K(t) = 1$ et t_2 est le nombre des 3−chaînes. Notons par T_i la somme de tous les t_i des forêts obtenues lorsque chaque arête de S et de V est choisie pour $\sum_{v\in H} d(v)(d(v) - 1)$, nous obtenons $\overline{R}(S) \geq cm + d$. En effet, nous avons :

$$
\begin{aligned}
\overline{R}(S) &\geq 1 + \frac{Mc(m-1) - m_b c + (2M - 4 + V/6)d + (1-c-d)T_1 + (5/3 - 2c - d)T_2}{M} \\
&\qquad \text{(Lemme 7.4.1)} \\
&\geq 1 + \frac{Mc(m-1) - c(m-1) + (2M - 4 + V/6)d + (8/3 - 3c - 2d)T_1 + (5/3 - 2c - d)(T_2 - T_1)}{M} \\
&\geq 1 + \frac{(M-1)c(m-1) + (2M - 4 + V/6)d + (8/3 - 3c - 2d)T_1 + (5/3 - 2c - d)V}{M} \\
&\qquad \text{(Lemme 7.4.2))} \\
&\geq 1 + \frac{(M-1)c(m-1) + (2M - 4)d + (8/3 - 3c - 2d)T_1}{M} \\
&\qquad \text{(coefficients positifs dans } V \geq 0) \\
&\geq 1 + \frac{(M-1)c(m-1) + (2M - 4)d + 4(8/3 - 3c - 2d)}{M} \\
&\qquad \text{(Lemme 7.4.3)} \\
&\geq 1 + \frac{(2m-2)c(m-1) + (4m - 6)d + 4(8/3 - 3c - 2d)}{(2m-1)} \\
&\qquad \text{(monotonie sur M)} \\
&= cm + d.
\end{aligned}
$$

□

Comme une conséquence immédiate de ce théorème nous avons :

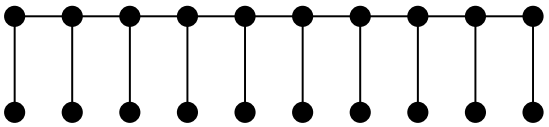


Fig. 23 – Un exemple de 9−pattes

Corollaire 7.4.1. *Pour tout arbre non vide T (autre qu'une* 3*−chaîne), nous avons :*

$$\Delta_{\mathcal{A}}(T) \geq c = \frac{136}{177}.$$

Pour l'instant nous ne savons pas si la borne inférieure c donnée par le corollaire ci-dessus est la plus serrée. Cependant, l'étude suivante montre que l'écart est petit.

Les n−pattes, la Fig. 23, sont des graphes définis par l'ensemble d'arêtes tel que : $\{(x_i, y_i)|0 \leq i \leq n\} \cup \{(x_i, x_{i+1})|0 \leq i < n\}$. Ils ont seulement un couplage maximal de taille $n + 1$. A chaque fois qu'une arête est choisie dans un rendez-vous, les arbres résiduels sont des x−pattes ; en conséquence, il est possible de calculer récursivement l'espérance du nombre de rendez-vous, noté par E_n. Cette espérance est donnée par une récurrence.

L'équation récursive qualifiant l'espérance du nombre de rendez-vous dans une n−pattes est :

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

qui peut s'écrire comme :

$$E_0 = 1, E_1 = 5/3 \text{ et}$$
$$(2n+1)E_n - (2n+1)E_{n-1} - 2E_{n-2} = 2, \ \forall n \geq 1.$$

La solution explicite de cette récurrence est :

$$E_n = \frac{1}{4}(2n+5)\sum_{k=1}^{n+2} \frac{(-1)^{k+1}2^k k!}{(2k+1)!}$$

et pour l'asymptotique nous avons :

$$E_n \ \sim \ (n + 5/2)(1 - e^{-1}\int_0^1 e^{t^2} dt)$$

Preuve. Si on pose $v_n = E_n + 1$, $w_n = v_n/(2n+5)$ et $d_n = w_n - w_{n-1}$
on trouve $(2n+5)d_n = -2d_{n-1}$

d_n peut s'exprimer comme le produit des $-\frac{2}{2n+5}$ qui s'exprime très bien avec des factoriels et des puissances de 2. De là, on obtient w_n comme une somme (moins simplifiable a priori) et par

la suite, nous on obtient E_n au pire sous la forme d'une somme :

$$E_n = \frac{1}{4}(2n+5)\sum_{k=1}^{n+2}\frac{(-2)^{k+1}}{\prod_{i=0}^{k}(2i+1)}.$$

Le produit $\prod_{i=1}^{k}(2i+1)$ se simplifie en $\frac{(2k+1)!}{2^k k!}$,

L'expression, par conséquent, devient : $E_n = \frac{1}{4}(2n+5)\sum_{k=1}^{n+2}\frac{(-1)^{k+1}2^k k!}{(2k+1)!}$.

□

Chapitre 8

Diffusion dans un réseau avec un temps de transmission aléatoire

Sommaire

Dans un réseau de communication, la diffusion (*broadcast*) est le processus qui consiste à envoyer un message à partir d'un noeud à tous les autres nœuds du réseau[1]. La diffusion par inondation (*flooding*) est un mécanisme classique de diffusion, où chaque nœud retransmet le message reçu à tous ses voisins. Dans le présent chapitre, nous présentons et nous analysons un modèle d'émission avec un délai aléatoire de transmission. Nous considérons des variations où les arêtes peuvent avoir différents taux de transmission. Le délai aléatoire de transmission considéré est le temps nécessaire pour transiter un message entre deux noeuds adjacents dans le réseau.

8.1 Introduction

La diffusion (un-à-tous) est un problème central de l'informatique répartie. Elle a plusieurs applications, par exemple l'élection d'un chef afin de relancer le processus de ré-initialisation ou la recherche d'une information particulière parmi les nœuds du réseau. Elle est également employée pour chercher un dossier ou une information dans les systèmes de calcul pair-à-pair.

[1] On parle aussi du problème de contamination

Le problème de diffusion est aussi lié au problème de la propagation des rumeurs ou des virus dans les réseau sociaux. Ainsi, un nœud est dit *contaminé* s'il connaît le message initial diffusé.

La radiodiffusion est un modèle de diffusion qui a été intensivement étudiée au cours de ces dernières années pour maintes topologies différentes et sous une grande variété de modèles [FDRU90, DHSZ04, PP05]. Soit un réseau de n processeurs reliés entre eux par des liens de communication, la diffusion est la tâche ayant pour objectif de faire connaître à tous les processeurs la totalité des données connues initialement par un seul processeur, appelé *initiateur*. Le lecteur intéressé pourra consulter [HHL88].

Selon le modèle de communication considéré, la diffusion pose différents types de problèmes. Elle a été étudiée sous différentes hypothèses : le *mode tous ports*[2] ; le message réçu est envoyé à tous les voisins, le *mode un port*[3] ; le message reçu est envoyé à un seul voisin, les réseaux de radio, communication synchrone ou asynchrone, liens unidirectionnels ou bidirectionnels (voir [PP05, HHL88] pour un aperçu plus étendu).

Dans ce chapitre, nous considérons une approche de diffusion par inondation "sans visbilité"[4] pour diffuser des messages à travers le réseau. La diffusion sans visibilité commence par le nœud source disséminant un message parmi ses voisins. Toutes les fois qu'un nœud reçoit le message pour la première fois, il envoie une copie à tous ses voisins à l'exception du noeud à partir duquel il a reçu le message. Le processus de diffusion s'arête lorsque tous les nœuds du réseau ont reçu au moins une copie du message initial à diffuser. La situation est modélisée par un graphe de communication, dans lequel les sommets représentent les processeurs, et les arêtes représentent les liaisons bidirectionnelles. Le modèle proposé ici est enrichi par des attributs assignés aux liens de communication qui correspondent aux délais aléatoires de transmission.

8.2 Le modèle

Un graphe simple $G = (V, E)$ non orienté représente un réseau de communication dans lequel les sommets correspondent aux membres du réseau et les arêtes correspondent aux liaisons reliant des paires de ses membres. Dans ce chapitre, nous considérons également le mode de diffusion (un-à-tous). Dans ce mode, toute arête du graphe ne peut être employée qu'une seule fois pour envoyer le même message, et par conséquent dans une seule direction. Autrement dit, tant que le sommet v envoie un message à un sommet u, le sommet u ne peut pas envoyer de message à v.

À chaque sommet, un message peut être instantanément dupliqué en de multiples copies, pour être envoyé en même temps sur les chemins multiples du graphe.

Ici, nous associons à chaque arête e une v.a. (variable aléatoire) X_e qui représente le temps requis pour qu'un message traverse cette arête. Nous supposons que X_e soit une v.a. exponentiellement distribuée de paramètre $\lambda(e) > 0$:

[2] *all-port-mode*
[3] *one-port-mode*
[4] *blind flooding*

$$Pr(X_e < x) = 1 - e^{-\lambda(e)x}, \quad \forall x \geq 0.$$

Pour de tels graphes nous définissons une *arête feuille* toute arête telle que, dès que cette arête est traversée par un message, le sommet récepteur de ce message n'expédie plus le message reçu sur aucune autre arête ; tous ses voisins ont vu le message.

8.3 Durée de la diffusion

La durée de transmission correspond au temps nécessaire pour que les messages envoyés traversent le réseau. Elle dépend fortement de la longueur du chemin entre deux entités communicantes. Par conséquent, pour obtenir le délai d'émission dans le réseau, il est souhaitable que le chemin de la source à la destination soit le plus long que possible.

A ma connaissance, la diffusion n'a pas été étudiée dans le modèle de transmission avec délai aléatoire.

Notre problème est de trouver une borne supérieure de l'espérance du temps nécessaire pour accomplir la diffusion sous l'hypothèse que toutes les arêtes ont un délai aléatoire de transmission.

8.4 Processus de Markov

Nous considérons un graphe simple, connexe et non orienté $G = (V, E)$ et une fonction de pondération positive λ sur E. Soit un sommet de départ $v_0 \in V$, le processus de la diffusion (ou de *contamination*) peut être modélisé comme un processus de Markov en temps continu comme suit.

L'ensemble $\mathcal{S}$ des états est l'ensemble de tous les sous-graphes connexes obtenus à partir de G et contenant le sommet v_0 : si l'ensemble des sommets contaminés à l'instant t est V', alors le processus a atteint l'état $G' = (V', E')$, où G' est le sous-graphe de G induit par V'.

Ce modèle est théoriquement simple et la proposition suivante le caractérise entièrement. Cependant, le système d'équations différentielles, qui fournit *en principe* la probabilité d'être dans l'état donné, est difficile à résoudre (sauf s'il s'agit de graphes extrêmement simples).

Dans la proposition suivante et pour un graphe $G' = (V', E') \in \mathcal{S}$, G'_+ dénote l'ensemble des arêtes ayant seulement une extrémité dans G'. Pour le même graphe et pour un sommet v de V', nous notons $G'(v)$ le sous-graphe de G' induit par l'ensemble $V' \setminus \{v\}$.

Nous supposons dans la suite que $|V| \geq 2$.

Proposition 8.4.1. *Soit $\{X_t\}_{t\in T}$ un processus de Markov comme présenté ci-dessus. Pour un graphe $H = (V', E') \in \mathcal{S}$, soit $P_H(t)$ dénote la probabilité que $X_t = H$ à l'instant t. Nous avons :*

(i) $$\frac{dP_{(\{v_0\},\emptyset)}(t)}{dt} = -P_{(\{v_0\},\emptyset)}(t) \sum_{e \text{ incidente à } v_0} \lambda(e),$$

(ii) $$\frac{dP_H(t)}{dt} = -P_H(t) \sum_{e \in H_+} \lambda(e) + \sum_{v \in H,\ v\neq v_0} P_{H(v)}(t) \sum_{e \in H,\ e \notin H(v)} \lambda(e), \text{ et}$$

(iii) $$\frac{dP_G(t)}{dt} = \sum_{v \in G,\ v\neq v_0} P_{H(v)}(t) \sum_{e \notin H(v)} \lambda(e),$$

avec la condition initiale $P_{(\{v_0\},\emptyset)}(0) = 1$.

Preuve. La démonstration utilise les mêmes techniques que celles utilisées dans la preuve de la proposition 4.3.3. Elle n'est en faite qu'une adaptation de la méthode d'analyse des probabilités de transitions entre les états voisins (voir par exemple Feller [Fel50], Chapitre XVII, section 5).

On considère l'évolution du processus dans l'intervalle $[t, t + h]$. En écrivant les probabilités pour que le processus soit dans l'état H, à l'instant $t+h$, à partir des états le précédants à l'instant t, on obtient pour tout état une équation qui se transforme en une équation différentielle. Ainsi dans le second membre de l'équation qui donne $\frac{dP_H(t)}{dt}$ figure une somme de signe positif correspondant aux états qui précèdent H et une somme de signe négatif correspondant aux états qui suivent H.

□

8.4.1 Diffusion dans une chaîne

Nous nous intéressons au cas particulier où le graphe de communication G est une chaîne. Dans la suite la taille ou la longueur d'une chaîne correspond au nombre de ses arêtes. Soit $C = (V_c, E_c)$ une chaîne de longueur n. Pour toute arête e_i, $1 \leq i \leq n$, soit X_i la v.a. exponentiellement distribuée de paramètre λ_i assignée à e_i. Soit aussi $\mathcal{B}_c$ la durée de la diffusion d'une extrémité à l'autre de cette chaîne.

Nous avons :

Proposition 8.4.2. *Si tous les λ_i sont différents, la v.a. $\mathcal{B}_c$ a la densité suivante :*

$$f_c(x) = \sum_{i=1}^{n} \Big(\prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{\lambda_j}{\lambda_j - \lambda_i}\Big) \lambda_i e^{-\lambda_i x}.$$

Dans le cas où $\lambda_i = \lambda$, pour tout i, nous avons :

Proposition 8.4.3. *$\mathcal{B}_c$ est la somme de n variables aléatoires exponentiellement distribuées indépendantes de paramètre λ, elle est Γ distribuée avec une densité :*

$$f_c(x) = \frac{\lambda}{\Gamma(n)}(\lambda x)^{n-1}e^{-\lambda x} \quad pour\ x \geq 0,$$

avec une espérance $\mathbb{E}(\mathcal{B}_c) = n/\lambda$ et une variance var $(\mathcal{B}_c) = n/\lambda^2$.

8.4.2 Diffusion dans les arbres enracinés

Le problème devient plus complexe quand nous avons une arborescence (arbre enraciné). Pour calculer l'espérance du temps nécessaire pour accomplir la diffusion, nous commençons le calcul sur une arête feuille. Cependant, à chaque sommet v, la fonction de distribution de la diffusion peut être exprimée comme le produit de la fonction de distribution des sous-arbres enracinés à v.

Soit $T = (V_T, E_T)$ un arbre enraciné dans lequel un processus de diffusion prend place. Le temps de la traversée pour chaque arête $e \in E_T$, selon notre modèle, est une v.a. ayant une distribution exponentielle de paramètre $\lambda(e)$.

D'abord considérons un exemple. Considérons l'arbre T_1 de figure 24.

Le sommet initiateur de la diffusion est le sommet coloré en noir. Soient A, B, C et D les v.a exponentiellement distribuées respectivement de paramètres a, b, c, et d. Ces variables sont indépendantes.

Le message diffusé est cessé d'être expédié à chaque sommet feuille, autre que le sommet de départ. Nous trouvons facilement la distribution de $\mathcal{B}_T = max\{A + C, B + C, D\}$.

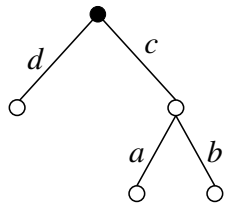


Fig. 24 – Arbre T_1

En utilisant des propriétés bien connues sur la somme et le maximum des variables aléatoires, nous obtenons la fonction de distribution et la valeur de l'espérance de diffusion :

$$F_{\mathcal{B}_T}(t) = (1 - e^{-dt})\left[1 - e^{-ct} - \frac{c(e^{-at} - e^{-ct})}{c - a} - \frac{c(e^{-bt} - e^{-ct})}{c - b} + \frac{c(e^{-(a+b)t} - e^{-ct})}{c - a - b}\right]$$

et

$$\begin{aligned}\mathbb{E}(\mathcal{B}_T) &= \frac{1}{c}+\frac{1}{d}-\frac{1}{c-a}-\frac{1}{c-b}-\frac{1}{c+d}+\frac{1}{c-a-b}+\frac{c}{a(c-a)}+\frac{c}{b(c-b)}\\ &\quad -\frac{c}{(a+b)(c-a-b)}-\frac{c}{(c-a)(a+d)}-\frac{c}{(c-b)(b+d)}+\frac{c}{(c-a)(c+d)}\\ &\quad +\frac{c}{(c-b)(c+d)}+\frac{c}{(c-a-b)(a+b+d)}-\frac{c}{(c-a-b)(c+d)}.\end{aligned}$$

On suppose que $a \neq c$, $b \neq c$ et $c \neq a+b$.

8.5 Simulation de l'algorithme

Dans le processus de diffusion, un sommet souhaitant disséminer un message à travers le graphe commence par envoyer une copie de ce message à tous ses voisins. Nous supposons que chaque arête e génère une v.a. exponentiellement distribuée qui détermine le temps exigé pour qu'un message traverse cette arête. Lorsqu'un sommet reçoit le message à diffuser pour la première fois, il fait suivre ce message à ses voisins, à l'exception du sommet voisin émetteur. En effet, un sommet u traite chaque message reçu d'un sommet voisin v, et si le même message est trouvé dans la file d'attente de transmission au sommet d'expéditeur v, il est éliminé de la file d'attente et ne sera jamais retransmis à v. Un sommet u peut recevoir le même message de plusieurs voisins avant de pouvoir le retransmettre aux voisins restants. Le nombre de messages envoyés est donc égal au nombre d'arêtes du graphe utilisé pour la diffusion, puisque chaque arête est employée exactement une fois dans le processus de diffusion.

Proposition 8.5.1. *Le temps exponentiellement distribué T de paramètre $\lambda(e)$, nécessaire pour traverser une arête e, n'est que*

$$\frac{-1}{\lambda(e)}\ln(x),$$

où x est une v.a. uniforme de valeurs dans l'intérvalle réel $[0,1]$.

Quatrième partie

Simulation

Chapitre 9

Simulation des algorithmes distribués

Sommaire

Dans ce chapitre nous allons présenter un outil de simulation des algorithmes distribués dont le but est de tester, vérifier les algorithmes distribués. Cet outil cache la complexité de construction d'applications distribuées au programmeur non spécialiste. En particulier, la gestion de bas niveau des communications est prise en charge par le simulateur.

9.1 Introduction

L'implémentation, le débogage, le test et l'expérimentation des algorithmes distribués sont des tâches assez complexes et délicates. Par conséquent, il est essentiel de comprendre les idées algorithmiques de haut niveau, indépendamment du langage et de la plate-forme de l'implémentation, pour mieux concevoir un outil qui répond au besoins spécifiés.

Plusieurs logiciels sont disponibles pour simuler les algorithmes distribués. Nous citons par exemple : VISIDIA [BGM01b, BGM+01a, BMMS02, MS, MS04, Sel04], VADE [MPT98], LYDIAN [KPT00]. A la différence de ces outils, l'outil présenté dans ce chapitre dispose d'un environnement adéquat pour simuler des algorithmes distribués probabilistes.

Dans ce qui suit, nous allons détailler quelques caractéristiques de ces outils.

VISIDIA (Visualization and Simulation of Distributed Algorithms) permet d'implémenter des algorithmes distribués et d'animer leurs exécutions. Son interface graphique permet à l'utilisateur de créer un réseau et de prototyper les algorithmes distribués. Cet outil comprend déja une bibliothèque d'algorithmes distribués. L'ajout d'un algorithme peut se faire de deux manières : soit l'implémenter à la main en utilisant les fonctionnalités de Visidia, soit le dessiner en utilisant une interface qui permet de créer des règles de réécriture simples avec contextes interdits.

VADE

VADE (Visualisation of Algorithms in Distributed Environnements) est un outil développé au Weizmann Institute of Science. VADE permet de visualiser les algorithmes distribués dans un système asynchrone. Il permet à l'utilisateur de visualiser l'exécution d'un algorithme dans une page Web tandis que l'exécution réelle se déroule sur un serveur distant. Ceci permet son utilisation à distance sans avoir besoin de l'installer.
Le système distribué utilisé par VADE est le suivant :

- le réseau est fiable ;
- les messages envoyés par un processus arrivent dans le même ordre de leur envoi (les canaux sont de type FIFO[1]) ;
- le réseau est asynchrone, chaque processeur possède sa propre horloge et il n'y a pas d'horloge globale.

Pour synchroniser l'exécution d'un algorithme, des événements sont envoyés pour modifier les états des noeuds et des liens en utilisant des messages de type *send synchronization* et *receive synchronization*.

LYDIAN

LYDIAN (Library of Distributed Algorithms and Animations) est un environnement pour la simulation et la visualisation des algorithmes distribués qui permet aux utilisateurs d'avoir un environnement expérimental pour tester et visualiser le comportement des algorithmes distribués. Lydian possède un éditeur de réseau et une autre fenêtre pour l'animation. Celle-ci est basée sur la trace de l'exécution des algorithmes distribués. L'environnement distribué utilise une horloge globale.

PARADE

PARADE (PARallel program Animation Development Environnement) est un environnement pour visualiser et pour déboguer les algorithmes parallèles et distribués. Il se compose de trois parties :

- programme de surveillance et de trace ;
- système de visualisation ;
- utilisation de la trace pour visualiser les actions du programme.

[1] First In First Out

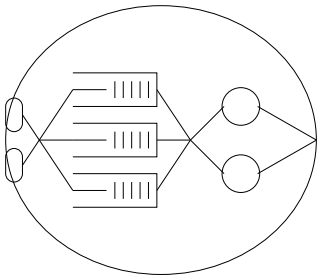


Fig. 25 – Un noeud

9.2 Description du modèle

Système distribué

Dans un système distribué, chaque entité dispose de canaux d'émission pour les messages sortants et de canaux de réception pour les messages entrants. Ces canaux sont en général finis et de types FIFO. Dans ce cas, chaque entité se comporte d'une façon indépendante des autres entités du système : l'émission (resp. la réception) d'un message consiste à le déposer (resp. le retirer) dans/de la file d'émission (resp. de réception).

La topologie

La topologie utilisée par notre simulateur est spécifiée par l'utilisateur. Elle peut être de n'importe quelle taille et de n'importe quelle disposition. La seule restriction (à part la limitation en mémoire) est que seulement un seul lien peut exister entre deux noeuds du réseau. Le réseau peut être uni ou bidirectionnel, et les messages sont échangés en mode asynchrone.

Le noeud

Le noeud représente l'élément de base du réseau. Il peut être vu comme un objet prenant en entrée un certain trafic, qui lui applique un certain traitement et enfin soit le redirige vers un autre noeud, soit tout simplement change son état.

Le noeud est décrit dans sa forme la plus générique possible selon nos modèles déjà utilisés et ceux à venir. Sa représentation visuelle est la suivante :

Le noeud est défini par sa distribution d'entrée, son type de service (algorithme à exécuter), ses ports d'entrée, ses files (boites aux lettres d'entrée et de sortie), ses ports de sortie.

Il possède en outre quelques fonctions permettant de spécifier son comportement qui concerne :

– Le choix de la file dans laquelle est stockée le message entrant *p*ush_msg

– Le choix de la file contenant le prochain message à servir et/ou le choix du message dans cette file *n*ext_msg

Le serveur est caractérisé par son type de service et les paramètres associés ainsi que par la méthode de choix d'une file pour le prochain message à traiter (discipline de service).

La file de messages[2] *q*ueue est l'objet qui permet de stocker les messages en attente de service. Elle est définie par sa capacité. Le type d'ordonnancement pour les files est FIFO. Il se fait par réécriture des fonctions d'ajout *a*dd et retrait *g*et de messages dans la file.

Les noeuds sont connectés entre eux par des arêtes comme c'est défini dans la topologie. Puisque la majorité d'algorithmes sont basés sur des noeuds spéciaux "initiateurs", certains noeuds peuvent être spécifiés d'être initiateur. Dans notre problème de diffusion, le noeud initiateur est spécifié par l'utilisateur alors que dans les autres problèmes que nous avons traité dans ce document, les noeuds initiateurs (les sommets simpliciaux ou les sommets feuilles) sont déterminés localement au cours de l'exécution.

Les liens

Les liens connectent les noeuds de réseau. Dans un réseau bidirectionnel, les liens existent dans les deux directions entre paire de noeuds. Pour tout lien, on peut associer un attribut (poids). Cependant, pour un réseau bidirectionnel, l'attribut dans les deux directions est le même.

Le temps

Les processus n'ont aucune connaissance de temps (i.e, il n'y a pas au sens physique une horloge globale).

Les processus

Les processus ont des programmes ou des actions à exécuter. Ils communiquent entre eux par passage de message via la "mémoire globale".

Les messages

Les messages sont des informations que deux noeuds voisins souhaitent échanger. Ils sont transmis sur les liens. Le délai entre l'envoi et la réception d'un message peut être configuré. En effet, il est nul dans nos problèmes d'élection de rendez-vous, mais dans notre problème de diffusion ce délai est de distribution exponentielle. Les messages sont envoyés via les canaux. En outre, ils ne peuvent passer qu'entre les noeuds qui sont en liaison directe.

[2]boite aux lettres

Un message *m*essage est caractérisé par une taille et sa priorité.

Passage de messages

Notre simulateur utilise des passages de messages en mode asynchrone. La raison principale pour celà est que dans la réalité, il n'existe pas des réseau synchrone. Le passage de messages est propre, i.e. la seule façon de perdre un message est dû aux défaillances.

Défaillances

Le simulateur des réseaux n'est pas complet sans la possibilité de simuler les défaillances des noeuds et des liens :

- Lorsqu'un lien est en panne, les messages de ce lien (i.e, tous les messages qui sont envoyés en empruntant ce lien, mais qui ne sont pas encore reçus) sont perdus.
- A chaque fois qu'un noeud est en défaillance toutes les informations, en mémoire, relatives à ce noeud sont perdues.

9.3 La simulation

La simulation repose sur un échéancier global contenant des événements classés par date. Un événement contient une ou plusieurs actions. Il peut-être un événement général (faire avancer les noeuds jusqu'à la date t, générer des statistiques, ...) un événement de type simulation (transférer ce message de cette file de ce noeud à une file d'un noeud voisin, ...). Chaque événement de l'échéancier est traité chronologiquement et spécifiquement selon son type. Nous avons deux types de simulation :

Stationnaire

La simulation stationnaire cherche à trouver un état stable de chaque élément du réseau de façon itérative en faisant durer la simulation jusqu'à ce que l'ensemble des états de ses composants soit stationnaire.

Transitoire

La simulation transitoire permet d'obtenir la réponse dynamique des composants du réseau à un changement des paramètres de simulation. Ces changements peuvent être dûs à une modification des paramètres des sources des flots de données d'un protocole de routage qui fait une mise à jour de ses tables ou d'un éventuel composant actif qui change d'état suite à un algorithme particulier.

9.3.1 Aspect de l'aléatoire

L'aspect de l'aléatoire est nécessaire pour mettre en œuvre les algorithmes distribués probabilistes. Or, il est difficile de fournir des nombres véritablement aléatoires dans un réseau d'ordinateurs par exemple. Parmi les exigences des algorithmes distribués, que nous avons présenté, est que les nombres aléatoires générés par les différents processus du réseau devrait être non corrélatifs. Présentement, nous n'avons pas des bonnes méthodes pour produire d'une manière distribuée des nombres pseudo-aléatoires non-corrélatifs.

Le générateur

La base de toute simulation est un générateur de nombres pseudo-aléatoires supposés distribués selon une loi uniforme sur l'intervalle réel $[0, 1]$, connaissant une valeur initiale (*seed* ou *semence*). Une classe très répandue de générateurs utilise une formule de récurrence à base de congruence (*prime modulus multiplicative generator*). D'autres s'inspirent de la suite de Fibonacci en additionnant deux valeurs précédentes ou font appel à des registres à décalage dans lesquels le résultat précédent est injecté après une transformation intermédiaire [Knu98]. Divers procédés permettent ensuite de générer une variable pseudo-aléatoire suivant une autre loi (normale, binomiale, poisson, exponentielle, gamma).

Pour générer des nombres pseudo-aléatoires nous avons utiliser des techniques de congruence :

$$X_{k+1} = (aX_k + c) mod\ m,\ = 0, 1, \ldots$$

où a est le multiplicateur, c l'incrément, m le modulo et X_0 la semence initiale. En choisissant judicieusement a, c, m et X_0. $\{X_k\}$ constitue une suite de nombre pseudo-aléatoire uniforme sur $[0, m-1]$ avec $0 \leq X_k \leq m-1$.

Des tests ($\chi 2$) permettent de déterminer si les coefficients ont été correctement choisis afin de garder l'indépendance des X_k et la fréquence du générateur.

En posant $U_k = \frac{X_k}{m-1}$, on obtient un générateur uniforme sur $[0, 1]$.

Pour passer à un générateur alétoire Z de distribution quelconque on utilise sa fonction de répartition $F(z) = Pr(Z \leq z)$ et on applique $Z = F^{-1}(U)$.

Dans le cas d'une distribution exponentielle, on a $F(z) = 1 - e^{-\lambda z}$, $\lambda > 0$ et donc $Z = \frac{-1}{\lambda} ln(1 - U)$.

Implémentation :

On a choisi les valeur suivantes afin d'obtenir un générateur ayant un bon comportement statistique (grande période = 2^{59}, peu de corrélations) :

$$\begin{aligned} m &= 576460752303423488LL \\ a &= 302875106592253LL \\ c &= 0LL \\ X_0 &= 0. \end{aligned}$$

Pour plus de détail sur la génération des nombres aléatoires, le lecteur pourra consulter l'ouvrage suivant [PTV92] chapitre 7. Une version éléctronique est aussi disonible sur le site *Numerical Receipes*[3].

La classe *g*enerator contient un générateur uniforme et une graine globale à tous les générateurs. Cette classe est ensuite dérivée afin d'obtenir le générateur voulu à partir de la variable uniforme.

L'événement

L'événement *e*vent est caractérisé par une date. Les différents événements régissant l'évolution de la simulation doivent spécialiser cette classe afin d'implémenter le traitement voulu.

Les événements sont classés par date dans un échéancier global (un arbre binaire équilibré).

Les événements principaux reçus des algorithmes distribués sont :

- l'envoi de message ;
- la modification de l'état d'un noeud ;
- le changement de l'état d'un canal de communication ;
- l'achèvement de l'exécution de l'algorithme.

Les événements principaux envoyés aux algorithmes distribués sont :

- l'arrêt provisoire de la simulation ;
- l'arrêt définitif de la simulation ;
- le redémarrage de l'exécution ;
- le changement de l'état d'un noeud ou d'un lien.

9.3.2 Principe simplifié de fonctionnement

La simulation est initiée par un graphe, puis le simulateur fait traiter les événements afin de faire évoluer le système jusqu'à la fin de la simulation.

[3] http ://www.nrbook.com/a/bookcpdf.html

Construction du réseau

Nous utillisons l'interface graphique fornit par Lyda afin de construire un réseau. Celle-ci permet de créer un fichier "image" représentant le schéma du réseau. Ainsi, par une simple manipulation de la souris, l'utilisateur peut ajouter, supprimer, ou choisir des processeurs, des liens de communications ou des sous-réseaux. Le réseau ainsi créé doit pouvoir être sauvegardé pour une utilisation ultérieure. Il est également possible d'importer un autre format de fichier ou d'exporter d'autres formats, en particulier en GML. Ceci permet d'utiliser d'autres éditeurs de graphes pour la génération de graphes qui peuvent être chargés dans par notre outils. Un réseau est donc un document du simulateur.

Création du réseau

La création du réseau est possible grâces aux différentes primitives disponible dans la classe *g*raph ainsi qu'aux constructeurs des éléments du réseau. Nous avons choisi une approche orientée objet pour les implémenter.

Graphe

Un graphe est composé de noeuds et d'arêtes.

- **vertex :** représente un noeud du graphe. Chaque noeud du réseau étant décrit par ses caractéristiques propres. Il a besoin d'une identité unique, de l'ensemble de ses voisins et de l'ensemble des arêtes incidentes à ce noeud. De même pour les graphes orientés, nous avons aussi l'ensemble des arêtes entrantes et sortantes.
- **edge :** une arête relie deux noeuds. Les deux sommets doivent être représentés dans l'objet arête.

L'objet *graphe* n'est qu'un ensemble d'objets noeud. Les principales méthodes de l'objet graphe sont :

- *nodes() :* ajoute l'ensemble des noeuds nécessaires à la simulation. Il convient de relier les noeuds correctement entre eux. Ceci permet d'assurer une adéquation entre le nombre de processeurs disponibles et le nombre de processus représentant le système à simuler.
- insert() : ajoute un noeud (ou une arête) passé en paramètre au graphe.
- *links() :* ajoute une arête entre deux noeuds passés en paramètre. Une autre méthode pour ajouter une arête orientée est *orientedLinks()*.
- *unlink() :* permet d'enlever une arête du graphe.
- *remove() :* permet d'enlever un noeud du graphe et toutes ces arêtes incidentes.
- *size() :* retourne la taille du graphe. La taille est le nombre de noeuds dans le graphe.

Principe d'évolution des noeuds

Une fois le réseau construit, il est nécessaire de simuler correctement son comportement en prenant en compte l'évolution de chacun des noeuds ainsi que leurs interactions.

Et effet, un sommet (processus) a besoin de connaître le nombre de ses voisins et ses canaux de communication. Il exécute séquentiellement les instructions de son service (l'objet algorithme que nous voulons simuler) en communiquant avec ses voisins par échange de messages. La communication sert à collecter des informations ou à envoyer des résultats afin de résoudre un problème commun à tous les processus du réseau.

Afin de rendre possible la communication entre les processus d'un réseau, des fonctions sont disponibles pour manipuler les messages. Un message est programmé par une classe `Message` qui contient toutes les informations indispensables. Les méthodes pour manipuler un message sont les suivantes :

- `sendTo` : envoie un message à un voisin.
- `receiveFrom` : (resp. `receive`) reçoit un message d'un voisin particulier (resp. le premier message dans une file d'attente du processeur).

9.3.3 Le Simulateur

Le simulateur est la partie intégrante de notre outil. C'est un "démon" qui corrèle et journalise les événements.

Pour assurer la communication entre le simulateur et ces entités, un objet de type arbre binaire équilibré (échéancier global) est obligatoire. Ce sont, en général, les messages envoyés qui ne sont pas encore mis dans les files d'attentes des noeuds destinataires i.e. ce sont les messages qui ne sont pas encore délivrés.

Les principales méthodes utilisées dans l'objet simulateur et leur fonctionnement sont :

- **startSimulation() :** c'est la méthode la plus importante car c'est ici que nous allons créer les Threads, mettre l'algorithme que nous voulons exécuter dans chacun.
- **sendTo() :** (resp. **sendToNext()**) envoie un message d'un sommet *id* vers un canal (le sommet suivant si le graphe est orienté).
- **getNextMessage() :** (resp. **getNextMessageFromPrevious()** vérifie la file d'attente du sommet, s'il existe un message ou bien s'il existe un message d'un certain port retourne le message, sinon le noeud est bloqué jusqu'à la réception d'un message.
- **getArity() :** pour un noeud donné, cette méthode retourne le degré de ce noeud dans le graphe,
- **sizeGraph() :** retourne la taille du graphe (le nombre de noeuds du graphe),
- **putVertexProperty() :** permet de modifier ou d'ajouter une propriété à un noeud donné. Une propriété est définie par un identifiant qui est une chaîne de caractères et par un objet qui est la valeur de cette propriété,

– **getVertexProperty() :** retourne la valeur d'une propriété donnée d'un noeud passé en paramètre,

9.3.4 Un mot sur la simulation

La simulation souffre, à son tour, de plusieurs inconvénients. En particulier, elle produit un volume important de résultats qui nécessitent d'être résumés pour une bonne interprétation. Ceux-ci, contrairement à des propriétés déduites d'une étude analytique, qui sont d'une certaine façon "universelles", sont étroitement applicables au système modélisé. Les résultats obtenus sont de nature statistique et peuvent donc être interprétés de manière équivoque. En dépit de ces inconvénients, la simulation reste un outil privilégié dans notre étude.

Bibliographie

[AG05] A. Arnold and I. Guessarian. *Mathématiques pour l'informatique - 2e année*. Ediscience, 4e edition edition, 2005.

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on theory of computing*, pages 82–93, 1980.

[Ber85] C. Berge. *Graphs and Hypergraphs*. North-Holland, 1985.

[BGM+01a] M. Bauderon, S. Gruner, Y. Métivier, M. Mosbah, and A. Sellami. Visualization of distributed algorithms based on labeled rewriting systems. In *Second International Workshop on Graph Transformation and Visual Modeling Techniques, Crete, Greece*, volume 50 of *ENTCS*, pages 229–239, 2001.

[BGM01b] M. Bauderon, S. Gruner, and M. Mosbah. A new tool for the simulation and visualization of distributed algorithms. In *MFI'01, Toulouse, France*, volume I, pages 165–177, May 21-23 2001.

[BLS99] A. Brandstädt, V.B. Le, and J.P. Spinard. *Graph Classes : A Survey*. SIAM Monographs in Discrete Mathematics and Applications, 1999.

[BM96] M. Bousquet-Mélou. A method for the enumeration of various classes of column-convex polygons. *Discrete Mathematics*, 154 :1–25, 1996.

[BM2a] M. Bousquet-Mélou. Convex polyominoes and heaps of segments. *J. Phys. A : Math. Gen.*, 25 :1925–1934, 1992a.

[BMMS02] M. Bauderon, Y. Métivier, M. Mosbah, and A. Sellami. From local computation to asynchronous message passing systems. Technical Report RR-1271-02, University of Bordeaux 1, 2002.

[BNRR95] D. Beauquier, M. Nivat, E. Remila, and J.M. Robson. Tiling figures of the plane with two bars, a horizontal and a vertical one. *Computational Geometry*, 5 :1–25, 1995.

[Bod93] H.L. Bodlaender. A tourist guide through treewidth. *Acta Cybernet*, 11 :1–25, 1993.

[Bol01] B. Bollobas. *Random graphs*. Cambridge University Press, 2001.

[Bre04] A. Bretto. *Introduction to Hypergraph Theory and its Applications to Image Processing. Mongraphy in : Advances in Imaging and Electron Physics*. Academic Press, March 2004.

[BSV+96] P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon. Symmetry breaking in anonymous networks : Characterizations. In *Israel Symposium on Theory of Computing Systems*, pages 16–26, 1996.

[DDD95] M. Delest, J.-P. Dubernard, and I. Dutour. Parallelogram polyominoes and corners. *J. Symbolic Comput.*, 20 :503–515, 1995.

[Dem98] E. Demaine. Protocols for non-deterministic communication over synchronous channels. In *Proceedings of the 12th International parallel processing symposium / 9th Symposium on parallel and distributed processing (IPPS/SPDP'98*, pages 24–30. IEEE Computer Society, 1998.

[DFJ+98] L. Devroye, A.M. Frieze, M. Jerrum, C. McDiarmid, M. Molloy, R. Motwani, P. Raghavan, B. Reed, and D. Welsh. Probabilistic methods for algorithmic discrete mathematics. Springer, 1998.

[DGFHR01] Carole Delporte-Gallet, Hugues Fauconnier, Jean-Michel Hélary, and Michel Raynal. Early stopping in global data computing. Technical Report 2001-12, LIAFA, Université Paris 7, 2001.

[DHSZ04] P. Duchon, N. Hanusse, N. Saheb, and A. Zemmari. Broadcast in the rendezvous model. In *STACS*, pages 559–570, 2004.

[Dij72] E.W. Dijkstra. Hierarchical ordering of sequential processes. *Operating Systems Techniques, Academic Press,*, pages 72–93, 1972.

[Dij74] E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11) :643–644, 1974.

[DIM95] S. Dolev, A. Israeli, and S. Moran. Analyzing expected time by scheduler-luck games. *IEEE Transactions on Software Engineering*, 21(5) :429–439, May 1995.

[DV84] M.-P. Delest and G. Viennot. Algebraic languages and polyominoes [sic] enumeration. *Theoret. Comput. Sci*, 34 :169–206, 1984.

[FDRU90] U. Feige, D.Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. In *SIGAL International Symposium on Algorithms*, pages 128–137, 1990.

[Fel50] W. Feller. *An Introduction to Probability Theory and Its Application, Volume I*. John Wiley and Sons, 1950.

[Fel66] W. Feller. *An Introduction to Probability Theory and Its Application, Volume II*. John Wiley and Sons, 1966.

[FHR72] J.-R. Fiksel, A. Holliger, and P. Rosenstiehl. Intelligent graphs. In R. Read, editor, *Graph theory and computing*, pages 219–265. Academic Press (New York), 1972.

[Fla97] P. Flajolet. A seating arrangement problem. Technical report, ALCOM-FT, 1997.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2) :374–382, 1985.

[FR80] N. Francez and M. Rodeh. A distributed abstract data type implemented by a probabilistic communication scheme. In *Proceedings of the 21st ACM Symposium on FOCS*, pages 133–138. ACM Press, 1980.

[Gar56] M. Gardner. *Mathematics, magic and mystery*. New York. Dover Publications, 1956.

[Gar95] M. Gardner. *New mathematical diversions*. Washington. Math. Assoc. Amer., 1995.

[GH01] J.L. Ganley and L.S. Heath. The pagenumber of $k-$trees is $o(k)$. *Discrete Applied Mathematics*, 109(Issue 3) :215–221, May 2001.

[GHP95] P. Galinier, M. Habib, and C. Paul. Chordal graphs and their clique graph. In M. Nagl (Ed.), editor, *Graph-Theoretic Concepts in Computer Science, WG'95*, volume 1017 of *Lecture Notes in Computer Science*, pages 358–371. 21st Internationnal Workshop WG'95, Springer, June 1995.

[GM02] E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible. In Mogens Nielsen and Uffe Engberg, editors, *FoSSaCS*, volume 2303 of *Lecture notes in comput. Sci.*, pages 159–171. Springer, 2002.

[God02] E. Godard. *Réécritures de graphes et algorithmique distribuée*. PhD thesis, Université Bordeaux I, 2002.

[Gol54] S.W. Golomb. Checker boards and polyominoes. *American Mathematical Monthly*, 1954.

[Gol96] S.W. Golomb. *Polyominoes : puzzles, patterns, problems, and packings*. Princeton University Press, 1996.

[GSB94] R. Gupta, S.A. Smolka, and S. Bhaskar. On randomization in sequential and distributed algorithms. *ACM Comput. Surv.*, 26(1) :7–86, Mar 1994.

[GY04] J.L. Gross and J. Yellen. *Handbook of Graph Theory*. CRC Press, 2004.

[HHL88] S.M. Hedetniemi, ST. Hedetniemi, and A.L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4) :319–349, 1988.

[HKP01] M. Hanckowiak, M. Karonski, and A. Panconesi. On the distributed complexity of computing maximal matchings. *SIAM J. Discrete Math.*, 15(1) :41–57, 2001.

[HMR+06] A. El Hibaoui, Y. Métivier, J.M. Robson, N. Saheb-Djahromi, and A. Zemmari. Analysis of a randomized dynamic timetable handshake algorithm. Technical Report 1402-06, LaBRI, Université Bordeaux 1, May 2006.

[HSD06] A. El Hibaoui and Y N. Saheb-Djahromi. Broadcast with random delay transmission. September 2006.

[HSZ05a] A. EL HIBAOUI, N. SAHEB, and A. ZEMMARI. Polyominoids and uniform election. *FPSAC : 17th International Conference on Formal Power Series and Algebraic Combinatorics*, June 2005.

[HSZ05b] A. EL HIBAOUI, N. SAHEB, and A. ZEMMARI. A uniform probabilistic election algorithm in $k-$trees. *IMACS : 17th IMACS World Congress : Scientific Computation, Applied Mathematics and Simulation*, Jully 2005.

[IJ93] A. Israeli and M. Jalfon. Uniform self-stabilizing ring orientation. *Information and Computation*, 104(2) :175–196, 1993.

[IR90] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1) :60–87, 1990.

[IS93] R.M. Idury and A.A. Schäffer. Triangulating three-colored graphs in linear time and linear spaces. *SIAM J. Disc. Math.*, 6(2) :289–293, 1993.

[Knu98] D.E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, 3rd edition edition, 1998. This is a full BOOK entry.

[KPT00] B. Koldehofe, M. Papatriantafilou, and Ph. Tsigas. Lydian : An extensible educational animation environment for distributed algorithms. In *5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE'2000)*, page 189. ACM press, 2000.

[KT81] S. Karlin and H.M. Taylor. *A second course in stochastic processes*. Academic Press, 1981.

[KY96] T. Kameda and M. Yamashita. Computing on anonymous networks : Part ii - decision and membership problems. *IEEE Transactions on parallel and distributed systems*, 7(1) :90–96, 1996.

[Lan77] G. Le Lann. Distributed systems – toward a formal approach. In *Proceedings of the IFIP Congress 77*, pages 155–160, 1977.

[Lav95] C. Lavault. *Évaluation des algorithmes distribués*. HERMES, 1995.

[Lav05] C. Lavault. Mathématiques pour l'informatique. 2004-2005.

[Lec02] B. Leclerc. Graphes d'arches. *Math. & Sci. hum.*, 40(157) :27–48, 2002.

[Lei92] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures - Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.

[LFNV02] A. Del Lungo, A. Frosini, M. Nivat, and L. Vuillon. Discrete tomography : reconstruction under periodicity constraints. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2002.

[LMS92] I. Litovsky, Y. Métevier, and E. Sopena. Definition and comparison of local computations on graphs and networks. In *MFCS'92*, volume 629 of *Lecture Notes in Comput. Sci.*, pages 364–373, 1992.

[LMS95] I. Litovsky, Y. Métivier, and E. Sopena. Different local controls for graph relabelling systems. *Math. Syst. Theory*, 28 :41–65, 1995.

[LMS99] I. Litovsky, Y. Métivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In H. Ehrig, H.J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of graph grammars and computing by graph transformation*, volume 3, pages 1–56. World Scientific, 1999.

[LMZ95] I. Litovsky, Y. Métivier, and W. Zielonka. On the recognition of families of graphs with local computations. *Inform. and Comput.*, 118 :110–119, 1995.

[LR81] D. Lehmann and M.O. Rabin. On the advantage of free choice : A symmetric and fully distributed solution to the dining philosophers problem. *Proc. of POPL*, pages 133–138, 1981.

[LR94] D. Lehmann and M.O. Rabin. On the advantages of free choice : A symmetric and fully distributed solution to the dining philosophers problem. Number chapter 20, pages 333–352, Prentice Hall, 1994.

[Lyn96] N.A. Lynch. *Distributed algorithms*. Morgan Kaufman, 1996.

[MPT98] Y. Moses, Z. Polunsky, and A. Tal. Algorithm visualization for distributed environments. In *Proceedings IEEE Symposium on Information Visualization*, pages 71–78, 1998.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS] M. Mosbah and A. Sellami. Visidia : A tool for the VIsualization and SImulation of DIstributed Algorithms. *http ://www.labri.fr/visidia/.*

[MS94] Y. Métivier and N. Saheb. Probabilistic analysis of an election algorithm in a tree. In *Colloquium on trees in algebra and programming*, volume 787 of *Lecture Notes in Comput. Sci.*, pages 234–246. Spinger-Verlag, 1994.

[MS96] Y. Métivier and N. Saheb. Medians and centres of polyominos. *Inform. Proc. Letters*, 57 :175–181., 1996.

[MS04] M. Mosbah and A. SELLAMI. Un environnement général pour étudier et implèmenter les algorithmes distribués. In *Eighth Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI'04)*, pages 161–172. Centre De Publication Universitaire, 2004.

[MSZ03a] Y. Métivier, N. Saheb, and A. Zemmari. Analysis of a randomized rendezvous algorithm. *Information and Computation*, 184 :109–128, 2003.

[MSZ03b] Y. Métivier, N. Saheb, and A. Zemmari. A uniform randomized election in trees (extended abstract). In *Proceedings of The 10th International Colloquium on Structural Information and Communication Complexity (SIROCCO 10)*, pages 259–274. Carleton university press, 2003.

[Mul89] S. Mullender. *Distributed Systems*. ACM Press, 1989.

[Pel00] D. Peleg. *Distributed Computing, a locality-sensitive approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[PP05] A. Pelc and D. Peleg. Feasibility and complexity of broadcasting with random transmission failures. In *PODC*, pages 334–341, 2005.

[PTV92] W.H Press, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[Rab82] M. Rabin. N-process mutual exclusion with bounded waiting by 4 log 2 n - valued shared variable. *Journal of Computer and System Sciences*, 25(1) :66–75, August 1982.

[Ray88] M. Raynal. *Networks and distributed computation : concepts, tools, and algorithms*. MIT Press, Cambridge, 1988.

[Ros74] D.J. Rose. On simple characterization of k-trees. *Discrete Mathematics*, 7 :317–322, 1974.

[Ros00] K.H. Rosen. *Handbook of discrete and combinatorial mathematics*. CRC Press, 2000.

[RRP97] F. Robin, M. Renaudin1, and G. Privat. Functionally asynchronous vlsi cellular array for morphological filtering of images. *Traitement du Signal*, 14(6) :655–64, 1997.

[RS82] J. Reif and P. Spirakis. Real time resource allocation in distributed systems. In *Proceedings of the 1th ACM Symposium on principles of distributed computing*, pages 84–94. ACM Press, 1982.

[Sed92] R. Sedgewick. *Algorithms in C++*. Addison-Wesley Co., 1st edition, 1992.

[Sel04] A. Sellami. *Des Calculs Lacaux aux Algorithmes Distribués*. PhD thesis, Université Bordeaux I, 2004.

[Tel91] G. Tel. Topics in distributed algorithms. 1991.

[Tel94] G. Tel. Introduction to distributed algorithms. 1994.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.

[Tem56] H.N.V. Temperley. Combinatorial problems suggested by the statistical mechanics of domains and of rubber-like molecules. *Phy.*, 103 :1–16, 1956.

[TvR85] Andrew S. Tanenbaum and Robert van Renesse. Distributed operating systems. *ACMCS*, 17(4) :419–470, Dec 1985.

[Val82] L.G. Valiant. A scheme for fast parallel communication. *SIAM J. Comput.*, 11(2) :350–361, 1982.

[YK96] M. Yamashita and T. Kameda. Computing on anonymous networks : Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1) :69–89, 1996.

[YK99] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transaction on Parallel and Distributed Systems*, 9(10) :878–887, 1999.

Index

T

V