Présentation et analyse de quelques algorithmes distribués probabilistes

Akka Zemmari

▶ **To cite this version:**

Akka Zemmari. Présentation et analyse de quelques algorithmes distribués probabilistes. Complexité [cs.CC]. Université de Bordeaux, 2009. tel-01879754

HAL Id: tel-01879754
https://theses.hal.science/tel-01879754

Submitted on 24 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Numéro d'ordre : 447

Université Bordeaux 1
Laboratoire Bordelais de Recherche en Informatique

HDR

présentée en vu d'obtenir l'habilitation à diriger les recherches,
spécialité « Informatique »

par

Akka ZEMMARI

Présentation et analyse de quelques algorithmes distribués probabilistes

Après avis des rapporteurs :

M. CHRISTIAN LAVAULT	Professeur	Université Paris 13
M. DAVID PELEG	Professeur	Weizmann Institute of Science
M. MASAFUMI YAMASHITA	Professeur	Kyushu University

HDR soutenue le 19 octobre 2009 devant le jury composé de :

M.	Christian Lavault	Professeur	Université Paris 13	(Rapporteur)
M.	Yves Métivier	Professeur	ENSEIRB	(Examinateur)
M.	David Peleg	Professeur	Weizmann Institute of Science	(Rapporteur)
M.	Franck Petit	Professeur	ENS Lyon	(Examinateur)
M.	John Michael Robson	Professeur	Université Bordeaux 1	(Examinateur)
M.	Nasser Saheb	MdC HDR	Université Bordeaux 1	(Examinateur)

À M., Z. et A., je vous aime
à mon père, tu me manques…

Remerciements

Je tiens à remercier les membres du jury pour l'intérêt qu'ils ont manifesté à ce travail. Je remercie particulèrement Messieurs les Professeurs Christian Lavault, David Peleg et Masafumi Yamashita d'avoir accepté d'être rapporteurs.

Mes remerciement très chaleureux à mes deux maîtres et amis : Yves Métivier et Nasser Saheb. Les différentes discussions, aussi bien au bureau qu'autour d'un café sont pour beaucoup dans l'aboutissement des travaux que j'ai pu mené depuis ma première année de thèse jusqu'à aujourd'hui.

Je remercie également Mike Robson, Mohamed Mosbah ainsi que Jean-François Marckert avec qui j'ai eu le plaisir de travailler ces dernières années.

Je tiens à exprimer ma gratitude à l'ensemble des membres du LaBRI, permanents, doctorants, administratifs et équipe dirigeante.

Enfin, un grand merci à mon grand soutien, Meriem, ma chère et douce épouse.

Talence, le 15 octobre 2009.

Table des matières

Préface

Ce document présente une partie de mes activités de recherche au sein de l'équipe Combinatoire et Algorithmes du Laboratoire Bordelais de Recherche en Informatique (LaBRI). Il est composé de six chapitres :

–le premier chapitre donne une introduction générale au domaine dans lequel s'inscrit ma recherche : l'analyse des algorithmes distribués, et en particulier des algorithmes distribués probabilistes. Il présente les définitions et notations utilisées dans ce mémoire, le modèle adopté ainsi que les outils communs aux résultats qui seront présentés dans les chapitres suivants.

–Le deuxième chapitre présente un algorithme distribué probabiliste permettant d'obtenir une élection équitable dans un graphe dont la topologie est un arbre. Par l'équité, nous entendons un algorithme donnant la même chance à tous les sommets de l'arbre d'être élus.

–Le chapitre suivant présente une extension de l'algorithme d'élection équitable dans les arbres à la classes des polyominoïdes.

–Le quatrième chapitre expose et analyse trois algorithmes distribués probabilistes permettant de construire un ensemble indépendant maximal. Le premier algorithme est basé sur l'envoi de nombres réels, le deuxième donne une implantation de l'algorithme précédent en utilisant des bits. Enfin le troisième algorithme améliore la bit complexité du précédent.

–Le chapitre cinq présente un algorithme optimal pour colorier les sommets d'un graphe. L'algorithme est à base d'envoi de messages de taille 1 bit.

–Les deux derniers chapitres présentent deux algorithmes permettant de réaliser des rendez-vous dans un graphe. Le premier algorithme utilise un temps discret et le second utilise les propriétés des variables aléatoires continues.

Les travaux présentés dans cette dissertation ont été réalisés en collaboration avec A. El Hibaoui, Y. Métivier, J. M. Robson et N. Saheb.

1 Définitions, modèles et résultats généraux

Ce chapitre présente les définitions et notations utilisées dans ce mémoire : les graphes (Ber85), (Roooo) et le modèle d'algorithmique distribuée (Teloo), (Lyn96) et (Lav95).

Il présente également quelques éléments de l'analyse d'algorithmes probabilistes.

1.1 Les graphes

Dans ce mémoire, nous utilisons les définitions et notations usuelles de la théorie des graphes. Un *graphe non orienté simple* est un couple $G = (V, E)$, où V est un ensemble de *sommets* et E est un ensemble de paires d'éléments *distincts* de V, appelés *arêtes*, la *taille* du graphe G est le nombre d'éléments de V. Si $e = \{u, v\} \in E$ est une arête de G, on dit que e est *incidente* à u et v et que u et v sont deux sommets *voisins*. Soit v un sommet du graphe G et $N(v)$ l'ensemble des voisins de v, le nombre d'éléments dans $N(u)$ est appelé *degré* du sommet u, et est noté $d(u)$.

Soit $G = (V, E)$ un graphe. Un *sous-graphe* du graphe G est un graphe $H = (V', E')$ dont l'ensemble des sommets V' (respectivement l'ensemble des arêtes E') est un sous-ensemble de V (respectivement de E). Un *sous-graphe couvrant* du graphe G est un sous-graphe de G qui contient tous les sommets de G.

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

1.2 Notations

Nous utilisons les notations suivantes (voir (SF96)). Soient f et g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}^+$. On dit que :

- $f(n) = O(g(n))$ si et seulement si il exite une constante c et un entier n_0 tels que $\forall n > n_0$, $f(n) \leq cg(n)$.
- $f(n) = \Omega(g(n))$ si et seulement si il exite une constante c et un entier n_0 tels que $\forall n > n_0$, $f(n) \geq cg(n)$.
- $f(n) = \Theta(g(n))$ si et seulement si il exite deux constantes $c_1 < c_2$ et un entier n_0 tels que $\forall n > n_0$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$.
- $f(n) = o(g(n))$ si $f(n)$ est négligeable devant $g(n)$, c'est-à-dire, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
- $f(n) \sim g(n)$ si $f(n)$ est asymptotiquement équivalente à $g(n)$, c'est-à-dire, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$.

1.3 Modèles

Dans cette section, nous rappelons quelques notions de l'algorithmique distribuée (Teloo), (Lyn96) et (Lav95).

Système distribué

Un *système distribué* est une collection d'unités physiques ou logiques autonomes - unités ayant leurs propres moyens de contrôle et de calcul - communiquant via des liens de communication. Ces unités peuvent être des ordinateurs ou des processeurs (unités physiques) ou simplement des processus (unités logiques).

Dans la suite, nous utiliserons assez souvent le terme « processus » pour désigner les unités du système distribué.

Système distribué synchrone

Dans un système distribué *synchrone*, une action (ou opération) de chaque processus se réalise en une suite de pas discrets appelée *round*. Dans un round, un processus envoie 0 ou plusieurs messages, reçoit 0 ou plusieurs messages et effectue un ensemble de calculs locaux. Les messages envoyés sont reçus pendant le même round, c'est-à-dire, si p envoie un message à q au $i^{\text{ème}}$ round, alors le message est reçu par q pendant le $i^{\text{ème}}$ round de q.

Système anonyme

Un système distribué est dit *anonyme* si on ne peut accéder aux identités des différentes unités du système. Dans la littérature (voir (Lav95)), on parle de *symétrie* : les processus jouent tous le même rôle et on ne dispose d'aucun moyen pour les distinguer. Un des grands paradigmes de l'algorithmique distribuée est de *briser* cette symétrie (Ang80, Lav95), et un moyen pour y parvenir est d'utiliser les *algorithmes probabilistes*.

Algorithme distribué

Soit $\mathcal{T}$ une tâche à réaliser et soit $\mathcal{S}$ un système distribué à $n \geq 1$ entités. Un *algorithme distribué* $\mathcal{A}$ pour la tâche $\mathcal{T}$ est un algorithme capable de faire collaborer tous, ou partie, des entités du système $\mathcal{S}$ afin de réaliser $\mathcal{T}$. L'algorithme $\mathcal{A}$ peut également être vu comme une collection d'algorithmes locaux $(\mathcal{A}_i)_{1 \leq i \leq n}$, chaque algorithme $\mathcal{A}_i$ étant exécuté par une des n entités du système $\mathcal{S}$.

Plus formellement, voir (Lav95, Teloo), un algorithme distribué exécuté sur un système distribué $\mathcal{S}$, est un système de transitions $\mathcal{A} = (\mathcal{C}, \rightarrow, \mathcal{I})$ tel que $\mathcal{C}$ est un ensemble de configurations de S, $\rightarrow \subseteq \mathcal{C} \times \mathcal{C}$ est une relation binaire de transition, $\mathcal{I} \subseteq \mathcal{C}$ est un sous-ensemble de configurations initiales de $\mathcal{S}$. Un *algorithme distribué local* exécuté sur un processus P de $\mathcal{S}$ est un système de transitions $(\mathcal{E}, \mathcal{J}, \rightarrow^i, \rightarrow^e, \rightarrow^r)$ tel que $\mathcal{E}$ est l'ensemble des états de P, $\mathcal{J} \subseteq \mathcal{E}$ un sous-ensemble d'états initiaux, et $\rightarrow^i, \rightarrow^e, \rightarrow^r$ trois relations de transition d'états de P correspondant, respectivement, aux actions internes, aux envois et enfin aux réceptions de messages.

Un *algorithme probabiliste* est un algorithme qui utilise 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'exécution de tels algorithmes. Les algorithmes probabilistes permettent de fournir des solutions parfois plus "efficaces" que les solutions déterministes, ou encore des solutions pour des problèmes qui n'admettent pas de solution déterministe.

Plus formellement, voir (Teloo) chapitre 9, un *processus probabiliste*, est défini par un quadruple $p = (\mathcal{E}, \mathcal{J}, \rightarrow^0, \rightarrow^1)$ où $\mathcal{E}$ et $\mathcal{J}$ sont définis comme ci-dessus et $\rightarrow^0$ et $\rightarrow^1$ sont des relations binaires sur $\mathcal{E}$ telles que le $i^{\text{ème}}$ pas du processus est exécuté selon la transition dans $\rightarrow^0$ si le résultat du tirage est 0 et selon la transition dans $\rightarrow^1$ sinon.

Un *algorithme probabiliste distribué* $\mathcal{A}_p$ est par conséquent une collection d'algorithmes locaux $\mathcal{A}_p = (\mathcal{A}_i)_{1 \leq i \leq n}$ telle que chaque algorithme $\mathcal{A}_i$ est un algorithme (ou processus) probabiliste.

Dans ce mémoire, nous considèrons un système distribué synchrone, communiquant par échange de messages : un processus envoie un message à son voisin en le déposant dans le lien correspondant à ce voisin. Nous modélisons le système par un graphe $G = (V, E)$ où V est un ensemble de sommets représentant l'ensemble des entités du système et E l'ensemble des arêtes représentant les liens de communication entre les entités.

Remarque 1.1 *Le modèle présenté dans cette section est le modèle commun à la plupart des chapitres présentés dans ce mémoire. En effet, les deux premiers chapitres et le chapitre 7 utilisent plutôt les caractéristiques des v.a. continues.*

1.4 Analyse des algorithmes distribués probabilistes

Les deux principales mesures auxquelles on s'intéresse sont le temps d'exécution de l'algorithme et le nombre ou la taille des messages échangés. On s'intéresse principalement à l'espérance mathématique de la valeur de la mesure considérée.

Une autre analyse consiste à comparer l'algorithme distribué probabiliste à un algorithme "idéal" et donc non nécessairement distribué ni probabiliste. Cette analyse est utilisée lorsque le but de l'algorithme est de construire des synchronisations locales dans le système distribué, ces synchronisations peuvent être des ensembles indépendants, des étoiles ou encore des poignées de mains (voir plus loin).

1.4.1 Complexité en temps

Dans ce mémoire, nous nous plaçons dans le cadre d'un système distribué synchrone. L'analyse de la complexité en temps des algorithmes proposés utilise l'hypothèse de synchronisme. C'est une hypothèse forte, néanmoins, comme c'est indiqué dans (Lav95), page 53, ceci permet d'obtenir une "bonne" estimation du temps d'exécution de l'algorithme si le système est supposé être asynchrone.

Complexité moyenne

On considère un système distribué synchrone $\mathcal{S}$ à $n \geq 1$ entités. Soit $\mathcal{T}$ une tâche à réaliser sur $\mathcal{S}$ et $\mathcal{A}_p = (\mathcal{A}_i)_{1 \leq i \leq n}$ un algorithme probabiliste permettant de réaliser la tâche $\mathcal{T}$. Sans perte de généralité, on considère que l'algorithme $\mathcal{A}_p$ démarre à l'instant $t = 0$. Nous définissons la variable aléatoire (v.a.) T comme le nombre d'unités de temps qui s'écoulent entre l'instant $t = 0$ et la plus petite valeur de t telle que $\forall i \in \{1, 2, \cdots, n\}$, l'algorithme $\mathcal{A}_i$ est terminé à l'instant t. La complexité moyenne de l'algorithme $\mathcal{A}_p$ est donc l'espérance mathématique de la v.a. T.

Dans l'analyse des algorithmes probabilistes, on s'intéresse principalement à l'espérance de T. Cependant, une analyse plus fine, et parfois plus complexe, permet de calculer, du moins asymptotiquement, la distribution de T. Dans beaucoup de cas, nous pouvons montrer que la probabilité que T dépasse $C(n)$ (une fonction de la taille n du graphe) est majorée par $1/n^{\alpha}$, avec $\alpha \geq 1$. On dit alors que $T \leq C(n)$ avec forte probabilité.

Cette section introduit quelques résultats communs aux algorithmes probabilistes proposés et analysés dans ce mémoire. Il s'agit d'algorithmes distribués permettant de calculer un MIS, un couplage ou encore un coloriage.

Ces algorithmes ont en commun le fait qu'ils peuvent être décomposés en phases. À la fin de chaque phase, un ensemble d'entités du système peut décider, en fonction de leur état et des états de leurs voisins, si leur participation à l'algorithme est terminée. Ils se retirent alors du système

et les algorithmes se poursuivent sur le système résiduel. Le processus continue ainsi jusqu'à ce que toutes les entités aient pris leurs décisions finales. Plus formellement, Soit $G = (V, E)$ un graphe connexe non orienté tel que $|V| = n$ et $|E| = m$ modélisant un système distribué. Soit $\mathcal{P}$ un problème de la même nature que le couplage, le MIS ou le coloriage, admettant une solution : un algorithme probabiliste distribué $\mathcal{A}$. On suppose que $\mathcal{A}$ fonctionne par phases : à la fin de chaque phase, un ensemble de sommets (et les arêtes adjacentes) sont supprimés du graphe et l'exécution de l'algorithme se poursuit sur le graphe résiduel jusqu'à ce que le graphe soit réduit à un seul sommet. On définit la suite de graphes $(G_i)_{i\geq 0}$ par : $G_0 = G$ et pour tout $i \geq 1$, G_i est le graphe obtenu à partir de G_{i-1} après la $i^{\text{ème}}$ phase de l'algorithme $\mathcal{A}$. Nous avons le lemme suivant :

Lemme 1.1 *Soit T la complexité en temps de l'algorithme $\mathcal{A}$. S'il existe une constante $\epsilon > 0$ telle que, à chaque phase, le nombre moyen d'arêtes (resp. de sommets) supprimées par l'algorithme $\mathcal{A}$ soit supérieur ou égal à ϵm (resp. ϵn), où m et n sont respectivement le nombre d'arêtes et le nombre de sommets du graphe au début de la phase, alors l'espérance mathématique de T vérifie $\mathbb{E}(T) = O(\log n)$.*

Preuve. Nous allons prouver le lemme en considérant que l'algorithme supprime une fraction d'arêtes. Le même raisonnement peut s'appliquer si on remplace les arêtes par les sommets. Soit $(G_i)_{i\geq 0}$ la suite de graphes définie ci-dessus. Soit X_i la v.a. qui correspond au nombre d'arêtes du graphe G_i pour tout $i \geq 0$, et soit Y_i le nombre d'arêtes supprimées à la $i^{\text{ème}}$ phase. D'après l'hypothèse du lemme, nous avons l'inégalité suivante :

$$\mathbb{E}(Y_i \mid G_i) \geq \epsilon X_i.$$

Il est aussi facile de voir que $X_{i+1} = X_i - Y_i$ pour tout $i \geq 0$. Donc,

$$\mathbb{E}(X_{i+1} \mid G_i) = X_i - \mathbb{E}(Y_i \mid G_i) \leq X_i \cdot (1 - \epsilon)$$

Pour tout $i \geq 0$, définissons la nouvelle v.a. Z_i par $Z_i = X_i/(1-\epsilon)^i$. Alors, $\mathbb{E}(Z_{i+1} \mid G_i) \leq Z_i$. Ainsi, la v.a. Z_i est une sur-martingale (voir (Wil93)), et donc

$$\mathbb{E}(Z_{i+1}) = \mathbb{E}(\mathbb{E}(Z_{i+1} \mid G_i)) \leq \mathbb{E}(Z_i).$$

Une application directe d'un théorème de (Wil93) chapitre 9, donne $\mathbb{E}(Z_i) \leq Z_0 = m$. Donc

$$\mathbb{E}(X_i) = (1-\epsilon)^i \mathbb{E}(Z_i) \leq n(1-\epsilon)^i.$$

L'algorithme $\mathcal{A}$ s'arrête quand G_i est réduit à une seule arête, c'est-à-dire $X_i = 1$. Ce qui implique que i est inférieur ou égal au rapport $\log(n)/\log(\frac{1}{1-\epsilon})$. Ce qui termine la preuve. □

Complexité avec grande probabilité

Le lemme 1.1 permet d'obtenir une borne supérieure logarithmique de la complexité en moyenne d'un algorithme probabiliste distribué. Dans cette

section, nous donnons un résultat général permettant, de montrer que la complexité est logarithmique avec forte probabilité.

Soit $\mathcal{A}$ un algorithme probabiliste distribué résolvant un problème $\mathcal{P}$ dans un graphe $G = (V, E)$. $\mathcal{A}$ est exécuté par tous les sommets de G et composé de plusieurs phases.

Lemme 1.2 *Soit $C(n)$ une fonction de n. Si chaque sommet v termine l'exécution de l'algorithme $\mathcal{A}$ en un temps T_v et si pour tout $v \in V$, $\mathbb{P}r\left(T_v \geq C(n)\right) = O\left(\frac{1}{n^{1+\varepsilon}}\right)$, avec $\varepsilon > 0$, alors*

$$\mathbb{P}r\left(T > C(n)\right) = o\left(1\right).$$

Preuve. Nous avons :

$$\begin{aligned} \mathbb{P}r\left(T > C(n)\right) &= \mathbb{P}r\left(\exists v \in V \mid T_v \geq C(n)\right) \\ &\leq \sum_{v \in V} \mathbb{P}r\left(T_v \geq C(n)\right) \\ &= nO\left(\tfrac{1}{n^{1+\varepsilon}}\right) = O\left(\tfrac{1}{n^{\varepsilon}}\right). \end{aligned}$$

Ce qui démontre le lemme. □

Corollaire 1.1 *Si $C(n) = K \log n$, alors si tout sommet v du graphe termine l'algorithme en $T_v = O\left(\log n\right)$ avec probabilité $o\left(\frac{1}{n^{1+\varepsilon}}\right)$, $(\varepsilon > 0)$, alors, $T = O\left(\log n\right)$ avec forte probabilité.*

Nous avons vu dans le lemme 1.1 que si chaque phase supprime, en moyenne, au moins une fraction du nombre d'arêtes (resp. de sommets) du graphe résiduel au début de la phase, alors l'espérance du nombre de phases nécessaires pour la suppression de toutes les arêtes (resp. de tous les sommets) est au plus logarithmique en m (resp. en n). Nous avons également le lemme suivant :

Lemme 1.3 *S'il existe deux constantes a et $b \in]0, 1[$, telles que pour toute phase $i \geq 1$,*

$$\mathbb{P}r\left(\frac{Y_i}{X_i} > a\right) > b,$$

alors, il existe $K > 0$ telle que

$$\mathbb{P}r\left(T \geq K \log n\right) = o(m^{-1}).$$

Preuve. Soit $S = \{\phi_1, \phi_2, \cdots, \phi_t\}$ une suite de phases aboutissant à la terminaison de l'algorithme $\mathcal{A}$. Une phase ϕ_i est dite *bonne* (resp. *mauvaise*) si $\frac{Y_i}{X_i} > a$ (resp. $\frac{Y_i}{X_i} \leq a$). Soit Y le nombre de mauvaises phases dans S. La v.a. Y est alors dominée par une v.a. binomiale de paramètres t et $1 - b$.

Par ailleurs, nous avons la propriété suivante (borne de Chernoff (HMRAR98)) :

pour tout $0 < a < n_0 p_0$:

$$\mathbb{P}r\left(\mid BIN(n_0, p_0) \mid\right) < 2e^{-a^2/3n_0p_0},$$

où $BIN(n_0, p_0)$ est une v.a. binomiale à paramètres n_0 et p_0.

Donc, en prenant $t = K \log_{\frac{1}{1-a}} m$ et $a = bt - \log_{\frac{1}{1-a}} m$, ($K$ fonction de a et b), nous avons :

$$\mathbb{P}r\left(Y \geq t - \log_{\frac{1}{1-a}} m\right) = o(m^{-1}).$$

Ainsi, la probabilité que la séquence S contienne au moins $\log_{\frac{1}{1-a}} m$ bonnes phases vaut $1 - o(m^{-1})$. □

Remarque 1.2 *Nous pouvons montrer que si $\mathbb{P}r(T \leq \log n) = 1 - o\left(n^{-1}\right)$, alors $\mathbb{E}(T) \leq \log n$. L'inverse n'est pas vraie. En effet, considérons l'exemple suivant. Soit $G = (V, E)$ un graphe de taille $n = 2m$ (un nombre pair de sommets) tel que $V = \{v_1, v_2, \cdots, v_n\}$ et $\forall i \in \{1, 2, \cdots, n\}$, $\{v_i, v_{i+1}\} \in E$ si et seulement si i modulo $2 = 1$. Considérons la tâche $\mathcal{T}$ consistant à briser la symétrie entre les sommets adjacents dans G. Pour cette tâche, nous proposons l'algorithme $\mathcal{A}$ suivant. Chaque sommet v exécute les actions suivantes : (1) v génère un bit $b_v \in \{0, 1\}$ de manière équiprobable, (2) v envoie b_v à son voisin, (3) v reçoit le bit b_u envoyé par son voisin u. Si $b_u \neq b_v$ alors v arête l'algorithme sinon v reprend le point (1). Il est alors clair que l'algorithme $\mathcal{A}$ réalise la tâche $\mathcal{T}$.*

Il est aussi clair que le nombre moyen d'arêtes supprimées par chaque phase est égal à la moitié du nombre d'arêtes dans le graphe au début de cette phase. On en déduit que le nombre moyen de phases nécessaires à l'exécution de l'algorithme $\mathcal{A}$ est $\mathbb{E}(T) = O(\log n)$.

Plus précisément, pour tout $k \geq 1$,

$$\mathbb{P}r(T > k) = 1 - \left(1 - \frac{1}{2^k}\right)^n.$$

Donc :

$$E(T) = \sum_{k \geq 0} \left(1 - \left(1 - \frac{1}{2^k}\right)^n\right).$$

En utilisant la transformée de Mellin (voir chapitre 5), nous obtenons

$$E(T) \sim \log n + \epsilon + Q(\log_2 n) + O\left(\frac{1}{n^2}\right),$$

où ϵ est une constante et Q est une série de Fourier de période 1 et d'amplitude $< 10^{-6}$.

Par ailleurs, pour $k = \log_2 n$, nous avons

$$\mathbb{P}r(T > \log_2 n) \sim 1 - e^{-1}.$$

L'exemple montre que l'espérance de la complexité de l'algorithme $\mathcal{A}$ est asymptotiquement égale à $\log n$ mais que la probabilité que la complexité soit plus que logarithmique en n est une constante.

1.4.2 Efficacité des algorithmes probabilistes distribués

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

Les définitions suivantes concernent l'efficacité des algorithmes probabilistes permettant d'implémenter des solutions pour les problèmes suivants :

- problème du rendez-vous (ou poignées de mains) : il s'agit de proposer des algorithmes permettant de synchroniser des paires (disjointes) de sommets adjacents dans le graphe,
- élections locales LC_1 : il s'agit de proposer des algorithmes permettant de réaliser des élections de sommets indépendants (sommets non voisins) dans le graphe,
- élections locales LC_2 : il s'agit de proposer des algorithmes permettant de réaliser des élections de sommets à distance 2 les uns des autres dans le graphe,

Nous verrons plus loin les définitions formelles de ces problèmes et les algorithmes proposés.

Définition 1.1 *(MSZ03)* Soit $\mathcal{A}$ un algorithme probabiliste pour les synchronisations de type rendez-vous fonctionnant sur les graphes. Son *efficacité* sur un graphe quelconque G est le rapport :

$$\Delta_{\mathcal{A}}(G) = \frac{\overline{M}_{\mathcal{A}}(G)}{K(G)}, \tag{1.1}$$

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

De la même manière, nous définissons l'efficacité d'un algorithme réalisant des synchronisations de sommets avec tous les voisins comme suit :

Définition 1.2 *(MSZ02)* Soit $\mathcal{A}$ un algorithme probabiliste quelconque permettant de réaliser des synchronisations de type LC_1. Son *efficacité* $\Delta_{\mathcal{A}}(G)$ sur un graphe G est le rapport :

$$\Delta_{\mathcal{A}}(G) = \frac{\overline{M}_{1\mathcal{A}}(G)}{\alpha(G)}, \tag{1.2}$$

où $\overline{M}_{1\mathcal{A}}(G)$ est le nombre moyen de LS_1-synchronisations réalisées par $\mathcal{A}$ et $\alpha(G)$ est la taille maximum d'un MIS.

Il est alors facile de voir que ces définitions peuvent être étendues à tous les algorithmes permettant de construire des sortes de composantes locales d'un graphe. L'idée étant, comme annoncé ci-dessus, de comparer le nombre moyen de composantes construites par l'algorithme probabiliste au nombre maximum de composantes que pourrait construire un algorithme idéal.

Élection probabiliste dans les arbres

2

Le travail présenté dans ce chapitre a été initié par Y. Métivier et N. Saheb dans (MS94). Dans cet article, les auteurs analysent différentes approches pour élire un sommet dans un arbre et étudient la probabilité d'être élu pour un sommet quelconque de l'arbre.

Dans (MSDZ03) et (MSDZ05), nous avons proposé un algorithme complètement distribué permettant de réaliser une élection équitable dans les arbres : si l'arbre est de taille $n > 0$, alors tous ses sommets ont la même probabilité $\frac{1}{n}$ d'être élus.

Deux questions naturelles se posent : 1. peut-on adapter notre algorithme à d'autres classes de graphes ? 2. peut-on adapter notre algorithme pour que la probabilité, pour un sommet de l'arbre, d'être élu soit guidée par une loi de probabilité donnée ?

Dans (ESDZ05a), nous avons proposé un algorithme donnant la même chance d'être élu à tous les sommets d'un k-arbre et dans (ESDZ05b) et (ERSDZ), nous avons étendu l'algorithme au cas des polyominoïdes ce qui répond, en partie, à la question 1. Pour la question 2. nous avons initié un travail et obtenu quelques résultats dans (MSDZ09).

2.1 Introduction

Le but d'une élection est de choisir un élément d'un ensemble : cet élément est appelé *élément élu*. Il peut être utilisé pour prendre des décisions dans un système réparti ; il peut également centraliser des informations. Ce problème est l'un des grands paradigmes de l'algorithmique distribuée. Il a été présenté pour la première fois par LeLann dans (Le 77), depuis, il est étudié sous différentes hypothèses : des recherches ont été menées pour présenter des solutions à ce problème, et d'autres recherches ont été menées pour caractériser les graphes dans lesquels des solutions déterministes peuvent être trouvées.

Notre travail consiste a proposer des solutions probabilistes à ce problème d'abord dans le cas des arbres, puis dans le cas d'autres classes de graphes.

2.2 Élection dans les arbres

Dans le cas des arbres, il est facile de décrire les algorithmes d'élection au moyen d'un système de réécriture de graphe $\mathcal{R}$ (voir (MS97a)). On suppose que tous les sommets et toutes les arêtes de l'arbre portent la même étiquette ; cette hypothèse codant l'anonymat du système réparti : les processus et les canaux sont indistinguables. Ensuite, on décrit les règles de réécriture à appliquer :

L'ensemble des étiquettes est l'ensemble $\{\mathbf{C}, \mathbf{P}, \mathbf{E}\}$: **C** pour candidat, **P** pour perdant et **E** pour élu. Initialement, tous les sommets de l'arbre sont étiquetés avec **C**. Le système de réécriture $\mathcal{R}$ a deux règles :

R_1 : un sommet x, étiqueté **C**, qui est adjacent à exactement **un** sommet étiqueté **C**, peut être réétiqueté avec **P**,

R_2 : si un sommet étiqueté **C** n'a aucun sommet étiqueté **C** dans son voisinage, alors il peut être réétiqueté **E**.

Nous avons (voir (MS94)) :

Proposition 2.1 *Si le système **R** est appliqué à un arbre à n sommets, qui sont initialement étiquetés avec **C**, alors, après n applications des règles de réécriture, on obtient un graphe irréductible avec un sommet unique étiqueté par **E**, qui sera considéré comme le* sommet élu.

L'algorithme décrit par le système de réécriture est un algorithme générique correspondant à un effeuillage de l'arbre jusqu'à ce que celui-ci soit réduit à un seul sommet : le sommet élu. L'objectif de l'étude de cet algorithme est de calculer pour chaque sommet, la probabilité qu'il soit élu et ceci selon les différentes hypothèses posées.

Analyse de l'algorithme générique

On considère un arbre $T = (V, E)$ de taille $n \geq 2$. D'après la proposition 2.1, si le système de réécriture $\mathcal{R}$ est appliqué à T (avec tous les sommets étiquetés **C**), après n applications des règles de réécriture, on obtient un arbre irréductible dont un seul sommet est étiqueté **E** (sommet élu). Le processus d'application des règles peut alors être vu comme une suite de suppression des feuilles de l'arbre, la feuille restante étant le sommet élu.

Tant que l'arbre n'est pas irréductible, il y a au moins deux possibilités à chaque étape. L'objectif de l'analyse du processus décrit ci-dessus est de calculer la probabilité pour chaque sommet d'être élu.

Dans (MS94), les auteurs ont étudié le processus sous deux approches :

i) toutes les suites de suppressions sont de même probabilité,

ii) à chaque étape, toutes les feuilles ont la même chance d'être supprimées, et cela indépendamment de tous les événements précédents.

Pour la première approche, les auteurs ont réalisé une analyse fine et complète de l'algorithme et ont caractérisé le sommet qui a la plus forte probabilité d'être élu. Il s'agit du (des) sommet(s) médian(s). Cependant, cette approche n'est pas implémentable de manière distribuée. A contrario, la

seconde approche présente l'avantage d'être facilement implémentable en distribué mais aucune caractérisation du ou des sommets qui ont la plus forte chance d'être élus n'a été établie.

Dans (MSDZ03) et (MSDZ05), nous avons proposé une implémentation distribuée du processus décrit ci-dessus donnant au final la même probabilité à tous les sommets de l'arbre d'être élus. Dans le reste de cette section, nous présentons cette implémentation ainsi que l'analyse de l'algorithme correspondant.

2.3 Algorithme général d'élection

Dans cette section, nous donnons un algorithme permettant de guider localement l'élection globale. L'algorithme exécuté par chaque sommet v est paramétré par une valeur $\lambda(v)$ déterminée localement par v en utilisant une fonction F. L'algorithme utilise aussi une routine $attente - exponentielle(\lambda(v))$; Cette routine génère un temps d'attente T suivant une distribution exponentielle de paramètre $\lambda(v)$, la valeur de T est donc un réel valant :

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

où x est un réel pris uniformément dans l'ensemble $[0,1]$.

Algorithme 1 : Élection aléatoire dans un arbre

1: **Var** $Neigh_v$: ensemble de sommets (* voisins de v *) ;
2: $rec_v[w]$ pour chaque $w \in Neigh_v$: booléen initialisé à faux ;
3: (*$rec_v[w]$ est vrai si v reçoit un message de w *) ;
4: $\lambda(v)$: le paramètre de la durée de vie, sa valeur initiale est la même pour tous les sommets ;
5: **Début**
6: **Tant que**$\#\{w : rec_v[w] \text{ est faux}\} > 1$ **faire**
7: receive $< tok, \lambda(v) >$ from w ;
8: $\lambda(v) := F(\lambda(v), \lambda(w))$ (*F est une fonction quelconque *) ;
9: $rec_v[w] := vrai$;
10: **Fin Tant que** ;
11: (* À présent, il y a *un seul* sommet w_0 avec $rec_v[w_0] = faux$; v est alors une feuille *) ;
12: $attente - exponentielle(\lambda(v))$;
13: **Si** un message $< tok, p >$ est arrivé **alors** $etat_v := elu$;
14: **Sinon** ;
15: **début** ;
16: $etat_v := perdant$;
17: envoyer $< tok, \lambda(v)$ à w_0 tel que $rec_v[w_0]$ est égal à faux ;
18: **fin** ;
19: **Fin** ;

Chaque sommet v a un paramètre $\lambda(v)$ et une liste $rec_v[]$ de booléens. Chaque élément $rec_v[w]$ indique si oui ou non v a reçu un message du voisin w. À la réception d'un message $< tok, \lambda(w) >$ d'un voisin w, v met à jour son paramètre $\lambda(v)$ en appliquant la formule $\lambda(v) = F(\lambda(v), \lambda(w))$. Le choix de la fonction F est l'élément déterminant dans l'orientation de l'élection et la distribution de probabilité qui en résulte.

Le processus d'élection est modélisé par un processus de mort à temps continu comme suit : Chaque sommet v qui devient une feuille génère une durée de vie selon une loi exponentielle de paramètre $\lambda(v)$, ce paramètre étant déterminé au moment où v devient feuille. Ce qui est équivalent au fait que si v devient feuille à l'instant t, alors, la probabilité que la mort de v survienne pendant l'intervalle $[t, t+h[$ est donnée par $h\lambda(v) + o(h)$ quand h tend vers 0 et ceci indépendamment de ce qui s'est passé auparavant ou ailleurs dans l'arbre. Une fois le temps de vie écoulé, la feuille disparaît et transmet son poids au sommet père dans l'arbre. Le processus continue ainsi jusqu'à ce que l'arbre soit réduit à un seul sommet. Ce modèle général laisse libre le choix de la valeur initiale de $\lambda(v)$ et de la fonction F. La seule contrainte (imposée par le fait que l'on désire avoir un algorithme distribué) est que $\lambda(v)$ soit déterminée de manière locale quand v devient une feuille. L'idée intuitive de l'algorithme est de contrôler le processus d'élection en contrôlant la valeur de $\lambda(v)$. Une remarque simple illustre ce propos : si $\lambda(v)$ augmente, alors la probabilité pour v d'être élu diminue.

Remarque. *Si le paramètre $\lambda(v)$ est (et demeure) le même pour tous les sommets de l'arbre, alors, à chaque étape, toutes les feuilles de l'arbre ont exactement la même probabilité d'être supprimées, ce qui donne une distribution particulière déjà étudiée dans (MS94).*

Dans la suite, nous verrons comment "bien" choisir la valeur initiale de $\lambda(v)$ ainsi que la fonction F pour obtenir une élection équitable dans l'arbre : tous les sommets de l'arbre ont la même probabilité d'être élus.

2.4 Propriétés simples du modèle général

Quand un sommet v devient feuille, sa mort se produit selon un processus markovien de paramètre $\lambda(v)$, paramètre déterminé au moment où v devient feuille. Chaque feuille v se comporte *indépendamment* des autres feuilles de l'arbre et si v est vivante à l'instant t, alors la probabilité que sa mort survienne dans l'intervalle $[t, t+h[$ est donnée par $\lambda(v)h + o(h)$ quand h tends vers 0. Dans la suite de ce chapitre, on nomme $\lambda(v)$ le *taux de mort* de v. L'espérance de la durée de vie de v est égale à $1/\lambda(v)$ (à partir du moment où v devient feuille). À noter que $\lambda(v)$ n'est pas fixé à l'avance mais calculé au moment où v devient feuille en fonction des valeurs transmises à v. En particulier, si v a plusieurs voisins alors, au moment où v devient feuille, il a un seul voisin encore vivant dans l'arbre.

Nous commençons par un rappel de quelques propriétés simples des processus markoviens (voir (Fel60), Chapitre XVII).

Proposition 2.2 *Soit v un sommet de l'arbre T qui devient une feuille à l'instant t. Un taux de mort $\lambda(v)$ est alors affecté à v à cet instant. Soit $L(v)$ la v.a. correspondant à la durée de vie de v (à partir de l'instant t). Alors*

1. *$L(v)$ est une v.a. ayant une distribution exponentielle de paramètre $\lambda(v)$:*

$$\mathbb{P}r\big(L(v) > x\big) = e^{-\lambda(v)x}, \quad \forall x \geq 0.$$

2. *L'espérance de $L(v)$ est égale à $\frac{1}{\lambda(v)}$.*
3. *$L(v)$ vérifie la propriété "sans mémoire"*

$$\mathbb{P}r\big(L(v) > x + y \mid L(v) > x\big) = \mathbb{P}r\big(L(v) > y\big) = e^{-\lambda(v)y}, \ \forall x, y \geq 0.$$

A un instant t, l'arbre T de taille n est réduit à un sous-arbre T' de taille n', $2 \leq n' \leq n$, avec $k \geq 2$ feuilles $v_1, v_2, \cdots, v_k$, chacune ayant respectivement un taux de mort $\lambda_1, \lambda_2, \cdots, \lambda_k$. Soit $L(v_i)$ la v.a. correspondant à la durée de vie de v_i, $1 \leq i \leq k$. Le taux de mort de l'arbre T' est $\lambda(T') = \sum_{i=1}^{k} \lambda(v_i)$, autrement dit, la probabilité que la mort de T' survienne durant l'intervalle $[t, t+h[$ est $h\sum_{i=1}^{k} \lambda(v_i) + o(h)$ quand h tends vers 0. La durée de vie $L(T')$ du sous-arbre T' (à partir de l'instant où T est réduit à T' jusqu'au moment où une des feuilles de T' disparaît) est une v.a. définie par $L(T') = \min_{1 \leq i \leq k} L(v_i)$ vue que la prochaine disparition de feuille survient à l'instant $t + \min_{1 \leq i \leq k} L(v_i)$. Il est aussi possible de caractériser sa distribution :

$$\mathbb{P}r\left(L(T') \leq x\right) = 1 - \mathbb{P}r\left(L(T') > x\right) = 1 - e^{-x\lambda(T')}, \forall x \geq 0.$$

La v.a. $L(T')$ a donc une distribution exponentielle de paramètre $\lambda(T')$ et son espérance est égale à $\frac{1}{\lambda(T')}$.

Pour caractériser la probabilité $p_T(v)$ pour un sommet v d'être élu dans l'arbre T, nous pouvons faire la somme des probabilités des transitions commençant avec l'arbre T et se terminant avec le sommet v. Nous allons d'abord caractériser les probabilités de transition d'un sous-arbre à un autre sous-arbre. Soit T' le sous-arbre à l'instant t, le prochain état du processus d'élection est un des sous-arbre $T_i = T' \setminus \{v_i\}$ obtenu en supprimant la feuille v_i et l'arête incidente, pour $i = 1, 2, \cdots, k$. Chaque transition a une probabilité donnée par :

Proposition 2.3 *Soit T' l'état du processus à un instant t. Soit $v_1, v_2, \cdots, v_k$, $k \geq 2$, les feuilles de T'. Alors, l'état suivant du processus est un des sous-arbres T_i obtenu à partir de T' en supprimant la feuille v_i (et l'arête incidente) avec la probabilité de transition :*

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

Remarque 2.1 *Si $\lambda(v) = \lambda$ pour tout v, alors la probabilité d'élection est la distribution q définie et étudiée dans (MS94), étant donné que toutes les transitions depuis un sous-arbre T' ont la même probabilité.*

Soit T un arbre de taille $n \geq 2$. Soit $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ une séquence de sous-arbres de T avec pour tout $2 \leq i \leq n$, σ_i est obtenu par une transition depuis σ_{i-1} (en supprimant une feuille). Il est alors facile de voir que la probabilité d'obtenir la séquence σ est donnée par

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

Si $\Omega(T)$ est l'ensemble de toutes les séquences possibles, on obtient la probabilité pour un sommet v d'être élu :

$$p_T(v) = \sum_{\sigma \in \Omega(T), \sigma_n = \{v\}} p(\sigma).$$

Nous pouvons aussi étudier un paramètre très intéressant : la durée de l'élection. Nous verrons par la suite comment caractériser l'espérance de la v.a. correspondante.

Pour une séquence $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle \in \Omega(T)$ donnée, sa durée $D(\sigma)$ n'est autre que la somme des durées de vie des feuilles disparaissant dans σ. L'espérance du temps d'élection est alors $D(T) = \sum_{\sigma \in \Omega(T)} p(\sigma)\mathcal{E}(D(\sigma))$, où $\mathcal{E}(D(\sigma))$ est l'espérance de $D(\sigma)$. Dans la section 2.6 , nous étudions cette espérance.

Le processus d'élection dans T est en fait un processus markovien à temps continu. L'ensemble de ses états est l'ensemble des sous-arbres de T, l'état initial est T et les transitions sont définies par les processus de suppression de feuilles décrits ci-dessus. Nous avons alors la proposition suivante :

Proposition 2.4 *Soit T' un sous-arbre et $P_{T'}(t)$ la probabilité que l'état du processus d'effeuillage à l'instant t soit T'. Alors*

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

(ii) *pour tout sous-arbre $T' \neq T$ de taille ≥ 2,* $\dfrac{dP_{T'}(t)}{dt} = -\lambda(T')P_{T'}(t) + \sum_v \lambda(v)P_{T' \cup \{v\}}(t)$, *où la somme est étendue à tous les sommets v adjacents à T' et qui n'appartient pas à T', et*

(iii) *pour tout sommet v,* $\dfrac{dP_{\{v\}}(t)}{dt} = \sum_{v' \text{ adjacent à } v} \lambda(v')P_{\{v,v'\}}(t)$,

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

Cette proposition est mathématiquement intéressante car elle permet de caractériser à la fois la probabilité pour un sommet d'être élu et la durée de l'élection en résolvant des équations différentielles. Cependant, il n'y a pas de solution "simple" à ces équations sauf pour des cas particuliers et en utilisant d'autres techniques. Ce que nous verrons dans les sections suivantes.

2.5 Élection uniforme

Dans cette section, nous nous intéressons à un cas particulier du modèle général. Soit $T = (V, E)$ un arbre. Initialement, tous les sommets ont le même *poids* $w(v) = 1, \forall v \in V$. Quand une feuille disparaît, son père récupère son poids et le rajoute à son propre poids, la fonction F est une simple somme et la ligne 9 : de l'algorithme 1 devient alors :

$$9 : \lambda(v) := \lambda(v) + \lambda(w);$$

Ainsi, quand un sommet v devient feuille, $\lambda(v)$ est le nombre de sommets qui ont disparu de son côté. Nous verrons dans cette section que cette stratégie permet d'obtenir une élection équitable dans l'arbre T.

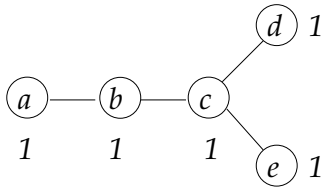


Figure 1

Exemple. *Soit T l'arbre de la Figure 1. Initialement, tous les sommets ont le même poids $w(v) = 1$. Il existe deux séquences aboutissant à l'élection du sommet a, ces deux séquences peuvent être identifiées par les séquences de suppressions de feuilles correspondantes : $\langle e, d, c, b\rangle$ et $\langle d, e, c, b\rangle$. Pour la première séquence, initialement, il y a 3 feuilles chacune ayant un poids 1 donc, la probabilité que e soit supprimé en premier est $\frac{1}{3}$ (proposition 2.3). Dans le sous arbre obtenu, il y a maintenant deux feuilles chacune ayant un poids 1 et donc la probabilité que d soit supprimé est $\frac{1}{2}$. À présent, dans le sous-arbre formé des trois sommets a, b, et c, la feuille a a un poids 1 alors que c a un poids 3. Sachant que la valeur du paramètre λ d'une feuille est son poids, (proposition 2.3), la probabilité de supprimer d'abord c est 3 fois plus grande que la probabilité de supprimer a, donc la probabilité de supprimer c à cet instant est égale à $\frac{3}{4}$. Le même raisonnement s'applique au dernier sous-arbre restant, formé des deux feuilles a et b, et la probabilité de supprimer b en premier est égale à $\frac{4}{5}$. Au final, la probabilité de la séquence $\langle e, d, c, b\rangle$ est égale à $\frac{1}{3} \times \frac{1}{2} \times \frac{3}{4} \times \frac{4}{5} = \frac{1}{10}$. On applique les mêmes calculs et on obtient la probabilité de la deuxième séquence : $\frac{1}{10}$. Ainsi, la probabilité que a soit élu est égale à $\frac{1}{5}$.*

Un calcul similaire montre que la probabilité pour chaque sommet de l'arbre d'être élu est égale à $\frac{1}{5}$.

Cet exemple illustre le fait que l'algorithme donne la même chance à tous les sommets de l'arbre d'être élus. Afin de prouver le résultat de manière générale, nous allons d'abords analyser l'algorithme appliqué aux arborescences (arbres orientés) ce qui nous permettra d'établir quelques résultats préliminaires et enfin d'arriver à prouver l'intuition donnée par l'exemple.

Soit $F = (V, E)$ une forêts d'arborescences. Soit le processus d'élection suivant : dans un intervalle $[t, t + h[$, une feuille (un sommet qui n'a pas de successeur dans l'arborescence) peut disparaître avec une probabilité proportionelle à son poids (initialement, tous les sommets ont un poids égal à 1). Si une feuille qui disparaît a un père, elle lui transmet son poids, le père met alors à jour son poids selon le même principe que dans le modèle précédent. Le processus continue ainsi dans la forêt restante jusqu'à ce que toute la forêt soit supprimée.

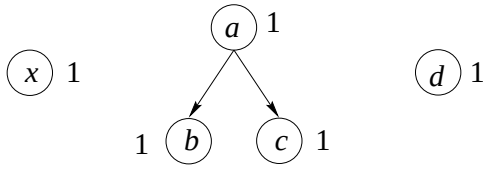


Figure 2

Exemple. *Soit la forêt de la Figure 2. Nous allons calculer la probabilité que le sommet x batte tous les autres sommets de la forêt (x est le sommet élu). Cet événement est la conséquence de l'une des huit séquences de suppression suivantes :* $\langle b, c, a, d\rangle$, $\langle b, c, d, a\rangle$, $\langle b, d, c, a\rangle$, $\langle d, b, c, a\rangle$, $\langle c, b, a, d\rangle$, $\langle c, b, d, a\rangle$, $\langle c, d, b, a\rangle$, $\langle d, c, b, a\rangle$. *Observons la première séquence. Initialement, il y a quatre feuilles chacune ayant un poids égal à 1. La probabilité que b soit supprimé en premier est égale à* $\frac{1}{4}$. *Dans la forêt résiduelle, la probabilité que c soit supprimé est égale à* $\frac{1}{3}$. *À présent, dans la forêt résiduelle, il y a trois feuilles : x de poids* 1, *d de poids* 1 *et a de poids* 3. *Donc la probabilité que a soit supprimé est égale à* $\frac{3}{5}$. *Il reste alors x et d chacun ayant un poids égal à* 1 *et donc la probabilité que x batte d est égale à* $\frac{1}{2}$. *La probabilité de la séquence* $\langle b, c, a, d\rangle$ *est donc égale à* $\frac{1}{4} \times \frac{1}{3} \times \frac{3}{5} \times \frac{1}{2} = \frac{1}{40}$. *Les probabilités des autres séquences peuvent être calculées de la même manière, et on obtient la valeur* $\frac{1}{5}$ *pour la probabilité que x bâte les autres sommets.*

Le nouveau processus de mort permet un raisonnement récursif. On peut aussi écrire les équations différentielles comme pour le modèle initial. Cependant, la résolution de ces équations reste difficile et nous allons plutôt étudier la distribution du temps d'attente avant qu'une forêt F disparaisse. Ensuite, nous prouvons que la probabilité qu'une forêt survive à une autre forêt ne dépend que des tailles des deux forêts. Nous obtenons enfin la propriété d'équité de l'algorithme.

Soit F une forêt, et $W(F)$ le temps d'attente avant que la forêt F disparaisse. $W(F)$ est une v.a. à valeurs réelles. Soit F_1 et F_2 deux forêts différentes, F_1 *survit* à (ou *bat*) F_2 si $W(F_1) > W(F_2)$.

Proposition 2.5 *Soit F une forêt de taille n. La fonction de distribution $G_F(t)$ de la v.a. $W(F)$ est donnée par :*

$$G_F(t) = \mathbb{P}r\left(W(F) \leq t\right) = (1 - e^{-t})^n, \quad \forall t \geq 0.$$

Preuve. Par récurrence sur la taille n de la forêt F. Si la forêt F est réduite à un seul sommet v, la durée de vie de v est une v.a. suivant une loi exponentielle de paramètre 1. La proposition est alors prouvée. Supposons à présent le résultat vrai pour toutes les forêts de taille inférieure à n. Soit F une forêt de taille n.

i) Si F est un ensemble de forêts $F_1, F_2, \cdots, F_k$ avec $k \geq 2$. Soit $n = n_1 + n_2 + \cdots + n_k$, avec n_i la taille de chaque F_i, $1 \leq i \leq k$. Dans ce cas, $W(F_i)$, $1 \leq i \leq k$, sont des v.a. deux à deux indépendantes. Il en résulte que :

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

ii) Sinon, supposons que la forêt F est de taille n et que F consiste en une racine r et un ensemble d'arborescences $A_1, A_2, \cdots, A_k$. Soit F' la forêt constituée des arborescences $A_1, A_2, \cdots, A_k$, elle est de taille $n - 1$ et donc, par l'hypothèse de récurrence, la fonction de distribution de $W(F')$ est $(1 - e^{-t})^{n-1}$. Or, $W(F)$ est la somme de deux v.a. $W(F')$ et la durée de vie de la racine r. D'autre part, la durée de vie de r est une v.a. exponentielle de paramètre n. Donc la fonction de distribution ((Fel66), p. 142. Theorem 2) de $W(F)$ est donnée par :

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

et ainsi,

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

Ce qui prouve le théorème. □

Un résultat direct de cette proposition :

Corollaire 2.1 **(i)** *Soit F_1 et F_2 deux forêts de tailles respectives n_1 et n_2. La probabilité que F_1 batte F_2 est égale à $\frac{n_1}{n_1+n_2}$. En particulier, la probabilité qu'un sommet isolé x batte une forêt de taille n est égale à $\frac{1}{n+1}$.*

(ii) *Soit F une forêt d'arborescences $A_1, A_2, \cdots, A_k$. Pour toute racine r_i de A_i, la probabilité d'élection de r_i est proportionelle à l'inverse de la taille de A_i $(1 \leq i \leq k)$.*

On peut alors énnoncer le théorème fondamental :

Théorème 2.1 *Soit T un arbre de taille n et v un sommet de T. Soit $q_T(v)$ la probabilité que v soit élu dans T, alors $q_T(v) = \frac{1}{n}$.*

Preuve. Si $n = 1$ ou $n = 2$ le théorème est évident. Sinon, soit $a_1, a_2, \cdots, a_k$ les sommets voisins de v et $A_1, A_2, \cdots, A_k$ les arborescences disjointes de racines respectives $a_1, a_2, \cdots, a_k$ et de tailles respectives $n_1, n_2, \cdots, n_k$. Pour tout i, on note aussi B_i l'arborescence de racine v obtenue de T en supprimant l'arborescence A_i. Le sommet v est alors battu dans T ssi une des racines a_i bat B_i. Or ces événements sont tous indépendants, donc

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

Cependant, d'après le corollaire 2.1, $q(a_i; \{A_i, B_i\})$ est égal à $\frac{n_i}{n}$ et donc, $1 - q_T(v) = \frac{n-1}{n}$. Or $\sum_{i=1}^{k} n_i = n - 1$. Le théorème est ainsi prouvé. □

2.6 Durée de l'élection dans le modèle uniforme

Une fois la distribution de la probabilité d'élection caractérisée, la durée de l'élection est une question naturelle. Il s'agit de mesurer le complexité en temps de l'algorithme présenté ci-dessus. Ce temps est en fait le temps d'absorption du processus markovien à temps continu considéré dans la section 2.4. Théoriquement, la proposition 2.4 détermine la probabilité d'être dans l'état T' (T' étant un sous-arbre de T) à l'instant t. Cependant, la résolution des équations différentielles permettant de déterminer cette probabilité est trop difficile.

Dans (MSDZ05), Nous avons conjecturé que la durée de l'élection est de $\ln n$ en moyenne et nous avons prouvé que ce résultat est vrai pour la chaîne et pour l'étoile. Dans (ERSDZ), nous avons prouvé cette conjecture. Nous allons donner ici la preuve. Dans la suite, D_n est la v.a. correspondant à la durée de l'élection dans un arbre de taille n.

Nous commençons par le résultat suivant :

Lemme 2.1 *Pour un arbre $T = (V, E)$ de taille n, toutes les $n - 1$ arêtes ont la même probabilité de survivre au processus d'éliminations successives jusqu'au dernier round. En d'autres termes, toutes les paires de sommets voisins u et v ont la même probabilité d'être finalistes.*

Nous obtenons alors le théorème suivant

Théorème 2.2 *La fonction de densité de D_n est donnée par :*

$$f(t) = n(n-1)e^{-2t}\left(1-e^{-t}\right)^{n-2}.$$

Preuve. Soit $e = \{v_1, v_2\}$ une arête de T. Soit T_i le sous arbre de T enraciné en v_i et de taille n_i ($i = 1, 2$). D'après la proposition 2.5, la probabilité qu'à l'instant t, e soit la seule arête encore vivante est égale à $\prod_{i=1}^{2}\left[(1-e^{-t})^{n_i-1} - (1-e^{-t})^{n_i}\right] = e^{-2t}(1-e^{-t})^{n-2}$. Cette probabilité, en outre, ne dépend pas de e. Ainsi, d'après le lemme précédant, la probabilité que le processus d'élection soit réduit à une seule arête est $(n-1)e^{-2t}(1-e^{-t})^{n-2}$. Donc si $F(t)$ est la probabilité d'être dans un état absorbant à l'instant t (autrement dit, $F(t)$ est la fonction de distribution de D_n), alors

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

Le théorème en résulte. □

Il est donc facile d'en déduire :

Corollaire 2.2 *L'espérance de la v.a. D_n est égale à $\sum_{i=2}^{n}\frac{1}{i} = H_n - 1$, où H_n est le $n^{ème}$ nombre harmonique.*

Preuve. Nous avons :

$$\mathbb{E}(D_t) = n(n-1)\int_0^{\infty} te^{-2t}(1-e^{-t})^{n-2}dt.$$

Un calcul simple (mais assez long), en effectuant un changement de variable $y = 1 - e^{-t}$, permet d'obtenir $\mathbb{E}(D_t) = H_n - 1$. □

Conclusion

Nous avons présenté un algorithme permettant de donner la même probabilité $\frac{1}{n}$ d'être élu à tous les sommets d'un arbre $T = (V, E)$ de taille $n \geq 1$.

L'équité de l'élection est obtenue grâce à (1) l'affectation d'un *même* poids initial à tous les sommets du graphe, (2) au mode de transmission et de prise en compte du poids des sommets qui disparaissent et enfin (3) aux propriétés de la loi exponentielle. Un sommet feuille v génère une durée de vie selon une loi exponentielle de paramètre $\lambda(v)$, (paramètre calculé, de manière locale, au moment où v devient feuille), quand sa durée de vie expire, v envoie son poids à son sommet père dans l'arbre. Un sommet qui reçoit le poids d'un sommet fils le rajoute à son propre poids.

Dans le chapitre suivant, nous présentons une extension de l'algorithme à une classe particulière de graphes : les graphes polyominoïdes.

Une deuxième piste de recherche consiste à ne pas affecter le même poids initial à tous les sommets du graphe mais un poids quelconque. Ce qui permettrait de guider la distribution de la loi de probabilité d'élection pour tout sommet v. Un travail a été fait dans ce sens et un article a été soumis (et est accepté) en collaboration avec J-.F. Marckert et N. Saheb (MSDZ09).

Élection dans les polyominoïdes

3

Dans ce chapitre, nous présentons une synthèse des travaux présentés dans (ESDZ05a) et (ERSDZ). Ce travail constitue une première tentative de généralisation du schéma de l'algorithme présenté dans la section 2.3 à d'autres classe de graphes.

Les points clés dans le processus correspondant à l'algorithme d'élection dans les arbres est de pouvoir d'abord décider localement quand un sommet est simplicial, c'est-à-dire, quand peut-il être retiré du graphe sans que le graphe résiduel perde ses propriétés (dans le cas des arbres, le sommet doit être une feuille), ensuite, une fois le sommet supprimé, qui de ses voisins récupère son poids.

Une première généralisation a été faite à la classe des k-arbres, (ESDZ05b), ensuite, nous avons étudié dans (ESDZ05a) et (ERSDZ) le cas des polyominoïdes. Ce chapitre présente cette dernière généralisation.

3.1 Introduction

Le schéma de l'algorithme d'élection présenté dans la section 2.3 peut être étendu à d'autres classes de graphes. En effet, nous observons que l'algorithme consiste à supprimer successivement des sommets du graphe jusqu'à ce que celui-ci soit réduit à un seul sommet, le sommet élu. Cependant, pour que l'algorithme soit appliqué à un graphe, il faut savoir répondre aux deux questions suivantes :

- à quel moment un sommet du graphe sait qu'il peut générer une durée de vie ? dans le cas de l'arbre, la réponse est simple : une fois que le sommet devient une feuille,
- quand un sommet dont la durée de vie est écoulée et que ce sommet doit disparaître, à quel sommet voisin doit-il envoyer son poids ? dans le cas des arbres c'est le sommet père de la feuille.

La généralisation et l'adaptation du schéma de l'algorithme peuvent être faites si nous savons répondre aux deux questions précédentes. Dans cette section, nous présentons une classe de graphes pour laquelle le schéma peut être adapté pour donner un algorithme d'élection équitable.

3.2 Les polyominoïdes

Un polyominoïde est un graphe dessiné sur la grille $\mathcal{Z} = \mathbb{Z} \times \mathbb{Z}$ en respectant un ensemble de contraintes. Les sommets d'un graphe dessiné dans $\mathcal{Z}$ sont définis par leurs coordonnées (x, y). Ses arêtes sont toutes de la forme $\{(x, y), (x + 1, y)\}$ ou $\{(x, y), (x, y + 1)\}$ pour $x \in \mathbb{Z}$ et $y \in \mathbb{Z}$. Ainsi, deux sommets $v = (x, y)$ et $v' = (x', y')$ adjacents vérifient $x = x'$ et $|y - y'| = 1$ ou bien $y = y'$ et $|x - x'| = 1$. Soit $\mathcal{T}$ l'ensemble de toutes les arêtes définies sur $\mathcal{Z}$, et soit $\mathbf{U}$ le graphe $\mathbf{U} = (\mathcal{Z}, \mathcal{T})$.

Une *cellule* est un sous graphe de $\mathbf{U}$, induit par un ensemble de quatre sommets $\{(x, y), (x + 1, y), (x + 1, y + 1), (x, y + 1)\}$.

Étant donné un cycle $\gamma = (x_0, y_0), (x_1, y - 1), \cdots, (x_k, y_k)$ avec $(x_0, y_0) = (x_k, y_k)$, un sommet (x, y) est *à l'intérieur* de γ si $card(\{i \mid y = y_i$ et $y \neq y_{i+1}$ et $x \leq x_i\})$ est impair (l'addition est modulo k).

Un *polyominoïde* (voir la figure 3.1) est un sous graphe $P = (V, E)$ de $\mathbf{U}$ vérifiant les conditions suivantes :

- V est fini,
- P est connexe,
- P ne contient pas de trou, i.e., pour tout cycle γ dans P, les sommets à l'intérieur de γ sont contenu dans V et si deux sommets sont à l'intérieur de γ, alors l'arête reliant les deux sommets est dans E.

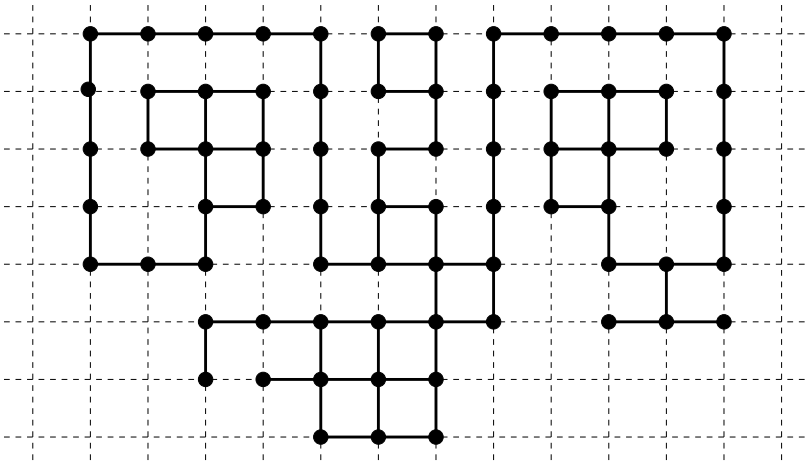


Fig. 3.1 – *Un exemple de polyominoïde.*

Un polyominoïde $Q = (V', E')$ est dit *sous polyominoïde* d'un polyominoïde $P = (V, E)$ si $V' \subseteq V$, $E' \subseteq E$ et $E' = E \cap \{\{u, v\} \mid u \in V', v \in V'\}$.

La classe des polyominoïdes peut être définie sur $\mathbf{U}$ par récurrence. Cette construction est complètement distribuée et peut être décrite par un ensemble de règles de réécritures ne nécessitant qu'une connaissance locale : celle d'une boule de rayon 2. Ainsi, l'ensemble des polyominoïdes peut être généré par une grammaire (context free). On définit inductive-

ment l'ensemble $\mathcal{P}$ des sous graphes partiels de **U** en utilisant les règles suivantes :

(a) Pour tout $(x, y) \in \mathcal{Z}$, $P = (\{(x, y)\}, \emptyset)$ appartient à $\mathcal{P}$.

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

(c) Soit $P = (V, E) \in \mathcal{P}$. Supposons que V contient 4 sommets voisins $v_1 = (x, y), v_2 = (x + 1, y), v_3 = (x + 1, y + 1), v_4 = (x, y + 1)$, situés dans une cellule de U, telles que trois des arêtes de la cellule soient dans E et la quatrième arête e n'appartienne pas à E. Alors $Q = (V, E \cup \{e\})$ appartient à $\mathcal{P}$.

Nous avons la proposition suivante :

Proposition 3.1 *Un sous graphe partiel $P = (V, E)$ de **U** est un polyominoïde ssi P appartient à $\mathcal{P}$.*

Preuve. La preuve de la proposition se fait de manière inductive grâce aux constructions données en (a), (b) et (c). □

3.3 Élection dans les polyominoïdes

Dans cette section, nous présentons l'algorithme d'élection dans les polyominoïdes. Cet algorithme est décrit par un système de réécriture de graphes (LMZ95).

Initialement, tous les sommets ont la même étiquette. Ensuite nous décrivons un système de réécriture de graphes noetherien qui, une fois appliqué sur le graphe et après un nombre fini de pas de réécritures, aboutit à un graphe irréductible où un sommet, et un seul, a une étiquette particulière. Ce sera le sommet élu.

Soit $P = (V, E)$ un polyominoïde et soit v un sommet de V. Nous introduisons un ensemble $\mathcal{L}$ d'étiquettes $\{\mathbf{N}, \mathbf{A}, \mathbf{P}, \mathbf{E}\}$ où **N** indique que le sommet est dans un état neutre, **A** un état actif, **P** un état perdant et enfin **E** l'état élu. Initialement, tous les sommets ont un poids $w = 1$ est sont étiquetés avec **N**.

L'algorithme d'élection est alors décrit par les règles suivantes :

$\mathbf{R}_0$: Si le degré de v est nul $(d(v) = 0)$, alors P est réduit au seul sommet v, ce sommet est alors élu (étiquette **E**).

$\mathbf{R}_1$: Si le degré de v est 1 $(d(v) = 1)$, alors v devient *actif* (étiquette **A**) et génère alors une durée de vie selon une loi exponentielle de paramètre le poids de v. Quand cette durée de vie est écoulée, v est supprimé (étiquette **P**), avec sa seule arête adjacente. Le sommet u voisin à v récupère alors le poids de v et rajoute $w(v)$ à son propre poids.

$\mathbf{R}_2$: Si $d(v) = 2$, alors si v est un coin gauche supérieur ou un coin gauche inférieur d'une cellule, v devient actif et quand sa durée de

vie expire, v est supprimé avec les deux arêtes adjacentes. Son voisin de droite récupère alors le poids de v. Plus formellement, soit $\{(x,y),(x+1,y),(x,y+1),(x+1,y+1)\}$ une cellule, si le degré de (x,y) est 2, alors (x,y) est $actif$ et quand sa durée de vie expire, son voisin $(x+1,y)$ récupère son poids.

$\mathbf{R}_3$: Si $d(v) = 3$, alors si v appartient à deux cellules et s'il n'a aucune arête à sa gauche, i.e., v n'a qu'une arête horizontale, alors son voisin de droite récupère son poids. Ainsi, soit $\{(x,y),(x+1,y),(x,y+1),(x+1,y+1)\}$ et $\{(x,y),(x+1,y),(x,y-1),(x-1,y-1)\}$ deux cellules. Si le degré de (x,y) est 3 alors (x,y) devient $actif$ et son voisin $(x+1,y)$ récupère son poids quand la durée de vie de (x,y) expire.

Dans le reste de ce chapitre, un sommet d'un polyominoïde P qui appartient à un cycle maximal dans P est appelé *sommet périphérique*. Nous avons alors le lemme suivant :

Lemme 3.1 *Soit v un sommet actif de degré 2 ou 3 dans un polyominoïde P. Alors v est un sommet périphérique.*

L'algorithme décrit par les règles $\mathbf{R}_0$, $\mathbf{R}_1$, $\mathbf{R}_2$ et $\mathbf{R}_3$ supprime les sommets une fois leur durée de vie expirée. Le processus continue sur le graphe résiduel jusqu'à ce que celui-ci soit réduit à un seul sommet. Il est alors fondamental de montrer que le graphe résiduel est toujours un polyominoïde :

Proposition 3.2 *Soit $P = (V,E)$ un polyominoïde de taille ≥ 2 et soit v un sommet actif dans P. Le graphe $P' = (V \setminus \{v\}, E \setminus \{\{v,u\}, u \in V\})$ est un polyominoïde.*

Preuve. Soit $P = (V,E)$ un polyominoïde, v un sommet actif dans P et $P' = (V \setminus \{v\}, E \setminus \{\{v,u\}, u \in V\})$. Il faut montrer que P' est connexe et sans trous.

–Si $d(v) = 1$, alors la suppression de v et de l'arête adjacente ne déconnecte pas le graphe et ne crée par de trou.

–Si $d(v) = 2$, soit v, v_1, v_2, et v_3 quatre sommets d'une cellule du polyominoïde P tels que v soit le sommet à supprimer. Soit $u \in V \setminus \{v, v_1, v_2, v_3\}$. Si le sommet u est accessible à un des sommets v_i par un chemin passant par v, alors u reste accessible à v_i par un autre chemin passant par $v_{j\neq i}$, $j = 1,2,3$. Par ailleurs, d'après le lemme 3.1, v est un sommet périphérique. Donc sa suppression ne crée pas de trou dans le polyominoïde.

–Si $d(v) = 3$, alors on applique le même raisonnement que dans le cas $d(v) = 2$.

□

3.4 Arbre couvrant standard

Soit $P = (V, E)$ un polyominoïde. Le graphe $T = (V, E)$ correspondant au processus de transmission de poids dans P est d'écrit comme suit :

- Si $e = \{(x, y), (x+1, y)\}$ est une arête de E alors e appartient à F, i.e. toutes les arêtes horizontales dans E appartiennent à F :

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

- Si $e = \{(x, y), (x, y+1)\}$ appartient à E et e n'est pas un côté gauche d'une cellule dans **P** alors e 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 de **P**.

Proposition 3.3 *Le graphe $T = (V, F)$ décrit ci-dessus est un arbre couvrant du polyominoïde P.*

Preuve. La preuve se fait par induction sur la construction du polyominoïde P.

–Si $P = (\{(x, y)\}, \emptyset)$, alors P est réduit à un seul sommet, donc la proposition est vraie.

–Soit $Q = (V, E)$ un polyominoïde et $T = (V, F \subseteq E)$ son arbre couvrant. Soit v et v' deux sommets voisins tels que $v \in V$ et $v' \notin V$. D'après les règles de la section 3.2, le graphe $P = (V \cup \{v'\}, E \cup \{\{v, v'\}\})$ est un polyominoïde. Il reste à prouver que $T' = (V \cup \{v'\}, F \cup \{\{v, v'\}\})$ est son arbre couvrant. Il est clair qu'aucun cycle n'est formé par l'ajout de la nouvelle arête $\{v, v'\}$. Donc le graphe résultant de l'union de l'arbre couvrant de Q et de l'arbre $(\{v, v'\}, \{\{v, v'\}\})$ est un arbre couvrant du polyominoïde P.

–Soit $Q = (V, E)$ un polyominoïde et soit $T = (V, F \subseteq E)$ son arbre couvrant. Supposons que V contienne quatre sommets voisins $v_1 = (x, y)$, $v_2 = (x+1, y)$, $v_3 = (x+1, y+1)$ et $v_4 = (x, y+1)$ tels que trois arêtes de la cellule appartiennent à E mais pas la quatrième (soit e cette arête). Il est encore facile de vérifier que si on applique les règles de la section 3.2, le graphe $P = (V, E \cup \{e\})$ est un polyominoïde. Il reste à montrer que la transmission du poids se fait via l'arbre couvrant. Dans une cellule, on note $e_N = \{(x, y+1), (x+1, y+1)\}$, $e_S = \{(x, y), (x+1, y)\}$, $e_E = \{(x+1, y), (x+1, y+1)\}$ et $e_W = \{(x, y), (x, y+1)\}$. Soit $T' = (V, F')$ le graphe représentant les arêtes par lesquelles la transmission des poids est faite dans P, nous avons :

- Si l'arête e se trouve du côté gauche de la cellule, alors l'arbre couvrant ne change pas :

$$e = e_W \Longrightarrow F' = F.$$

- 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

$$F' = F \setminus \{e_W\} \cup \{e_N\}.$$

On voit donc que le graphe T' est connexe et aucun cycle n'est formé avec ces instructions. Il en résulte que T' est un arbre couvrant de P. □

Remarque 3.1 *L'arbre couvrant construit par les règles ci-dessus est unique.*

Définition 3.1 *L'arbre $T = (V, F)$ ainsi construit est* l'arbre couvrant standard *du polyominoïde P.*

Exemple 1. La figure 3.2 donne l'arbre couvrant standard du polyominoïde de la figure 3.1.

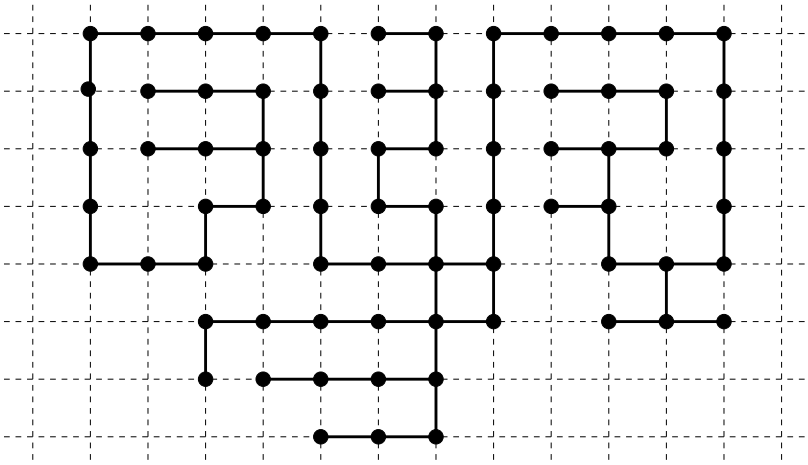


FIG. 3.2 – *L'arbre couvrant standard du polyominoïde de la figure 3.1.*

Proposition 3.4 *Soit $P = (V, E)$ un polyominoïde et $T = (V, F)$ son arbre couvrant standard. Le sommet $v \in V$ est actif dans P ssi c'est une feuille dans T.*

Preuve. La preuve de la proposition se fait également par induction sur la construction du polyominoïde. □

Nous prouvons aussi la proposition suivante :

Proposition 3.5 *Soit P un polyominoïde de taille ≥ 2. Soit v un sommet actif dans P et T l'arbre couvrant standard de P. Soit P' le polyominoïde résiduel obtenu une fois v et ses arêtes adjacentes supprimés et T' l'arbre couvrant standard de P'. Alors T' peut être obtenu à partir de T en supprimant la feuille v et son arête adjacente.*

Preuve. Soit P un polyominoïde et T son arbre couvrant standard. Il est clair que si on supprime un sommet v actif dans P, donc feuille dans T, alors l'arbre T' résiduel de T est un arbre couvrant de P. Il reste à montrer que T' est l'arbre standard de P' :

–Les arêtes horizontales de P' sont dans T'.

–Les arêtes verticales résiduelles de P', qui ne se trouvent pas du côté gauche d'une cellule de P', satisfont la même condition dans P, donc, sont dans T. Ainsi, elles sont dans l'arbre couvrant standard de P'. □

Comme résultats des discussions ci-dessus, nous obtenons l'algorithme distribué probabiliste permettant l'élection dans un polyominoïde :

Algorithme 2 : Élection probabiliste dans un polyominoïde.

1: **Tant que** P n'est pas réduit à un seul sommet **faire**
2: Tout sommet qui est actif ou qui devient actif (règles R_0-R_3) génère sa durée de vie selon son poids ;
3: Une fois la durée de vie écoulée, le sommet est supprimé avec ses arêtes adjacentes et son voisin dans l'arbre couvrant standard récupère son poids ;
4: **Fin tant que**

3.5 Analyse de l'algorithme

L'algorithme d'élection dans un polyominoïde correspond à un processus d'élection dans son arbre couvrant standard : la proposition 4 permet de voir qu'un sommet actif dans le polyominoïde est en fait un sommet feuille dans l'arbre standard.

Étant donné un polyominoïde $P = (V, E)$. Tous les sommets de V ont initialement un même poids égal à 1. D'après les règles de réécritures de la section 3, quand un sommet disparaît, son successeur récupère son poids et l'ajoute à son poids actuel. À l'instant t où v devient actif dans le graphe résiduel P' (qui est aussi un polyominoïde) son poids est égal au nombre de sommets qui ont disparu de son côté. Nous avons alors le théorème suivant :

Théorème 3.1 *La stratégie décrite par les règles R_0, R_1, R_2, et R_3 aboutit à une élection probabiliste quitable : dans un polyominoïde de taille $n \geq 1$, tous les sommets ont la même probabilité $1/n$ d'être élus.*

La preuve du théorème est une adaptation des preuves présentées dans le chapitre précédent. En effet, il suffit d'appliquer le raisonnement à une élection dans l'arbre standard défini ci-dessus.

Conclusions

Dans ce chapitre, nous avons présenté une extension de l'algorithme permettant de réaliser une élection équitable dans un arbre à la classe des polyominoïdes. Il est alors interessant d'étudier la faisabilité de l'algorithme dans des classes plus générales. En effet, la difficulté semble résider dans :

(1) la possibilité pour un sommet de décider s'il est simplicial et ceci sans connaissance globle qur le graphe, (2) la mode de transmission du poids une fois le sommet supprimé.

Une étude a été faite pour les $k-$arbres (ESDZ05b), mais d'autres classes semblent intéressantes à considérer : la classe des graphes triangulés semble être un bon candidat à étudier.

Calcul distribué d'un MIS 4

Ce chapitre présente le travail réalisé en collaboration avec Y. Métivier, J.M. Robson et N. Saheb autour du problème de l'ensemble indépendant maximal (MIS).

J'ai commencé à travailler sur ce problème durant ma thèse. J'avais étudié des procédure probabilistes permettant de réaliser des exclusions mutuelles lors de l'application des règles de réécriture codant des algorithmes distribués.

Une de ces procédures dite LE_1 permet de réaliser des élections locales dans les boules de rayon 1 (voir (MSZ02)). Ce qui permet d'implémenter des calculs locaux nécessitant une exclusion mutuelle dans des étoiles ouvertes.

Il est alors facile de voir que si on répète plusieurs fois l'exécution de cette procédure, et qu'à chaque itération, on supprime du graphe les sommets élus et leurs voisins, les sommets élus forment un ensemble indépendant maximal.

Ce chapitre présente trois algorithmes permettant d'obtenir un MIS. Le premier algorithme est à base d'échange de nombres réels. Sa complexité en moyenne est de $O(\log n)$ mais la taille des messages utilisés est (théoriquement) infinie. Le deuxième algorithme présente une simulation des échanges de réels par échange de messages de taille 1 bit, sa complexité en moyenne est de $O(\log^2 n)$. Enfin, nous étudions une version "désynchronisée" de cet algorithme permettant d'obtenir un algorithme calculant un MIS de complexité moyenne $O(\log n)$ et n'utilisant que des messages de taille 1 bit, ce qui en fait un algorithme optimal (dans un sens que nous préciserons plus loin).

4.1 Introduction

Soit $G = (V, E)$ un graphe simple et connexe. Un ensemble indépendant de G est un sous-ensemble I de V tel que pour tous sommets u et v de I, l'arête $\{u, v\}$ n'appartient pas à E. Un ensemble indépendant I est maximal (MIS) si pour tout sommet w de V, il existe $u \in I$ tel que l'arête $\{u, w\}$ appartient à E.

La construction d'un MIS dans un graphe constitue une brique de base pour la réalisation de plusieurs algorithmes distribués permettant de résoudre des problèmes tels que le routage, le contrôle de la topologie, le

coloriage, etc. Plusieurs travaux ont été menés pour étudier ce problème et plusieurs résultats sont présentés en particulier dans (Lyn96) (chapitre 4, p. 71-76) et dans (Peloo) (chapitre 8).

Dans ce chapitre, nous montrons comment un algorithme glouton peut être implémenté de manière distribuée afin de calculer un MIS dans un réseau de processeurs échangeant des messages. Nous nous intéressons à la fois à la complexité en bits et à la complexité en temps de l'algorithme.

Le calcul d'un MIS a été étudié massivement sous différentes hypothèses et différentes approches (ABI86, Lub86, AGLP89, Lin92). Karp et Widgerson (KW84) ont prouvé que calculer un MIS est NC. Kuhn et al. dans (KMNW05) ont étudié le problème pour des classes particulières de graphes et Moscibroda et al. dans (MW05) ont étudié le problème pour les réseaux radio. Le tableau suivant présente une comparaison entre notre résultat (l'algorithme $\mathcal{C}$) et les autres algorithmes connus dans la littérature.

	Connaissance préalable	Temps (en moyenne)	Taille des messages (nombre de bits)	Bit complexité (par canal)
Luby (Lynch)	Taille du graphe	$O(\log n)$	$\log n$	$O(\log^2 n)$
Luby (Peleg)	Degrée maximum dans le 2-voisinage	$O(\log^2 n)$	$\log n$	$O(\log^3 n)$
Luby (Wattenhofer)	Degré maximum des sommets voisins	$O(\log n)$	$\log n$	$O(\log^2 n)$
Alon, Babai and Itai	Degré maximum des sommets voisins	$O(\log n)$	$\log n$	$O(\log^2 n)$
Algorithme $\mathcal{C}$	Pas de connaissance	$O(\log n)$	1	$O(\log n)$

Remarque 4.1 *Dans la colonne "Connaissance préalable" :*
- *la "Taille du graphe" correspond à la taille initiale du graphe,*
- *le "Degré maximum des sommets voisins" et le "Degrée maximum dans le 2-voisinage" sont des connaissances calculées au début de chaque phase.*

4.2 Algorithme $\mathcal{A}$: un algorithme à base de réels

Le premier algorithme probabiliste distribué que nous avons étudié, noté $\mathcal{A}$, est à base d'échanges de réels. Il se décompose en plusieurs phases. À chaque phase, chaque processeur u encore dans le graphe génère un nombre réel aléatoire $x(u)$. Un processeur u est inclus dans le MIS si son x est un minimum local : $x(u) < x(v)$ pour tout voisin v de u. Nous considèrons que les nombres générés sont uniformes dans l'ensemble $[0, 1)$.

Quand tous les minimums locaux ont été inclus dans le MIS, chaque processeur survivant (non inclus dans le MIS et non voisin d'un sommet

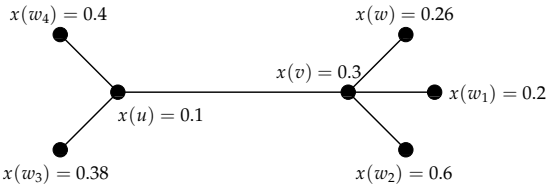


FIG. 4.1 – *Le sommet u supprime préventivement le sommet v, et l'arête* $\{v, w\}$.

dans le MIS), génère une nouvelle v.a. et encore une fois, les minimums locaux sont inclus dans le MIS et ainsi de suite. L'algorithme s'arrête quand il n'y a plus de processeur survivant.

Nous obtenons le lemme suivant :

Lemme 4.1 *À la fin de chaque phase, le nombre moyen d'arêtes supprimées du graphe résiduel G est supérieur ou égal à la moitié du nombre d'arêtes dans G.*

Preuve. Un sommet u supprime *prématurément* un voisin v si $x(u)$ est le minimum des $x(w)$ pour tout w dans l'ensemble $\{u, v\} \cup \{w \text{ voisins de } u \text{ ou } v\}$, (voir la figure 4.1).

Si tel est le cas, u sera inclus dans le MIS et v et toutes les arêtes $\{v, w\}$ seront supprimés du graphe. Les arêtes $\{v, w\}$ sont également dites supprimées *préventivement*. Si $d(u)$ et $d(v)$ sont respectivement les degrés de u et de v, alors la probabilité que u supprime préventivement v est égale à $1/(d(u) + d(v))$. On en déduit (par la linéarité de l'espérance mathématique) que le nombre moyen d'arêtes supprimées prématurément est supérieur ou égal à $\left(\sum_{\{u,v\}\in E}\left(\frac{d(v)}{d(u)+d(v)} + \frac{d(u)}{d(v)+d(u)}\right)\right)/2$ puisque $d(v)$ arêtes sont supprimées si u supprime v et $d(u)$ sont supprimées si v supprime u et qu'une arête $\{v, w\}$ peut être supprimée prématurément deux fois. La somme fait $\left(\sum_{\{u,v\}\in E} 1\right)/2$, elle est égale à $m/2$. □

Nous en déduisons :

Corollaire 4.1 *Il existe deux constantes k_1 et K_1 telles que pour tout graphe $G = (V, E)$ à n sommets, le nombre de phases pour supprimer toutes les arêtes de G est*

1. *inférieur à $k_1 \log n$ en moyenne,*
2. *inférieur à $K_1 \log n$ avec la probabilité $1 - o(n^{-1})$.*

Preuve. Le premier point résulte du lemme 4.1 et du lemme 1.1.
Pour le deuxième point, nous utilisons le lemme 1.3. Pour cela, il suffit de prendre $a = 1/4$ et $b = 1/2$. Rappelons qu'une phase est alors dite *bonne* (resp. *mauvaise*) si elle supprime au moins un quart (resp. moins de un quart) des arêtes ; rappelons également que pour chaque phase i, X_i est le nombre d'arêtes supprimées du graphe et r_i le rapport entre X_i et le nombre Y_i d'arêtes dans le graphe résiduel. Alors :

$$\mathbb{P}r\left(i^{\text{ème}} \text{ phase est bonne}\right) \geq \frac{1}{3}.$$

En effet, si $\mathbb{P}r\left(i^{\text{ème}} \text{ phase est bonne}\right) < \frac{1}{3}$ alors

$$\mathbb{E}\left(r_i\right) < \frac{2}{3} \times \frac{1}{4} \times \frac{1}{3} \times 1 = \frac{1}{2},$$

ce qui contredit le lemme 4.1. Une application directe du lemme 1.3 permet alors de prouver le lemme.

□

Enfin :

Théorème 4.1 *L'algorithme $\mathcal{A}$ calcule un MIS pour tout graphe de taille n en $O(\log n)$ unités de temps avec la probabilité $1 - o(n^{-1})$.*

4.3 Algorithme $\mathcal{B}$: un algorithme simulant l'envoi de réels

Dans cette section, nous présentons et analysons un algorithme basé sur l'envoie de messages de taille 1 bit. L'algorithme simule le tirage et l'envoie de réels par le tirage et l'envoie de bits. Le principe est le suivant : une phase est découpée en rounds. Pendant un round, un sommet v génère (uniformément) un bit 0 ou 1 et l'envoie à tous ses voisins encore actifs. À la réception des messages envoyés par ses voisins, et en fonction de leurs contenus, il décide soit d'être dans le MIS, soit de se déclarer en dehors du MIS, soit de continuer la phase actuelle, soit encore d'attendre la phase suivante.

4.3.1 Algorithme

Nous considérons un algorithme simulant l'envoie de réels par l'envoie de bits (l'algorithme $\mathcal{B}$). Dans cet algorithme, les processeurs échangent des messages de taille finie ; en effet, les seuls messages échangés sont les suivants :

- un bit 0 ou 1 de *DONNÉES*
- $In - MIS$
- $Not - In - MIS$
- *Ineligible* : quand un sommet v envoie ce message, il indique que jusqu'à la fin de la phase courante, v ne sera pas dans le MIS.

Au début d'une phase, un processeur connaît ses voisins encore dans le graphe résiduel et construit un ensemble de sommets $actifs$ initialisé avec ces sommets. Le statut d'un sommet peut être soit *Eligible* : le sommet peut encore être inclus dans le MIS pendant la phase courante, ou *Ineligible* : le sommet ne peut plus être inclus dans le MIS pendant la phase courante. Tous les processeurs encore dans le graphe ont initialement un statut *Eligible*.

Une phase d'échange de réels est remplacée par une phase composée d'un ensemble de rounds (Algorithme 3). Chaque round est composé

d'un envoi, d'une réception et d'un ensemble d'actions internes. Un message est de l'un des quatre types cités ci-dessus. Les messages de type *DONNÉES* contiennent les bits qui permettent de calculer le nombre réel généré par le sommet. Les autres messages permettent au processeur d'arrêter d'envoyer les messages de type *DONNÉES* dès que le nombre de bits échangés est suffisant pour calculer les minimums locaux.

Dans chaque round, chaque processeur u génère un bit aléatoire $b(u)$ et l'envoie à chaque sommet actif v. Ensuite, il effectue les opérations suivantes :

- Si $b(u) = 0$ et u reçoit 1 de chaque voisin actif, u est un minimum local. Il met $\eta(u)$ à 1 et envoie *In-MIS* à tous ses voisins. Le processus u ne participe plus à aucune des phases suivantes.
- Si $b(u)$ est différent du bit reçu de v, il supprime v de la liste de ses voisins actifs pendant la phase actuelle.
- Si $b(v) = 1$ et u reçoit 0 d'un voisin, u sait qu'il n'est pas un minimum local. Son état, pour cette phase, devient *Ineligible*. Le processus u envoie *Ineligible* à tous ses voisins actifs et supprime de sa liste de voisins actifs tous les voisins inéligibles.
- Si u reçoit *In-MIS* d'un voisin, il met $\eta(u)$ à 0 et envoie *Not-In-MIS* à tous les autres voisins. Le processus u ne prend plus part au phases suivantes mais continue à générer et à envoyer des bits tant qu'il a encore des voisins actifs.
- Si u reçoit *Not-In-MIS* d'un voisin v, il note que v est non eligible et sera supprimé du graphe après la phase actuelle. Si u est lui même ineligible, il supprime v de sa liste de voisins actifs.
- Si u reçoit *Ineligible* d'un voisin v, il note que v n'est pas eligible pendant cette phase. Si u est lui même ineligible, il supprime v de sa liste de voisins actifs.
- Si u est ineligible et n'a pas de voisins actifs eligibles, c'est la fin de la phase actuelle. Il arrête de participer à la phase actuelle et est prêt pour démarrer une nouvelle phase si $\eta(u) = -1$.

Initiallement, pour tout sommet v du graphe : $\eta(v) = -1$; *active-set*(v), *Not-In-MIS-set*(v) et *In-MIS-set*(v) sont des ensembles de sommets. Au début de chaque phase, pour chaque sommet v tel que $\eta(v) = -1$: *status*(v) = *Eligible*, *active-set*(v) contiennent l'ensemble des voisins w de v satisfaisant $\eta(w) = -1$, *Not-In-MIS-set*$(v) = \emptyset$, *In-MIS-set*$(v) = \emptyset$ et *Ineligible-set*$(v) = \emptyset$.

Pour plus d'explications sur le fonctionnement et le déroulement de l'algorithme, le lecteur peut consulter (MRSDZ09).

4.3.2 Analyse de l'algorithme

On commence par le lemme suivant :

Lemme 4.2 *À la fin de chaque phase, chaque processeur a une probabilité supérieure à* $1/4$

Algorithme 3 : Une phase de l'algorithme $\mathcal{B}$.

Tant que ($\eta(v) \neq 1$ et $\eta(v) \neq 0$ et $active - set(v)$ est non vide) **faire**
  Tirer uniformément un bit $b(v)$;
  Envoyer $b(v)$ à tous les voisins actifs ;
  Recevoir $b(w)$ de tout voisin actif w ;
  Si $b(v) = 0$ **alors** ;
    Si tout voisin actif w a tiré $b(w) = 1$ **alors** ;
      $\eta(v) := 1$;
      Envoyer $In - MIS$ à chaque voisin ;
    Fin Si ;
  Sinon ;
    Si il existe au moins un voisin actif w qui a tiré $b(w) = 0$ **alors** ;
      $status(v) := Ineligible$;
      Envoyer $Ineligible$ à tous les voisins actifs ;
    Fin Si ;
  Fin Si ;
  Recevoir un message $mess(w)$ de chaque voisin actif w ;
  Pour chaque voisin actif w tel que $mess(w) = Ineligible$ faire $Ineligible - Set(v) := Ineligible - Set(v) \cup \{w\}$;
  Pour chaque voisin actif w tel que $mess(w) = In - MIS$ faire $In - MIS - Set(v) := In - MIS - Set(v) \cup \{w\}$;
  Si il existe un voisin actif w tel que $mess(w) = In - MIS$ **alors** ;
    $\eta(v) := 0$;
    Envoyer $Not - In - MIS$ à chaque voisin ;
  Fin Si ;
  Recevoir un message $mess(w)$ de chaque voisin actif w ;
  Pour chaque voisin actif w tel que $mess(w) = Not - In - MIS$ faire $Not - In - MIS - Set(v) := Not - In - MIS - Set(v) \cup \{w\}$;
  Si $status(v) = Ineligible$ **alors** ;
    Pour chaque voisin w tel que $w \notin Not - In - MIS - Set(v) \cup In - MIS - Set(v) \cup Ineligible - Set(v)$ faire $active - set(v) := active - set(v) \setminus \{w\}$
  Fin Si ;
Fin Tant que
Supprimer des voisins de v tous les sommets dans $In - MIS - Set(v)$ ou dans $Not - In - MIS - Set()$;

d'être dans l'un des états : $In - MIS$, $Not - In - MIS$ ou $End - of - phase$ (le processeur a terminé la phase actuelle).

Preuve. La preuve du lemme est assez technique. Le lecteur peut consulter (MRSDZ09) pour la preuve complète. □

On obtient alors le corollaire suivant :

Corollaire 4.2 *Il existe deux constantes k_2 et K_2 telles que le nombre maximum de bits DONNES générés par tout processeur u dans toutes les phases est*

1. *inférieur à $k_2 \log n$ en moyenne,*
2. *inférieur à $K_2 \log n$ avec la probabilité $1 - o(n^{-2})$.*

Remarque 4.2 *Le corollaire ci-dessus donne une majoration de la complexité en moyenne de l'algorithme pour chaque sommet u du graphe.*

On en déduit le théorème suivant :

Théorème 4.2 *L'algorithme $\mathcal{B}$ calcule un MIS pour tout graphe de taille n en $O(\log^2 n)$ avec la probabilité $1 - o(n^{-1})$.*

Preuve. Application directe du lemme 1.2. □

4.4 Algorithme $\mathcal{C}$: un algorithme optimal en nombre de bits

Cette section présente un algorithme améliorant la borne $O(\log^2 n)$ de l'algorithme $\mathcal{B}$. Le nouvel algorithme utilise des messages de taille 1 bit et a une complexité (mesurée en nombre de bits) égale en moyenne à $O(\log n)$ et à $O(\log n)$ avec la probabilité $1 - o(n^{-1})$.

L'idée principale de cet algorithme (l'algorithme $\mathcal{C}$) est de *désynchroniser* les différentes phases sur un sommet. En effet, dans l'algorithme $\mathcal{B}$, un sommet v génère un bit et l'envoie à tous les sommets voisins avec lesquels la symétrie n'est pas encore brisée. Une phase de l'algorithme $\mathcal{B}$ se termine lorsque la symétrie est brisée avec *tous* les voisins de v. Dans l'algorithme $\mathcal{C}$, on opère par anticipation : si v brise la symétrie avec un voisin v_1 mais pas avec le voisin v_2, alors v considère la phase actuelle terminée avec v_1 mais pas avec v_2. Il commence alors immédiatement la phase suivante avec v_1 mais continue la phase actuelle avec v_2.

Ainsi, dans un même round, le sommet v peut envoyer le bit b_1, correspondant à la phase t_1, au sommet v_1 et le bit b_2, correspondant à la phase t_2, au sommet v_2, avec $t_1 \neq t_2$.

Remarque 4.3 *Pour l'analyse de l'algorithme, l'observation fondamentale est qu'à chaque round, la symétrie est brisée sur une arête avec la probabilité $1/2$.*

Description générale de l'algorithme

Chaque sommet v exécute en alternance un round du processus **calc_win** et un round du processus **calc_mis**. Le processus **calc_win** calcule pour chaque paire de sommets voisins et pour chaque phase, lequel des deux sommets a la plus petite valeur dans la phase. Le processus **calc_mis** utilise le résultat de **calc_win** pour décider si le sommet doit être supprimé du graphe et si c'est le cas, en informe les sommets voisins.

Algorithme 4 : Algorithme $\mathcal{C}$.

1: **Tant que** ($\eta(v) \neq 1$ et $\eta(v) \neq 0$ et $active - set(v)$ est non vide) **faire**
2: un round de **calc_win** ;
3: un round de **calc_mis** ;
4: **Fin Tant que**

Variables de l'algorithme $\mathcal{C}$

Chaque sommet u utilise les variables suivantes :

- pour tout voisin v de u, $phase_u(v)$ est un entier positif correspondant au numéro de la phase actuelle entre u et v ; initialement, $phase_u(v)$ est égal à 1, par symétrie, nous avons $phase_u(v) = phase_v(u)$;
- pour tout voisin v de u, $bit_u(v)$ est un entier positif correspondant au numéro du bit qui sera envoyé par u à v lors de la phase courante ; initialement, $bit_u(v)$ est égal à 1 ;
- X_u est un tableau de dimension 2 tel que pour tout voisin v de u, $X_u[phase_u(v), j]$ est un bit ;
- pour chaque voisin v de u, et pour chaque numéro de phase t, $win_u(v)(t)$ est un booléen qui vaudra *vrai* si dans la phase t, la symétrie est brisée pour l'arête entre u et v et que u ait la plus petite valeur.

Soit u un sommet et v un voisin de u ; soit t le numéro d'une phase ; on note par $x_u(v,t)$ la mot défini par les bits envoyés par u à destination de v depuis le début de la phase t. La longueur de la phase t, notée $l(t)$, est égale à $Max\{\mid x(v,t) \mid\mid v$ est un voisin de $u\}$, où $\mid x(v,t) \mid$ est la longueur du mot $x(v,t)$. Initialement, $l(t) = 0$.

Soit u un sommet, la phase t est *active* s'il existe un sommet actif v voisin de u tel que $t = phase_u(v)$ et $x_u(v,t) = x_v(u,t)$.

Processus calc_win

Au début d'un round du processus **calc_win**, le processus u tire uniformément un nouveau bit $b(t)$ pour toute phase active t de u et ajoute $b(t)$ dans X, c'est-à-dire $X[t, l(t)+1] := b(t)$.

Pour chaque voisin v, u récupère $b(v) = X[phase_u(v), bit_u(v)]$, envoie $b(v)$ à v et reçoit le bit $b(u)$ de u.

Si $b(v) = b(u)$ alors il est nécessaire de considérer les bits suivants pour différentier $x_u(v, phase_u(v))$ et $x_v(u, phase_v(u))$ de la phase $phase_u(v) = phase_v(u)$, donc u exécute l'instruction $bit_u(v) := bit_u(v) + 1$. Sinon, le résultat est enregistré et on démarre la phase suivante, le sommet u exécute les instructions suivantes :

- $win_u(v)(phase_u(v)) := (b(v) = 0)$;
- $phase_u(v) := phase_u(v) + 1$;
- $bit_u(v) := 1$.

Processus calc_mis

Pour un sommet u, si u est actif pendant une phase t, il effectue le calcul en trois étapes. Initialement, u connaît les sommets v qui sont actifs pendant la phase t et attend jusqu'à ce qu'il connaisse toutes ses variables $win_u(v)(t)$. Il sait alors s'il est ou non inclus dans un MIS pendant cette phase et envoie un message d'1 bit *in* à chaque sommet v. Puis, il attend jusqu'à ce qu'il ait reçu un message *in* de chaque voisin v. Il sait alors s'il est exclu du graphe dans cette phase et envoie un message d'1 bit *out* à chaque voisin v. Dans la dernière étape, il attend jusqu'à ce qu'il reçoive un message *out* de chaque voisin v. Il sait alors quels sont les sommets voisins encore actifs au début de la phase $t+1$. Il met à jour sa variable $\eta(u)$ et son ensemble de voisins actifs. Il est alors prêt pour démarrer la phase $t+1$ s'il est encore actif.

Analyse de l'algorithme $\mathcal{C}$

On sait qu'en moyenne et avec forte probabilité, le nombre de phases T est au plus $k_1 \log n$. On en déduit qu'après $O(\log n)$ tous les sommets ont calculé leurs valeurs $win_u(v)(t)$ pour tout $t \leq T$, puisque, avec la probabilité $1/2$, chaque round est un succès sur chaque arête $\{u,v\}$. En prenant $K \log n$ rounds avec K suffisamment grand, on a une probabilité $o(n^{-3})$ qu'une arête $\{u,v\}$ n'atteigne pas T. Ainsi, la probabilité que cela se passe est $o(n^{-1})$.

Nous avons alors le résultat principal de cette section :

Théorème 4.3 *L'algorithme $\mathcal{C}$ calcule un MIS pour tout graphe de taille n en un temps $O(\log n)$ avec la probabilité $1-o(n^{-1})$, chaque message étant de taille 1 bit.*

Preuve. Soit T le nombre de phases. Le nombre de rounds (d'échanges de bits) dans le processus **calc_win** ne peut dépasser T que si le nombre de phases dans l'algorithme $\mathcal{B}$ dépasse $T/5$ ou si, pour une arête $\{u,v\}$, u et v génèrent les mêmes bits dans au moins $\lceil 4T/5 \rceil$ des premiers T rounds . Donc

$$\begin{aligned} \mathbb{P}r(\text{ plus de } T \text{ rounds }) \quad \leq \quad & \mathbb{P}r(\text{ plus de } T/5 \text{ phases }) \\ & + m\binom{T}{\lceil 4T/5 \rceil}(1/2)^{\lceil 4T/5 \rceil}, \end{aligned}$$

où m est le nombre d'arêtes. Le second terme est asymptotiquement égal à $m(3125/4096)^{T/5}$. Pour $T > 40 \log n$, $(3125/4096)^{T/5} = O(n^{-2}T^{-2})$; ce qui permet d'obtenir directement une borne supérieure du nombre de rounds avec la probabilité $1-o(n^{-1})$. Pour l'espérance mathématique,

noua avons :

$$\begin{aligned}
\mathbb{E}\left(\#rounds\right) &= \sum_{T=1}^{\infty} \mathbb{P}r\left(\#rounds \geq T\right) \quad (\# \text{ symbolise le nombre}) \\
&= \sum_{T=1}^{40\log n} \mathbb{P}r\left(\#rounds \geq T\right) + \sum_{T=40\log n+1}^{\infty} \mathbb{P}r\left(\#rounds \geq T\right) \\
&\leq 40\log n + \sum_{T=40\log n+1}^{\infty} \mathbb{P}r\left(\#rounds \geq T\right) \\
&\leq 40\log n \\
&\quad + \sum_{T=40\log n+1}^{\infty} \left(\mathbb{P}r\left(\#rounds \geq T/5\right) + m \times O(n^{-2}T^{-2})\right) \\
&\leq 40\log n + 5\mathbb{E}(T) + o(\log n) \\
&= O(\log n).
\end{aligned}$$

□

Ensuite, le calcul des sommets à ajouter dans les MIS à chaque phase peut se faire en deux rounds supplémentaires et ainsi $2k_1 \log n$ rounds sont suffisants pour que tous les sommets terminent l'algorithme. Nous avons finalement :

Corollaire 4.3 *La bit complexité par canal de l'algorithme $\mathcal{C}$ est $O(\log n)$.*

L'algorithme $\mathcal{C}$ est optimal. En effet, Kothapalli et al, dans (KSOS06), montrent que si on ne s'autorise que des messages de taille 1 bit, alors tout algorithme distribué nécessite, avec forte probabilité, au moins $\Omega(\log n)$ rounds pour colorier un anneau anonyme de taille n avec un nombre fini de couleurs. Or, Wattenhofer, dans (Wato7) page 36, montre que tout algorithme de MIS peut être transformé en un algorithme de coloriage. Si on se limite au cas des graphes cycle, cette transformation produit un algorithme de coloriage ayant la même complexité, à une constante multiplicative près, en bit et en temps que la complexité du calcul du MIS.

4.5 Indépendance asymptotique

Cette section étudie l'impact de l'insertion d'un sommet v du graphe G dans le MIS sur la probabilité d'insertion d'un sommet $u \neq v$ dans le MIS. Nous réalisons l'étude pour l'algorithme $\mathcal{C}$.

Il est clair que si un sommet v est inséré dans le MIS, la probabilité d'insérer un voisin u est égale à 0. De la même manière, cette insertion augmente la probabilité d'inclusion des sommets à distance 2. Qu'en est-il alors des sommets à une distance ≥ 3 de v ? Nous verrons que cette influence disparaît si la distance entre v et u est assez grande et que le graphe est à degré maximal borné. Nous exhibons également un exemple montrant que cette affirmation est fausse dans le cas général. Les résultats sont présentés ici sans preuve. Le lecteur peut consulter (MRSDZ09) pour avoir les différentes preuves.

Un sommet v *survit* à la $k^{\text{ème}}$ phase s'il n'est ni choisi ni supprimé jusqu'à la fin de la $k^{\text{ème}}$ phase. Nous avons le lemme suivant :

Lemme 4.3 *Pour tout sommet v de degré d, le nombre de phases auxquelles v survit est dominé par une v.a. géométrique de paramètre $1/(d+1)$.*

Proposition 4.1 *Soit u et v deux sommets à distance l dans G. Supposons que u et v soient de degrés finis fixés. Soit $\mathbb{P}r(v \mid u)$ la probabilité que v soit inclus dans le MIS conditionnée par l'inclusion de u et soit $\mathbb{P}r(v)$ la probabilité du même événement sans conditionnement. Nous avons :*

$$\mathbb{P}r(v \mid u) = \mathbb{P}r(v) + O(\delta^l), \quad \textit{quand } l \to \infty,$$

pour un δ avec $\mid \delta \mid < 1$.

La proposition précédente reste vraie si on suppose l'hypothèse (plus faible) suivante : les degrés restent négligeables comparés à la distance. Cependant, l'exemple suivant montre que la proposition n'est pas vérifiée en général.

Exemple. Considérons le graphe $G = (V, E)$ suivant avec deux sommets u et v à distance $l = 4l' + 1$ l'un de l'autre :

- $V = \{u_{i,j}, v_{i,j} \mid i = 0, \cdots, 2^{l'}, j = 1, \cdots, l^{3i}\}$ avec $u_{0,1} = u$ et $v_{0,1} = v$,
- $E = \{(u_{i,j}, u_{i,k}), (v_{i,j}, v_{i,k}), (u_{i,j}, u_{i+1,k}), (v_{i,j}, v_{i+1,k})$ et $(u_{2l',j}, v_{2l',k})$, pour tous i, j, k pour lesquels ces sommets existent.

Il est alors démontré, (MRSDZ09), que dans le graphe G ainsi construit, l'inclusion de l'un des deux sommets u et v dans le MIS influence la probabilité d'inclusion de l'autre sommet.

Dans (MRSDZ09), nous avons également étudié cette indépendance pour les algorithmes de Luby et les présentations qu'en font Lynch dans (Lyn96) et Wattenhofer dans (Wat07).

Conclusion

Nous avons présenté un algorithme permettant de construire un MIS en utilisant un échange de messages de taille 1 bit. Cet algorithme a une complexité moyenne égale à $O(\log n)$, ce qui en fait un algorithme optimal. La technique utilisée consiste à simuler le tirage et l'échange de nombres réels par des tirages et des échanges de bits, et à utiliser un mécanisme de desynchronisation de phases.

Cette technique semble généralisable à d'autres problèmes. Une première piste serait d'étudier sa faisabilité pour le problème de couverture de graphes par des étoiles fermées. En effet, dans (MSZ02), nous avons étudié un algorithme à base de tirage de réels permettant de résoudre ce problème. Néaumoins, sa bit complexité reste non bornée.

Coloriage de graphe

5

Ce chapitre présente un algorithme probabiliste simple pour réaliser une coloration d'un graphe quelconque. L'algorithme fonctionne par échange de messages de taille 1 bit, et a une complexité en moyenne en $O(\log n)$ avec probabilité $1 - o(n^{-1})$. Ce travail est réalisé en collaboration avec Y. Métivier, J.-M. Robson et N. Saheb.

Le graphe est supposé anonyme, et aucune connaissance globale sur le graphe n'est à la disposition des différents sommets. Nous supposons par contre qu'un sommet est capable de distinguer entre ses voisins en utilisant ses différents liens de communication.

Nous montrons que notre algorithme est optimal et améliore la *complexité en bits* des autres algorithmes probabilistes distribués connus permettant le coloriage des graphes. Nous étudions également d'autres paramètres de l'algorithme tels que le nombre total de bits générés dans tout le graphe et le nombre moyen de bits générés par sommets. Une étude détaillée est aussi faite pour des classes particulières de graphes tels que les cycles, les graphes complets et les graphes aléatoires.

5.1 Introduction

Le problème

Soit $G = (V, E)$ un graphe simple non orienté. Une *sommet-coloration* de G est une fonction qui affecte une couleur $c(v)$ à chaque sommet v de G telle que pour tous $\{u, v\} \in E$, $c(u) \neq c(v)$.

La suite de ce chapitre présente un algorithme probabiliste simple et efficace (en temps et en nombre de bits) permettant de réaliser une sommet-coloration de tout graphe G.

L'algorithme fonctionne en phases synchrones : au début de chaque phase, un sommet non encore colorié connaît l'ensemble de ses voisins encore actifs (sommets qui n'ont pas encore leur couleur définitive) et crée un ensemble contenant l'ensemble de ces sommets. Chaque phase est composée d'un envoi, d'une réception et d'un ensemble d'actions internes.

À chaque round, chaque sommet u génère (uniformément) un bit aléatoire et l'envoie à tous ses voisins encore actifs, il reçoit ensuite un bit de chaque voisin actif et supprime de l'ensemble de ses voisins actifs les sommets qui lui ont envoyé un bit différent de celui généré par u. Soit $couleur_u$

le mot formé des bits générés par u ($couleur_u[i]$ est le $i^{\text{ème}}$ bit généré par u). Si u n'a plus de voisins actifs, alors u a sa couleur définitive : $couleur_u$.

Kothapalli et al. ont montré dans (KOSS06) que, si on ne s'autorise que des messages de taille 1 bit par phase, alors tout algorithme distribué réalisant le coloriage a besoin d'au moins $\Omega(\log n)$ rounds avec forte probabilité pour colorier un cycle de taille n. Nous en déduisons par conséquent, que l'algorithme que nous présentons dans ce chapitre est optimal.

La couleur définitive d'un sommet u est le mot $couleur_u$ constitué des différents bits générés par u. Ce mot peut être interprété comme un entier positif. Nous montrons que ce nombre est de l'ordre de $O(d(v))$ en moyenne.

5.2 Algorithme *FS_Color*

L'algorithme opère en rounds. À la fin chaque round, les sommets qui obtiennent leurs couleurs définitives arrêtent d'exécuter l'algorithme, ils sont alors supprimés du graphe avec leurs arêtes adjacentes. Les autres sommets continuent d'exécuter l'algorithme dans le graphe résiduel.

Formellement, chaque sommet u maintient une liste $actifs_u$ de sommets voisins actifs, c'est-à-dire la liste des sommets voisins non encore coloriés et avec lesquels la symétrie n'est pas encore brisée. Initialement, $actifs_u$ est égale à $N(u)$. La couleur $couleur_u$ d'un sommet u est initialement le mot vide, à chaque round, u génère un bit b_u, le rajoute à la fin de $couleur_u$ et envoie b_u à tous les sommets dans l'ensemble $actifs_u$. Il reçoit ensuite les bits envoyés par ses voisins encore actifs et met à jour la liste $actifs_u$. Le sommet u répète cet ensemble d'actions jusqu'à ce que la symétrie soit brisée avec *tous ses voisins*, sa couleur est alors le mot $couleur_u$.

Algorithme 5 : L'algorithme *FS_Color*.

```
1:  var :
2:      couleur_v : mot Init mot-vide ;
3:      actifs_v : ⊆ N(v) Init N(v) ;
4:      b_v : ∈ {0,1}} ;
5:  Tant que actifs_v ≠ ∅ faire
6:      b_v ← flip(0,1) ;
7:      couleur_v ← b_v ⊕ couleur_v ;
8:       Pour tout u ∈ actifs_v Faire
9:          envoyer b_v à u ;
10:         recevoir b_u de u ;
11:         Si b_v ≠ b_u alors
12:             active_v ← active_v \ {u} ;
13:         Fin Si
14:      Fin Pour
15: Fin Tant que
```

Remarque 5.1 *La couleur d'un sommet u est la concaténation de tous les bits générés par u depuis le début de l'exécution de l'algorithme. Cette couleur peut donc être interprétée comme un entier.*

5.3 Analyse de l'algorithme

Espérance du temps d'exécution

Chaque sommet u, non encore colorié, génère un bit b_u, envoie b_u à tous les voisins de u encore actifs et reçoit b_v de chaque voisin actif v. Si $b_u \neq b_v$, alors u met à jour sa liste de sommets actifs en supprimant v de l'ensemble $actifs_u$. En terme de structure de graphe, cela revient à supprimer l'arête $\{u, v\}$ du graphe G. Or, la probabilité que l'événement $b_v \neq b_u$ se produise est égale à $1/2$. Nous en déduisons :

Lemme 5.1 *À la fin de chaque round, l'espérance mathématique du nombre d'arêtes supprimées du graphe résiduel G est égale à la moitié du nombre d'arêtes dans G.*

On en déduit, d'après le lemme 1.1, le corollaire suivant :

Corollaire 5.1 *Il existe deux constantes k_1 et K_1 telles que pour tout graphe G à $n \geq 1$ sommets, le nombre de rounds nécessaires pour supprimer toutes les arêtes de G est*
- *inférieur à $k_1 \log n$ en moyenne,*
- *inférieur à $K_1 \log n$ avec grande probabilité.*

Preuve. Le preuve du corollaire utilise les mêmes arguments que la preuve du corollaire 4.1. (Voir (MRSDZ) pour plus de détails). □

On en déduit alors le théorème principal :

Théorème 5.1 *L'algorithme FS_Color calcule une coloration de tout graphe de taille n en $O(\log n)$ rounds avec forte probabilité en n'utilisant que des messages de 1 bit.*

5.3.1 Le nombre total de bits générés

Dans cette section, nous nous intéressons au nombre total de bits générés dan tout le graphe. Soit v un sommet quelconque et soit L_v la v.a. qui compte le nombre de bits générés par le sommet v. Si on note par B_G le nombre total de bits générés dans tout le graphe, alors $B_G = \sum_{v \in V} L_v$. Or

$$\mathbb{E}(L_v) = \sum_{k \geq 1} \left(1 - \left(1 - \frac{1}{2^k}\right)^{d(v)}\right).$$

Ce qui permet de prouver que :

Lemme 5.2 *Le nombre total de bits générés par tous les sommets du graphe vérifie*

$$\mathbb{E}(B_G) \leq n \sum_{k \geq 0} \left(1 - \left(1 - \frac{1}{2^k}\right)^{2m/n}\right).$$

On a alors le corollaire suivant :

Corollaire 5.2
- *Pour tout graphe $G = (V, E)$ avec $\mid V \mid = n$ et $\mid E \mid = m$ tel que $m/n \to \infty$, nous avons $\mathbb{E}(B_G) = O(n \log n)$.*
- *Si G est un arbre ou un cycle, alors $\mathbb{E}(B_G) \leq \frac{8n}{3}$.*

5.3.2 Complexité locale

Dans cette section, nous nous intéressons à la complexité locale de l'algorithme *FS_Color* : l'espérance mathématique du nombre de bits générés par sommet. Soit v un sommet de degré $d(v) = d$ et soit L_d le nombre de bits générés par le sommet v et soit $l_d = \mathbb{E}(L_d)$ son espérance mathématique. Soit $I(v)$ les arêtes adjacentes à v. À chaque round, chaque arête $e \in I(v)$ est supprimée avec probabilité $1/2$, il en résulte que $l_d = O(\log d)$. Cependant, pour obtenir une valeur plus précise qu'une borne supérieure, on utilise la transformée de Mellin. En effet, nous avons :

Proposition 5.1 *Soit $G = (V, E)$ un graphe connexe et $v \in V$ avec $d(v) = d \geq 1$. Soit l_d l'espérance mathématique du nombre de bits générés par v avant d'obtenir sa couleur définitive. On a :*

$$l_d = \log_2(d) + \frac{1}{2} + \frac{\gamma}{\log 2} + Q(\log_2(d)) + O\left(d^{-2}\right),$$

où γ est la constante d'Euler-Mascheroni et Q est une série de Fourier de période 1 et dont l'amplitude ne dépasse pas 10^{-6}.

Preuve. Pour tout $d \geq 1$, nous avons la récurrence suivante :

$$l_d = 1 + \sum_{i=0}^{d} \binom{d}{i} \frac{1}{2^d} l_{d-i},$$

avec la condition initiale $l_0 = 0$.

Pour résoudre cette récurrence, nous introduisons la fonction génératrice exponentielle suivante :

$$L(z) = \sum_{d \geq 1} l_d \frac{z^d}{d!}.$$

Un calcul simple permet de montrer que :

$$L(z) = e^z - 1 + e^{z/2} L(z/2).$$

Donc, si on note $a(z) = e^z - 1$, on résout cette équation en utilisant la technique de l'itération :

$$\begin{aligned} L(z) &= a(z) + e^{z/2} L(\tfrac{z}{2}) \\ &= a(z) + e^{z/2} a(z/2) + e^{3z/4} L(\tfrac{z}{4}) \\ &= \ldots \\ &= \textstyle\sum_{k \geq 0} e^{z(1-\frac{1}{2^k})} a(\frac{z}{2^k}) \\ &= \textstyle\sum_{k \geq 0} e^{z(1-\frac{1}{2^k})} (e^{\frac{z}{2^k}} - 1). \end{aligned}$$

Après avoir développé les fonctions exponentielles, nous obtenons une forme explicite pour l_d :

$$l_d = \sum_{k\geq 0} \left(1 - \left(1 - \frac{1}{2^k}\right)^d\right).$$

Par ailleurs, on a $(1-a)^n \sim e^{-an}$, donc, si on pose

$$F(x) = \sum_{k\geq 0} \left(1 - e^{-x/2^k}\right),$$

nous obtenons facilement $l_d \sim F(d)$.

La forme exacte de $F(x)$ peut être obtenue en utilisant la transformée de Mellin (voir (FS)). La transformée de Mellin de $F(x)$ est :

$$F^*(s) = -\frac{\Gamma(s)}{1-2^s},$$

avec le contour fondamental $< -1, 0 >$. Par ailleurs, nous avons :

$$\Gamma(s) = \frac{e^{-\gamma s}}{s} \prod_{k=1}^{\infty} \left(1 + \frac{s}{k}\right)^{-1} e^{s/k}.$$

Donc $F^*(s)$ est méromorphe, admet un pôle double en $s = 0$ et des pôles imaginaires en $s = \chi_k = \frac{2ik\pi}{\log 2}$, pour tout $k \in \mathbb{Z} \setminus \{0\}$. Il en résulte le développement suivant de $F^*(s)$ dans le contour $< -1/2, 2 >$:

$$F^*(s) = \frac{1}{\log 2}\frac{1}{s^2} - \frac{\gamma + \frac{1}{2}\log 2}{s \log 2} + \frac{1}{\log 2} \sum_{k\in\mathbb{Z}\setminus\{0\}} \frac{\Gamma(\chi_k)}{s - \chi_k}.$$

D'où :

$$F(x) = \frac{\log x}{\log 2} + \frac{1}{2} + \frac{\gamma}{\log 2} + Q\left(\log_2(x)\right) + O\left(x^{-2}\right),$$

où $Q(u) = -\frac{1}{\log 2} \sum_{k\in\mathbb{Z}\setminus\{0\}} \Gamma(\chi_k)\, e^{-2ik\pi u}$. Ce qui termine la preuve. □

Nous avons alors le corollaire :

Corollaire 5.3 *Soit v un sommet tel que $d(v) = d \to \infty$. Si $c(v)$ est la couleur de v, alors $c(v) = O(d)$.*

En outre, nous obtenons la distribution de probabilité de la v.a. L_d :

Lemme 5.3 *Soit $d \geq 1$, nous avons :*

- $\mathbb{P}r\,(L_d = 0) = 0$, *et*
- $\mathbb{P}r\,(L_d = k) = \left(1 - \frac{1}{2^k}\right)^d - \left(1 - \frac{1}{2^{k-1}}\right)^d$, *si* $k \geq 1$.

Ce qui permet de calculer le premier et le deuxième moment de la v.a. L_d et d'énoncer :

Lemme 5.4 *Si on note par $\mathbb{V}ar$ la variance, alors :*

$$\begin{aligned}\mathbb{V}ar\left(L_d\right) &= \left(\frac{1}{\log 2}-1\right)\log_2(d)+\frac{1}{12}+\frac{\pi^2}{6(\log(2))^2}\\ &\quad -P\left(\log_2(d)\right)+O\left(\frac{1}{d^2}\right),\end{aligned}$$

où $P(u)=Q(u)^2+\left(2u+\frac{2\gamma}{\log(2)}-\frac{2}{\log(2)}\right)Q(u)$ et Q est la série défine dans le lemme 5.1.

Preuve. On commence par calculer le second moment de la v.a. L_d :

$$\begin{aligned}\mathbb{E}\left(L_d^2\right) &= \sum_{k\geq 0}k^2\mathbb{P}r\left(L_d=k\right)\\ &= \sum_{k\geq 1}k^2\left[\left(1-\frac{1}{2^k}\right)^d-\left(1-\frac{1}{2^{k-1}}\right)^d\right]\\ &= \sum_{k\geq 0}(2k+1)\left(1-\left(1-\frac{1}{2^k}\right)^d\right).\end{aligned}$$

En utilisant la même approximation exponentielle que pour le lemme précédent, nous avons $\mathbb{E}\left(L_d^2\right)=G(d)$ où :

$$G(x)=\sum_{k\geq 0}(2k+1)\left(1-e^{-x/2^k}\right).$$

La transformée de Mellin de G est donnée par :

$$\begin{aligned}G^*(s) &= \int_0^\infty x^{s-1}G(x)dx\\ &= \sum_{k\geq 0}(2k+1)\int_0^\infty x^{s-1}\left(1-e^{-x/2^k}\right)dx\\ &= -\Gamma(s)\sum_{k\geq 0}(2k+1)2^{ks}\\ &= -\Gamma(s)\frac{2^s+1}{(1-2^s)^2},\end{aligned}$$

avec comme contour fondamental $\langle -1,0\rangle$.

Donc, la fonction $G^*(s)$ est méromorphe en 0 et admet un pôle d'ordre 3 en $s=0$ ainsi que des pôles imaginaires en $s=\chi_k=\frac{2ik\pi}{\log 2}$, pour tout $k\in\mathbb{Z}\setminus\{0\}$. Finalement :

$$\begin{aligned}G^*(s) &= -\frac{2}{(\log 2)^2}\frac{1}{s^3}+\left(\frac{1}{\log 2}+\frac{2\gamma}{(\log 2)^2}\right)\frac{1}{s^2}\\ &\quad -\left(\frac{1}{3}+\frac{\gamma}{\log 2}+\frac{\pi^2}{6(\log 2)^2}+\frac{\gamma^2}{(\log 2)^2}\right)\frac{1}{s}+\frac{2}{(\log 2)^2}\sum_{k\in\mathbb{Z}\setminus\{0\}}\frac{\Gamma(\chi_k)}{(s-\chi_k)^2}.\end{aligned}$$

Donc, en utilisant le "reverse mapping theorem" de (FS), il vient :

$$\begin{aligned}G(x) &= (\log_2 x)^2+\left(1+\frac{2\gamma}{\log 2}\right)\log_2 x\\ &\quad +\frac{1}{3}+\frac{\gamma}{\log 2}+\frac{\pi^2}{6(\log 2)^2}+\left(\frac{\gamma}{\log 2}\right)^2+\Delta\left(\frac{\log x}{\log 2}\right)+O\left(\frac{1}{x^2}\right),\end{aligned}$$

où $\Delta(u)=-\frac{2}{(\log 2)^2}\sum_{k\in\mathbb{Z}\setminus\{0\}}\Gamma\left(\chi_k\right)e^{-2ik\pi u}$.

Nous obtenons la valeur du second moment :

$$\begin{aligned}\mathbb{E}\left(L_d^2\right) &= (\log_2 d)^2 + \left(1+\tfrac{2\gamma}{\log 2}\right)\log_2 d \\ &\quad +\tfrac{1}{3}+\tfrac{\gamma}{\log 2}+\tfrac{\pi^2}{6(\log 2)^2}+\left(\tfrac{\gamma}{\log 2}\right)^2+\Delta\left(\tfrac{\log d}{\log 2}\right)+O\left(\tfrac{1}{d^2}\right).\end{aligned}$$

D'où

$$\begin{aligned}\mathbb{V}ar\left(L_d\right) &= \mathbb{E}\left(L_d^2\right)-\mathbb{E}\left(L_d\right)^2 \\ &= \left(\tfrac{1}{\log 2}-1\right)\log_2 d+\tfrac{1}{12}+\tfrac{\pi^2}{6(\log 2)^2} \\ &\quad -P\left(\log_2 d\right)+O\left(\tfrac{1}{d^2}\right),\end{aligned}$$

où $P(u)=Q(u)^2+\left(2u+\frac{2\gamma}{\log 2}-\frac{2}{\log 2}\right)Q(u)$.

Ce qui prouve le lemme. □

Nous avons alors :

Proposition 5.2 *Le ratio entre L_d et $\log_2 d$ tend en probabilité vers 1 quand d tend vers ∞.*

Preuve. Soit la v.a. $R_d=\frac{L_d}{\log_2 d}$. Nous avons :

$$\mathbb{E}\left(R_d\right)=1+\left(\frac{1}{2}+\frac{\gamma}{\log 2}\right)/\log_2 d+\frac{1}{\log_2 d}\left(Q\left(\log_2 d\right)-O\left(\frac{1}{d^2}\right)\right),$$

et

$$\begin{aligned}\mathbb{V}ar\left(R_d\right) &= \mathbb{V}ar\left(L_d\right)/(\log_2 d)^2 \\ &= \left(\tfrac{1}{\log 2}-1\right)/\log_2 d+\left(\tfrac{1}{12}+\tfrac{\pi^2}{6(\log 2)^2}\right)/(\log_2 d)^2 \\ &\quad -\tfrac{1}{(\log d)^2}P\left(\log_2 d\right)+O\left(\tfrac{1}{d^2}\right).\end{aligned}$$

En utilisant l'inégalité de Tchebychev nous obtenons :

$$\forall\varepsilon>0,\mathbb{P}r\left(\mid R_d-1\mid>\varepsilon\right)<\frac{\mathbb{V}ar\left(R_d\right)}{\varepsilon^2}\to 0 \text{ quand } d\to\infty,$$

ce qui termine la preuve. □

On a alors un résultat plus précis que le corollaire 5.1 :

Corollaire 5.4 *Soit v un sommet et supposons que son degré d tend vers l'infini et soit $c(v)$ sa couleur. Avec grande probabilité $1-o(1/n^2)$, $c(v)=O(d)$.*

5.4 Cas particuliers

Cette section étudie les valeurs des différents paramètres introduits plus haut dans les cas particuliers suivants : les cycles, les graphes aléatoires et les graphes complets.

5.4.1 Les cycles

Soit $G = (V, E)$ un cycle de taille $n \geq 3$. Soit $v \in V$ un sommet quelconque. Un calcul simple donne :

Lemme 5.5 $\mathbb{E}(L_v) = \frac{8}{3}$.

Pour la complexité globale de l'algorithme, soit T la v.a. qui représente le temps nécessaire pour colorier tous les sommets du cycle G, c'est-à-dire, le nombre de rounds. Si n est impair, alors il faut au moins un round pour colorier tous les sommets de G. Donc $\mathbb{P}r(T > 1) = 1$. Sinon, c'est-à-dire, si n est pair, il est facile de voir que $\mathbb{P}r(T > 1) = 1 - \frac{1}{2^{n-1}}$.

Plus généralement, en utilisant le principe d'inclusion-exclusion, (Roooo), pour tout $k \geq 2$:

$$\begin{aligned}\mathbb{P}r(T > k) &= \sum_{i=1}^{n-1}(-1)^{i+1}\binom{n}{i}\left(\tfrac{1}{2^i}\right)^k + (-1)^{n+1}\left(\tfrac{1}{2^{n-1}}\right)^k \\ &= 1 - \left(1 - \tfrac{1}{2^k}\right)^n + (-1)^{n+1}\left(\left(\tfrac{1}{2^{n-1}}\right)^k - \left(\tfrac{1}{2^n}\right)^k\right).\end{aligned}$$

Donc, avec $k = 2\log_2 n$, nous obtenons :

$$\mathbb{P}r(T > k) \sim 1 - e^{-1/n} \to 0 \text{ lorsque } n \to \infty.$$

Par ailleurs, étant donné que $\mathbb{E}(T) = \sum_{k\geq 1}\mathbb{P}r(T > k)$, nous avons :

$$\begin{aligned}\mathbb{E}(T) &= \mathbb{P}r(T > 1) + \mathbb{P}r(T > 2) \\ &\quad + \sum_{k\geq 3}\left[1 - \left(1 - \tfrac{1}{2^k}\right)^n + (-1)^{n+1}\left(\left(\tfrac{1}{2^{n-1}}\right)^k - \left(\tfrac{1}{2^n}\right)^k\right)\right].\end{aligned}$$

En utilisant les mêmes arguments que dans la section 5.3.2, nous avons :

$$\mathbb{E}(T) = \log_2 n + \frac{5}{2} + \frac{\gamma}{\log 2} + Q(\log_2(n)) + O\left(n^{-2}\right).$$

Nous en déduisons alors :

Lemme 5.6 *Soit T le nombre de rounds nécessaires pour colorier tous les sommets d'un cycle de taille $n \geq 3$.*
- *L'espérance de T est asymptotiquement égale à $\log_2 n + \frac{5}{2} + \frac{\gamma}{\log 2}$,*
- *elle est inférieure à $2\log_2 n$ avec forte probabilité.*

On peut également calculer la distribution de la v.a. T :

Lemme 5.7 *Soit $G = (V, E)$ un cycle de taille $n \geq 3$ et T la v.a. définie ci-dessus. Alors*
- $\mathbb{P}r(T = 0) = 0$,
- *Si n est impaire, alors :*
  - $\mathbb{P}r(T = 1) = 0$,
  - $\mathbb{P}r(T = 2) = \frac{3^n - 3}{4^n}$.
- *Si n est pair, alors :*
  - $\mathbb{P}r(T = 1) = \frac{1}{2^{n-1}}$,

– $\mathbb{P}r\,(T=2) = \frac{3^n+3}{4^n} - \frac{1}{2^{n-1}}$.

et, pour tout $k > 2$:

$$\begin{aligned}\mathbb{P}r\,(T=k) &= \left(1-\tfrac{1}{2^k}\right)^n - \left(1-\tfrac{1}{2^{k-1}}\right)^n \\ &\quad +(-1)^{n+1}\left(\left(\tfrac{1}{2^{n-1}}\right)^{k-1} - \left(\tfrac{1}{2^n}\right)^{k-1} - \left(\tfrac{1}{2^{n-1}}\right)^k + \left(\tfrac{1}{2^n}\right)^k\right).\end{aligned}$$

Par un raisonnement similaire à celui de la preuve de la proposition 5.2, il vient :

Proposition 5.3 *Le rapport entre T et* $\log_2 n$ *tend vers* 1 *en probabilité quand* $n \to \infty$.

5.4.2 Graphes aléatoires

Un graphe aléatoire est un graphe obtenu en commençant par un ensemble de n sommets et en ajoutant, de manière aléatoire, des arêtes entre ces sommets. Différents modèles de graphes aléatoires produisent différentes distributions de probabilité sur les graphes. Le modèle le plus étudié est le modèle $G_{n,p}$: chaque arête est rajoutée (indépendamment des autres) avec probabilité p (et n'est donc pas ajoutée avec probabilité $q = 1-p$).

Soit $G_{n,p} = (V, E)$ un graphe aléatoire et soit v un sommet quelconque. Conditionné par $d(v) = k \geq 0$, on a :

$$\mathbb{E}\,(L_v \mid d(v) = k) = \sum_{i\geq 0}\left(1-\left(1-\frac{1}{2^i}\right)^k\right).$$

Or $\mathbb{P}r\,(d(v)=k) = \binom{n-1}{k}p^k q^{n-1-k}$, donc

$$\mathbb{E}\,(L_v) = \sum_{k=0}^{n-1}\left[\binom{n-1}{k}p^k q^{n-1-k}\sum_{i\geq 0}\left(1-\left(1-\frac{1}{2^i}\right)^k\right)\right].$$

Ce qui permet d'obtenir :

$$\mathbb{E}\,(L_v) = \sum_{i\geq 0}\left(1-\left(1-\frac{p}{2^i}\right)^k\right).$$

Donc, en utilisant le même raisonnement que dans la section 5.3.2 :

Proposition 5.4

$$\begin{aligned}\mathbb{E}\,(L_v) &= F\,((n-1)p) \\ &= \log_2\,((n-1)p) + \tfrac{1}{2} + \tfrac{\gamma}{\log 2} + Q\,(\log_2((n-1)p)) + O\left((np)^{-2}\right).\end{aligned}$$

Remarque 5.2 *Pour* $p = \frac{\alpha \log n}{n}$, *avec* $\alpha > 1$, *c'est-à-dire, avec forte probabilité, G est connexe, alors :*

$$\mathbb{E}\,(L_v) = \log_2\,(\log(n)) + \log_2\,\alpha + \frac{1}{2} + \frac{\gamma}{\log 2} + Q\,(\log_2\,(\alpha\log(n))) + O\left(\frac{1}{(\log n)^2}\right).$$

5.4.3 Graphes complets

Si $G = (V, E)$ est un graphe complet, alors on peut voir que $G = G_{n,1}$. Donc, pour la complexité locale, nous avons pour tout sommet $v \in V$, $\mathbb{E}(L_v) = \log_2(n-1) + \frac{1}{2} + \frac{\gamma}{\log 2} + Q(\log_2(n-1)) + O\left(n^{-2}\right)$.

Pour étudier la complexité globale, on peut observer que le cas étudié correspond au problème bien connu du calcul de l'espérance mathématique de la hauteur d'un *trie*, (CFV01).

Un trie est une structure de données utilisée pour construire des dictionnaires d'ensembles de mots produits par une source. Dans notre cas, le trie est construit comme suit : à chaque phase, chaque sommet génère un bit 0 ou 1. Les sommets ayant généré 0 sont regroupés dans un nouveau sommet du trie et ceux ayant généré 1 sont regroupés dans un autre sommet. Le processus est alors répété sur les nouveaux sommets jusqu'à ce que les sommets deviennent des singletons. Il est donc facile d'observer que l'espérance mathématique de la hauteur du trie ainsi construit est la complexité globale de notre algorithme.

Remarque 5.3 *Clément et al., dans (CFV01), ont étudié ce paramètre h_n (hauteur d'un trie de taille n) dans un contexte plus général : l'alphabet peut contenir plus que deux symboles et chaque symbole s_i est généré avec probabilité p_i. Dans notre cas particulier, $s_1 = 0$, $s_2 = 1$ et $p_1 = p_2 = 1/2$, ainsi :*

$$\mathbb{E}(h_n) \sim 2\log_2 n.$$

De plus, ils ont montré que h_n a une distribution asymptotique doublement exponentielle. Il est alors possible de déduire le premier point du corollaire 5.1 de leur résultat puisque la complexité globale de l'algorithme de coloriage dans tout graphe est majorée par le temps nécessaire au coloriage d'un graphe complet dans notre algorithme.

Conclusion

Dans ce chapitre, nous avons présenté un algorithme simple et efficace pour colorier un graphe de taille n. Les sommets du graphe échangent des messages de taille 1 bit. Nous avons montré que la complexité de l'algorithme est en moyenne $O(\log n)$ et avec forte probabilité égale à $O(\log n)$.

Notre algorithme est donc optimal en bit complexité puisqu'il n'utilise que des messages de taille 1 bit. Cependant, le nombre de couleurs utilisé par l'algorithme reste trop grand comparé à l'algorithme de Johanson (Joh99). Il serait, par conséquent intéressant d'étudier la possibilité de réduire, de manière significative, le nombre de couleurs utilisées.

Problème du rendez-vous

6

Dans un réseau de processeurs, un rendez-vous (ou poignée de main) permet de réaliser des communications exclusives entre des paires de sommets voisins. Ce qui permet d'implémenter des algorithmes distribués codés par des systèmes de réécriture impliquant des règles sur les arêtes, ou encore, de réaliser (après un nombre d'itérations) un couplage maximal du graphe représentant le réseau.

Ce chapitre présente des algorithmes probabilistes distribués permettant de réaliser des rendez-vous. Une partie des résultats obtenus sur le sujet ont été établis en collaboration avec A. El Hibaoui, Y. Métivier, J.- M. Robson, et N. Saheb.

6.1 Introduction

6.1.1 Le problème

Dans un réseau de processeurs communiquant par échange de messages en mode synchrone, l'émetteur et le récepteur doivent être tous les deux prêts pour communiquer. On dit que les deux processeurs ont un rendez-vous, ils sont alors prêt à réaliser une communication de manière exclusive.

Plusieurs algorithmes distribués codés par des règles de réécriture nécessitent cette exclusivité de communication pour être implémentés, voir (MS97b) pour quelques exemples.

Ce chapitre présente les solutions que nous avons apporté au problème du rendez-vous. Le premier algorithme fait l'hypothèse d'un temps discret et le second utilise des propriétés des variables aléatoires continues.

6.1.2 Algorithme $S\_RdV()$

Le premier algorithme étudié est noté $S\_RdV()$. Chaque noeud v choisit au hasrad et uniformément parmi ses voisins un voisin w, il lui envoie un message 1 et envoie un message 0 à tous ses autres voisins différents de w. Il y a alors rendez-vous entre deux sommets voisins u et v si, et seulement si, u envoie 1 à v et v envoie 1 à u (voir la figure 6.1). Plus formellement, la suite d'instructions est décrite par l'algorithme $S\_RdV()$.

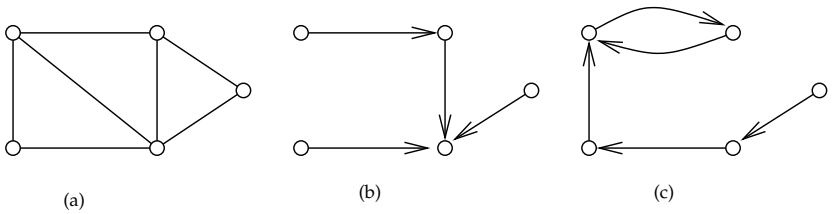


FIG. 6.1 – *(a) un graphe G, (b) une exécution de l'algorithme S_RdV() sur G ne donnant pas lieu à un rendez-vous, (c) une exécution de l'algorithme S_RdV() sur G donnant lieu à un rendez-vous*

Algorithme 6 : L'algorithme $S\_RdV()$

1: Chaque sommet v répète infiniment :
2: le sommet v choisit, aléatoirement et uniformément, un de ses voisins $c(v)$;
3: le sommet v envoie 1 à $c(v)$;
4: le sommet v envoie 0 à ses voisins différents de $c(v)$;
5: le sommet v reçoit les messages de tous ses voisins ;
(* Il y a un rendez-vous si v reçoit 1 de $c(v)$ *)

6.1.3 Analyse de l'algorithme $S\_RdV()$

Cette section présente l'analyse de l'algorithme $S\_RdV()$. Quelques résultats sont donnés ici sans preuve. Le lecteur peut se référer à (MSZ00, MSZ02, MSZ03, Zem08) pour plus de détails.

Le premier paramètre est la probabilité d'obtenir au moins un rendez-vous dans le graphe, ensuite, nous nous sommes intéressés à la probabilité d'avoir un rendez-vous sur une arête, ce qui permet par la suite de calculer le nombre moyen de rendez-vous dans le graphe et d'étudier l'efficacité de l'algorithme $S\_RdV()$ comparé à un algorithme "idéal" et donc non nécessairement probabiliste ni distribué.

6.2 Probabilité de succès

Définition 6.1 Soit $G = (V, E)$ un graphe. Un *appel* sur G est une fonction c de V dans V qui associe à chaque sommet un de ses voisins.

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

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

On peut représenter un appel sur un graphe G par le graphe orienté $G_c = (V, A)$, où A est tel que pour tout $\{v, w\} \in E$, l'arc (v, w) appartient à A si et seulement si $w = c(v)$, voir la figure 6.1, partie (c).

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

De plus, il est facile de voir que :

Lemme 6.1 *Soit c un appel sur le graphe G, c est un échec si et seulement si G_c est sans cycle de longueur 2.*

Lemme 6.2 *Si G est un arbre, alors tout appel sur G est un succès.*

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

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

$$\alpha(G) = \prod_{v \in V} \frac{1}{d(v)}. \tag{6.1}$$

Soit $s(G)$ la probabilité de succès et $f(G)$ celle d'échec. Du lemme 6.1, on déduit que :

Lemme 6.3 *Nous avons :*

$$f(G) = \alpha(G)N(G) \tag{6.2}$$

et

$$s(G) = 1 - \alpha(G)N(G), \tag{6.3}$$

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

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

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

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

$$\mathbb{P}r(e^{(1)}) = \mathbb{P}r(\{v, w\}^{(1)}) = \frac{1}{d(v)d(w)}. \tag{6.4}$$

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

$$\mathbb{P}r(M) = \mathbb{P}r(e_1^{(1)} \wedge e_2^{(1)} \wedge ... \wedge e_k^{(1)}) = \prod_{\{v,w\} \in M} \frac{1}{d(v)d(w)} = \prod_{e \in M} \mathbb{P}r(e^{(1)}). \quad (6.5)$$

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

Soit

$$q_k = \sum_{M \in \mathcal{M}_k} \mathbb{P}r(M),\ k = 0, 1, ..., \lfloor n/2 \rfloor. \quad (6.6)$$

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

Proposition 6.1 *Soit la suite $q_k, k = 0, 1, ..., \lfloor n/2 \rfloor$ comme définie ci-dessus pour un graphe connexe G de taille n. Alors, pour tout entier l entre 1 et $\lfloor n/2 \rfloor$, la probabilité d'obtenir au moins l rendez-vous sur G est :*

$$P_l = \sum_{0 \leq i \leq \lfloor n/2 \rfloor - l} (-1)^i q_{l+i}. \quad (6.7)$$

En particulier, la probabilité de succès est :

$$s(G) = P_1 = \sum_{0 \leq i \leq \lfloor n/2 \rfloor - 1} (-1)^i q_{i+1}. \quad (6.8)$$

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

Exemple 6.1 Soit G un cycle de taille $n \geq 2$. Le nombre $N(G)$, introduit dans le lemme 6.3, est égal à 2. Ainsi :

$$f(G) = \frac{1}{2^{n-1}}, \quad (6.9)$$

et

$$s(G) = 1 - \frac{1}{2^{n-1}}. \quad (6.10)$$

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

$$(1-\frac{1}{2^{n-1}})+2\frac{1}{2^{n-1}}(1-\frac{1}{2^{n-1}})+3(\frac{1}{2^{n-1}})^2(1-\frac{1}{2^{n-1}})+... \tag{6.11}$$

c'est-à-dire :

$$\frac{2^{n-1}}{2^{n-1}-1}. \tag{6.12}$$

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

- Si on ajoute une arête à un arbre, la probabilité d'avoir au moins un rendez-vous ne peut que diminuer.
- Le graphe G de la figure 6.2 dû à H. Austinat et V. Diekert (AD), montre que l'ajout d'une arête peut augmenter cette probabilité. En effet, pour ce graphe, nous avons $s(G) = 1156/1600 = 0.7225$. Soit G' le graphe obtenu à partir de G en ajoutant l'arête $\{1,2\}$. Nous avons $s(G') = 4742/6400 = 0.7409....$

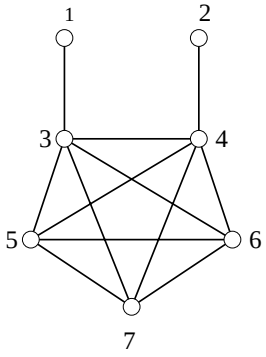


FIG. 6.2 – *Un exemple où l'ajout d'une arête augmente la probabilité d'obtenir au moins un rendez-vous.*

Des résultats plus précis ont été obtenus pour des classes particulières de graphes. En effet, nous avons :

Proposition 6.2 *Soit $G = (V, E)$ un graphe à degré d-borné, et $s(G)$ la probabilité de succès. Alors*

$$s(G) \geq 1 - \left(1 - \frac{1}{d^2}\right)^{|E|}. \tag{6.13}$$

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

$$f(G) \leq e^{\frac{-|E|}{d^2}}. \tag{6.14}$$

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

Corollaire 6.1 *Soit G un graphe $d-$régulier, la probabilité d'échec $f(G)$ vérifie :*

$$f(G) \leq e^{\frac{-n}{2d}}. \tag{6.15}$$

Dans le cas où, G est un graphe complet, nous avons :

Proposition 6.3 *Soit K_n le graphe complet de taille n, alors :*
- $s(K_n) = \sum_{k\geq 1}(-1)^{k+1}\frac{n!}{k!2^k(n-2k)!}\frac{1}{(n-1)^{2k}}$,
- $s(K_n)$ *est asymptotiquement* $1 - e^{-1/2}$,
- *et le nombre moyen d'appels nécessaires pour obtenir un succès vaut asymptotiquement* $\frac{\sqrt{e}}{\sqrt{e}-1}$.

La proposition 6.2 donne une borne inférieure pour la probabilité de succès si le graphe est à degré borné par d. Le corollaire 6.1 montre combien cette borne est importante si d est suffisamment petit par rapport à $n = |V|$. Cependant, cette borne devient trop large si d est trop grand et si $|E|$ n'est pas suffisamment grand. Il est donc intéressant de trouver une borne uniforme ne dépendant ni de d, ni de $|E|$. Cette section a pour objectif de donner un tel minorant. En effet, nous avons le théorème suivant :

Théorème 6.1 *La probabilité $s(G)$ de succès dans un appel sur tout graphe $G = (V, E)$ est minorée par $1 - e^{-\overline{M}(G)}$, où $\overline{M}(G)$ est le nombre moyen de rendez-vous dans G.*

En utilisant le théorème 6.1 et la proposition 6.5, il vient :

Corollaire 6.2 *La probabilité $s(G)$ d'un succès sur tout graphe $G = (V, E)$ est minorée par $1 - e^{-1/2}$.*

Et par conséquent :

Corollaire 6.3 *Le nombre moyen d'appels nécessaires pour obtenir un succès est majoré par $\frac{\sqrt{e}}{\sqrt{e}-1}$.*

Remarque 6.1 *Dans (Die02) et (DT05), les auteurs ont prouvé que le graphe complet K_n minimise la probabilité de succès pour les graphes de taille n.*

6.3 Nombre moyen de rendez-vous

Soit X le nombre de rendez-vous pour un appel sur G, le nombre moyen de rendez-vous sur G, noté $\overline{M}(G)$, est $E(X)$: c'est l'espérance mathématique de X. Ce paramètre peut être considéré comme une mesure du degré de parallélisme réalisé par l'algorithme de rendez-vous.

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

$$\chi_e = \begin{cases} 1 & \text{s'il y a rendez-vous sur } e, \\ 0 & \text{sinon.} \end{cases} \tag{6.16}$$

On a

$$X = \sum_{e \in E} \chi_e. \tag{6.17}$$

Donc :

$$E(X) = \sum_{e \in E} E(\chi_e). \tag{6.18}$$

Or

$$E(\chi_{\{v,w\}}) = \frac{1}{d(v)d(w)}, \tag{6.19}$$

donc :

$$E(X) = \sum_{\{v,w\} \in E} \frac{1}{d(v)d(w)}, \tag{6.20}$$

et finalement :

Proposition 6.4 *Le nombre moyen de rendez-vous sur G est :*

$$\overline{M}(G) = \sum_{\{v,w\} \in E} \frac{1}{d(v)d(w)}. \tag{6.21}$$

Considérons les cas particuliers suivants :

Exemple 6.2 Si G est le graphe complet de taille $n \geq 2$, alors

$$\overline{M}(G) = \binom{n}{2} \frac{1}{(n-1)^2} = \frac{n}{2(n-1)}. \tag{6.22}$$

Exemple 6.3 Si G est le cycle de taille $n \geq 3$, alors

$$\overline{M}(G) = \frac{n}{4}. \tag{6.23}$$

Exemple 6.4 Si $G = (V, E)$ est de degré borné par d, alors

$$\overline{M}(G) \geq \frac{|E|}{d^2}. \tag{6.24}$$

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

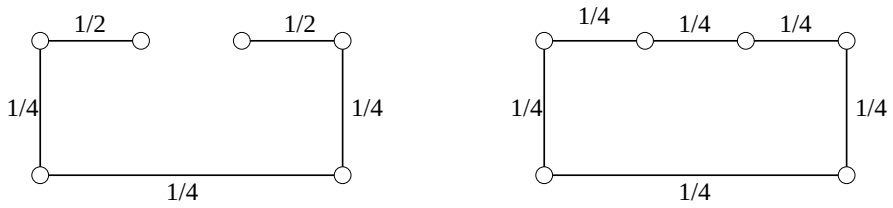
$$\overline{M}(T) \geq \frac{n-1}{d^2}. \tag{6.25}$$

Dans le cas des graphes réguliers de degré d :

$$\overline{M}(G) = \frac{n}{2d}, \tag{6.26}$$

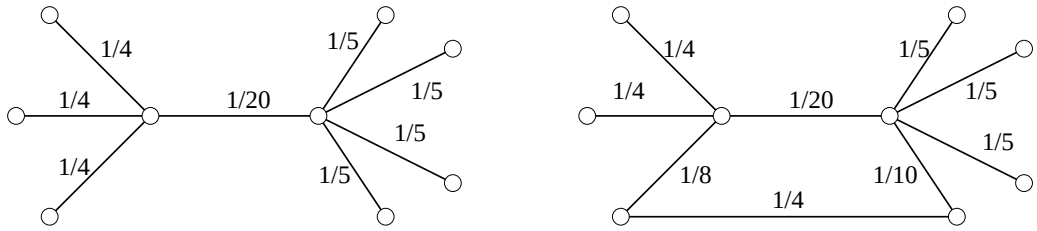
où n est la taille du graphe.

Nous nous intéressons maintenant à l'impact de l'ajout d'une nouvelle arête sur le comportement de $\overline{M}(G)$. Les exemples suivants illustrent le fait que le nombre d'arêtes ne favorise pas nécessairement les rendez-vous. En effet, les figures 6.3 et 6.4 montrent que le nombre moyen de rendez-vous n'est pas monotone en fonction du nombre d'arêtes.



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

FIG. 6.3 – *Un exemple où le nombre moyen de rendez-vous diminue si on ajoute une arête.*



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

FIG. 6.4 – *Un exemple où le nombre moyen de rendez-vous augmente si on ajoute une arête.*

La proposition 6.5 donne un minorant du nombre moyen de rendez-vous.

Proposition 6.5 *Pour un entier positif fixé n, le graphe complet K_n minimise le nombre moyen de rendez-vous sur les graphes de taille n. La valeur moyenne minimale réalisée par K_n est $\frac{n}{2(n-1)}$.*

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

$$\overline{M}(G) = \frac{1}{2} \sum_{v \in V} p(v), \tag{6.27}$$

où $p(v)$ désigne la probabilité que v obtienne un rendez-vous.

Sachant que :

$$p(v) = \frac{1}{d(v)} \sum_{\{w,v\} \in E} \frac{1}{d(w)}, \tag{6.28}$$

et que $d(w) \leq n - 1$, nous avons $p(v) \geq \frac{1}{n-1}$. En sommant sur tous les sommets, nous obtenons

$$\overline{M}(G) \geq \frac{n}{2(n-1)}. \tag{6.29}$$

Or, dans le cas du graphe complet de taille n :

$$\overline{M}(G) = \frac{n}{2(n-1)}. \tag{6.30}$$

D'où la proposition. □

6.4 Distribution du nombre de rendez-vous

Cette section s'intéresse à la distribution asymptotique du nombre de rendez-vous pour quelques classes de graphes. Essentiellement, on étudie le comportement asymptotique de ce nombre pour des classes particulières de graphes de taille très grande. Si le graphe est une étoile, ce nombre est toujours égal à 1. Si c'est un cycle de taille n, cette variable prend ses valeurs dans l'intervalle $[0, \frac{n}{2}]$ et son espérance mathématique est $\frac{n+1}{4}$. Dans le cas des graphes complets, ses valeurs possibles sont dans l'intervalle $[0, \frac{n}{2}]$, et son espérance mathématique est asymptotiquement $\frac{1}{2}$. Cependant, même si, en principe, le calcul de la distribution est possible, il n'y a pas de technique facile et simple, et une application naïve de la technique classique basée sur l'énumération est assez compliquée.

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

6.4.1 Cas des cycles et des chaînes

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

Lemme 6.4 *Pour tout entier k, la probabilité d'obtenir exactement k rendez-vous est*

$$\mathbb{P}r(X_n = k) = \frac{1}{2^{n-2}}\binom{n-1}{2k-1}. \tag{6.31}$$

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

$$\left\{ \begin{array}{rcll} \phi_n(x) & = & \frac{1}{2}\phi_{n-1}(x) + \frac{1}{2}x\psi_{n-1}(x), & \forall n \geq 2 \\ \psi_n(x) & = & \frac{1}{2}\psi_{n-1}(x) + \frac{1}{2}\phi_{n-1}(x), & \forall n \geq 2 \end{array} \right. \tag{6.32}$$

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

$$\phi_n(x) = \frac{2\sqrt{x}}{1+\sqrt{x}}\left(\frac{1+\sqrt{x}}{2}\right)^n - \frac{2\sqrt{x}}{1-\sqrt{x}}\left(\frac{1-\sqrt{x}}{2}\right)^n, \tag{6.33}$$

on en déduit alors que

$$[x^k]\phi_n(x) = \frac{1}{2^{n-2}}\binom{n-1}{2k-1}. \tag{6.34}$$

D'où le lemme. □

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

Corollaire 6.4 *La variance de la variable aléatoire X_n est égale à $\frac{n-1}{16}$.*

D'autre part, la fonction génératrice $\phi_n(x)$ peut être utilisée pour montrer que le comportement asymptotique de la v.a. X_n est normal. Plus précisément :

Théorème 6.2 *La v.a. normalisée définie par*

$$Y_n = \frac{4X_n - n}{\sqrt{n}}, \tag{6.35}$$

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

$$\mathbb{P}r(a < Y_n \leq b) \to \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx. \tag{6.36}$$

Preuve. Pour tout entier k, soit $j = 2k-1$, et $l = n - 2k + 1$. Le lemme 6.4 montre que

$$\mathbb{P}r(X_n = k) = \frac{1}{2} \frac{n!}{j!l!} \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^l. \tag{6.37}$$

En appliquant la formule de Stirling, nous obtenons :

$$\mathbb{P}r(X_n = k) \sim \frac{1}{2} \sqrt{\frac{n}{2\pi jl}} \left(\frac{n}{2j}\right)^j \left(\frac{n}{2l}\right)^l, \tag{6.38}$$

et

$$\frac{jl}{n} \sim \frac{(\sqrt{n}x + n)(n - \sqrt{n}x)}{4n} \sim \frac{4}{n}, \text{ où } x = \frac{4k-n}{\sqrt{n}}. \tag{6.39}$$

Ainsi

$$\mathbb{P}r(X_n = k) = \frac{1}{\sqrt{2\pi n}} \left(\frac{n}{2j}\right)^j \left(\frac{n}{2l}\right)^l \tag{6.40}$$

et

$$\ln \left(\frac{n}{2j}\right)^j \left(\frac{n}{2l}\right)^l \sim -\frac{x^2}{2}. \tag{6.41}$$

Le théorème est alors prouvé par un raisonnement similaire a celui de ((Loè55), p. 22). □

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

Théorème 6.3 *Définissons la v.a. normalisée par* $V_n = \frac{4Z_n - n}{\sqrt{n}}$. *Alors, si n tend vers l'infini :*

$$\mathbb{P}r(a < V_n \leq b) \rightarrow \frac{1}{\sqrt{2}\pi} \int_a^b e^{-x^2/2} dx \tag{6.42}$$

pour tout intervalle réel $[a, b]$.

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

$$R_i = \begin{cases} 1 & \text{s'il y a un rendez-vous sur } \{i, i+1\}, \\ 0 & \text{sinon.} \end{cases} \tag{6.43}$$

Nous avons :

$$\sum_{i=2}^{n-2} R_i \leq X_n \leq \sum_{i=2}^{n-2} R_i + 2. \tag{6.44}$$

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

$$S_i = \begin{cases} 1 & \text{s'il y a un rendez-vous sur } \{i, i+1\}, \\ 0 & \text{sinon.} \end{cases} \tag{6.45}$$

Nous avons alors

$$\sum_{i=2}^{n-2} S_i \leq Z_n \leq \sum_{i=2}^{n-2} S_i + 3. \tag{6.46}$$

Or

$$\sum_{i=2}^{n-2} R_i = \sum_{i=2}^{n-2} S_i, \tag{6.47}$$

donc

$$|Z_n - X_n| \leq 3. \tag{6.48}$$

D'où

$$|V_n - Y_n| \leq \frac{12}{\sqrt{n}}. \tag{6.49}$$

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

6.4.2 Nombre de rendez-vous dans le graphe complet

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

On commence par le lemme suivant :

Lemme 6.5 *Soit m un entier positif. Si $n \to \infty$, alors la probabilité d'avoir au moins m rendez-vous sur K_n, $\mathbb{P}r(X_n \geq m)$ tend vers $\sum_{k \geq m} (-1)^{k+m} \frac{1}{k!2^k}$.*

Preuve. D'après la proposition 6.1, et en utilisant un raisonnement similaire à celui de la preuve du premier point de la proposition 6.3, la probabilité d'avoir au moins m rendez-vous sur K_n vaut :

$$\mathbb{P}r(X_n \geq m) = \sum_{k \geq m} (-1)^{k+m} \frac{n!}{k!2^k (n-2k)!} \frac{1}{(n-1)^{2k}}. \tag{6.50}$$

D'où le lemme par une application de la formule de Stirling. □

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

Théorème 6.4 *Pour tout entier positif m, la probabilité $\mathbb{P}r(X_n = m)$ pour que X_n soit égal à m tend vers*

$$2\frac{(-1)^m}{\sqrt{e}} - \frac{1}{m!2^m} - 2(-1)^m \sum_{k<m} (-1)^k \frac{1}{k!2^k} \tag{6.51}$$

quand $n \to \infty$. Et pour $m = 0$,

$$\mathbb{P}r(X_n = 0) \longrightarrow \frac{1}{\sqrt{e}}, \tag{6.52}$$

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

$$\mathbb{P}r(X = m) = \mathbb{P}r(X \geq m) - \mathbb{P}r(X \geq m+1). \tag{6.53}$$

Le théorème est alors une conséquence du lemme 6.5. □

6.5 Graphes aléatoires

Cette section étudie l'algorithme *S_Rdv*() appliqué à des graphes aléatoires. Le modèle retenu est le modèle $G_{n,p}$: On a n sommets différents, et pour toute paire de sommets u et v, l'arête $\{u, v\}$ est ajoutée dans le graphe avec probabilité p, (et n'est donc pas ajoutée avec probabilité $q = 1 - p$).

6.5.1 Probabilité d'un rendez-vous

Soit $G_{n,p} = (V, E)$ un graphe aléatoire et soit u et v deux sommets différents. S'il n'y a pas d'arête entre u et v c'est-à-dire, si $\{u, v\} \notin E$ alors il ne peut y avoir de rendez-vous entre u et v. Donc, on suppose que $e = \{u, v\} \in E$ et on étudie la probabilité d'un rendez-vous sur e conditionnée par l'événement $e \in E$. Soit $\mathcal{H}(e)$ l'événement « *il y a rendez-vous sur e* » . Alors, nous avons :

$$\begin{aligned}&\mathbb{P}r(\mathcal{H}(e) \mid e \in E)\\ &= \sum_{k_1=1}^{n-1}\sum_{k_2=1}^{n-1} \mathbb{P}r\left(\mathcal{H}(e) \mid e \in E \wedge d(u) = k_1 \wedge d(v) = k_2\right) \mathbb{P}r\left(d(u) = k_1 \wedge d(v) = k_2 \mid e \in E\right).\end{aligned}$$

Or, $\mathbb{P}r\left(\mathcal{H}(e) \mid e \in E \wedge d(u) = k_1 \wedge d(v) = k_2\right) = \frac{1}{k_1 k_2}$ et

$$\mathbb{P}r\left(d(u) = k_1 \wedge d(v) = k_2 \mid e \in E\right) = \mathbb{P}r\left(d(u) = k_1 \mid e \in E\right) \mathbb{P}r\left(d(v) = k_2 \mid e \in E\right). \quad (6.54)$$

Un raisonnement probabiliste permet de voir que :

$$\mathbb{P}r\left(d(u) = k \mid e \in E\right) = \binom{n-2}{k-1} p^{k-1} q^{n-k-1}. \quad (6.55)$$

Donc, nous dérivons de (6.54) :

$$\mathbb{P}r\left(d(u) = k_1 \wedge d(v) = k_2 \mid e \in E\right) = \binom{n-2}{k_1-1}\binom{n-2}{k_2-1} p^{k_1+k_2-2} q^{2n-k_1-k_2-2}. \quad (6.56)$$

Ainsi

$$\mathbb{P}r(\mathcal{H}(e) \mid e \in E) = \left(\sum_{k=1}^{n-1} \frac{1}{k}\binom{n-2}{k-1} p^{k-1} q^{n-k-1}\right)^2. \quad (6.57)$$

Un calcul simple faisant intervenir les série génératrices permet d'établir le lemme suivant :

Lemme 6.6 *Soit $G_{n,p} = (V, E)$ un graphe aléatoire et soit u et v deux sommets. Alors*

$$\mathbb{P}r(\mathcal{H}(e)) = \frac{(1 - q^{n-1})^2}{(n-1)^2 p}. \quad (6.58)$$

Alors :

Corollaire 6.5 *Soit $G_{n,p}$ un graphe aléatoire et soit v un sommet quelconque. Désignons par $\mathcal{HS}(v)$ l'événement « v est impliqué dans un rendez-vous ». Alors*

$$\mathbb{P}r(\mathcal{HS}(v)) = \frac{\left(1 - q^{n-1}\right)^2}{(n-1)p}.$$

Preuve. Soit $v \in V$ un sommet non isolé et soit $u \in N(v)$ tel que $c(v) = u$. Il y a un rendez-vous entre v et u ssi $c(u) = v$. Donc,

$$\begin{aligned}&\mathbb{P}r\left(\mathcal{HS}(v) \mid v \text{ n'est pas isolé }\right)\\ &= \sum_{k=1}^{n-1} \mathbb{P}r(c(u) = v \mid (u, v) \in E \wedge d(u) = k)\mathbb{P}r(d(u) = k \mid (u, v) \in E)\\ &= \sum_{k=1}^{n-1} \tfrac{1}{k}\tbinom{n-2}{k-1} p^{k-1} q^{n-k-1} = \tfrac{1-q^{n-1}}{(n-1)p}.\end{aligned}$$

Or $\mathbb{P}r(v \text{ n'est pas isolé }) = 1 - q^{n-1}$, donc

$$\mathbb{P}r\left(\mathcal{HS}(v)\right) = \frac{(1-q^{n-1})^2}{(n-1)p}$$

Le corollaire en résulte. □

6.5.2 Le nombre moyen de rendez-vous

Soit $(u,v) \in V \times V$ tel que $u \neq v$, et soit $\chi_{u,v}$ la v.a. définie par : $\chi_{u,v} = 1$ s'il y a rendez-vous entre u et v et $\chi_{u,v} = 0$ sinon. Donc, par la linéarité de l'espérance, si on note par $X_{n,p}$ le nombre de rendez-vous dans $G_{n,p}$ alors

$$\begin{aligned}\mathbb{E}\left(X_{n,p}\right) &= \tfrac{1}{2}\textstyle\sum_{(u,v)\in V\times V \text{ tel que } u\neq v} \mathbb{E}\left(\chi_{u,v}\right) \\ &= \tfrac{1}{2}\textstyle\sum_{(u,v)\in V\times V \text{ tel que } u\neq v} \mathbb{P}r\left(\chi_{u,v}=1\right),\end{aligned}$$

Ainsi, en vertu du lemme 6.6 :

Lemme 6.7

$$\mathbb{E}\left(X_{n,p}\right) = \frac{n(n-1)}{2}\frac{\left(1-q^{n-1}\right)^2}{(n-1)^2 p} = \frac{n}{2(n-1)p}\left(1-q^{n-1}\right)^2.$$

Remarque 6.2

1. *Si p est une constante, alors $\left(1-q^{n-1}\right)^2 \to 1$ quand $n \to \infty$. Donc, $\mathbb{E}\left(X_{n,p}\right) \sim \frac{1}{2p}$ quand $n \to \infty$.*
2. *Si $np = \lambda + o(1) > 0$, alors le degré moyen de tout sommet v est une constante, donc $q^{n-1} \sim e^{-\lambda}$. Donc,*

$$\exists \alpha > 0 \text{ tel que } \mathbb{E}(X_{n,p}) \sim \alpha n.$$

3. *Si $n^2 p = \lambda + o(1)$, alors*

$$\exists \beta > 0 \text{ tel que } \mathbb{E}(X_{n,p}) \sim \beta.$$

4. *Si $n^\gamma p = \lambda + o(1)$, avec $\gamma > 2$ alors*

$$\mathbb{E}(X_{n,p}) \to 0, \text{ quand } n \to \infty.$$

6.5.3 Probabilité de succès

Soit $G_{n,p} = (V,E)$ un graphe aléatoire, et soit $v \in V$ un sommet quelconque. Si v n'est pas isolé, alors tous ses voisins ont la même probabilité $\frac{1}{d(v)}$ d'être choisis par v. Dans cette section, nous étudions la probabilité de succès dans le graphe $G_{n,p}$, c'est-à-dire, la probabilité d'au moins un rendez-vous dans $G_{n,p}$.

Nous commençons par calculer le probabilité que le graphe $G_{n,p}$ contienne au moins un sommet isolé.

Lemme 6.8 *Soit $G_{n,p}$ un graphe aléatoire. Si on note $\mathcal{P}(G_{n,p})$ la probabilité que $G_{n,p}$ contienne au moins un sommet isolé, alors :*

$$\mathcal{P}(G_{n,p}) \leq nq^{n-1}.$$

Preuve. Soit v un sommet de V. La probabilité que v soit isolé est donnée par $\mathbb{P}r\left(d(v)=0\right)=q^{n-1}$. Donc, si $V=\{v_1,v_2,\cdots,v_n\}$, alors

$$\mathcal{P}(G_{n,p}) = \mathbb{P}r\left(\bigvee_{i=1}^{n} d(v_i)=0\right)$$

Or, pour tout $i \geq 1$, soit $\{v_{j_1},v_{j_2},\cdots,v_{j_i}\}$ i sommets différents. Nous avons :

$$\begin{aligned}
&\mathbb{P}r\left(d(v_{j_1})=0 \wedge d(v_{j_2})=0 \wedge \cdots \wedge d(v_{j_i})=0\right)\\
&= \mathbb{P}r\left(d(v_{j_1})=0\right)\mathbb{P}r\left(d(v_{j_2})=0 \mid d(v_{j_1})=0\right)\cdots\mathbb{P}r\left(d(v_{j_i})=0 \mid \textstyle\bigwedge_{l=1}^{i-1} d(v_{j_l})=0\right)\\
&= q^{n-1}q^{n-2}\cdots q^{n-i}\\
&= q^{\frac{i(i-1)}{2}+i(n-i)}.
\end{aligned}$$

Donc, en utilisant le principe d'inclusion-exclusion (Roooo), on obtient

$$\mathcal{P}(G_{n,p}) = \sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}q^{\frac{i(i-1)}{2}+i(n-i)}.$$

Le lemme est une conséquence immédiate de cette égalité. □

On en déduit le corollaire suivant :

Corollaire 6.6 *Soit $\alpha \geq 2$. Si $p=\frac{\alpha \log n}{n-1}$ alors, avec grande probabilité, $G_{n,p}$ ne contient pas de sommet isolé.*

Preuve. Utilisant le lemme précédent, nous avons $\mathcal{P}(G_{n,p}) \leq nq^{n-1} = n\left(1-p\right)^{n-1} \leq ne^{-(n-1)p}$. Donc, si $p=\frac{\alpha \log n}{n-1}$ alors

$$\mathcal{P}(G_{n,p}) \leq \frac{1}{n^{\alpha-1}}.$$

Ce qui veut dire que la probabilité que $G_{n,p}$ ne contienne pas de sommet isolé est supérieure ou égale à $1-o(\frac{1}{n})$. Le corollaire en résulte. □

Dans le reste de cette section, nous supposons que $p=\frac{\alpha \log n}{n-1}$, donc, avec forte probabilité, $G_{n,p}$, ne contient pas de sommet isolé. Soient $(V_i)_{i=1,..,k}$ et $(E_i)_{i=1,..,k}$ tels que

- $\forall i \neq j$, $V_i \cap V_j = \emptyset$ et $E_i \cap E_j = \emptyset$
- $V = V_1 \cup V_2 \cup ... \cup V_k$, $E = E_1 \cup E_2 \cup ... \cup E_k$, et
- $\forall i$, $G_i = (V_i, E, i)$ est un graphe connexe.

Il est facile de voir que $s(G_{n,p}) = \max\{s(G_i) \mid i \in \{1,2,...,k\}\}$ et en utilisant le théorème 1.1 de (Dieo2), $s(G_i) \geq s(K_{|V_i|})$, pour tout $i \in \{1,2,...,k\}$. Or, dans (DT05), l'auteur prouve aussi que

$$s(K_n) \geq s(K_{n+1}), \text{ for any } n \geq 2.$$

Ainsi

$$s(K_n) \geq 1 - e^{-1/2}, \text{ for any } n \geq 2.$$

Nous avons donc :

Lemme 6.9 *Si $p = \frac{\alpha \log n}{n}$ pour tout $\alpha \geq 2$ alors, avec forte probabilité, la probabilité de succès dans $G_{n,p}$ est minorée par une constante :*

$$s(G_{n,p}) \geq 1 - e^{\frac{1}{2}}.$$

6.5.4 Distribution du nombre de rendez-vous

Dans cette section, nous étudions le nombre de rendez-vous dans un graphe $G_{n,p}$. Soit $X_{n,p}$ la v.a. qui compte le nombre de rendez-vous. On montre, en particulier, que la v.a. $X_{n,p}$ a une distribution asymptotique de Poisson avec comme paramètre $\lambda = \frac{1}{2p}$.

Nous avons alors le théorème suivant :

Théorème 6.5 *La probabilité d'obtenir exactement m rendez-vous dans $G_{n,p}$ vaut asymptotiquement :*

$$\mathbb{P}r\left(X_{n,p} = m\right) \sim \sum_{k=m}^{n/2} (-1)^{k+m} \binom{k}{m} \frac{n!}{k! 2^k (n-2k)!} p^k \left(\frac{1-q^{n-1}}{(n-1)p}\right)^{2k}.$$

Preuve. Soit M un couplage de taille $l \geq 1$. Si $M = \{e_1, e_2, \cdots, e_l\}$, alors, nous avons :

$$\begin{aligned} \mathbb{P}r\left(M\right) &= \mathbb{P}r\left(e_1 \wedge e_2 \wedge \cdots \wedge e_l\right) \\ &= \mathbb{P}r\left(e_1\right) \mathbb{P}r\left(e_2 \mid e_3\right) \cdots \mathbb{P}r\left(e_l \mid e_1 \wedge e_2 \cdots \wedge e_{l-1}\right) \end{aligned}$$

où, pour tout $i \in \{1, 2, \cdots, l\}$, e_i est l'événement : il y a un rendez-vous sur e_i.

Par ailleurs, M étant un couplage, les arêtes de M ne sont pas adjacentes. Donc, avoir un rendez-vous sur une arête dans M n'affecte pas la probabilité d'avoir un rendez-vous sur les autres arêtes de M.

On a donc :

$$\mathbb{P}r\left(M\right) = \prod_{i=1}^{l} \mathbb{P}r(e_i).$$

Or, pour tout i, $\mathbb{P}r(e_i) = \left(\frac{1-q^{n-1}}{(n-1)p}\right)^2$. D'où :

$$\mathbb{P}r\left(M\right) = \left(\frac{1-q^{n-1}}{(n-1)p}\right)^{2l}.$$

Soit, à présent, $\mathcal{M}_k$ l'ensemble de tous les couplages de taille k, et soit la suite $\{q_k\}$ définie par :

$$q_k = \sum_{M \in \mathcal{M}_k} \mathbb{P}r\left(M\right).$$

Alors

$$q_k = \mid \mathcal{M}_k \mid \left(\frac{1-q^{n-1}}{(n-1)p}\right)^{2k}.$$

Cependant, pour tout $k \geq 1$, on peut construire un couplage de taille k dans $G_{n,p} = (V, E)$ ssi $\mid E \mid \geq k$ et $\forall e_1, e_2 \in E$, e_1 et e_2 ne sont pas adjacentes. Un raisonnement combinatoire permet de calculer la probabilité et le nombre de couplages de taille k : $\frac{n!}{k!2^k(n-2k)!}p^k$. Ainsi,

$$q_k = \frac{n!}{k!2^k(n-2k)!}p^k\left(\frac{1-q^{n-1}}{(n-1)p}\right)^{2k}.$$

Une application directe du principe d'inclusion-exclusion permet d'obtenir :

$$\mathbb{P}r\left(X_{n,p} = m\right) = \sum_{k=m}^{n/2}(-1)^{k+m}\binom{k}{m}\frac{n!}{k!2^k(n-2k)!}p^k\left(\frac{1-q^{n-1}}{(n-1)p}\right)^{2k}.$$

□

Corollaire 6.7 *Si p est une constante, alors pour tout entier positif m, la probabilité que $X_{n,p}$ soit égal à m tend vers $\frac{1}{m!}\left(\frac{1}{2p}\right)^m e^{-\frac{1}{2p}}$, quand $n \to \infty$.*

6.6 Efficacité de l'algorithme *S_Rdv()*

Cette section s'intéresse à l'efficacité de l'algorithme *S_Rdv()*. Soit $G = (V, E)$ un graphe avec $|V| = n$ et $|E| = m$, et soit M sa matrice d'incidence, (Roooo). Pour une énumération des sommets $v_1, \cdots, v_n$, et des arêtes $e_1, \cdots, e_m$, elle est définie par :

$$M[i,j] = \begin{cases} 0 & \text{si } v_i \text{ n'est pas extrémité de } e_j \\ 1 & \text{si } v_i \text{ est une extrémité de } e_j, \end{cases} \tag{6.59}$$

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

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

$$x[j] = \begin{cases} 0 & \text{si } x_j \in F \\ 1 & \text{si } x_j \notin F, \end{cases} \tag{6.60}$$

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

Donc la taille maximale d'un couplage de G, $K(G)$ est la valeur optimale de la fonction objective du programme linéaire en nombres entiers suivant (Roooo) :

$$\begin{array}{ll} \text{maximiser :} & < \mathbf{1}_m, x > \\ \text{sous les contraintes :} & Mx \leq \mathbf{1}_n \\ & x \in \{0,1\}^m. \end{array} \tag{6.61}$$

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

$$\begin{array}{ll} \text{minimiser :} & < \mathbf{1}_n, y > \\ \text{sous les contraintes :} & M^T y \geq \mathbf{1}_m \\ & y \in \mathbb{R}^n, \;\; y \geq 0 \end{array} \tag{6.62}$$

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

Énonçons maintenant le théorème principal de cette section :

Théorème 6.6 *L'efficacité $\Delta_{RV}(T)$ de l'algorithme probabiliste $S\_Rdv()$ proposé sur tout arbre T est strictement supérieure à $\frac{1}{3}$.*

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

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

Soit T' l'arbre induit par suppression de a et de ses fils de l'arbre enraciné T. Une sommet-couverture de T peut être obtenue par l'ajout de a à une sommet-couverture de T'. Ainsi, si le théorème est vérifié pour T', on doit prouver que $\overline{M}(T) - \overline{M}(T') \geq \frac{1}{3}$.
un calcul simple permet d'obtenir la différence. En effet, nous avons :

$$\overline{M}(T) - \overline{M}(T') = \frac{d-1}{d} + \frac{1}{dd'} - \sum_{i=1}^{d'-1} [\frac{1}{(d'-1)\delta_i} - \frac{1}{d'\delta_i}], \tag{6.63}$$

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

Or, il est facile de voir que

$$\sum_{i=1}^{d'-1} [\frac{1}{(d'-1)\delta_i} - \frac{1}{d'\delta_i}] < \frac{1}{d'}. \tag{6.64}$$

Ainsi,

$$\overline{M}(T) - \overline{M}(T') \geq \frac{d-1}{d} + \frac{1}{dd'} - \frac{1}{d'} = (1 - \frac{1}{d})(1 - \frac{1}{d'}). \tag{6.65}$$

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

$$\frac{1}{2} + \frac{1}{4} - \frac{1}{2\delta} \tag{6.66}$$

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

Remarque 6.3 *Pour le graphe de la figure 6.5, l'efficacité est inférieure à $\frac{1}{2}$. Nous ne conaissons pas de borne plus grande que $\frac{1}{3}$.*

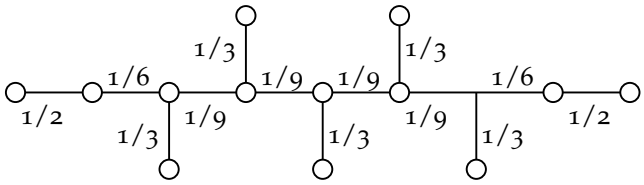


FIG. 6.5 – *Un graphe G tel que $\overline{M}(G) = 31/9 < K(G)/2 = 7/2$.*

Un algorithme à base d'agenda dynamique

7

Ce chapitre présente un algorithme permettant de réaliser des rendez-vous mais dont le principe est différent de l'algorithme étudié dans le chapitre précédent. L'idée de l'algorithme présenté ici est d'utiliser un agenda dynamique et de procéder par anticipation.

Le travail présenté ici a été réalisé en collaboration avec A. El Hibaoui, Y. Métivier, J.-M. Robson et N. Saheb

7.1 Introduction

L'algorithme présenté dans ce chapitre est à base d'agendas dynamiques. Il s'agit d'un algorithme synchrone basé sur un mécanisme de *timer* : le processeur a accès à une horloge physique et comme indiqué dans (Teloo) page 87 :

1. on suppose que le timer est un temps global continu auquel tous les processeurs ont accès : c'est une variable continue à valeurs réelles dont la valeur augmente en fonction du temps,
2. on fait l'hypothèse d'un temps global, chaque événement a lieu à un instant donné et la durée de l'événement est supposée être nulle.

On suppose que les communications ne prennent pas de temps, ainsi, le temps qui s'écoule entre le moment où un sommet prend une décision et le moment où il la transmet à ses voisins est égal à 0.

7.1.1 Algorithme *Dynamic_Rdv*()

Chaque processus p tire, pour chaque voisin q un temps $t_p(q)$: une v.a. uniforme continue dans l'intervalle $[0, 1[$. Si les deux processus p et q sont tous les deux disponibles à l'instant $t_p(q)$, un rendez-vous a lieu entre p et q à cet instant (voir l'algorithme 7).

Remarque 7.1 *Étant donné que le temps est continu, un des trois événements se produit avec probabilité 1.*

Algorithme 7 : L'algorithme *Dynamic_RdV*()

1: Chaque processus p répète infiniment souvent l'ensemble d'actions suivantes :
2: p génère $t_p(q)$: un temps aléatoire choisi uniformément dans l'intervalle de réels $[0,1[$ pour chaque voisin q ;
3: p attend jusqu'à ce que l'un de ces évènements se produise : ;
4: 1. p reçoit 1 d'un voisin r : ;
5: p envoie 0 à tous les autres voisins (*Il y a un rendez-vous entre p et r*) ;
6: 2. $t = t_p(q)$ et p n'a reçu aucun signal de q : ;
7: p envoie 1 à q et 0 à tous les autres voisins (*Il y a un rendez-vous entre p et q *) ;
8: 3. $t = 1$ (* le round est terminé sans que p ne prenne part à un rendez-vous*) ;

7.2 Nombre de rendez-vous

Soit $G = (V, E)$ un graphe de taille $\mid V \mid = n > 1$ et tel que $\mid E \mid = m > 0$. L'algorithme *Dynamic_Rdv*() commence par générer $2|E|$ v.a. réelles indépendantes dans l'intervalle $[0,1[$: 2 v.a. pour chaque arête. Nous supposons que toutes les $(2|E|)!$ permutations possibles sont équiprobables. Pour que cette hypothèse soit vraie, il suffit de considérer que pour chaque arête $e = \{u, v\}$ de G, l'algorithme génère deux v.a. $X_e(u)$ et $X_e(v)$, correspondant aux valeurs $t_u(v)$ et $t_v(u)$ dans l'algorithme.

Le premier rendez-vous aura lieu sur une arête $e = \{u, v\}$ si une des v.a. $X_e(u)$ ou $X_e(v)$ est minimale dans tout le graphe. Ainsi, pour le premier rendez-vous dans G, toutes les arêtes ont la même chance $1/|E|$ d'être choisies.

L'affectation du premier rendez-vous à u et v sur $\{u, v\}$, implique que ces deux sommets et toutes leurs arêtes adjacentes seront supprimées du graphe G. Le processus continue alors dans le graphe résiduel (sans regénérer de nouvelles dates de rendez-vous) jusqu'à ce que toutes les arêtes soient supprimées du graphe. Le nombre de rendez-vous dans un round est donc simplement le nombre total d'arêtes sur lesquelles un rendez-vous a lieu. Soit $R(G)$ ce nombre, il s'agit d'une v.a. entière dont la valeur est égale à 0 avec probabilité 1 si $E = \emptyset$, à 1 avec probabilité 1 si $|E| = 1$ et à une valeur dans l'ensemble des cardinaux des couplages maximaux du graphe G avec une probabilité à déterminer.

Il est alors facile de prouver le fait simple suivant :

Fait. Supposons que le premier rendez-vous ait lieu sur l'arête $e = \{u, v\}$. Soit $G_e = (V', E')$ le graphe résiduel obtenu après suppression des sommets u et v et de leurs arêtes adjacentes. Alors toutes les $(2|E'|)!$ permutations des v.a. générées dans G_e, conditionnées par la minimalité de $X_e(u)$ or $X_e(v)$, ont la même probabilité $1/(2|E'|)!$. Ce qui implique que garder

les $2|E|$ v.a. pour les arêtes de G_e est équivalent à un tirage de nouvelles v.a. sur les arêtes de G_e.

Ce fait permet de calculer, récursivement, la distribution de probabilité de la v.a. $R(G)$. Soit $G = (V, E)$, si $E = \emptyset$, alors $R(G) = 0$. On peut prouver facilement la proposition suivante :

Proposition 7.1 *Soit $G = (V, E)$ avec $n = |V| \geq 2$ et $m = |E| \geq 1$. Alors, nous avons :*

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

À noter que si $m \geq 1$, alors la probabilité de succès est égale à 1 dans un round, ce qui n'est pas le cas de l'algorithme *S_Rdv*().

- Si G est un graphe complet de taille $\mid V \mid = n$, alors, avec probabilité 1, $R(G) = \lfloor n/2 \rfloor$.
- Si G est une étoile de taille $n \geq 2$, alors, avec probabilité 1, $R(G) = 1$.

À présent, soit $M_e(G)$ la v.a. égale à 1 si un rendez-vous a lieu sur $e = \{u, v\}$ et à 0 sinon. la v.a. $R(G)$ peut être écrite comme la somme $\sum_{e \in E} M_e(G)$?. De plus, nous avons :

Proposition 7.2 *Soit $F(e)$ l'ensemble des sommets de G non adjacents à e. La v.a. peut alors être caractérisée récursivement comme suit :*

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

Dans le chapitre précédent, nous avons vu que, si $e = \{u, v\}$, alors le probabilité que $M_e(G)$ soit égale à 1 vaut $1/(d(u)d(v))$. La proposition suivante montre que l'algorithme *Dynamic_Rdv*() est au moins aussi efficace que l'algorithme *S_Rdv*().

Proposition 7.3 *Soit $e = \{u, v\}$ une arête dans le graphe $G = (V, E)$. Alors*

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

En réalité, dans plusieurs cas, la probabilité d'un rendez-vous sur une arête avec l'algorithme *Dynamic_Rdv*() est largement plus grande que celle obtenue avec l'algorithme *S_Rdv*(). En effet, il est possible d'exhiber des exemples dans lesquels $d(u)d(v)$ est plus grand que m et, donc, la borne inférieure $1/m$ pour $Pr(M_e(G) = 1)$ dépasse la valeur $1/(d(u)d(v))$.

7.2.1 Nombre moyen de rendez-vous

Cette section s'intéresse au nombre moyen de rendez-vous réalisés par l'algorithme *Dynamic_Rdv*(). Soit $\overline{R}(G)$ l'espérance mathématique du nombre de rendez-vous dans le graphe G. Nous avons :

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

De la proposition 7.3, nous déduisons la proposition suivante :

Proposition 7.4 *Pour tout graphe, l'espérance mathématique du nombre de rendez-vous réalisés par l'algorithme S_Rdv() est inférieure à l'espérance mathématique du nombre de rendez-vous réalisés par l'algorithme Dynamic_Rdv().*

1. Graphes complets. Si K_n est un graphe complet de taille n, alors $\overline{R}(K_n) = \lfloor n/2 \rfloor$. Alors que l'algorithme *S_Rdv()* réalise en moyenne $n/(2(n-1))$ rendez-vous.

2. Graphes étoiles. Si G est une étoile de taille $n \geq 2$, alors $\overline{R}(G) = 1$.

3. Graphes Chaînes. Comme dernier exemple, on considère le cas de la chaîne. Soit $G_n = (V_n, E_n)$ une chaîne de taille $|V_n| = n$, et soit $R_n = \overline{R}(G_n)$ le nombre moyen de rendez-vous dans G_n. Nous avons la proposition suivante :

Lemme 7.1 *On a :*

1. *Le nombre moyen de rendez-vous dans une chaîne satisfait la récurrence suivante :*

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

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

Remarque 7.2 *La fonction génératrice $R(z)$ est la fonction correspondant au problème « Seating Arrangement Problem » proposé et résolu par Flajolet dans (Fla97).*

La FGO $R(z)$ peut également être utilisée pour donner l'expression exacte de R_n. En effet :

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

En outre, nous avons

Proposition 7.5 *La valeur asymptotique de R_n vaut :*

$$R_n \sim (1-e^{-2})\frac{n}{2} \approx 0.432332\, n \qquad (n \to \infty).$$

7.2.2 Impact de l'ajout d'une arête

Considérons les graphes de la figure 7.1. G_2 (resp. G_3) est obtenu en supprimant de (resp. en ajoutant dans) G_1 une arête. Un calcul simple permet de voir que $\overline{R}(G_1) = 9/5$ et $\overline{R}(G_2) = \overline{R}(G_3) = 2$. Ces exemples montrent que le rajout d'une arête dans un graphe connexe n'a pas d'effet monotone sur le nombre moyen de rendez-vous.

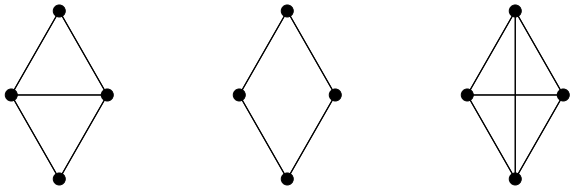


FIG. 7.1 – *Trois graphes connexes ayant le même ensemble de sommets.*

7.2.3 Impact de l'ajout d'un sommet

Cette section s'intéresse à l'impact de l'ajout d'un sommet à un graphe connexe sur le nombre moyen de rendez-vous dans un graphe connexe. Nous montrons que cet impact est positif, et plus exactement, nous avons :

Proposition 7.6 *Soit $G = (V, E)$ un graphe et G' un graphe obtenu à en rajoutant un sommet x et zéro ou plus d'arêtes entre x des sommets de G. Alors*

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

Preuve. Par récurrence sur $n = |V|$. Pour $n = 0$, la proposition est triviale. Supposons la proposition vraie pour tout graphe de taille inférieure à n. Soit G un graphe de taille n et soit e la première arête sur laquelle un rendez-vous a lieu dans G'. On a alors deux cas selon que x est adjacent ou non à e. Pour prouver la proposition, il suffit de montrer que l'inégalité est vérifiée conditionnée par les deux événements $\mathcal{E}_1$ et $\mathcal{E}_2$ suivants :

1. **Événement $\mathcal{E}_1$.** Soit $e = \{x, v\}$ pour un sommet $v \in V$. On a par ailleurs $\overline{R}(G'/\mathcal{E}_1) = \overline{R}(G'_e) + 1$ et, par l'hypothèse de récurrence : $\overline{R}(G'_e) \leq \overline{R}(G) \leq \overline{R}(G'_e) + 1$. Ce qui permet d'obtenir : $\overline{R}(G) \leq \overline{R}(G'/\mathcal{E}_1) \leq \overline{R}(G) + 1$.
2. **Événement $\mathcal{E}_2$.** Soit e une arête de G. D'après l'hypothèse de récurrence, on a $\overline{R}(G_e) \leq \overline{R}(G'_e) \leq \overline{R}(G_e) + 1$, conditionné par le choix de $e \in E$. 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 $\overline{R}(G'_e) + 1$, e variant dans E. Donc $\overline{R}(G) \leq \overline{R}(G'/\mathcal{E}_2) \leq \overline{R}(G) + 1$.

La proposition en résulte. □

7.2.4 Nombre de rendez-vous dans les graphes aléatoires

Considèrons un graphe aléatoire de degré moyen $c > 0$. Nous calculons d'abords la valeur asymptotique de la probabilité d'un rendez-vous sur une arête du graphe. Soit G un graphe aléatoire de taille n et de degré moyen c. Dans ce modèle, la probabilité que deux sommets différents soient liés par une arête est égale à $c/(n-1)$, ce qui permet d'obtenir un degré moyen égal à c. Soit $\{a, b\}$ une arête existante dans G. Le but

de cette section est d'établir une expression asymptotique de la probabilité $P(n,c)$ d'un rendez-vous sur $\{a,b\}$. Nous avons alors le théorème suivant :

Théorème 7.1 *La probabilité $P(n,c)$ qu'un rendez-vous ait lieu sur une arête existante $\{a,b\}$ dans un graphe aléatoire G de taille n, et de degré moyen égal à c tend vers $1/(1+c)$, quand $n \to \infty$.*

Soit $\mathcal{R}(n,c)$ le nombre moyen de rendez-vous dans le graphe aléatoire considéré dans cette section. Alors :

Proposition 7.7 *$\mathcal{R}(n,c)$ vaut asymptotiquement $(n/2)(1-1/(1+c))$, quand n tend vers ∞.*

Conclusion et perspéctives

Ce mémoire a présenté mes activités de recherche en algorithmique distribuée et en analyse des algorithmes distribués probabilistes. Plusieurs problèmes restent ouverts. Ce dernier chapitre présente quelques pistes de recherche qui méritent d'être explorées.

7.3 Autour de l'élection

Nous avons présenté un algorithme permettant de réaliser une élection équitable dans tout arbre de taille $n \geq 1$. Nous avons vu que l'algorithme peut être étendu à deux classes de graphes plus générales : les $k-$arbres et les polyominoïdes. Serait-il possible de trouver d'autres classes de graphes pour lesquelles l'algorithme produirait une élection équitable ? Nous pensons que les graphes triangulés constituent un bon candidat pour cette extension.

Nous avons également présenté une généralisation de l'algorithme pour guider l'élection de manière à obtenir une distribution de probabilité déterminée par avance et fonction d'un poids initial. Serait-il possible de réaliser toute distribution de probabilité ?

7.4 Autour des algorithmes sur les graphes

7.4.1 Autres problèmes

Nous avons étudié les trois problèmes suivants : le problème du MIS, le problème du coloriage, et le problème de la poignée de main. Nous avons présenté des algorithmes probabilistes permettant de les résoudre et nous avons étudié leurs complexités (en nombre de phases, et en bit complexité). D'autres problèmes sont de la même nature que ces trois problèmes :

- coloriage à distance 2 : un $2-$coloriage pour un graphe $G = (V, E)$ est une fonction c qui attribue à chaque sommet v de V une couleur $c(v)$ telle que,

  $$\forall (u, v) \in V \times V,\ \ d(u, v) \leq 2 \Rightarrow c(u) \neq v(v)$$

  nous avons montré qu'un coloriage de G peut se faire en moyenne en $O(\log n)$ phases et avec forte probabilité tout en n'échangeant que des messages de taille 1 bit. Une question simple est la suivante : peut-on faire de même pour réaliser un $2-$coloriage de G ?

– couverture maximale par des étoiles fermées : étant donné un graphe $G = (V, E)$, une couverture de G par des étoiles fermées est un sous-ensemble I de V tel que

$$\forall (u, v) \in I \times I,\ \ d(u, v) \geq 2$$

une couverture I est maximale si $\forall v \in V \setminus I$, $I \cup \{v\}$ n'est pas une couverture par des étoiles fermées.
Nous avons vu que pour le MIS, il existe un algorithme optimal permettant de construire un MIS, pour tout graphe de taille n, en $O(\log n)$ phases en moyenne et avec forte probabilité. Cet algorithme n'utilisant que des messages de taille 1 bit. Existe-il un algorithme permettant de calculer une couverture maximale en $O(\log n)$ phases ?

Ces deux problèmes acquiert une importance considérable quand on sait qu'il n'existe pas d'algorithme deterministe ou de type Las vegas permettant de détecter la coïncidence à distance ≥ 3 (voir (MSZ02)).

Un autre problème intéressant de la théorie des graphes est le problème de l'ensemble dominant minimal : un sous-ensemble S de V est dominant dans $G = (V, E)$ si $\forall v \in V \setminus S$, il existe $u \in S$ tel que $\{u, v\} \in E$. L'ensemble S est dominant si S est le plus petit sous-ensemble de V qui vérifie cette définition. Ce problème est assez proche, au moins dans son énnoncé, du problème du MIS. Existe-t-il un algorithme de complexité moyenne $O(\log n)$ permettant de construire un tel ensemble ?

7.4.2 Efficacité des algorithmes

Dans (MSZ02), nous avons étudié l'efficacité d'un algorithme à base de tirage de réels pour la construction en *une seule* phase d'un ensemble indépendant. Nous avons montré que l'efficacité de cet algorithme est $\geq 1/3$ si le graphe est un arbre mais que son efficacité peut être très médiocre dans le cas général. L'algorithme que nous avons présenté dans ce mémoire opère en $O(\log n)$ phases en moyenne et construit un MIS. Il serait alors inétressant de calculer l'efficacité de ce nouvel algorithme.

7.5 Autour de la diffusion

Dans (DHSZ04) et (DHSZ06), nous avons étudié le problème de la diffusion dans un réseau communiquant par handshakes. Nous avons montré que si les handshakes sont réalisés en utilisant la procédure étudiée dans (MSZ03), alors la complexité de la diffusion dans un graphe de taille n est $O(n\Delta)$.

Dans (EMR$^+$), nous avons étudié une procédure à base d'agendas dynamiques réalisant des handshakes. Nous avons montré que cette nouvelle procédure est plus efficace que celle de (MSZ03). Une piste de recherche consisterait à étudier la diffusion dans un graphe où les handshakes sont réalisés avec cette nouvelle procédure.

Annexe A. Curriculum Vitae

Nom patronymique : ZEMMARI
Prénom : Akka
Date et lieu de naissance : le 01 mars 1973 à Irklaouen (Maroc)
Nationalité : Française
Situation de famille : Marié, un enfant
Adresse personnelle : 9, rue des arbousiers, 33600 Pessac.
Téléphone : 05.56.90.13.48 (domicile) - 05.40.00.60.93 (professionnel)

Situation actuelle : Maître de Conférences en Informatique
Établissement : Université Montesquieu Bordeaux IV.

Titres universitaires (Université Bordeaux 1) :

1997 D.E.A. d'informatique, option Architectures et applications massivement parallèles. Mémoire sur le routage compact et adaptatif, sous la direction du professeur Cyril Gavoille.

2000 Thèse de Doctorat d'informatique, sous la direction d'Yves Métivier et Nasser Saheb : *"Contribution à l'analyse d'algorithmes distribués"*.

Activités professionnelles :

2000 - 2001 Attaché Temporaire d'Enseignement et de Recherche (ATER) en Informatique, Université Bordeaux 1.

2001- Maître de Conférences en Informatique, université Montesquieu Bordeaux IV.

Responsabilités administratives :

– Depuis septembre 2008, je suis directeur des études de la MIAGE de Bordeaux. J'ai en charge la participation à la définition du contenu pédagogique de la formation et la gestion de la coordination entre les responsables des trois années de formation (L3, M1 et M2) ainsi

que le représentation de la formation au sein des universités de tutelle : l'université Bordeaux 1 et l'université Montesquieu Bordeaux IV.
– Entre 2003 et 2007, j'ai été responsable du parcours informatique de gestion (ancienne MIAGE) de la Licence Informatique. J'ai eu pour charge de coordonner les enseignements entre les deux universités de tutelle : l'université Bordeaux 1 et l'université Montesquieu Bordeaux IV, de participer à la définition du contenu pédagogique et d'organiser les différents jurys.

Activités d'encadrement

Mémoires de Master 2 :

2004-2005 Shehla Abbas, *"Distributed Calculation of a Spanning Tree in a Dynamic Graph"*.

2006-2007 Jonathan Thiebault, *"Localité et coordination dans les réseaux de capteurs mobiles"*.

Thèses :

2005-2008 J'ai co-encadré avec le professeur Mohamed Mosbah, à hauteur de 50%, la thèse de Madame Shehla Abbas, thèse financée par une bourse de l'organisme SFERE et dont le titre est "Distributed Calculations using Mobile Agents". Cette thèse a débutée en septembre 2005 et à été soutenue le 15 décembre 2008.

Animation scientifique

– Membre du comité d'organisation des $9^{\text{ème}}$ rencontres francophones sur les aspects algorithmiques de télécommunications (AlgoTel 2007).
– Membre du comité d'organisation de la conférences DISC 2008.
– J'ai arbitré plusieurs articles : pour des conférences STACS, SPAA, SIROCCO, NOTERE, DISC, ... et pour des journaux TCS, JDA, ...

Annexe B. Liste des publications

Thèse :

[1] A. Zemmari. *Contribution à l'analyse d'algorithmes distribués.* Thèse de Doctorat de l'université Bordeux 1, 2000.

Articles acceptés dans des revues d'audience internationale avec comité de sélection :

[1] A. El Hibaoui, Y. Métivier, J.M. Robson, N. Saheb-Djahromi and A. Zemmari. Analysis of a Randomised Dynamic Timetable Handshake Algorithm. *Pure Mathematics and Applications (Algebra and Theoretical Computer Science)*. À paraître, 2009.

[2] B. Derbel, M. Mosbah and A. Zemmari. Sublinear Fully Distributed Partition with Applications. *Theory of Computing Systems*. À paraître, 2009.

[3] A. Zemmari. On Handshakes in Random Graphs. *Information Processing Letters*. 108 :119-123, 2008.

[4] Philipe Duchon, Nicolas Hanusse, N. Saheb and A. Zemmari. Broadcast in the Rendezvous Model. *Information and Computation*. 204 :697-712, 2006.

[5] Y. Métivier, N. Saheb and A. Zemmari. Locally Guided Randomized Elections in Trees : the Totally Fair Case. *Information and Computation*, 198 :40-55, 2005.

[6] Y. Métivier, N. Saheb and A. Zemmari. Analysis of a Randomized Rendez Vous Algorithm. *Information and Computation*, 184 :109-128, 2003.

[7] Cyril Gavoille and A. Zemmari. The compactness of adaptive routing tables. *Journal of Discrete Algorithms*, 1 :237-254, 2003.

[8] Y. Métivier, N. Saheb and A. Zemmari. Randomized Local Elections. *Information Processing Letters*, Volume 82, Issue 6.313-320, 2002.

Articles acceptés dans des conférences d'audience internationale avec comité de sélection et actes :

[9] Y. Métivier, J.M. Robson, N. Saheb-Djahromi, A. Zemmari. *An Optimal Bit Complexity Randomised Distributed MIS Algorithm*. In the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2009). May 25-27, 2009. Piran, Slovenia.

[10] J.-F. Marckert, N. Saheb, A. Zemmari.*Election Algorithms with Random Delays in Trees*. In the 21st International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2009).

[11] A. El Hibaoui, Y. Métivier, J.M. Robson, N. Saheb-Djahromi, A. Zemmari. *Analysis of a Randomized Dynamic Timetable Handshake Algorithm*. In GASCOM 2008. Bibbiena (Arezzo-ITALY). June 16-19, 2008.

[12] S. Abbas, M. Mosbah and A. Zemmari. *A Probabilistic Model for Distributed Merging of Mobile Agents*. In 2nd International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS 2008). Leeds, U.K. July 2-3, 2008.

[13] S. Abbas, M. Mosbah and A. Zemmari. *A Generic Distributed Algorithm for Computing by Random Mobile Agents*. In 10th Pacific Rim International Workshop On Multi-Agents (PRIMA'2007). Bangkok, Nov 21 - 23, 2007. Springer, Lecture Notes in Computer Science, volume 5044, pages 392-397.

[14] B. Hamid, M. Mosbah and A. Zemmari. *A Self-stabilizing Distributed Algorithm For Resolving Conflicts*. In 2nd International Workshop On Reliability in Decentralized Distributed Systems(RDDS'2007). Vilamoura, Algarve, Portugal, Nov 25 - 30, 2007. Springer, Lecture Notes in Computer Science, volume 4806, pages 1042-1051.

[15] S. Abbas, M. Mosbah and A. Zemmari. *Merging Time of Random Mobile Agents*. In (LDIC'2007). Bremen, Allemagne, August 28th - 30th, 2007. Procedings of LDIC. Pages 179-189, Springer Verlag.

[16] S. Abbas, M. Mosbah and A. Zemmari. *Collecte d'informations par des agents mobiles*. In New Technologies for Distributed Systems (NOTERE'2007). Marrakech, Maroc, Juin 4-8, 2007. To appear in Hermes.

[17] M. Mosbah, A. Sellami and A. Zemmari. *Using Graph Relabelling Systems for Resolving Conflicts*. In New Technologies for Distributed Systems (NOTERE'2006). 6-9 juin, 2006 Toulouse, France. Pages 283-294. Edition Lavoisier, Hermes.

[18] S. Abbas, M. Mosbah and A. Zemmari. *Distributed Computation of a Spanning Tree in a Dynamic Graph by Mobile Agents*. In IEEE International Conference on Engineering of Intelligent Systems (ICEIS 2006). April 22-23, 2006. Islamabad, Pakistan.

[19] B. Derbel, M. Mosbah and A. Zemmari. *Fast Distributed Graph Partition and Application*. In 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006). April 25, 2006. Rhodes Island, Greece.

[20] A. El Hibaoui, N. Saheb-Djahromi and A. Zemmari. *Polyominoids and Uniform Election*. In 17th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2005). June 20, 2005. Taormina, Sicily, Italy.

[21] A. El Hibaoui, N. Saheb-Djahromi and A. Zemmari. *A Uniform Probabilistic Election Algorithm in k-Trees*. In 17th IMACS World Congress : Scientific Computation, Applied Mathematics and Simulation (IMACS 2005). July 11 - 15, 2005. Paris, France ;

[22] P. Duchon, N. Hanusse, N. Saheb and A. Zemmari. *Broadcast in the Rendezvous Model*. In 21st International Symposium on Theoretical Aspects of Computer Science (STACS 2004). March 25-27, 2004. Montpellier, France.

[23] P. Duchon, N. Hanusse, N. Saheb and A. Zemmari. *Broadcast Using Rendezvous (Extended Abstract)*. Breif announcement In 17th International Symposium on Distributed Computing (DISC 2003). October 1-3, 2003. Sorrento, Italy.

[24] Y. Métivier, N. Saheb and A. Zemmari. *A Uniform Randomized Election in Trees (Extended Abstract)*. In 10th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2003). Carleton University Press, June 2003.

[25] Y. Métivier, N. Saheb and A. Zemmari. *Uniform Randomized Election in Trees*. In GASCOM'2001. November 2001. Certosa di Pontignano (Sienne), Italie.

[26] N. Saheb and A. Zemmari. *Methods for Computing the Concurrency Degree in Commutation Monoids*. In 12-th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2000), pages 731-742. Springer, June 2000. Moscow, Russie.

[27] C. Gavoille and A. Zemmari. *The Compactness of Adaptive Routing Tables*. In 7-th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000), pages 127-140. Carleton Scientific, June 2000. Aquila, Italie.

[28] Y. Métivier, N. Saheb and A. Zemmari. *Randomized Rendezvous. In Colloquium on Mathematics and Computer Science*. Trends in Mathematics series, pages 183-194. Birkhauser, September 2000. Paris, France.

Articles soumis :

[29] Y. Metivier, J.- M. Robson, N. Saheb-Djahromi, A. Zemmari. *An analysis of an optimal bit complexity randomised distributed vertex colouring algorithm*. Soumis.

[30] A. El Hibaoui, N. Saheb-Djahromi, J.- M. Robson, A. Zemmari. *Uniform Election in Polyominoids and Trees*. Soumis.

[31] S. Abbas, M. Mosbah, A. Zemmari.*Merging Mobile Agents : a Random Walk Approach*. Soumis.

Bibliographie

[ABI86] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms, 7 (1986), pp. 567–583. (Cité page 34.)

[AD] H. Austinat and V. Diekert, communication privée. (Cité page 59.)

[AGLP89] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin, *Network decomposition and locality in distributed computation*, in FOCS, IEEE, 1989, pp. 364–369. (Cité page 34.)

[Ang80] D. Angluin, *Local and global properties in networks of processors*, in Proceedings of the 12th Symposium on theory of computing, 1980, pp. 82–93. (Cité page 4.)

[Ber85] C. Berge, *Graphs and Hypergraphs*, North-Holland, 1985. (Cité page 3.)

[CFV01] J. Clément, P. Flajolet, and B. Vallée, *Dynamical sources in information theory : A general analysis of trie structures*, Algorithmica, 29 (2001), p. 307369. (Cité page 54.)

[DHSZ04] P. Duchon, N. Hanusse, N. Saheb, and A. Zemmari, *Broadcast in the rendezvous model*, in STACS, V. Diekert and M. Habib, eds., vol. 2996 of Lecture Notes in Computer Science, Springer, 2004, pp. 559–570. (Cité page 82.)

[DHSZ06] P. Duchon, N. Hanusse, N. Saheb, and A. Zemmari, *Broadcast in the rendezvous model*, Inf. Comput., 204 (2006), pp. 697–712. (Cité page 82.)

[Die02] M. Dietzfelbinger, *The probability of a rendezvous is minimal in complete graphs*, in ISAAC, P. Bose and P. Morin, eds., vol. 2518 of Lecture Notes in Computer Science, Springer, 2002, pp. 55–66. (Cité pages 60 et 70.)

[DT05] M. Dietzfelbinger and H. Tamaki, *On the probability of rendezvous in graphs*, Random Struct. Algorithms, 26 (2005), pp. 266–288. (Cité pages 60 et 70.)

[EMR+] A. El Hibaoui, Y. Métivier, J. Robson, N. Saheb-Djahromi, and A. Zemmari, *Analysis of a randomized dynamic timetable handshake algorithm*, Pure Mathematics and Applications (PuMA). à paraître. (Cité page 82.)

[ERSDZ] A. El Hibaoui, J. Robson, N. Saheb-Djahromi, and A. Zemmari, *Uniform election in polyominoids and trees*. soumis à Discrete Applied Mathematics. (Cité pages 13, 22 et 25.)

[ESDZ05a] A. El Hibaoui, N. Saheb-Djahromi, and A. Zemmari, *Polyominoids and uniform election*, in 17th Formal Power Series and Algebraic Combinatorics (FPSAC), 2005. (Cité pages 13 et 25.)

[ESDZ05b] A. El Hibaoui, N. Saheb-Djahromi, and A. Zemmari, *A uniform probabilistic election algorithm in k-trees*, in 17th IMACS World Congress : Scientific Computation, Applied Mathematics and Simulation (IMACS), 2005. (Cité pages 13, 25 et 32.)

[Fel60] W. Feller, *An Introduction to Probability Theory and Its Application, Volume I. 2nd edition*, John Wiley and Sons, 1960. (Cité page 17.)

[Fel66] W. Feller, *An Introduction to Probability Theory and Its Application, Volume II 2nd edition*, John Wiley and Sons, 1966. (Cité page 21.)

[Fla97] P. Flajolet, *A seating arrangement problem*, tech. rep., ALCOM-FT, 1997. (Cité page 78.)

[FS] P. Flajolet and R. Sedgewick, *The average case analysis of algorithms : Mellin transform asymptotics*, Tech. Rep. RR-2956. (Cité pages 49 et 50.)

[HMRAR98] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, eds., *Probabilistic methods for algorithmic discrete mathematics*, vol. 16 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 1998. (Cité page 8.)

[Joh99] O. Johanson, *Simple distributed $(\Delta+1)$-coloring of graphs*, Information Processing Letters, 70 (1999), pp. 229–232. (Cité page 54.)

[KMNW05] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, *Fast deterministic distributed maximal independent set computation on growth-bounded graphs.*, in DISC, 2005, pp. 273–287. (Cité page 34.)

[KOSS06] K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer, *Distributed coloring in $O(\sqrt{\log n})$ bit rounds*, in 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece, IEEE, 2006. (Cité page 46.)

[KSOS06] K. Kothapalli, C. Scheideler, M. Onus, and C. Schindelhauer, *Distributed coloring in o/spl tilde/(/spl radic/(log n)) bit rounds*, in IPDPS, IEEE, 2006. (Cité page 42.)

[KW84] R. M. Karp and A. Widgerson, *A fast parallel algorithm for the maximal independent set problem*, in Proceedings of the 16th ACM Symposium on Theory of computing (STOC), ACM Press, 1984, pp. 266–272. (Cité page 34.)

[Lav95] C. Lavault, *Évaluation des algorithmes distribués*, Hermes, 1995. (Cité pages 3, 4, 5 et 6.)

[Le 77] G. Le Lann, *Distributed systems - towards a formal approach*, in IFIP Congress, 1977, pp. 155–160. (Cité page 13.)

[Lin92] N. Linial, *Locality in distributed graph algorithms*, SIAM J. Comput., 21 (1992), pp. 193–201. (Cité page 34.)

[LMZ95] I. Litovsky, Y. Métivier, and W. Zielonka, *On the recognition of families of graphs with local computations*, Inf. Comput., 118 (1995), pp. 110–119. (Cité page 27.)

[Loè55] M. Loève, *Probability Theory*, D. Van Nostrand Company, Inc., 1955. (Cité page 65.)

[Lub86] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053. (Cité page 34.)

[Lyn96] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996. (Cité pages 3, 4, 34 et 43.)

[MRSDZ] Y. Métivier, J. Robson, N. Saheb-Djahromi, and A. Zemmari, *An optimal bit complexity randomised distributed vertex colouring algorithm*. soumis. (Cité page 47.)

[MRSDZ09] Y. Métivier, J.-M. Robson, N. Saheb-Djahromi, and A. Zemmari, *A bit complexity efficient randomized distributed mis algorithm*, in 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2009), 2009. (Cité pages 37, 38, 42 et 43.)

[MS94] Y. Métivier and N. Saheb, *Probabilistic analysis of an election algorithm in a tree*, in CAAP, S. Tison, ed., vol. 787 of Lecture Notes in Computer Science, Springer, 1994, pp. 234–245. (Cité pages 13, 14, 16 et 18.)

[MS97a] Y. Métivier and E. Sopena, *Graph relabelling systems : A general overview*, Computers and Artificial Intelligence, 16 (1997). (Cité page 14.)

[MS97b] Y. Métivier and E. Sopena, *Graph relabelling systems : A general overview*, Computers and Artificial Intelligence, 16 (1997). (Cité page 55.)

[MSDZ03] Y. Métivier, N. Saheb-Djahromi, and A. Zemmari, *A uniform randomized election in trees*, in SIROCCO, J. F. Sibeyn, ed., vol. 17 of Proceedings in Informatics, Carleton Scientific, 2003, pp. 259–274. (Cité pages 13 et 15.)

[MSDZ05] Y. Métivier, N. Saheb-Djahromi, and A. Zemmari, *Locally guided randomized elections in trees : The totally fair case*, Inf. Comput., 198 (2005), pp. 40–55. (Cité pages 13, 15 et 22.)

[MSDZ09] J.-F. Marckert, N. Saheb-Djahromi, and A. Zemmari, *Election algorithms with random delays in trees*, in 21st Formal Power Series and Algebraic Combinatorics (FPSAC), 2009. à paraître. (Cité pages 13 et 24.)

[MSZ00] Y. Métivier, N. Saheb, and A. Zemmari, *Randomized rendezvous*, in Mathematics and computer science : Algorithms, trees, combinatorics and probabilities, Trends in mathematics, 2000, pp. 183–194. (Cité page 56.)

[MSZ02] Y. Métivier, N. Saheb, and A. Zemmari, *Randomized local elections*, Inf. Process. Lett., 82 (2002), pp. 313–320. (Cité pages 10, 33, 43, 56 et 82.)

[MSZ03] Y. Métivier, N. Saheb, and A. Zemmari, *Analysis of a randomized rendezvous algorithm*, Inf. Comput., 184 (2003), pp. 109–128. (Cité pages 10, 56 et 82.)

[MW05] T. Moscibroda and R. Wattenhofer, *Maximal independent set in radio networks*, in Proceedings of the 25 Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, 2005, pp. 148–157. (Cité page 34.)

[Pel00] D. Peleg, *Distributed computing : a locality-sensitive approach*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. (Cité page 34.)

[Roooo] *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, 2000. (Cité pages 3, 52, 70, 72 et 73.)

[SF96] R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*, Addison-Wesley Publishing Company, 1996. 512 pages. (ISBN 0-201-40009-X). (Cité page 3.)

[Teloo] G. Tel, *Introduction to distributed algorithms*, Cambridge University Press, 2000. (Cité pages 3, 4, 5 et 75.)

[Wato7] R. Wattenhofer, *http ://dcg.ethz.ch/lectures/fso8/distcomp/lecture/chapter4.pdf.*, 2007. (Cité pages 42 et 43.)

[Wil93] D. Williams, *Probability with Martingals*, Cambridge, 1993. (Cité page 7.)

[Zemo8] A. Zemmari, *Handshakes in random graphs*, Inf. Process. Lett., 108 (2008), pp. 119–123. (Cité page 56.)

Ce document a été préparé à l'aide de l'éditeur de texte GNU Emacs et du logiciel de composition typographique LaTeX 2ε.