Deliverable 5
Probabilistic incremental proof-based development

31 janvier 2010 à 11:58 A.M.

Projet RIMEL

ANR-06-SETI-015

Equipe MOSEL, LORIA, Université Henri Poincaré Nancy 1
ClearSy
LABRI, Université de Bordeaux & CNRS
http://rimel.loria.fr
Années 2007-2008-2009

Avertissement
Ce document est rédigé par Dominique Méry avec les contributions de Akka, Mohammed....

Contents

List of Figures

Chapter 1

Présentation générale

Sommaire

1.1 Introduction

Le développement incrémental dirigé par la preuve de systèmes informatiques vise à mettre en œuvre la démarche de correction par construction (correct-by-construction)[20] dans le cadre de systèmes qui sont à logiciels prépondérants. Cette construction repose sur l'écriture de modèles événementiels dans le langage de modélisation EVENT B et la progression de la construction est fondée sur le raffinement qui permet de préciser de plus en plus ce qui sera un modèle final. Cette démarche progressive est validée par la relation de raffinement mais aussi par la preuve mathématique de conditions de vérification permettant de garantir que le modèle concret satisfait les mêmes propriétés que le modèle abstrait. La démarche générale consiste donc à expliciter un modèle abstrait qui répond bien au problème posé et que le concepteur va progressivement modifier pour le rendre de plus en plus proche de la solution finale visée. Cette solution finale peut être un algorithme séquentiel, un algorithme réparti ou un système à logiciel prépondérant. Si l'histoire commence ainsi pour ce livrable, le contexte est différent pour la suite.
Ce rapport dresse le bilan du travail sur les problèmes posés par l'intégration d'hypothèses probabilistes dans un développement fondé sur le raffinement. Il apporte quelques points de progrès mais un travail encore important reste à mener et ces progrès sont possibles dans la mesure où l'expression du temps dans le développement incrémental est réaliste et réalisé notamment dans la tâche 2 et le livrable 2[5] de ce projet. La thèse [34] de Joris Rehm apporte des éléments qu'il faudra sans doute développer et qui sont déjà très largement utilisés par la communauté EVENT B . Le point de départ de ce travail est le développement [3, 11, 14] prouvé et incrémental de l'algorithme d'élection du leader utilisé dans le standard IEEE 1394 parfois appelé FIREWIRE. Le développement de l'algorithme conduit à proposer la résolution de la contention qui constitue une configuration possible de non-terminaison de cet algorithme. En effet, l'utilisation de model checkers comme UPPAAL conduit à exhiber une suite infinie de configurations de cette sorte. Cette suite infinie ayant une probabilité infime d'arriver, cela nous conduit à conclure à la terminaison de l'algorithme et à l'élection du leader. Notre travail a été motivé par les travaux de Caroll Morgan et Annabel McIver [24] qui ont été instanciés dans le cadre du formalisme EVENT B lors de la parution de l'article [29]. Poursuivant ce travail, Stefan Hallestede et Thai Son Hoang [13] proposent une substitution probabiliste dans le cadre de EVENT B , en conservant la philosophie de développement de ce formalisme par préservation des concepts passés.
Une des idées développées dans le cadre de ce projet est celle de patron de conception prouvé et cette idée se fonde sur des observations faites au cours des études de cas réalisées pour construire des modèles de systèmes. Jean-Raymond Abrial[2] a souligné l'intérêt d'une telle démarche capable de capitaliser des preuves parfois difficiles mais qui sont réutilisables dans d'autres dévelopements.
Pour ce livrable, nous faisons le point sur les travaux menés par les collègues de Bordeaux sur les algorithmes probabilistes et sur quelques solutions apportées pour prendre en compte les probabilités dans le développement incrémental et prouvé. Ce document apporte beaucoup de questions qui ne sont pas encore complètement résolues et nécessitera d'être poursuivi dans un groupe associant les spécialistes du LABRI pour les aspects *algorithmique probablistique* et les spécialistes de EVENT B pour les aspects EVENT B . Notre travail a consisté à analyser les algorithmes probabilistes et à poursuivre l'exemple de l'élection du leader pour prendre en compte les probabilités dans le développement. La tâche a permis aussi de considérer quelques exemples de tels algorithmes afin de mesurer les besoins.

1.2 Intégration des aspects probabilistes dans le développement incrémental prouvé EVENT B

1.2.1 Challenge du B probabiliste

Le calcul de raffinement de Back [8, 7, 9] repose sur la sémantique par transformateurs de prédicats. La relation de raffinement $a \sqsubseteq b$ est interprétée par l'expression logique suivante: $\forall \varphi . \varphi \in \mathcal{L} \Rightarrow [a]\varphi \Rightarrow [b]\varphi$ où $[c]\varphi$ est la *weakest precondition* de c par rapport à φ. Le travail de Caroll Morgan et Annabelle McIver [24] s'appuie en premier lieu sur une extension du calcul des plus faibles préconditions dans un cadre probabiliste. Dans le cas du calcul des préconditions classiques, le transformateur de prédicats exprime un programme comme une transformation d'une *postcondition* en une *précondition* suffisante pour atteindre ces *postconditions*. L'idée est d'utiliser une notion plus générale qu'une condition comme une situation qui apporte des éléments permettant d'atteindre un état ou des états après l'exécution d'un programme.

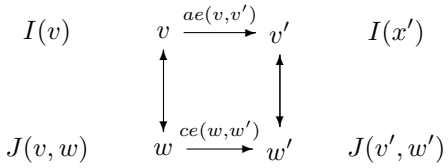
Le terme anglais utilisé est une *pre-expectation*. Ainsi, le langage des probabilités est introduit dans le raisonnement sur les actions et les programmes en considérant une transformateur de prédicats appliqués à ces cas. On parlera de transformateurs de prédicats probabilistes. Puisque la méthode B est fondée sur le calcul des transformateurs de prédicats, il est donc assez immédiat d'appliquer cette extension pour traiter les programmes probabilistes dans un cadre EVENT B adapté intégrant cette extension. Le challenge [29] du EVENT B probabiliste est donc posé. Comme nous l'avons souligné auparavant, les études de cas sont importantes et Morgan et al [29] considère l'algorithme probabiliste d'exclusion mutuelle de Rabin [33] comme illustration des éléments du problème posé par ces algorithmes. Cet algorithme fonctionne comme suit; il utilise trois variables partagées: un sémaphore assure la propriété d'exclusion mutuelle, un nombre loterie résout la question de compétition et une autre variable résout les questions liées à la compétition. Le nombre loterie suit une loi de Bernouilli et de ce fait la probabilité de choisir k est $1/2^k$ et a la propriété que la probabilité d'avoir plus d'un processus ayant choisi le maximum est bornée par une constante. Finalement, la probabilité d'entrer en section critique est de $1/N$ où N est le nombre de processus en compétition. Cet exemple est décrit dans le cadre d'une interprétation probabiliste du langage EVENT B et le raffinement est analysé dans ce cadre. Il s'agit de poser le problème du développement d'un tel algorithme dans le cadre de EVENT B .

1.2.2 Vers un EVENT B probabiliste

A la suite de ce travail, Hallestede et Hoang [13] ont envisagé de prendre en compte la modélisation probabilsite dans le langage EVENT B en veillant à faire une extension conservative du cadre EVENT B . Ce travail introduit les conditions de vérification propres à la notion de *terminaison presque certain*. Le point délicat est d'exprimer dans une condition de vérification, le fait que la convergence d'un événement nouveau est dominée par une valeur donnée. Tout nouvel événement ne doit pas augmenter la borne supérieure bornant le système en développement. Enfin, les choix possibles pour faire décroître une variable associée à la convergence du système doivent être en nombre fini.
Pour résumer les conditions de vérifications ajoutées, nous considérons un événement ea raffiné par un événement ca.

EVENT ea
ANY t
$G(t, v)$ WHERE
$S(t, v)$
END

EVENT ca
ANY t
$H(t, v)$ WHERE
$T(t, v)$
END



La condition de vérification classique est:

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v, w) \ \wedge \ BA(ce)(w, w') \ \Rightarrow \ \exists v'.(BA(ae)(v, v') \ \wedge \ \mathrm{J}(v', w'))$$

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v, w) \ \wedge \ H(u, v) \ \wedge \ T(u, w, w') \ \Rightarrow \ \exists t, v'.(G(t, v) \ \wedge \ S(t, v, v') \ \wedge \ \mathrm{J}(v', w'))$$

Un nouvel événement est supposé raffiner *skip* mais aussi ne pas produire une divergence dans le nouveau modèle et donc le variant V doit être borné.

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v, w) \ \wedge \ H(u, v)/ \Rightarrow \ V(w) \in \mathbb{N}$$

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v,w) \ \wedge \ H(u,v) \ \wedge \ T(u,w,w') \ \Rightarrow \ V(w') < V(w)$$

Les deux conditions ci-dessus constituent des conditions de convergence pour les nouveaux événements et cette convergence doit être assurée aussi pour le cas des modèles probababilistes EVENT B . Une première observation conduit à noter qu'une action probabiliste a le même effet par rapport à la préservation de l'invariant: le comportement est démoniaque. Pour ce qui est de la preuve de décroissance de V, la condition de vérification est exprimée comme suit:
L'action $T(u,w,w')$ peut décroître le variant $V(w)$ dans le sens angélique:

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v,w) \ \wedge \ H(u,v) \ \Rightarrow \ (\exists w'.T(u,w,w') \ \wedge \ V(w') < V(w))$$

Pour la convergence presque certaine, les auteurs imposent que le variant soit borné pour tous les événements, par une valeur $U(w)$:

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v,w) \ \wedge \ H(u,v) \ \Rightarrow \ (V(w) < U(w))$$

Cette borne doit aussi décroître pour tous les événements de la machine concrète:

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v,w) \ \wedge \ H(u,v) \ \wedge \ T(u,w,w') \ \Rightarrow \ (U(w') < U(w))$$

Enfin, la probabilité minimale des actions satisfaisant les conditions précédentes ne doit pas être nulle et il est imposé la finitude des choix possibles pour w':

$$\mathrm{I}(v) \ \wedge \ \mathrm{J}(v,w) \ \wedge \ H(u,v) \ \Rightarrow \ finite(\{w'|T(u,w,w')\})$$

Cette dernière condition est rencontrée dans tous les exemples traités, notamment l'élection du leader où *in fine* seuls deux processus sont en compétition pour le leadership. L'idée est de proposer un traitement le plus simple possible par rapport au traitement du raffinement probabiliste dans un contexte EVENT B par une simple adaptation des conditions énoncées précédemment et si on peut introduire des événements probabilistes qui raffinent des événements non-déterministes, il devient complexe d'appliquer le raffinement probabiliste tel qu'il est défini par C. Morgan sans conduire à des modifications importantes de EVENT B et des outils.

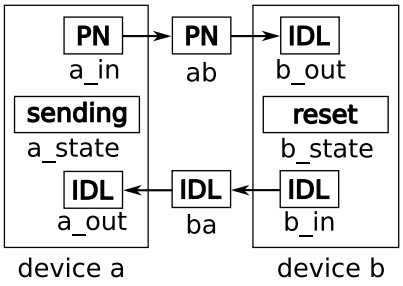
1.2.3 Analyse d'une solution

La question de la prise en compte des algorithmes probabilistes provient de l'étude [3] de cas du développement correct incrémental du protocole d'élection du leader de l'IEEE 1394. Dans cette étude, il est clair que le modèle final satisfaisait une propriété de sûreté sur la progression possible des calculs et qu'il indiquait qu'une forêt convergeait vers un arbre mais que ces calculs pouvaient conduire à une situation dite de *contention*. La contention est l'état connu par deux processus voisins qui ont demandé à l'autre d'être le leader et qui ont fait cette demande dans un temps conduisant à ce que les deux soient demandeurs et demandés. Une solution pourrait de choisir le processus de numéro le plus petit ou le plus grand mais ce serait de supposer que des numéros existent et ce n'est pas toujours possible. Une autre solution est faire attendre chaque processus un temps *court* ou *long* avant de redemander. Dans la mesure où le choix est différent si on attend suffisamment, cela signifie qu'ils peuvent être dans une situation d'attente *long* pour l'un et *cours* pour l'autre. Bien sûr, une modélisation en UPPAAL a conduit à exhiber une trace infinie ne résolvant pas le problème mais il s'agit d'une trace *peu probable*. La situation de *contention* est modélisée par les gardes des événements exprimant que x envoie une demande à y à condition que y n'a pas déjà demandé à x mais

il se peut que la même action soit exécutée par y à l'intention de x et donc ils se retrouvent alors tous les deux en demande. L'idée est alors de changer de problème. En effet, le nouveau problème est de résoudre le problème de l'élection du leader pour deux sites. Cette solution a été prise par J. Rehm dans sa thèse [34] et dans le livrable 2 [5] mais pour introduire la notion de temps dans le développement. Le premier événement du nouveau dppement est simplement l'événement de choix d'un des deux nœuds comme leader:

EVENT accept
ANY x
WHERE
$x \in N$
$leader = \varnothing$
THEN
$leader := \{x\}$
END

Les communications entre les deux processus se font via des canaux et on a plusieurs étapes avant d'avoir une communication effective:



Puis, on peut justifier l'introduction de deux temps pour résoudre la question de la symétrie:

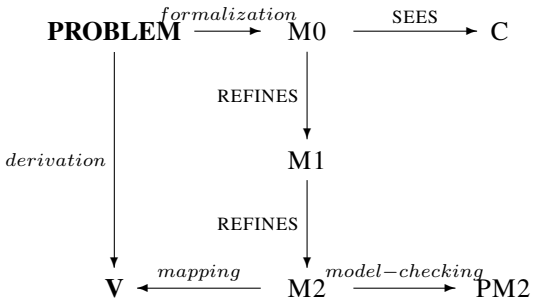
- Toutes les conditions abstraites présentes dans les gardes sont remplacées par l'utilisation des deux délais.

- Deux nouvelles constantes : st (*short time*) et lt (*long time*) qui sont des nombres entiers non nuls. Pour que le protocole se déroule correctement, leurs valeurs doivent respecter: $st \geq prop \times 2$ et $lt \geq prop \times 2 + st - 1$. On étend l'utilisation du patron par deux nouvelles valeurs pour l'ensemble ACT avec a_awake, b_awake. Comme précédemment nous définissons : $at\_a\_awake = at(a\_awake)$ et $at\_b\_awake = at(b\_awake)$. Enfin, les deux variables supplémentaires a_sleept et b_sleept permettent de noter le délai choisi par chaque appareil.

- une situation typique de contention (avec les valeurs $prop = 3$, $st = 6$ and $lt = 11$).

- L'élection sera un succès si les délais choisis sont différents.

Les événements sont modifiés et prennent en compte les temps introduits. Il est alors clair que la phase importante pour résoudre la question de la convergence est d'introduire le temps dans le système. En effet, il paraît difficile de traiter ce protocole sans gérer le temps. Enfin, le traitement des aspects probabilistes peut alors être géré par le codage des automates produits pour des paires de sites voisins et on peut utiliser un *model checker* probabiliste pour analyser la situation devenue traitable localement. Nous donnons le schéma permettant de montrer le processus de raffinement pour la résolution du problème de l'élection du leader en intégrant dans le raffinement les aspects liés au temps et en définissant une étape complémentaire pour appliquer le model checking probabiliste sur le système constitué de deux nœuds quelconques voisins et en contention.



1.3 Etudes de cas

Les travaux menés sur les algorithmes probabilistes visaient d'une part à étudier les problèmes résolus par ces algorithmes et d'autre part à acquérir un savoir-faire sur les types d'algorithmes probabilistes. L'objectif reste d'intégrer des aspects probabilistes dans le développement incrémental EVENT B . Deux problèmes sont développés dans les chapitres suivants.

1.3.1 Construction MIS

Cet algorithme permet de construire un MIS en utilisant un échange de messages de taille 1 bit. Il 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 [28], nous avons étudié un algorithme à base de tirage de réels permettant de résoudre ce problème. Néaumoins, sa bit complexité reste non bornée.

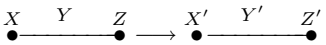
1.3.2 Coloriage de graphe

Dans ce cas, 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 [15]. Il serait, par conséquent intéressant d'étudier la possibilité de réduire, de manière significative, le nombre de couleurs utilisées.

1.4 Conclusion

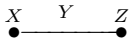
Les travaux menés dans cette tâche se focalisent sur l'intégration des éléments probabilistes du raisonnement effectué quand on conçoit un algorithme réparti où des situations qui peuvent être atteintes requièrent des arguments probabilistes pour conclure. L'exemple de l'élection du leader IEEE 1394 conduit à une configuration dite de contention dans laquelle un raisonnement localisé sur les deux sites ou processus en contention est nécessairement probabiliste pour justifier de la terminaison probabiliste de l'élection. Cependant, cette configuration s'appuie sur un modèle correct par rapport à une chronologie introduite dans le raffinement. Cette chronologie conduit à des modèles qui peuvent être étudiés par rapport à la question probabiliste et dans cet ordre uniquement. On obtient ainsi un traitement complet du développement de l'algorithme d'élection du leader en intégrant les travaux initiaux [3] et ceux sur le temps [34, 5]. Quant à considérer les algorithmes MIS et de coloriage, il reste à analyser précisèment l'intégration des probabilités dans le modèles des calculs locaux mais il s'agit là d'un point futur. Initialement, les travaux ont montré que le modèle des calculs locaux pouvait être très facilement codé dans EVENT B :

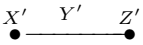
- Règle de ré-étiquettage

  with $X' = f_1(X, Y, Z)$, $Y' = f_2(X, Y, Z)$ and $Z' = f_3(X, Y, Z)$.

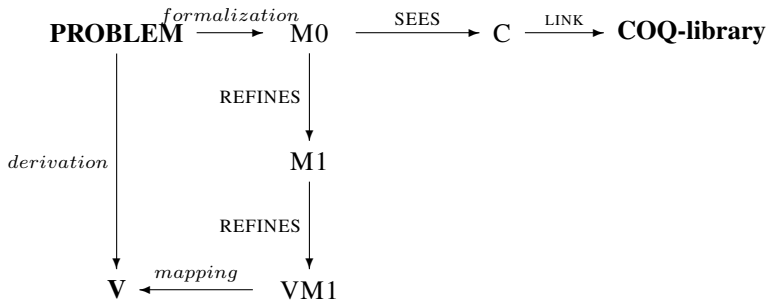
- Traduction en Event B

  WHEN
  

  THEN
  

  $X, Y, Z : |(X' = f_1(X, Y, Z) \wedge Y' = f_2(X, Y, Z) \wedge Z' = f_3(X, Y, Z))$
  END

Cela conduit à un schéma de ce type qui se généralise pour l'intégration des aspects probabilistes dans le schéma suivant:



Les travaux futurs pourraient s'articuler selon les directions suivantes:

- Etude des modèles locaux par rapport au raffinement probabiliste.
- Développement d'études de cas, en illustrant le plongement du modèle des calculs locaux dans EVENT B .
- Intégration logicielle du model checking probabiliste dans EVENT B par interaction avec PRISM [32], par exemple.

Chapter 2

Calcul distribué d'un MIS

Ce chapitre présente le travail réalisé autour du problème de l'ensemble indépendant maximal (MIS) et publié dans [27].

2.1 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 [23] (chapitre 4, p. 71-76) et dans [31] (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 [4, 22, 6, 21]. Karp et Widgerson [16] ont prouvé que calculer un MIS est NC. Kuhn et al. dans [19] ont étudié le problème pour des classes particulières de graphes et Moscibroda et al. dans [30] 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. *Dans la colonne "Connaissance préalable", la "Taille du graphe" correspond à la taille initiale du graphe et le "Degré maximum des sommets voisins" et le "Degrée maximum dans le* 2*-voisinage" sont des connaissances calculées au début de chaque phase.*

2.2 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 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 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 $\{u, v\} \cup \{w$ voisins de u ou $v\}$.
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 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 $1 - o(n^{-1})$.*

Enfin :

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

2.3 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.

2.3.1 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 1). 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 : $\eta(v) = -1$; $active\text{-}set(v)$, $Not\text{-}In\text{-}MIS\text{-}set(v)$ et $In\text{-}MIS\text{-}set(v)$ sont des ensembles de sommets. Au début de chaque phase, pour chaque sommet v tel que $\eta(v) = -1$: $status(v) = Eligible$, $active\text{-}set(v)$ contiennent l'ensemble des voisins w de v satisfaisant $\eta(w) = -1$, $Not\text{-}In\text{-}MIS\text{-}set(v) = \varnothing$, $In\text{-}MIS\text{-}set(v) = \varnothing$ et $Ineligible\text{-}set(v) = \varnothing$.
Pour plus d'explications sur le fonctionnement et le déroulement de l'algorithme, le lecteur peut consulter [27].

2.3.2 Analyse de l'algorithme

On commence par le lemme suivant:

Lemme 2. *À la fin de chaque phase, chaque processeur a une probabilité supérieure à* $1/4$ *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 [27] pour la preuve complète. □

On obtient alors le corollaire suivant :

Algorithm 1: 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 $Not-In-MIS-Set()$;

Corollaire 2. *Il existe deux constantes k_2 et K_2 telles que le nombre maximum de bits $DONNEES$ 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. *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 2. *L'algorithme $\mathcal{B}$ calcule un MIS pour tout graphe de taille n en $O(\log^2 n)$ avec la probabilité $1 - o(n^{-1})$.*

2.4 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. *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.

Algorithm 2: 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 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

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

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 :

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

□

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 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 [18], 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 [35] 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.

2.5 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 [27] 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 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 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, [27], 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 [27], nous avons également étudié cette indépendance pour les algorithmes de Luby et les présentations qu'en font Lynch dans [23] et Wattenhofer dans [35].

2.6 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 [28], nous avons étudié un algorithme à base de tirage de réels permettant de résoudre ce problème. Néaumoins, sa bit complexité reste non bornée.

Chapter 3

Coloriage de graphe

Ce chapitre présente un algorithme probabiliste simple pour réaliser une coloration d'un graphe quelconque. Ce travail a donné lieu à la publication de l'article [25].

3.1 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 [17] 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.

3.2 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 $couleur_u$.

Algorithm 3: L'algorithme FS_Color.

var:
  $couleur_v$: mot **Init** $mot\text{-}vide$;
  $actifs_v$: $\subseteq N(v)$ **Init** $N(v)$;
  b_v: $\in \{0,1\}\}$;
Tant que $actifs_v \neq \varnothing$ **faire**
  $b_v \leftarrow flip(0,1)$;
  $couleur_v \leftarrow b_v \oplus couleur_v$;
  Pour tout $u \in actifs_v$ **Faire**
    envoyer b_v à u;
    recevoir b_u de u;
    Si $b_v \neq b_u$**alors**
      $active_v \leftarrow active_v \setminus \{u\}$;
    Fin Si
  Fin Pour
Fin Tant que

Remarque 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.*

3.3 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 4. *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 le corollaire suivant :

Corollaire 4. *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 1. (Voir [26] pour plus de détails). □

On en déduit alors le théorème principal :

Théorème 4. *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.*

3.3.1 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. *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.
- *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 $\mathbb{E}(B_G) \leq \frac{8n}{3}$.*

3.3.2 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 2. *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 [12]). 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 6. *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 6. *Soit $d \geq 1$, nous avons :*

- $\mathbb{P}r\left(L_d = 0\right) = 0$, *et*
- $\mathbb{P}r\left(L_d = k\right) = \left(1 - \frac{1}{2^k}\right)^d - \left(1 - \frac{1}{2^{k-1}}\right)^d$, *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 7. *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 définie dans le lemme 2.

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 :

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

Donc, en utilisant le "reverse mapping theorem" de [12], il vient :

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

où $\Delta(u) = -\frac{2}{(\log 2)^2}\sum_{k\in\mathbb{Z}\setminus\{0\}}\Gamma(\chi_k)\,e^{-2ik\pi u}$.
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 3. *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 4 :

Corollaire 7. *Soit v un sommet et supposons que son degré d tend vers l'infini et soit $c(v)$ sa couleur. Avec grande probabilité $1 - o(1/n^2)$, $c(v) = O(d)$.*

3.4 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.

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

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, [1], 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 3.3.2, 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 9. *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 10. *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 3, il vient :

Proposition 4. *Le rapport entre T et $\log_2 n$ tend vers 1 en probabilité quand $n \to \infty$.*

3.4.2 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 :

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

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

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

Ce qui permet d'obtenir :

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

Donc, en utilisant le même raisonnement que dans la section 3.3.2 :

Proposition 5.

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

Remarque 2. *Pour $p = \frac{\alpha \log n}{n}$, avec $\alpha > 1$, c'est-à-dire, avec forte probabilité, G est connexe, alors :*

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

3.4.3 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 $v \in V$, $\mathbb{E}\left(L_v\right) = \log_2(n-1) + \frac{1}{2} + \frac{\gamma}{\log 2} + Q\left(\log_2(n-1)\right) + O\left(n^{-2}\right)$.
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 $trie$, [10].
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 3. *Clément et al., dans [10], 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 :*

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

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 4 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.

3.5 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 [15]. Il serait, par conséquent intéressant d'étudier la possibilité de réduire, de manière significative, le nombre de couleurs utilisées.

Bibliography

[1] *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 2000.

[2] Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2009.

[3] Jean-Raymond Abrial, D. Cansell, and D. Méry. A Mechanically Proved and Incremental Development of IEEE 1394 Tree Identify Protocol. *Formal Aspects of Computing*, 14(3):215–227, 2003.

[4] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

[5] Projet ANR-RIMEL. Intégration du temps dans le développement incrémental prouvé. Livrable RIMEL, LORIA, Juillet 2008.

[6] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *FOCS*, pages 364–369. IEEE, 1989.

[7] Ralph Back. On correct refinement of programs. *Journal of Computer and System Sciences*, 23(1):49–68, 1979.

[8] Ralph Back. A calculus of refinements for program derivations. *Acta Informatica*, 25:593–624, 1998.

[9] Ralph Back and Joachim von Wright. *Refinement Calculus A Systematic Introduction*. Graduate Texts in Computer Science. Springer, 1998.

[10] J. Clément, Ph. Flajolet, and B. Vallée. Dynamical sources in information theory: A general analysis of trie structures. *Algorithmica*, 29(1/2):307369, 2001.

[11] M. Devillers, D. Griffioen, J. Romin, and F. Vaandrager. Verification of a Leader Election Protocol: Formal Methods Applied to IEEE 1394. *Formal Methods in System Design*, 16:307–320, 2000. Kluwer Academic Publishers.

[12] P. Flajolet and R. Sedgewick. The average case analysis of algorithms: Mellin transform asymptotics. Technical Report RR-2956.

[13] Stefan Hallerstede and Thai Son Hoang. Qualitative probabilistic modelling in event-b. In Jim Davies and Jeremy Gibbons, editors, *IFM*, volume 4591 of *Lecture Notes in Computer Science*, pages 293–312. Springer, 2007.

[14] IEEE. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*, August 1995.

[15] Ö. Johansson. Simple distributed $(\Delta + 1)$-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.

[16] 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)*, pages 266–272. ACM Press, 1984.

[17] 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.

[18] 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.

[19] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *DISC*, pages 273–287, 2005.

[20] Gary T. Leavens, Jean-Raymond Abrial, Don Batory, Michael Butler, Alessandro Coglio, Kathi Fisler, Eric Hehner, Cliff Jones, Dale Miller, Simon Peyton-Jones, Murali Sitaraman, Douglas R. Smith, and Aaron Stump. Roadmap for enhanced languages and methods to aid verification. In *Fifth Intl. Conf. Generative Programming and Component Engineering (GPCE 2006)*, pages 221–235. ACM, October 2006.

[21] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21:193–201, 1992.

[22] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1053, 1986.

[23] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[24] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement And Proof For Probabilistic Systems (Monographs in Computer Science)*. SpringerVerlag, 2004.

[25] Y. Métivier, J.-M. Robson, N. Saheb-Djahromi, and A. Zemmari. Analysis of an optimal bit complexity randomised distributed vertex colouring algorithm. In *13th International Conference On Principles Of DIstributed Systems (OPODIS 2009)*, 2009. à paraître.

[26] Y. Métivier, J.-M. Robson, N. Saheb-Djahromi, and A. Zemmari. Analysis of an optimal bit complexity randomised distributed vertex colouring algorithm. In *13th International Conference On Principles Of DIstributed Systems (OPODIS 2009)*, 2009. à paraître.

[27] 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.

[28] Yves Métivier, Nasser Saheb, and Akka Zemmari. Randomized local elections. *Inf. Process. Lett.*, 82(6):313–320, 2002.

[29] Carroll Morgan, Thai Son Hoang, and Jean-Raymond Abrial. The challenge of probabilistic *event b* - extended abstract. In Helen Treharne, Steve King, Martin C. Henson, and Steve A. Schneider, editors, *ZB*, volume 3455 of *Lecture Notes in Computer Science*, pages 162–171. Springer, 2005.

[30] 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)*, pages 148–157. ACM Press, 2005.

[31] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[32] Prism. http://www.prismmodelchecker.org/.

[33] Michael O. Rabin. N-process mutual exclusion with bounded waiting by $4log_2$ n-valued shared variable. *J. Comput. Syst. Sci.*, 25(1):66–75, 1982.

[34] Joris Rehm. *Gestion du temps par le raffinement*. PhD thesis, Université Henri poincaré Nancy 1, Decembre 2009.

[35] R. Wattenhofer. http://dcg.ethz.ch/lectures/fs08/distcomp/lecture/chapter4.pdf. 2007.